

EG'07 | **Eurographics 2007**
PRAGUE | **September 3-7, 2007**
Prague, Czech Republic

Tutorial Notes

Volume 1

Content: T1, T2, T5, T6, T9, T10

Monday, 3rd September 2007

Karol Myszkowski and Vlastimil Havran
Tutorial Chairs, Eurographics 2007

Published by
The Eurographics Association
ISSN 1017-4656

©2007	The Eurographics Association
ISSN	1017-4656
Produced by:	Computer Graphics Group Department of Computer Science and Engineering Faculty of Electrotechnical Engineering Czech Technical University in Prague Karlovo nám. 13, 12135 Prague 2, Czech Republic
Print Preparation:	Vlastimil Havran
Cover Design:	Diana Deléková
Printed by:	APOLYS, Kolbenova 609/40, Prague 9

Preface

Over the years, tutorials have built a strong reputation of being an important part of the Eurographics conference. In these tutorials, conceptual and implementation aspects of recent techniques are analyzed in depth by leading experts in the field.

This year the tutorial program is composed of 12 half-day tutorials, which span a wide range of topics including: shape modeling, physically-based modeling and simulation, haptic interfaces, virtual crowds, 3D video, reflectance acquisition, computational photography, rendering, high-performance parallel computing, and visual data mining. While the selected topics are very broad, several tutorials are provided in the context of virtual human modeling. Another group of tutorials present new trends in imaging and rendering. Thus, we informally split the tutorial program into three tracks: Virtual Humanoid, Imaging and Rendering, and Miscellaneous.

We are grateful to everyone who has submitted a tutorial proposal to EG'07. We also would like to thank all reviewers for their valuable comments and expertise, which greatly helped in the tutorial selection process. We want to express our gratitude to the members of the technical team Stefanie Behnke and René Berndt for their continuous support with the on-line submission system. Finally, we are indebted to Jiří Žára, EG'07 co-chair, for his continuous support and advice.

Karol Myszkowski and Vlastimil Havran

August 2007

List of Reviewers

Aubrecht, Petr	Kordík, Pavel
Baciu, George	Lensch, Hendrik
Barbagli, Federico	Magnor, Marcus
Becks, Andreas	Matusik, Wojciech
Belyaev, Alexander	Mikovec, Zdenek
Benthin, Carsten	Orkisz, Maciej
Berka, Roman	Otaduy, Miguel A.
Čadík, Martin	Reiterer, Harald
Cheung, German	Scholz, Volker
Chrysanthou, Yiorgos	Slater, Mel
Decoret, Xavier	Smolic, Aljoscha
Gagalowicz, Andre	Tecchia, Franco
Goesele, Michael	Teschner, Matthias
Gumhold, Stefan	Theisel, Holger
HO, Edmond S. L.	Ward, Greg
Klatzky, Roberta	Žára, Jiří
Klíma, Martin	Ziegler, Gernot

Table of Contents - Volume 1

T1 Haptic Simulation, Perception and Manipulation of Deformable Objects	Vol.1/1
Organizers Nadia Magnenat-Thalmann (MIRALab, University of Geneva) Ugo Bonanni (MIRALab, University of Geneva)	
Speakers Nadia Magnenat-Thalmann (MIRALab, University of Geneva) Ian Summers (Biomedical Physics Research Group, University of Exeter) Massimo Bergamasco (PERCRO, Scuola Superiore SantAnna)	
T2 Populating Virtual Environments with Crowds	Vol.1/25
Organizers Daniel Thalmann (EPFL, Lausanne, Switzerland) Carol O'Sullivan (Trinity College, Dublin, Ireland)	
Speakers Daniel Thalmann (EPFL, Lausanne, Switzerland) Carol O'Sullivan (Trinity College, Dublin, Ireland) Barbara Yersin (EPFL, Lausanne, Switzerland) Jonathan Maim (EPFL, Lausanne, Switzerland) Rachel McDonnell (Trinity College, Dublin, Ireland)	
T5 New Trends in 3D Video	Vol.1/125
Organizers Christian Theobalt (Stanford University, USA) Stephan Würmlin (LiberoVision AG, Switzerland)	
Speakers Christian Theobalt (Stanford University, USA) Stephan Würmlin (LiberoVision AG, Switzerland) Edilson de Aguiar (MPI Informatik, Germany) Christoph Niederberger (LiberoVision AG, Switzerland)	
T6 Capturing Reflectance - From Theory to Practice	Vol.1/365
Organizers Hendrik P. A. Lensch (MPI Informatik) Michael Goesele (University of Washington) Gero Müller (Bonn University)	
Speakers Hendrik P. A. Lensch (MPI Informatik) Michael Goesele (University of Washington) Gero Müller (Bonn University)	
T9 Programming the Cell BE for High Performance Graphics	Vol.1/439
Organizers Michael McCool (RapidMind Inc.) Bruce D'Amora (IBM T.J. Watson Research Center)	
Speakers Michael McCool (RapidMind Inc.) Bruce D'Amora (IBM T.J. Watson Research Center)	
T10 Advanced Topics in Virtual Garment Simulation	Vol.1/531
Organizers Bernhard Thomaszewski (WSI/GRIS, University of Tübingen) Markus Wacker (Computer graphics, University of Applied Sciences Dresden) Wolfgang Straßer (WSI/GRIS, University of Tübingen)	
Speakers Bernhard Thomaszewski (WSI/GRIS, University of Tübingen) Markus Wacker (Computer graphics, University of Applied Sciences Dresden) Nadia Magnenat-Thalmann (MiraLab, Geneva) Etienne Lyard (MiraLab, Geneva)	

Table of Contents - Volume 2

T3 Towards the Virtual Physiological Human	Vol.2/593
Organizers Nadia Magnenat-Thalmann (MIRALab, University of Geneva)	
Speakers Nadia Magnenat-Thalmann (MIRALab/University of Geneva) Benjamin Gilles (MIRALab/University of Geneva) Hervé Delingette (INRIA, Asclepios) Andrea Giachetti (CRS4, Visual computing group) Marco Agus (CRS4, Visual computing group)	
T4 Inverse Kinematics and Kinetics for Virtual Humanoids	Vol.2/643
Organizers Ronan Boulic (EPFL) Richard Kulpa (M2S)	
Speakers Ronan Boulic (EPFL) Richard Kulpa (M2S)	
T7 Computational Photography	Vol.2/743
Organizers Ramesh Raskar (MERL) Jack Tumblin (Northwestern University)	
Speakers Ramesh Raskar (MERL) Jack Tumblin (Northwestern University)	
T8 Applications of Information Theory to Computer Graphics	Vol.2/813
Organizers Mateu Sbert (University of Girona, Spain) Miquel Feixas (University of Girona, Spain)	
Speakers Mateu Sbert (University of Girona, Spain) Miquel Feixas (University of Girona, Spain) Jaume Rigau (University of Girona, Spain) Ivan Viola (University of Bergen, Norway) Miguel Chover (Jaume I University, Spain)	
T11 Visual Mining of Text Collections	Vol.2/929
Organizers Rosane Minghim (University of Sao Paulo, Brazil) Haim Levkowitz (University of Massachusetts Lowell, USA)	
Speakers Rosane Minghim (University of Sao Paulo, Brazil) Haim Levkowitz (University of Massachusetts Lowell, USA)	
T12 3D Shape Description and Matching Based on Properties of Real Functions	Vol.2/1023
Organizers Bianca Falcidieno (CNR - IMATI - GE, Italy) Michela Spagnuolo (CNR - IMATI - GE, Italy)	
Speakers Bianca Falcidieno (CNR - IMATI - GE, Italy) Daniela Giorgi (CNR - IMATI - GE, Italy) Simone Marini (CNR - IMATI - GE, Italy) Giuseppe Patan (CNR - IMATI - GE, Italy) Michela Spagnuolo (CNR - IMATI - GE, Italy)	

Haptic Simulation, Perception and Manipulation of Deformable Objects

Eurographics 2007 Tutorial T1

Organizers

Nadia Magnenat-Thalmann (MIRALab, University of Geneva)
Ugo Bonanni (MIRALab, University of Geneva)

Speakers

Nadia Magnenat-Thalmann (MIRALab, University of Geneva)
Ian Summers (Biomedical Physics Research Group, University of Exeter)
Massimo Bergamasco (PERCRO, Scuola Superiore SantAnna)

EG:2

Haptic Simulation, Perception and Manipulation of Deformable Objects

N. Magnenat-Thalmann¹, P. Volino¹, U. Bonanni¹, I.R. Summers², A. C. Brady², J. Qu²,
D. Allerkamp³, M. Fontana⁴, F. Tarri⁴, F. Salsedo⁴, M. Bergamasco⁴

¹ MIRALab, University of Geneva, Switzerland

² Biomedical Physics Research Group, University of Exeter, U.K.

³ WelfenLab, Leibniz Universität Hannover, Germany

⁴ PERCRO, Scuola Superiore Sant'Anna, Italy

Abstract

This tutorial addresses haptic simulation, perception and manipulation of complex deformable objects in virtual environments (VE). We first introduce HAPTEX, a research project dealing with haptic simulation and perception of textiles in VEs. Then, we present state-of-the-art techniques concerning haptic simulation and rendering, ranging from physically based modelling to control issues of tactile arrays and force-feedback devices. In the section on cloth simulation for haptic systems we describe techniques for simulating textiles adapted to the specific context of haptic applications. The section concerning tactile aspects of virtual objects shows how arrays of contactors on the skin can be used to provide appropriate spatiotemporal patterns of mechanical excitation to the underlying mechanoreceptors. Finally, the last section addresses the problem of developing suitable force feedback technologies for the realistic haptic rendering of the physical interaction with deformable objects, addressing the design of novel force feedback systems, innovative concepts for curvature simulation and control algorithms for accuracy improvement.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Virtual Reality

1. Introduction

Research on multimodal simulation in virtual environments faces the challenge of reproducing the aspect and behaviour of real objects. The simulation should be as realistic as possible and take place within a virtual reality (VR) system which provides the user with multiple interfacing modes (such as vision, audio, and interaction devices). Multimodality typically addresses the stimulation of different channels of perception. In this context, some perceptual channels have been more exploited than others. The achievement of a high degree of **visual** realism is increasingly becoming more popular in the entertainment industry, where video games are offering an always improved experience to the user. This is particularly supported by the establishment of dedicated Graphics Processing Units (GPUs) included in high-end graphics cards featuring programmable shaders. Also 3D spatialised **sound** has become common in the last years, and audio surround facilities in CAVE systems and even in home theatres are widely used. The ability to **touch** virtual objects, however, has not been fully exploited so far. The integration of realistic force-feedback and tactile stimulation within virtual reality applications is far less satisfying than audio-visual integration, and still at the beginning.

This tutorial deals with the reproduction of the sense of touch within virtual reality environments. In this context, we will present how to simulate, perceive and manipulate complex deformable objects such as virtual textiles both from the visual and the haptic viewpoint.

1.1 Reproducing the sense of touch

The discipline dealing with technology interfacing the user via the sense of touch is called **haptics**. A main obstacle to the widespread adoption of haptic devices within ordinary VR systems is currently represented by the unavailability of efficient-and-affordable haptic devices. But there are also other factors which preclude the application development in this domain, such as the high complexity and computational costs linked to haptic simulation.

In order to be performed accurately, multimodal simulation addressing vision and touch involves a high load on the computer's processing units. It is therefore important to find the best trade-off between the simulation's realism (in terms of visual and physical accuracy) and performance (in terms of response latency). To optimize the resource management, the visual and the haptic sensory channels can be processed in separate layers, since they have different requirements in terms of update rates or relevant physical properties to be simulated. However, this practice requires a robust and stable coupling between the two modalities [AH98]. The synchronization between layers must occur in real time, because delays or asynchronous behaviour can strongly affect the believability of the user experience.

1.2 Rendering complex deformable objects

The research concerning new ways of rendering virtual objects both visually and haptically in a fast and stable way represents a particular challenge when dealing with

physically based, complex deformable objects. In this field, researchers need to reproduce the object's aspect and behaviour in a physically accurate way and provide a simulation model able to calculate the deformations of the object occurring during interaction. Typically, the simulated interaction is tool-based, i.e. the user interacts with the object indirectly, feeling the forces arising during manipulation only at one specific point.

Complexity increases in the case of multipoint interactions, as collisions and deformations must be computed for each contact point. Direct haptic interaction, e.g. the simulation of the real contact between the human hand and a deformable object, is very demanding not only because of the number of deformations arising all over the contact surface and affecting each other, but also because of the technical difficulties in rendering the contact forces over a distributed area.

The haptic response rendered to the user of a multimodal simulation system can be of different nature. In this tutorial, we deal mainly with tactile and kinaesthetic feedback. Tactile arrays can reproduce the properties and small scale details of the object properties by selectively stimulating the mechanoreceptors under the skin of the fingertips. Tactile arrays give a feedback to the user, but do not allow him to exert actively forces. Force-feedback devices, in contrast, allow the user to add energy to the multimodal simulation system and actively manipulate virtual objects. Moreover, force-feedback devices reflect forces acting on the simulated object and allow the user to obtain kinaesthetic feedback about the performed operations.

2. Touching virtual textiles

Textile is an ideal deformable object to render in the context of haptic simulation. Humans are inherently familiar with clothes, and used to handle clothing materials since prehistoric ages. However, while cloth simulation is a popular topic in computer graphics, there have been very little attempts to render textiles haptically [GPU*03] [Hua02]. In these cases, only static cloth has been taken into consideration. Animated textiles were not taken into account because of the high requirements posed by the real time animation of cloth.

Interestingly, these early attempts have tried to combine the results of studies on physical properties of fabrics done in the textile industry with garment simulation. However, only a very limited amount of fabric parameters was taken into consideration. This can be easily ascribed to one of the limitations we face today in the field of computer graphics and haptics: the high number of physical properties we are able to feel and discriminate with our sensorial system can't be realistically and exhaustively reproduced by a multimodal simulation system today. These limitations concern both the visual and the haptic aspect of the simulation, and affect not only textiles but all kinds of physically based deformable objects. It is therefore necessary to simplify the usage of physical parameters by identifying a finite number of properties that can be considered the most relevant for an approximate but realistic simulation of the handling of a specific deformable object. Moreover, this simplification is necessary for each channel of perception, since the description of an object's behaviour from the visual, tactile or kinaesthetic aspect has different requirements.

2.1 The HAPTEX Project

The European research project HAPTEX (HAPtic sensing of virtual TEXtiles) tackles several of the above mentioned

challenges. The goal of HAPTEX is to provide a multimodal system able to simulate virtual textiles in real time, allowing multipoint haptic interaction with a piece of virtual fabric [SFR*05]. In the HAPTEX system, haptic manipulation takes place through a novel haptic interface, which provides both force- and tactile feedback and aims to reproduce the feeling of touching a cloth surface with two fingertips.

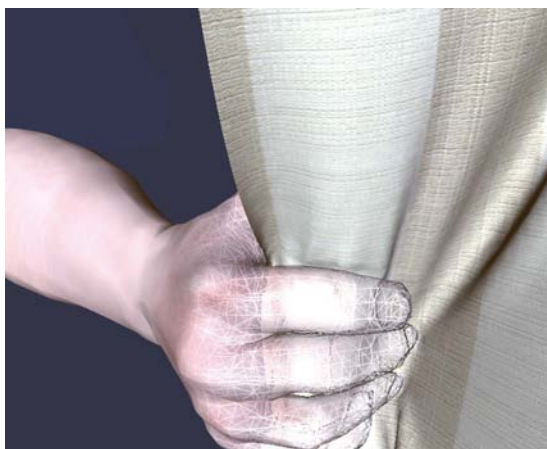


Figure 1: The HAPTEX showcase

The system simulates the large-scale motion of a square of fabric hanging from a stand in virtual space, accurately described by its mechanical properties (such as stiffness or elasticity). These large-scale forces are returned by a force feedback device which allows for haptic manipulation. Moreover, tactile stimuli are derived from the fabric's small scale properties (texture and roughness) and rendered by piezoelectric tactile arrays integrated in the force-feedback device.

Users of the HAPTEX System can perform actions such as touching, stroking and stretching the virtual garment, selecting the simulated textile from a range of samples and feeling the different physical characteristics between them. The HAPTEX System allows to perform different textile handling actions, depicted in the following figure.

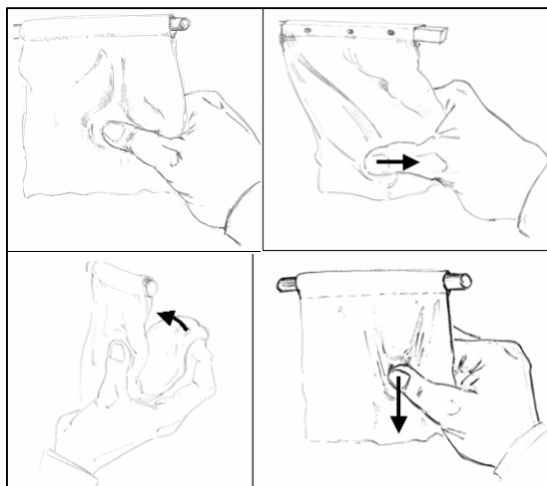


Figure 2: Envisaged handling actions

2.2 Design of the haptic system

In the HAPTEX System, both the visual simulation and the haptic rendering are based on the physical properties of real textiles. Modelling the behaviour of textiles is a complex task because of its dependency on several parameters such as flexibility, compressibility, elasticity, resilience, density, surface contour (roughness, smoothness), surface friction and thermal character [MMLMT05]. The process of handling fabrics to understand their properties and structure is called “fabric hand”. Understanding the way people are used to handle the objects to simulate (in this case, textiles) is of crucial relevance for designing and developing a haptic system. The HAPTEX approach is to analyze all perceptual and practical implications of fabric hand, in order to derive a set of requirements to the system. From these ideal requirements, the system is realized according to the possibilities offered by today’s technology [Hap06a]. The research and developments done in the context of the HAPTEX project cover textile measurements, real-time cloth simulation, tactile interfaces, force-feedback devices, haptic rendering (both tactile and force-feedback) and the integration of the complete haptic system.

2.3 Components of the HAPTEX System

The HAPTEX System is mainly composed by the following components [MTB06] [Hap05a]:

Measured physical parameters: The “Kawabata Evaluation System for Fabrics” (KES-F) is one of the main standards in the field of objective measurements of fabric hand [Kaw80]. The KES-F equipment is able to test for textile properties and extract physical parameters of textiles. These vary depending on the fibre type or fabric type and dimension [Hap05b]. Alternatively, other equipments such as tensile testers can be used to obtain specific physical parameters of fabric samples [MMLMT05]. The physical parameters are used by the cloth simulation, the tactile renderer and the force-feedback renderer.

Cloth simulation: The HAPTEX textile simulation is driven by a mechanical model which takes as input part of the mechanical parameters obtained from measurements on fabrics [VDB*07]. See Section 3 for more details.

Tactile component: The tactile array generates impulses of mechanical excitation for the mechanoreceptors underlying the skin of the user’s fingertip. These spatiotemporal stimuli evoke the sensation of stroking the finger over a surface and feeling its patterns and edges. A tactile renderer returns drive signals for the array on the basis of the user’s movements and a model of the finger/object interaction [Hap06a] [Hap06b]. See Section 4 for more details.

Force-feedback component: The force-feedback device returns to the user the forces acting on the manipulated piece of textile. A force-feedback renderer takes care of the computation of forces to be returned on the basis of the mechanical model of the cloth simulation. The force-feedback device also hosts the tactile component. Different configurations of the force-feedback component have been realized in the context of the HAPTEX Project [Hap06a] [Hap06b]. See Section 5 for more details.

3. Cloth Simulation in Haptic Systems

As any simulation system, cloth simulation requires significant computational resources for being performed accurately. Its integration in real-time simulation systems

requires the implementation of state-of-the-art techniques, for mechanical models as well as for numerical integration. Furthermore, haptic applications require high robustness for dealing with approximate tracking and highly variable frame rates inherent to the performance and artefacts of current motion tracking devices. In this section, we describe some techniques for cloth simulation adapted to the specific context of haptic applications.

3.1 Overview

Garment simulation for interactive applications still remains a challenge, and the challenge is mainly to combine state-of-the-art simulation techniques that offer the best trade-off between computational speed, accuracy and robustness.

The particular challenges are described in the following sections. These are mainly:

- * The design of a fast simulation system for simulating the tensile and bending elastic properties of cloth materials, which may possibly be anisotropic and nonlinear.
- * The implementation of an efficient numerical integrator that offers robust simulation ensuring stability despite possible irregular frame rates and other artefacts related to motion tracking techniques.

3.2 Simulating the Mechanics of Cloth

3.2.1 Mechanical Properties of Cloth

The mechanical properties of deformable surfaces can be grouped into four main families:

- * **Elasticity**, which characterizes the internal forces resulting from a given geometrical deformation.

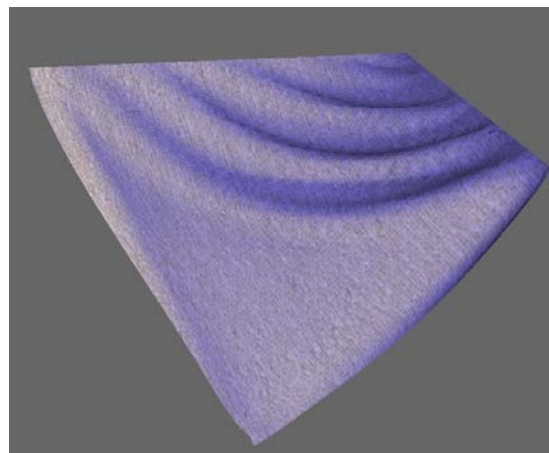


Figure 3: Piece of textile simulated in real-time

- * **Viscosity**, which includes the internal forces resulting from a given deformation speed.
- * **Plasticity**, which describes how the properties evolve according to the deformation history.

Most important are the elastic properties that are the main contributor of mechanical effects in the usual contexts where cloth objects are used. In the context of haptic applications, the motion of the cloth also depends on

dissipative effects related to viscosity and plasticity, which therefore have to be taken into account up to some degree of approximation by the simulation system.

Depending on the amplitude of the mechanical phenomena under study, the curves expressing mechanical properties exhibit shapes of varying complexity. If the amplitude is small enough, these shapes may be approximated by straight lines. This **linearity** hypothesis is a common way to simplify the characterization and modelling of mechanical phenomena.

It is common in elasticity theory to consider that the orientation of the material has no effect on its mechanical properties (**isotropy**). This however is inappropriate for cloth, as its properties depend considerably on their orientation relative to the fabric thread.

Elastic effects can be divided into several contributions:

- * **Metric elasticity**, deformations along the surface plane.
- * **Bending elasticity**, deformations orthogonally to the surface plane.

Metric elasticity is the most important and best studied aspect of fabric elasticity. It is usually described in terms of strain-stress relations. For linear elasticity, the main laws relating the strain ϵ to the stress s involve three parameters, which are:

- * **The Young modulus E**, summarizing the material's reaction along the deformation direction.
- * **The Poisson coefficient ν** , characterizing the material's reaction orthogonal to the deformation direction.
- * **The Rigidity modulus G**, pertaining to oblique reactions.

Along the two orthogonal directions \mathbf{i} and \mathbf{j} , these relations, named **Hook's Law**, **Poisson Law** and **Simple Shear Law** relating the stress ϵ to the strain σ are respectively expressed as follows:

$$\epsilon_{ii} = \frac{1}{E_i} \sigma_{ii} \quad \epsilon_{jj} = \frac{\nu_{ij}}{E_i} \sigma_{ii} \quad \epsilon_{ij} = \frac{1}{G_{ij}} \sigma_{ij} \quad (1)$$

Cloth materials are two-dimensional surfaces for which two-dimensional variants of the elasticity laws are suitable. They are not isotropic, but the two orthogonal directions defined by the thread orientations can be considered as the main orientations for any deformation properties. In these **orthorhombic** cloth surfaces, the two directions are called **weft (u)** and **warp (v)**, and they have specific Young modulus and Poisson coefficients, E_u, ν_u and E_v, ν_v , respectively. The elasticity law can be rewritten in terms of these directions as follows:

$$\begin{bmatrix} \sigma_{uu} \\ \sigma_{vv} \\ \sigma_{uv} \end{bmatrix} = \frac{1}{1 - \nu_u \nu_v} \begin{bmatrix} E_u & \nu_v E_u & 0 \\ \nu_u E_v & E_v & 0 \\ 0 & 0 & G(1 - \nu_u \nu_v) \end{bmatrix} \begin{bmatrix} \epsilon_{uu} \\ \epsilon_{vv} \\ \epsilon_{uv} \end{bmatrix} \quad (2)$$

Energetic considerations imply the above matrix to be symmetric, and therefore the products $E_u \nu_v$ and $E_v \nu_u$ are equal. Considering isotropic materials, we also have the following relations:

$$E_u = E_v \nu_u = \nu_v G = \frac{E}{2(1 - \nu)} \quad (3)$$

A similar formulation can be obtained for bending elasticity. However the equivalent of the Poisson coefficient for bending is usually taken as null. The relation between the curvature strain γ and stress τ is expressed using the flexion modulus B and the flexion rigidity K (often taken as null) as follows:

$$\begin{bmatrix} \tau_{uu} \\ \tau_{vv} \\ \tau_{uv} \end{bmatrix} = \begin{bmatrix} B_u & 0 & 0 \\ 0 & B_v & 0 \\ 0 & 0 & K \end{bmatrix} \begin{bmatrix} \gamma_{uu} \\ \gamma_{vv} \\ \gamma_{uv} \end{bmatrix} \quad (4)$$

While elasticity expresses the relation between the force and the deformation, viscosity expresses the relation between the force and the deformation speed in a very similar manner. To any of the elasticity parameters can be defined a corresponding viscosity parameter obtained by substitution of the stresses ϵ and γ by their derivatives along time ϵ' and γ' .

While the described linear laws are valid for small deformations of the cloth, large deformations usually enter the nonlinear behaviour of cloth, where there is no more proportionality between strain and stress. This is practically observed by observing a "limit" in the cloth deformation as the forces increases, often preceding rupture (resilience), or remnant deformations observed as the constraints are released (plasticity). A common way to deal with such nonlinear models is to assume weft and warp deformation modes as still being independent, and replace each linear parameter E_u, E_v, G, B_u, B_v by nonlinear strain-stress behaviour curves.

3.2.2 Measuring the Mechanical Properties of Cloth

The garment industry needs the measurement of major fabric mechanical properties through normalized procedures that guarantee consistent information exchange between garment industry and cloth manufacturers. The **Kawabata Evaluation System for Fabric (KES)** is a reference methodology for the experimental observation of the elastic properties of the fabric material. Using five experiments, fifteen curves are obtained, which then allow the determination of twenty-one parameters for the fabric, among them all the linear elastic parameters described above, except for the Poisson coefficient.

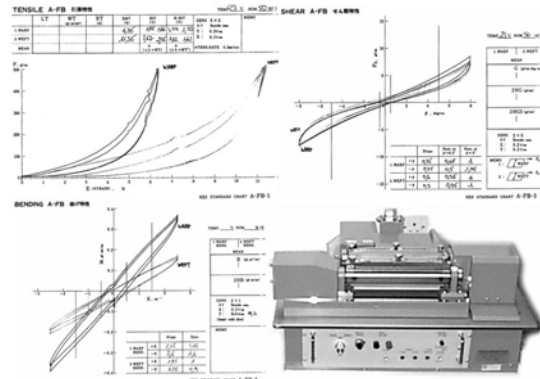


Figure 4: Measuring cloth properties using KES.

Five standard tests are part of KES for determining the mechanical properties of cloth, using normalized measurement equipment. The **tensile test** measures the force/deformation curve of extension for a piece of fabric of normalized size along weft and warp directions and allows the measurement of \mathbf{E}_u and \mathbf{E}_v along with other parameters assessing nonlinearity and hysteresis. The **shearing test** is the same experiment using shear deformations, which allows the measurement of \mathbf{G} . The **bending test** measures the curves for bending deformation in a similar way, and allows the measurement of \mathbf{B}_u and \mathbf{B}_v . Finally, the **compression test** and the **friction test** allow the measurement of parameters related to the compressibility and the friction coefficients.

While the KES measurements allow determination of parameters assessing the nonlinearity of the behaviour curves and some evaluation of the plasticity, other methodologies, such as the FAST method, use simpler procedures to determine the linear parameters only.

While the Kawabata measurements and similar systems summarize the basic mechanical behaviours of fabric material, the visual deformations of cloth, such as buckling and wrinkling, are a complex combination of these parameters with other subtle behaviours that cannot be characterized and measured directly.

In order to take these effects into account, other tests focus on more complex deformations. Among them, the draping test considers a cloth disk of given diameter draped onto a smaller horizontal disc surface. The edge of the cloth will fall around the support, and produce wrinkling. The wrinkle pattern can be measured (number and depth of the wrinkles) and used as a validation test for simulation models.

Tests have also been devised for measuring other complex deformation of fabric material, mostly related to bending, creasing and wrinkling.

3.3 Cloth Simulation Systems

Cloth being approximated as a thin surface, its mechanical behaviour is decomposed in in-plane deformations (the 2D deformations along the cloth surface plane) and bending deformation (the 3D surface curvature).

The in-plane behaviour of cloth is described by relationships relating, for any cloth element, the stress $\boldsymbol{\sigma}$ to the strain $\boldsymbol{\epsilon}$ (for elasticity) and its speed $\boldsymbol{\epsilon}'$ (for viscosity) according the laws of viscoelasticity. For cloth materials, strain and stress are described relatively to the weave directions weft and warp following three components: weft and warp elongation (\mathbf{uu} and \mathbf{vv}), and shear (\mathbf{uv}). Thus, the general viscoelastic behaviour of a cloth element is described by strain-stress relationships as follows:

$$\begin{aligned} \sigma_{uu}(\epsilon_{uu}, \epsilon_{vv}, \epsilon_{uv}, \epsilon'_{uu}, \epsilon'_{vv}, \epsilon'_{uv}) \\ \sigma_{vv}(\epsilon_{uu}, \epsilon_{vv}, \epsilon_{uv}, \epsilon'_{uu}, \epsilon'_{vv}, \epsilon'_{uv}) \\ \sigma_{uv}(\epsilon_{uu}, \epsilon_{vv}, \epsilon_{uv}, \epsilon'_{uu}, \epsilon'_{vv}, \epsilon'_{uv}) \end{aligned} \quad (5)$$

Assuming to deal with an orthotropic material (usually resulting from the symmetry of the cloth weave structure relatively to the weave directions), there is no dependency between the elongation components (\mathbf{uu} and \mathbf{vv}) and the shear component (\mathbf{uv}). Assuming null Poisson coefficient as well (a rough approximation), all components are independent, and the fabric elasticity is simply described by

three independent elastic strain-stress curves (weft, warp, shear), along with their possible viscosity counterparts.

In the same manner, viscoelastic strain-stress relationships relate the bending momentum to the surface curvature for weft, warp and shear. With the typical approximations used with cloth materials, the elastic laws are only two independent curves along weft and warp directions (shear is neglected), with their possible viscosity counterparts.

The issue is now to define a model for representing these mechanical properties on geometrical surfaces representing the cloth. These curved surfaces are typically represented by polygonal meshes, being either triangular or quadrangular, and regular or irregular.

Continuum mechanics are one of the schemes used for accurate representation of the cloth mechanics. Mechanical equations are expressed along the curved surface, and then discretised for their numerical resolution. Such accurate schemes are however slow and not sufficiently versatile for handling large deformations and complex geometrical constraints (collisions) properly. Finite Element methods express the mechanical equations according to the deformation state the surface within well-defined elements (usually triangular or quadrangular). Their resolution also involves large computational charges. Another option is to construct a model based on the interaction of neighbouring discrete points of the surface. Such particle systems allow the implementation of simple and versatile models adapted for efficient computation of highly deformable objects such as cloth.

3.3.1 Spring-Mass Models

The simplest particle system one can think of is spring-mass systems. In this scheme, the only interactions are forces exerted between neighbouring particle couples, similarly as if they were attached by springs (described by a force/elongation law along its direction, which is actually a rigidity coefficient and a rest length in the case of linear springs). Spring-mass schemes are very popular methods, as they allow simple implementation and fast simulation of cloth objects. There has also been recent interest in this method as it allows quite a simple computation of the Jacobian of the spring forces, which is needed for implementing semi-implicit integration methods (see Section 3.4).

The simplest approach is to construct the springs along the edges of a triangular mesh describing the surface. This however leads to a very inaccurate model that cannot model accurately the anisotropic strain-stress behaviour of the cloth material, and also not the bending. More accurate models are constructed on regular square particle grids describing the surface. While elongation stiffness is modelled by springs along the edges of the grid, shear stiffness is modelled by diagonal springs and bending stiffness is modelled by leapfrog spring along the edges. This model is still fairly inaccurate because of the unavoidable cross-dependencies between the various deformation modes relatively to the corresponding springs. It is also inappropriate for nonlinear elastic models and large deformations. More accurate variations of the model consider angular springs rather than straight springs for representing shear and bending stiffness, but the simplicity of the original spring-mass scheme is then lost.

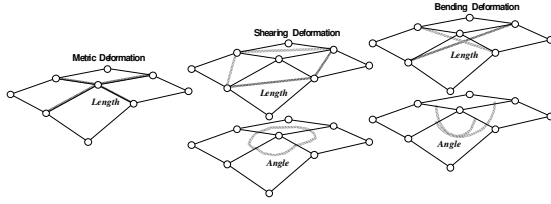


Figure 5: Using length or angle springs for simulating cloth with a square particle system grid.

3.3.2 Accurate Particle System for Tensile Viscoelasticity

Because of the real need of representing accurately the anisotropic nonlinear mechanical behaviour of cloth in garment prototyping applications, spring-mass models are inadequate, and we need to find out a scheme that really simulates the viscoelastic behaviour of actual surfaces. For this, we have defined a particle system model that relates this accurately over any arbitrary cloth triangle through simultaneous interaction between the three particles which are the triangle vertices. Such a model integrates directly and accurately the strain-stress model defined in Part 2.1 using polynomial spline approximations of the strain-stress curves, and remains accurate for large deformations.

In this model, a triangle element of cloth is described by 3 2D coordinates $(\mathbf{u}\mathbf{a}, \mathbf{v}\mathbf{a})$, $(\mathbf{u}\mathbf{b}, \mathbf{v}\mathbf{b})$, $(\mathbf{u}\mathbf{c}, \mathbf{v}\mathbf{c})$ describing the location of its vertices \mathbf{A} , \mathbf{B} , \mathbf{C} on the weft-warp coordinate system defined by the directions \mathbf{U} and \mathbf{V} with an arbitrary origin. They are orthonormal on the undeformed cloth (Figure M4). Out of them, a precomputation process evaluates the following values:

$$\begin{aligned} R_{ua} &= d^{-1} (vb - vc) & R_{va} &= -d^{-1} (ub - uc) \\ R_{ub} &= -d^{-1} (va - vc) & R_{vb} &= d^{-1} (ua - uc) \\ R_{uc} &= d^{-1} (va - vb) & R_{vc} &= -d^{-1} (ua - ub) \end{aligned}$$

with

$$d = ua (vb - vc) + ub (vc - va) + uc (va - vb) \quad (6)$$

During the computation process, the current deformation state of the cloth triangle is evaluated using the current 3D direction and length of the deformed weft and warp direction vectors \mathbf{U} and \mathbf{V} . They are computed from the current positions $\mathbf{P}\mathbf{a}$, $\mathbf{P}\mathbf{b}$, $\mathbf{P}\mathbf{c}$ of its supporting vertices as follows:

$$\begin{aligned} \mathbf{U} &= R_{ua} \mathbf{P}\mathbf{a} + R_{ub} \mathbf{P}\mathbf{b} + R_{uc} \mathbf{P}\mathbf{c} \\ \mathbf{V} &= R_{va} \mathbf{P}\mathbf{a} + R_{vb} \mathbf{P}\mathbf{b} + R_{vc} \mathbf{P}\mathbf{c} \end{aligned} \quad (7)$$

The current in-plane strains $\boldsymbol{\epsilon}$ of the cloth triangle is then computed with the following formula:

$$\begin{aligned} \epsilon_{uu} &= |\mathbf{U}| - 1 & \epsilon_{vv} &= |\mathbf{V}| - 1 \\ \epsilon_{uv} &= \frac{|\mathbf{U} + \mathbf{V}|}{\sqrt{2}} - \frac{|\mathbf{U} - \mathbf{V}|}{\sqrt{2}} \end{aligned} \quad (8)$$

We have chosen to replace the traditional shear deformation evaluation based on the angle measurement between the thread directions by an evaluation based on the length of the diagonal directions. The main advantage of this is a better accuracy for large deformations (the computation of the behaviour of an isotropic material under large deformations remains more axis-independent).

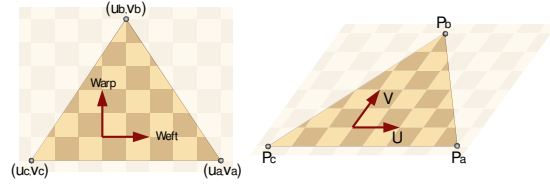


Figure 6: A triangle of cloth element defined on the 2D cloth surface (left) is deformed in 3D space (right) and its deformation state is computed from the deformation of its weft-warp coordinate system.

For applications that model internal in-plane viscosity of the material, the "evolution speeds" of the weave direction vectors are needed as well. They are computed from the current triangle vertex speeds $\mathbf{P}\mathbf{a}'$, $\mathbf{P}\mathbf{b}'$, $\mathbf{P}\mathbf{c}'$ as follows:

$$\begin{aligned} \mathbf{U}' &= R_{ua} \mathbf{P}\mathbf{a}' + R_{ub} \mathbf{P}\mathbf{b}' + R_{uc} \mathbf{P}\mathbf{c}' \\ \mathbf{V}' &= R_{va} \mathbf{P}\mathbf{a}' + R_{vb} \mathbf{P}\mathbf{b}' + R_{vc} \mathbf{P}\mathbf{c}' \end{aligned} \quad (9)$$

Then, the current in-plane strain speeds $\boldsymbol{\epsilon}'$ of the triangle is computed:

$$\begin{aligned} \epsilon'_{uu} &= \frac{\mathbf{U} \cdot \mathbf{U}'}{|\mathbf{U}|} & \epsilon'_{vv} &= \frac{\mathbf{V} \cdot \mathbf{V}'}{|\mathbf{V}|} \\ \epsilon'_{uv} &= \frac{(\mathbf{U} + \mathbf{V}) \cdot (\mathbf{U}' + \mathbf{V}')}{|\mathbf{U} + \mathbf{V}| \sqrt{2}} - \frac{(\mathbf{U} - \mathbf{V}) \cdot (\mathbf{U}' - \mathbf{V}')}{|\mathbf{U} - \mathbf{V}| \sqrt{2}} \end{aligned} \quad (10)$$

At this point, the in-plane mechanical behaviour of the material can be expressed for computing the stresses $\boldsymbol{\sigma}$ out of the strains $\boldsymbol{\epsilon}$ (elasticity) and the strain speeds $\boldsymbol{\epsilon}'$ (viscosity) using the curves discussed in Part 2.2. Finally, the force contributions of the cloth triangle to its support vertices computed from the stresses $\boldsymbol{\sigma}$ as follows:

$$\begin{aligned} \mathbf{F}\mathbf{a} &= -\frac{d}{2} \left((R_{ua} \sigma_{uu} + R_{va} \sigma_{vv}) \frac{\mathbf{U}}{|\mathbf{U}|} + (R_{ua} \sigma_{uv} + R_{va} \sigma_{vu}) \frac{\mathbf{V}}{|\mathbf{V}|} \right) \\ \mathbf{F}\mathbf{b} &= -\frac{d}{2} \left((R_{ub} \sigma_{uu} + R_{vb} \sigma_{vv}) \frac{\mathbf{U}}{|\mathbf{U}|} + (R_{ub} \sigma_{uv} + R_{vb} \sigma_{vu}) \frac{\mathbf{V}}{|\mathbf{V}|} \right) \\ \mathbf{F}\mathbf{c} &= -\frac{d}{2} \left((R_{uc} \sigma_{uu} + R_{vc} \sigma_{vv}) \frac{\mathbf{U}}{|\mathbf{U}|} + (R_{uc} \sigma_{uv} + R_{vc} \sigma_{vu}) \frac{\mathbf{V}}{|\mathbf{V}|} \right) \end{aligned} \quad (11)$$

It is important to note that when using semi-implicit integration schemes (see Section 3), the contribution of these forces in the Jacobian $\partial \mathbf{F} / \partial \mathbf{P}$ and $\partial \mathbf{F} / \partial \mathbf{P}'$ can easily be computed out of the curve derivatives $\partial \boldsymbol{\sigma} / \partial \boldsymbol{\epsilon}$ and the orientation of the vectors \mathbf{U} and \mathbf{V} .

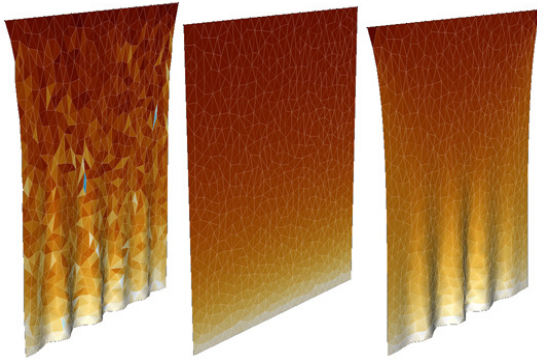


Figure 7: *Drape accuracy between a simple spring-mass system along the edges of the triangle mesh (left) and the proposed accurate particle system model (centre). Colour scale shows deformation. The spring-mass model exhibits inaccurate local deformations, along with an excessive "Poisson" behaviour. This is not the case with the accurate model, which may still model the "Poisson" effect if needed (right, with a Poisson coefficient 0.5). The spring-mass model is also unable to simulate anisotropic or nonlinear models accurately.*

3.3.3 Linear Particle System for Bending Elasticity

Unlike tensile stiffness, bending stiffness necessitates the action of out-of-plane forces that are usually more expensive to compute than in-plane forces.

Several solutions have been proposed in the literature, representing two main approaches. The first is to use crossover springs that extend the surface, opposing transversal bending. This approach, which integrates nicely in any simulation system based on spring-mass, is however very inaccurate. The second is to evaluate precisely the angle between adjacent mesh elements and to create between them normal forces that oppose this angle through opposite bending momentum. This approach can reach similar accuracy as grid continuum-mechanics and grid particle system derivatives which are fairly complex to evaluate. The third is to obtain a completely linear formulation of bending forces by computing it directly as a weighted sum of particle positions, without any consideration of the surface normals or any other complex geometric computation.

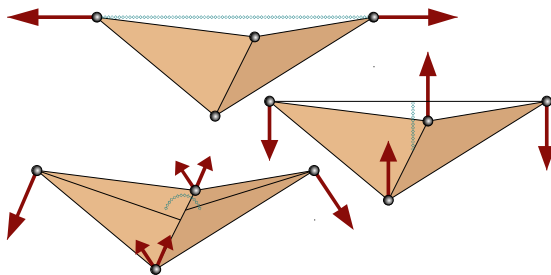


Figure 8: *Three ways for creating bending stiffness in a triangle mesh: Using tensile crossover springs over mesh edges (top), using forces along triangle normals (bottom), and, as we propose, using forces evaluated from a weighted sum of vertex positions (right).*

The idea of the linear model is the following: First, a "bending vector" that represents the bending of the surface is computed through a simple linear combination of particle positions. Then, it is redistributed as particle forces according to the bending stiffness of the surface. This scheme preserves total translational and rotational momentum conservation without the need of recomputing the distribution coefficients according to the current position of the particles. This leads to a very simple computation process which is perfectly linear, and thus very well adapted to implicit numerical integration.

We start from two adjacent triangles (P_A, P_C, P_D) and (P_B, P_D, P_C) . Their common edge has a length noted l and their respective heights relatively to vertices P_A and P_B are noted h_A and h_B .

The two adjacent triangles approximate a curved surface that contains the four vertices of the two triangles, and we assume that the surface is only curved *orthogonally* to the edge (P_C, P_D) (Figure M2 left). This is indeed not an obvious assumption, since *any kind* of surface curvature may produce some bending around the edge. However, this choice can be assumed as being the best, as the direction of the edge bending matches the curvature of the surface.

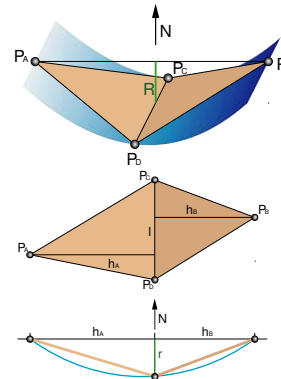


Figure 9: *The curved surface equivalent to two adjacent triangles (left), and the computation of its curvature (right).*

Our goal is now to estimate this curvature from the *height difference* noted r between the edges (P_A, P_B) and (P_C, P_D) . As we restrict ourselves to linear bending, we assume that the bending stiffness remains constant whatever the amount of curvature, and therefore we evaluate it assuming the edge angle between the adjacent triangles remain small. In these conditions, we can evaluate the curvature γ of the surface as follows:

$$\gamma = \frac{2r}{h_A h_B} \quad (12)$$

Now, we need to compute the height difference r from the current position of the triangles. Again in the context of small edge angle, this is approximated through the projected length of a *bending vector* \mathbf{R} on the approximate normal of the surface \mathbf{N} (normalized to unit length) so as:

$$r = \mathbf{R} \cdot \mathbf{N} \quad (13)$$

The bending vector \mathbf{R} indeed represents a kind of "second-order deformation difference" between the two elements, and its normal component represents the actual surface bending. It can be computed as a simple linear combination of vertex positions, as follows:

$$R = \alpha_A P_A + \alpha_B P_B + \alpha_C P_C + \alpha_D P_D \quad (14)$$

With:

$$\alpha_A = \frac{|N_B|}{|N_A| + |N_B|} = \frac{h_B}{h_A + h_B} \quad \alpha_C = -\frac{|N_D|}{|N_C| + |N_D|} \quad (15)$$

$$\alpha_B = \frac{|N_A|}{|N_A| + |N_B|} = \frac{h_A}{h_A + h_B} \quad \alpha_D = -\frac{|N_C|}{|N_C| + |N_D|}$$

Using the normals:

$$N_A = (P_A - P_C) \wedge (P_A - P_D) \quad N_C = (P_C - P_B) \wedge (P_C - P_A) \quad (16)$$

$$N_B = (P_B - P_D) \wedge (P_B - P_C) \quad N_D = (P_D - P_A) \wedge (P_D - P_B)$$

The main idea of our linear bending stiffness scheme is to apply forces on the particles that directly oppose the bending vector \mathbf{R} of the current deformation, *without* projection along \mathbf{N} , or any other intermediate computations that would explicitly evaluate the actual values of the bending strain and stress.

Thus, we consider that the bending forces $\mathbf{F}_A, \mathbf{F}_B, \mathbf{F}_C, \mathbf{F}_D$ are applied on the vertices $\mathbf{P}_A, \mathbf{P}_B, \mathbf{P}_C, \mathbf{P}_D$ respectively along the vector \mathbf{R} . That can be done as follows, with a stiffness coefficient λ that would bring adequate scaling:

$$F_A = -\lambda \alpha_A R \quad F_C = -\lambda \alpha_C R \quad (17)$$

$$F_B = -\lambda \alpha_B R \quad F_D = -\lambda \alpha_D R$$

This distribution, which uses the same coefficients as (14), has been chosen for satisfying total mechanical momentum conservation in the system.

Finally, we need to compute the value of the stiffness coefficient λ according to the linear bending stiffness modulus μ of the surface and the geometry of the triangles.

The bending momentum created by the forces \mathbf{F}_A and \mathbf{F}_B applied on respectively \mathbf{P}_A and \mathbf{P}_B around the edge $(\mathbf{P}_C, \mathbf{P}_D)$ can be expressed from the height difference \mathbf{r} , through (17), (15) and (13) as follows:

$$h_A F_A \cdot N = -h_B F_B \cdot N = \lambda \frac{h_A h_B}{h_A + h_B} R \cdot N = \lambda \frac{h_A h_B}{h_A + h_B} r \quad (18)$$

The bending momentum also results from the bending stiffness modulus μ of the bent surface of curvature γ applied over the length l of the edge $(\mathbf{P}_C, \mathbf{P}_D)$. From this, using (2):

$$l \mu \gamma = \frac{2 l \mu r}{h_A h_B} \quad (19)$$

A non-obvious issue is to take into account how adjacent edge bends combine together for describing the actual surface curvature. Energetic considerations, mainly detailed by [GHDS03] suggest that λ should be evaluated by equating (18) with *one third* of (19). Therefore:

$$\lambda = \frac{2}{3} \frac{h_A + h_B}{(h_A h_B)^2} l \mu \quad (20)$$

It can be demonstrated that Only (14) and (17) are required for having exactly total mechanical momentum conservation, both translational and rotational, *whatever the current position of the particles* and whatever the actual way of computing coefficients, provided that their sum is null. Thus, momentum conservation is not broken by having the coefficients $\alpha_A, \alpha_B, \alpha_C, \alpha_D$ computed *independently*

from the current position of the particles. Therefore, these should be precomputed using the initial shape of the mesh, or the parametric coordinates of the vertices on the cloth. In these conditions, the Jacobian of the bending stiffness forces is simple and straightforward to compute from (14) and (17) without any approximation. This is done as follows, with \mathbf{I} denoting the identity matrix, and with any \mathbf{J} and \mathbf{K} among $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$:

$$\frac{\partial F_J}{\partial P_K} = -\lambda \alpha_J \alpha_K I \quad (21)$$

Thanks to a perfectly linear bending model, the Jacobian of all bending forces is constant and totally independent from the current particle positions. This allows efficient numerical resolution through usual implicit numerical resolution methods with good convergence properties, along with possible algorithmic optimizations for performing the computation quickly.

3.4 Numerical Integration

The equations resulting from the mechanical formulation of particle systems do usually express particle forces \mathbf{F} depending on the state of the system (particle positions \mathbf{P} and speeds \mathbf{P}'). In turn, particle accelerations \mathbf{P}'' is related to particle forces \mathbf{F} and masses \mathbf{M} by Newton's 2nd law of dynamics. This leads to a second-order ordinary differential equation system, which is turned to first-order by concatenation of particle position \mathbf{P} and speed \mathbf{P}' into a state vector \mathbf{Q} . A vast range of numerical methods has been studied for solving this kind of equations.

We have conducted extensive tests for benchmarking numerous integration methods, using performance, accuracy, stability and robustness as criteria. We have selected three candidates, each of which performs best in its own context:

- * *1st-order semi-implicit Backward Euler*, which seems to be the best robust general-purpose method for any relaxation task (garment assembly and draping) [BHW94] [MDDB00].
- * *2nd-order semi-implicit Backward Differential Formula*, which offers increased dynamic accuracy along time (garment simulation on animated characters), at the expense of robustness (unsuited for draping during interactive design) [EDC96].
- * *5th-order explicit Runge-Kutta with timestep control*, which offers very high non-dissipative dynamic accuracy (accurate simulation of viscous and dissipative parameters in animated garments), at the expense of computation time (requires small time steps depending on the numerical stiffness, unsuited for stiff materials and refined discretisations) [BWH*06].

Our implementation integrates these three methods, and dynamically switches between them depending on the simulation context.

3.4.1 Discussing Integration Methods

3.4.1.1. Implicit Integration Methods

The most widely-used method for cloth simulation is currently the semi-implicit Backward Euler method, which was first used by Baraff et al in the context of cloth simulation. As any implicit method, it alleviates the need of high accuracy for the simulation of stiff differential equations, offering convergence for large timesteps rather

than numerical instability (a step of the semi-implicit Euler method with "infinite" timestep is actually equivalent to an iteration of the Newton resolution method).

The formulation of a generalized implicit Euler integration is the following:

$$Q_{(t+dt)} - Q_{(t)} = Q'_{(t+\alpha dt)} dt \quad (22)$$

The derivative value is not known at a moment after t , and is then extrapolated from the value at moment t using the Jacobian, leading to the semi-implicit expression which requires the resolution of a linear system:

$$Q_{(t+dt)} - Q_{(t)} = \left(I - \alpha \frac{\partial Q'}{\partial Q_{(t)}} dt \right)^{-1} Q'_{(t)} dt \quad (23)$$

We have introduced the coefficient α so as to modulate the "implicitness" of the formula. Hence, $\alpha = 1$ is the regular implicit Backward Euler step (stable), whereas $\alpha = 0$ is the explicit Forward Euler step (unstable), and $\alpha = 1/2$ is the 2nd-order implicit Midpoint step (most accurate, at the threshold of stability).

The α parameter is a good handle for adjusting the compromise between stability and accuracy. While maximum robustness is obviously observed for large values, reducing its value increases accuracy (reduces numerical damping) at the expense of stability, and speeds up the computation as well (better conditioning of the linear system to be resolved).

Better accuracy can also be obtained through the use of the 2nd-order Backward Differential Formula (BDF-2), as described by Hauth et al. This uses the previous state of the system for enhancing accuracy up to 2nd-order, with a minimal impact on the computation charge. Its generalized implicit expression is:

$$Q_{(t+dt)} - Q_{(t)} = \beta (Q_{(t)} - Q_{(t-dt)}) + Q'_{(t+\alpha dt)} \delta t$$

with
$$\beta = \frac{2\alpha - 1}{2\alpha + 1} \quad \text{and} \quad \delta t = \frac{2}{2\alpha + 1} dt \quad (24)$$

And its semi-implicit expression is:

$$Q_{(t+dt)} - Q_{(t)} = \left(I - \alpha \frac{\partial Q'}{\partial Q_{(t)}} \delta t \right)^{-1} (\beta (Q_{(t)} - Q_{(t-dt)}) + Q'_{(t)} \delta t) \quad (25)$$

While $\alpha = 1$ is the regular implicit BDF-2 step, $\alpha = 0$ is the explicit Leapfrog method, and $\alpha = 1/2$ is again the implicit Midpoint method. Best accuracy is offered for $\alpha = 1/\sqrt{3}$, where the method is 3rd-order (moderately stable).

Compared to Backward Euler, the main interest of the BDF-2 method is that it exhibits better accuracy for dynamic simulation over time (less numerical damping) for moderately stiff numerical contexts (at the expense of reduced robustness for nonlinear situations). For very stiff contexts however, this benefit disappears. While it is possible to implement higher-order BDF methods, their interest is reduced by their lack of stability, and high accuracy could be more efficiently reached using high-order explicit methods. Stability of implicit methods is also affected by the nonlinearities of the mechanical model.

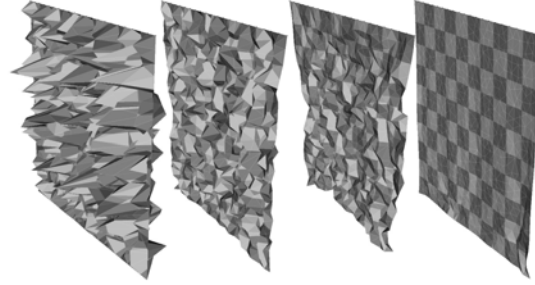


Figure 10: Stability test: A square of cloth is initially deformed with large random perturbations, and then simulated using various timesteps.

3.4.1.2. Explicit Integration Methods

Unlike implicit methods, explicit methods do not offer convergence to equilibrium if the timestep is too large compared to the numerical stiffness of the equations. On the other hand, they are very simple to implement, and much compute much faster than their implicit counterpart for reaching a given accuracy. This is particularly true for high-order methods, which offer very high accuracy if the timestep is small enough, but diverge abruptly if it exceeds a threshold (related to the stiffness of the equations). This is why an efficient timestep control scheme is essential for the implementation of these methods.

While the explicit 1st-order Euler and 2nd-order Midpoint methods should be restricted to simple applications (beside their simplicity, they have no benefits compared to their implicit counterparts), a popular choice is the 5th-order Runge-Kutta scheme with embedded error evaluation. It is a six-stage iteration process where the computed error magnitude can be used for controlling the adequate timestep requirements. Unlike implicit methods, this method yields a very good guaranteed accuracy (resulting from the high-order, but which may require very small timesteps), which is particularly important for problems where energy conservation is a key issue (for example, evaluating the effect of viscous parameters in the motion of fabrics). On the other hand, explicit methods are quite unsuited for the fast relaxation of the static cloth draping applications.

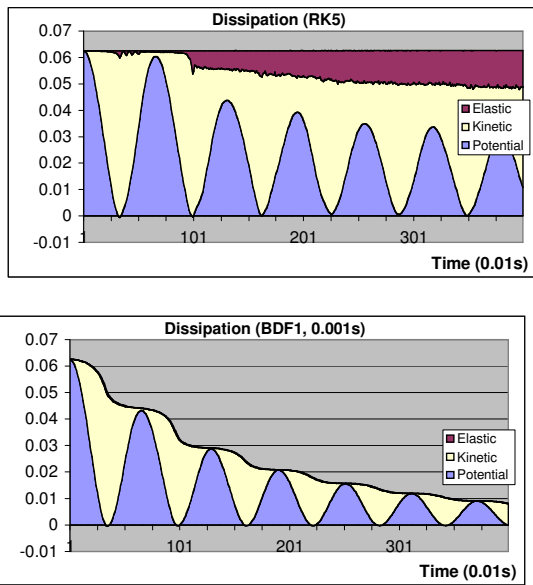


Figure 11: Evaluation of numerical damping of various integration methods using energy dissipation plots along time (50cm x 50cm square cloth, initially horizontal, attached along one edge, linear isotropic 100N/m, 100g/m², 2cm² elements, no dissipative parameters). 5th-order Runge-Kutta (up) accurately preserves the total energy along time, a good amount of it being transferred to elastic energy through small-scale mesh jittering (timesteps between 0.0001s and 0.00001s). Implicit methods such as Inverse Euler (down) damp small-scale motion.

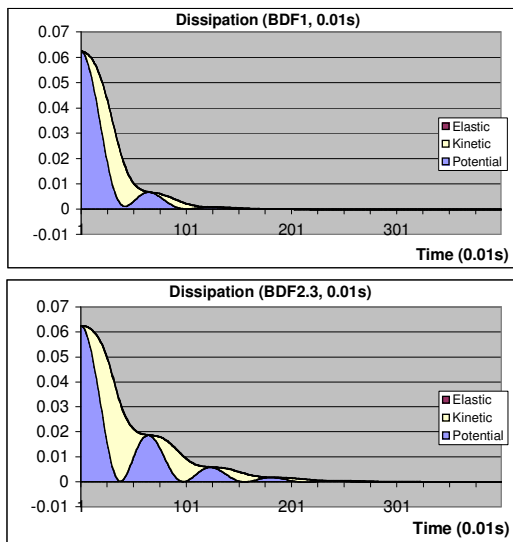


Figure 12: The 3rd-order BDF2 variation (down) preserves energy significantly better than Inverse Euler (up).

3.4.2 Implementation Issues

While there are no particular issues related to the implementation of explicit integration methods, semi-implicit methods require the resolution of large sparse linear equations systems, which are mainly constructed from the Jacobian of the mechanical law $\partial \mathbf{F} / \partial \mathbf{P}$ and $\partial \mathbf{F} / \partial \mathbf{P}'$ (their sparse structure relates the mechanical dependency between the particles). Among possible speed-up approximations, the Jacobian terms generated by the non-stiff forces can be neglected (they are then explicitly integrated).

A choice candidate for resolving this linear system is the Conjugate Gradient method, which is iterative and thus offers compromise between computation charge and symmetric accuracy, and which also allows efficient implementation for sparse systems.

Among possible optimizations are linearization schemes aimed at performing the computation using a constant approximation of the Jacobian, so as to implement pre-processing optimizations in the resolution. While giving reasonable benefits for draping applications, these approximations however generate large "numerical damping" that slow down convergence and alter highly the motion of the cloth along time.

The only solution for simulating the accurate motion of cloth was indeed to use real value of the Jacobian corresponding to the current state of the system. We have taken advantage of the Conjugate Gradient method which only needs the Jacobian matrix products with given vectors to compute these products "on the fly" directly from the system state, skipping the sparse explicit storage of the matrix for each frame. Our system actually allows performing partial linearization of the Jacobian, so as to use the linearization ratio offering the best trade-off between motion accuracy and stability, depending on the simulation context.

3.5 Collision Processing

Collision detection is indeed one of the most time-consuming tasks when it comes to simulate virtual characters wearing complete garments. In high-performance systems, this task is usually performed through an adapted bounding-volume hierarchy algorithm, which uses a constant Discrete-Orientation-Polytope hierarchy constructed on the mesh, and optimization for self-collision detection using curvature evaluation on the surface hierarchy. This algorithm is fast enough for allowing full collision and self-collision detection between all objects of the scene with acceptable impact on the processing time (rarely exceeds 20% of the total time). Thus, body and cloth meshes are handled totally symmetrically by the collision detection process, ensuring perfect versatility of the collision handling between the body and the several layers of garments.

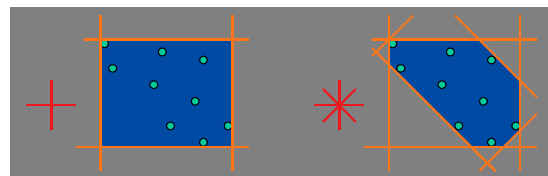


Figure 13: Discrete-Orientation-Polytopes are a generalisation of axis-aligned Bounding Boxes.

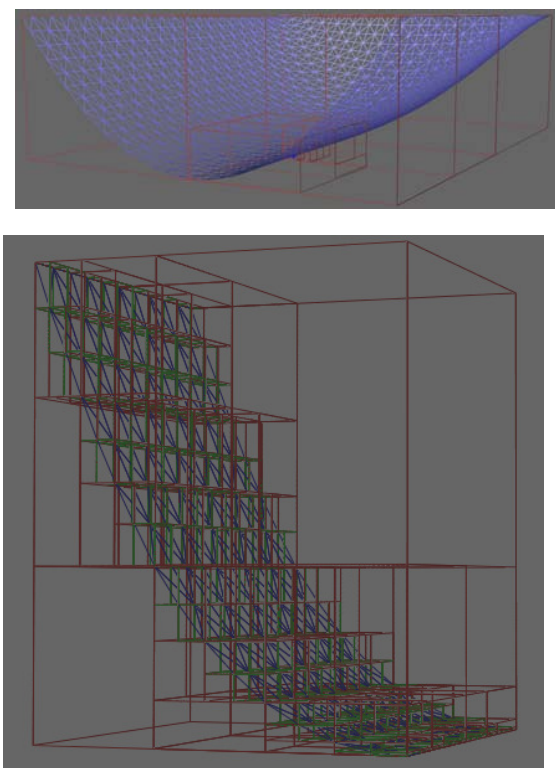


Figure 14: Bounding-volume hierarchy for collision detection on a cloth mesh.

Collision response is handled robustly using a geometrical scheme based on correction of mesh position, speed and acceleration. This scheme ensures good accuracy and stability without the need of large nonlinear forces that alter the numerical resolution of the mechanical model.

4. Tactile perception of Synthetic Surfaces

Tactile aspects of a virtual object can be represented as a spatial distribution of synthetic touch sensations over the fingertips. An array of contactors on the skin may be used to provide appropriate spatiotemporal patterns of mechanical excitation to the underlying mechanoreceptors. Tactile rendering software can generate drive signals for the array on the basis of the user's movements and a model of the finger/object interaction.

4.1 Overview

This section is concerned with the tactile aspects of a virtual object, represented as a spatial distribution of synthetic touch sensations over the fingertips. These sensations can provide information about the surface texture of the virtual object and about the contact between object and skin (contact area and position of edges/ corners).

To excite the skin mechanoreceptors, an array of contactors on the skin may be used to provide spatiotemporal patterns of mechanical input to the skin surface. In practice, such an array is one component of a haptic interface, integrated with a force-feedback component which represents the gross mechanical properties of the virtual object. Encounters with virtual objects, during active exploration of the workspace by the

user, produce appropriate patterns of tactile stimulation on the fingertips. This is illustrated schematically in Figure 15.

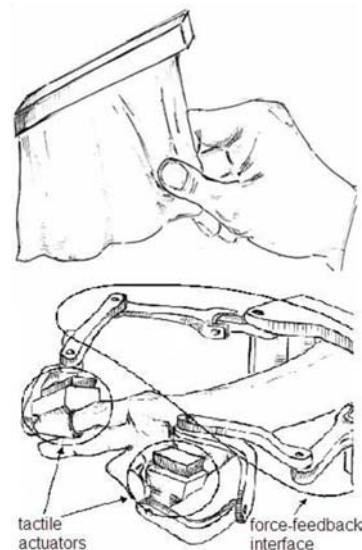


Figure 15: Schematic diagram of a compound haptic interface. The upper picture shows a virtual scenario (from the HAPTEX project) in which a fabric sample is evaluated in terms of overall mechanical properties and surface properties; the lower picture indicates how the overall mechanical properties are represented by force-feedback and the surface properties are represented by tactile stimulation from actuators on the fingertips.

When presenting the tactile aspects of a virtual object, the intention is not to reproduce the significant features of the small-scale surface topology of the object in terms of a virtual surface – that would require micron-scale resolution and is probably beyond the scope of current technology. Instead, the intention is to reproduce the perceptual consequences of small-scale features of the surface topology, i.e., appropriate excitation patterns over the various populations of touch receptors in the skin. (Shape displays have been developed to reproduce larger-scale features of an object's surface topology [WLH02, WLH04], at millimetre-scale resolution, but these are not the subject of the present discussion.)

4.2 Design of a stimulator array

As outlined above, an array of contactors on the skin may be used to provide spatiotemporal patterns of mechanical excitation to the underlying mechanoreceptors. The design requirements for such a stimulator array – contactor spacing, working bandwidth and output amplitude – are largely determined by the response of the mechanoreceptors.

The hairless skin which is found on the fingertips and the palms of the hands contains four populations of mechanoreceptors: pacinian receptors and three types of non-pacinian receptor [JYV00]. These populations differ in terms of their frequency response and their temporal response [GBH01]. Figure 16 indicates the distribution of pacinian receptors in the fingertip of a young adult.

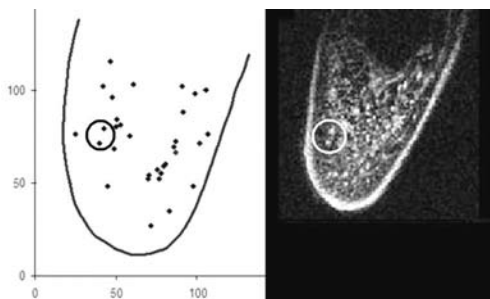


Figure 16: Pacinian receptors in the fingertip. The left panel shows a 2D projection of receptor locations (140 μm units), identified from a set of MRI slices, one of which is shown in the right panel. For further details, see [CSB*06].

The optimal spacing of contactors in a simulator array is determined by the spatial acuity of the sense of touch – around 1 mm on the fingertip [JYV00]. However, a contactor spacing of 1 mm equates to around 100 contactors over the fingertip, each of which requires independent control. This is realistic for a passive (non-moving) device [SC02, KBD*07] but is difficult to implement in an active device, for which a spacing of around 2 mm (i.e., around 25 contactors on the fingertip) may be a better choice. (There is some evidence [SCS*01] that perceptions available from an array with 2 mm spacing are not very different than those from an array with 1 mm spacing.)

In order to produce “realistic” touch sensations, a working bandwidth of around 10 to 500 Hz is required for the drive mechanism of each contactor, corresponding to the frequency range over which the various mechanoreceptors are sensitive [GBH01]. Pacinian receptors are expected to respond most strongly to frequencies in the upper part of this frequency range (100 to 500 Hz, say); stimulation at lower frequencies is expected to stimulate mainly non-pacinian receptors.

It is difficult to closely specify the amplitudes of contactor movement which are required to produce particular levels of touch sensation, because sensation level varies with the extent of the area stimulated, particularly when pacinian receptors are involved [Ver63]. However, it is possible to give approximate figures: “comfortable” sensation levels are produced by amplitudes of a few microns at frequencies around 300 Hz and a few tens of microns at frequencies around 50 Hz. (In one of very few studies on this topic, Verrillo *et al.* [VFS69] determined equal-sensation contours for a fixed area of stimulation on the palm of the hand. It is not clear how well these data may be applied to the case of a variable area of stimulation on the fingertip.)

A further consideration is the direction of movement of the contactors in a stimulator array. In “real” touch perception, the interaction between skin and object produces normal and tangential forces on the skin surface (with two orthogonal components for the latter). However, stimulator arrays are constrained by present technologies to produce input forces in only a single direction (i.e., normal to the skin surface, or tangential in only one of the two available directions). It is difficult to assess the significance of this constraint. Since the aim is to produce appropriate excitation patterns over the various populations of touch receptors, the question is whether receptors that typically respond to forces in a particular direction in the “real” situation (e.g., receptors which respond to skin stretch caused by tangential forces) can be excited by forces in a

different direction in the virtual situation. It is difficult to answer this question with certainty because the existing literature on mechanoreceptor transduction is very limited. In practice, it seems that the direction of contactor movement does not make a great difference to the nature of the available tactile sensation, and so an acceptable stimulator can be designed on the basis of either tangential or normal movement.

Design requirements for contactor spacing, working bandwidth and output amplitude may be satisfied by a variety of electromechanical drive mechanisms. Hafez and colleagues [BHA*03, HB04] have developed arrays of drivers, based on shape-memory alloy or moving-coil technology, which apply normal forces to the skin. Hayward and colleagues [PH03, LPH07] have used piezoelectric-bimorph actuators to apply tangential forces. Summers *et al.* [SBS*05] have used similar actuators to apply normal forces, as have Kyung *et al.* [KAK*06].

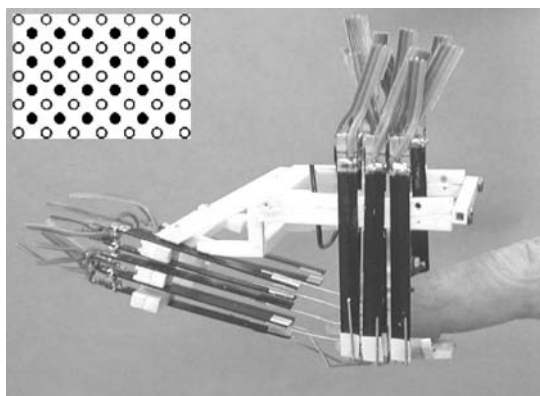


Figure 17: Stimulator array developed for the HAPTEX project. The contactor surface lies under the finger – contactors are driven by piezoelectric bimorphs (appearing as black rectangles). The inset shows the arrangement of 24 moving contactors, interspersed between fixed contactors.

Looking at one design of drive mechanism in more detail: the stimulator array developed for the HAPTEX project is shown in Figure 17. Piezoelectric bimorphs are used to drive 24 contactors in a 6×4 array on the fingertip, with a spacing of 2 mm between contactor centres. It can be seen that the drive mechanism is placed to the side of the finger and ahead of the finger, rather than below the contactor surface (which, at first sight, appears to be the most convenient location). With one such array on the index finger and one on the thumb, this positioning of the drive mechanism allows the finger to move close to the thumb so that a virtual textile can be manipulated between the tips of finger and thumb.

The contactor surface delivers to the fingertip the small forces associated with touch stimuli, but it must also deliver the larger forces associated with the overall mechanical properties of the virtual object, represented by the output of the force-feedback system (see Figure 15). However, the moving contactors which provide touch stimuli are driven by delicate piezoelectric mechanisms and so they are unsuitable for delivering the force-feedback output, which may involve forces of considerable magnitude. Consequently, the contactor surface includes an additional set of contactors (“fixed” contactors – see inset to Figure 17) which deliver the force-feedback output, in parallel with the tactile stimulation from the moving contactors. This parallel delivery is shown schematically in Figure 18.

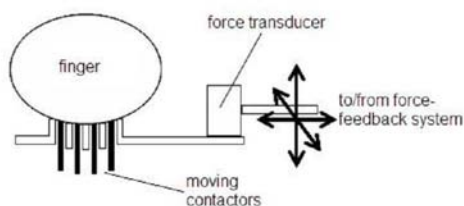


Figure 18: Schematic diagram of the integration of the stimulator array and the force-feedback system. The finger rests on the contactor surface, composed of moving contactors (shown in black) which deliver tactile stimuli and “fixed” contactors (shown in white) which connect to the force-feedback system via a 3D force transducer.

4.3 Drive signals for a stimulator array

During active exploration of a virtual tactile environment it is necessary to generate in real time a drive waveform for each contactor of the stimulator array(s) which are in contact with the user’s fingertip(s). The amount of data which must be generated “on the fly” is thus considerable. For example, the HAPTEX system has 24-contactor arrays on finger and thumb, requiring 48 analogue drive signals, in principle each with a bandwidth of around 500 Hz. However, because of the limited temporal resolution, frequency resolution and phase sensitivity of human touch perception [She85, RHD*87, VGV90, FMF*92, SWM*05], there are possibilities for a significant reduction in the data flow. For example, each drive signal may be reduced to the sum of a limited number of sinusoidal components, distributed across the working bandwidth (10 to 500 Hz – see above). The drive signal may then be simply specified in terms of the amplitudes of these components, which require an update every 20 ms or so.

In the HAPTEX project, a cut-down version of this scheme has been used, in which the drive signal to each contactor is the sum of components at only two frequencies: 40 Hz and 320 Hz. Following the suggestion of Bernstein [BED89], the higher frequency was selected (at 320 Hz) to target pacinian receptors and the lower frequency was selected (at 40 Hz) to target non-pacinian receptors. Each drive signal is specified by the amplitudes A_{40} and A_{320} of the two signal components. These are updated every 25 ms. A virtual tactile surface is specified in terms of an amplitude map for each of the two frequency components that make up the stimulus.

The spatial resolution available from pacinian receptors is expected to be worse than that available from non-pacinian receptors, i.e., spatial discrimination for 320 Hz stimuli is expected to be worse than spatial discrimination for 40 Hz stimuli. In fact, results from psychophysics experiments on this type of array [SCS*01, SC02] suggest the converse: spatial discrimination is better for perception of stimuli at 320 Hz than at 40 Hz. Thus, although it seems likely that different receptor mechanisms are targeted by the different stimulation frequencies, there is some doubt about which receptor populations are involved.

4.4 Tactile rendering

As outlined above, during exploration of a virtual tactile environment a drive waveform is specified for each contactor of the stimulator array(s). The intention is to produce time-varying excitation patterns in the various populations of mechanoreceptors in the skin, so as to

reproduce the touch sensations which are experienced during “real” tactile exploration.

A significant problem is the current lack of knowledge on the origin and nature of excitation patterns in real situations of tactile exploration of an object. The mechanical stimulation of a given receptor has a complicated relation to the mechanical properties and topology of the object’s surface, to the mechanical properties of the skin and its local topology (especially skin ridges, i.e., fingerprints), and to the precise nature of the exploratory movement (speed, contact pressure and direction). Although it may be possible to produce an accurate software model of an object’s surface, it is not at present possible to augment this with an accurate model of the skin/surface interaction. This situation may change in the near future: research is currently underway to develop an “artificial finger” with embedded transducers to mimic mechanoreceptors; improved finite-element models may also provide useful data.

For the particular case of the manipulation of textiles, the situation is more promising: Information on the nature of the mechanical input to the skin’s mechanoreceptors is available from the Kawabata system for evaluation of textiles [Kaw80]. This provides a range of data on the textile sample under test, including surface roughness and surface friction profiles which are direct measures of the mechanical excitations produced when a probe is moved over the textile surface. The probe and associated instrumentation are designed so that the measured quantities correlate well with subjective assessment of the textile surface. Hence the Kawabata surface measurements provide an approximation to the “perceived surface”, i.e., the surface after it has been “filtered” through the surface/skin interface. They thus provide a good basis for specifying drive signals for a stimulator array, in order to provide the tactile component for a virtual textile. Kawabata measurements have been used in this way by Govindaraj *et al.* [GGR*03]; they have also been used to provide source data for the tactile rendering developed within the HAPTEX project.

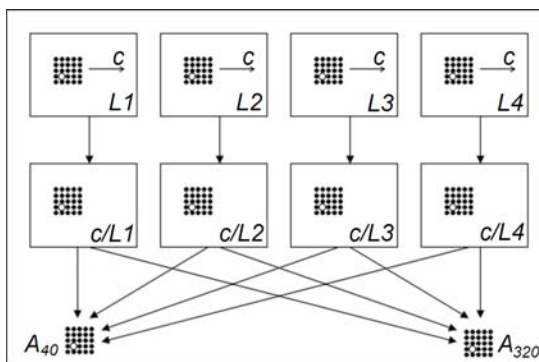


Figure 19: A simple scheme for tactile rendering.

Figure 19 outlines a simple scheme for tactile rendering, by which the drive signal to each point in the stimulator array is specified in terms of amplitude A_{40} at 40 Hz and an amplitude A_{320} at 320 Hz, and these amplitudes are in turn specified by the interaction between the virtual object and the exploratory movements of the user. The rendering is based on a pseudo-topology (i.e., an estimate of the surface after it has been filtered through the surface/skin interface), described in terms of amplitude distributions at length

scales $L1$, $L2$, etc.. These may be considered as amplitude distributions in frequency ranges $c/L1$, $c/L2$, etc., where c is the speed of exploration. [For a stimulator array with 2 mm spacing, the amplitude distributions might be specified at a resolution (“pixel size”) of 1 mm, with effective feature widths of ≥ 2 mm.] For each point in the array, e.g., the point indicated by a white dot in Figure 19, the amplitudes in the different frequency ranges are combined by appropriate filter functions to produce the drive amplitudes A_{40} and A_{320} .

A similar rendering scheme developed within the HAPTEX project is summarised in Figure 20. For each digit, the tactile renderer generates 24 drive signals for the 24 contactors of the stimulator array. Input and output data specified in 25 ms timesteps. The input data are:

- a small-scale description of the object surface, represented as 2D k -space, derived from a pseudo-topology at 0.1 mm resolution over an area of a few mm^2 ;
- a large-scale description of the object surface: a representation of the non-uniformity of the surface, specified as pseudo-amplitudes at 1 mm resolution over an area of several tens of cm^2 ;
- position and orientation of the finger pad on the virtual surface;
- speed and direction of the movement of the finger pad over the virtual surface.

The operation of the renderer is as follows:

Taking account of the direction of movement, a spatial-frequency spectrum is calculated from the 2D k -space of the small-scale description of the virtual surface. Information about the speed of movement of the finger pad is used to convert spatial-frequency components into temporal-frequency components. The resulting temporal-frequency spectrum is reduced to only two amplitudes, A_{40} and A_{320} , by application of appropriate bandpass filter functions, corresponding to the 40-Hz and 320-Hz channels. (It should be noted that the signal-processing operations to this point may be performed only once per 25-ms timestep, i.e., they may be common to all 24 output channels.) Amplitudes for the 40-Hz component in the drive signals for each of the 24 channels are obtained from A_{40} by weighting according to data from the large-scale description of the virtual surface, for the 24 locations on the finger at which the contactors of the tactile stimulator are positioned. Similarly, amplitudes for the 320-Hz component in the drive signals for each of the 24 channels are obtained from A_{320} by weighting according to data from the large-scale description of the virtual surface. (Note that, in principle, different large-scale descriptions of the virtual surface may be used in the 40-Hz and 320-Hz channels, to allow for the observed difference in spatial resolution on the fingertip at the two frequencies.)

4.5 Discussion

When using stimulator arrays and rendering schemes as described above, the intention is to present time-varying spatial patterns of tactile stimuli which have two perceptual dimensions: one relating to intensity and one relating to spectral distribution. In order to establish the potential for such a system, it is necessary to determine whether a two-dimensional perceptual space can indeed be created in this way – it is very likely that the intensity dimension is available to the user, but less obvious that the spectral dimension is available. (A spectral dimension is available in

the case of passive perception via a single contactor [She85, BED89], but this observation is not necessarily transferable to the case of active perception via an array of contactors.) However, recent results from Kyung *et al.* [KK07] demonstrate that test subjects can detect changes of frequency when stimuli are presented via a stimulator array in an active task, so it seems that spectral information is indeed available in such a scenario.

Initial evaluations of the HAPTEX system (Figure 20) also suggest that a 2D perceptual space can be achieved. For uniform stimuli (i.e., stimuli with no spatial variation over the skin), the spectral dimension appears relatively weak – changes in spectral balance at constant subjective intensity tend to be less noticeable than changes in subjective intensity at constant spectral balance. (There are perhaps 4 to 5 discriminable steps of spectral balance along an equal-intensity contour.)

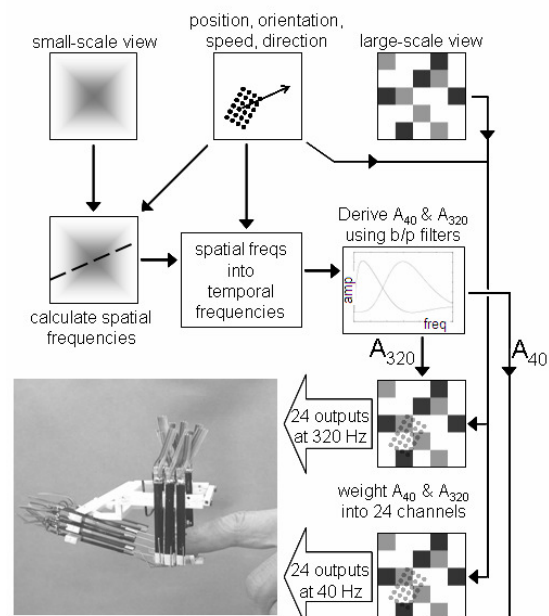


Figure 20: The tactile renderer. Input and output data are specified in 25 ms timesteps.

Perhaps the most interesting observation when using the HAPTEX system is a strong interaction between the perceived spatial aspects of the texture and the stimulation frequency. If the stimulation frequency is changed from 40 Hz to 320 Hz, the perceived sensation during active exploration changes much more if the texture is spatially non-uniform than if it is spatially uniform. It is clear that the spectral dimension provides a significant enhancement to the available range of tactile sensations.

Using data from the Kawabata system for a selection of real fabrics, the HAPTEX system has been used to simulate the tactile aspects of those fabrics. Given the apparent mismatch between the real situation (fingertip touching a textile) and the virtual situation (fingertip touching the metallic contactors of a stimulator array), results are surprisingly good – in some cases test subjects are able to match real and virtual textiles in terms of their tactile qualities. Prospects appear encouraging for the development of virtual textiles with acceptable tactile properties and it

seems likely that similar techniques can be successfully applied to other types of virtual object.

5. Force Feedback Technologies for the Rendering of the Direct Interaction with Deformable Objects

When addressing the problem of developing suitable force feedback technologies for the realistic haptic rendering of the physical interaction with deformable objects, a major distinction has to be recognized among the cases of "indirect interactions", i.e. interactions mediated by tools held by the user (for example in the case of the simulation of surgical operations) and the cases of "direct interactions", in which the limbs and the skin of the user come directly in contact with the surface of the deformable object, for example in the case of rubbing a textile. The latter case poses technical challenges that are dramatically more demanding than the former from both the qualitative and quantitative points of view. In this section we illustrate three examples of research works that aim at the improvement of the quality of direct contact simulation. These themes deal with the three important aspects of design of novel force feedback systems, innovative concepts for curvature simulation and control algorithms for accuracy improvement.

5.1 Introduction

While the simulation of the indirect interaction can satisfactorily be addressed using an accurately force controlled robotic manipulator, having an end-effector shaped like the tool used in the simulation (for example a cutter, see Figure 21), the realistic simulation of the direct interaction implies the development of suitable technologies, able to comply with the extremely sensitive and sophisticated human haptic sensorial system that can perceive a large number of features characterizing the local contact with the object, like for example the global location/orientation with respect to the skin of the contact area(s) and its (their) extension, the local mechanical impedance(s) of the object (i.e. the relationship between the local displacement, speed, acceleration and the corresponding reaction force), the local large scale (e.g. the curvature), medium scale (e.g. bumps and edge) and small scale (e.g. the roughness) geometry, the local temperature, etc. (see Figure 22).



Figure 21: Picture of a force feedback device for the simulation of indirect interactions.

The mechanisms underlying the perception of these features are currently not well understood and large research efforts are still required for identifying the stimuli

that have to be generated in order to elicit the perception of a specific contact feature. The exploitation of haptic illusions seems a promising approach for the simplification of the stimuli to be generated, but also in this case in depth researches are needed for clarifying the quantitative dependencies of the equivalent stimuli with the intended perception to be elicited.

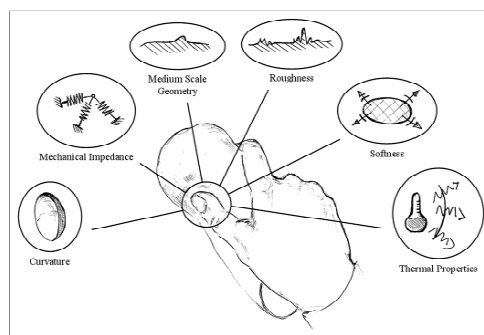


Figure 22: Elements characterizing the local direct contact with a deformable object.

As a broad result of the psychophysics researches carried on up today on the human haptic sensorial system, it can be said that there is a clear evidence of the major role of the mechanoreceptors located in the skin in the perception of the contact features, even if the proprioceptive sensors located in the physiological joints and in the tendons are used by the brain for the reconstruction of the global geometry and characteristics of the object, by space integration of the local perceptions.

From the technical point of view, the most accepted conceptual scheme for a haptic interface, able to render the direct interaction with virtual deformable objects, envisages the integration of a number of tactile arrays, i.e. a relatively high spatial resolution array of pins movable normally to a specified fixed surface in a relatively small range of motion, mounted on the end effectors of a force feedback device, i.e. an actuated and sensed mechanism able to move its end-effector(s) in a relatively large workspace and to generate controlled forces on it (them).

The first step in the definition of the requirements of the two different devices is the attribution of their roles in the generation of the required stimuli.

The most straightforward approach is to ascribe to the force feedback device the role of providing the means for the perception of the global characteristics of the local contacts, such as the locations/orientations of the contact areas with respect to the skin and their mechanical impedances, while the tactile actuator should be in charge of generating the stimuli that can change inside the contact area, like for example the small and medium scale geometry and the temperature. This approach leaves undefined the attribution of the generation of some border features like for example the global curvature and the extension of the contact area (that in fact in some way are each other dependent). In principle the global curvature of the local contact could be reproduced by the tactile array, but due to possible limitations to the maximum available displacement of the pins the alternative solution envisaging a global deformation of the surface where the pins are located should be also considered (see Figure 23).

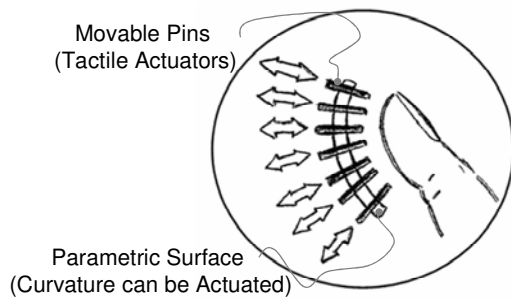


Figure 23: Scheme of a the tactile array with deformable backing surface for the simulation of different local curvature

Focusing on the basic functional requirements of the force feedback device, on the basis of what discussed above they can be defined as the followings:

1. allow the modification of both the global location and the orientation of the contact areas (and therefore of the tactile arrays) with respect to the user's skin;
2. allow the modification of the global mechanical impedances of the contact areas. This implies the generation of global reaction forces in function of the global displacement of the contact area

It is worth to add some additional considerations and requirements in order to better understand the terms of the challenge and identify the major technological components that have to be developed.

The first point is to define how many independent contact areas should be managed at the same time by the device. In the general case, during an interaction with a deformable object a virtually infinite number of independent contact areas can be simultaneously generated (let's think for example to the case of the manipulation of a textile with the whole hand). This is clearly very far from being concretely achievable with the present technology and a reasonable simplification is required. Considering that the hand is the most important organ of human haptic exploration of the external world and that the fingertips are the most sensitive portions of the hand's skin, a reasonable simplification is to require that the device could manage simultaneously at least five independent contact areas, one for each fingertip. Furthermore, in order to exploit at best the sensitivity of the mechanoreceptors, the device should be able of reproducing the transition from the non-contact to the contact phases, i.e. the stimuli have to be generated only when a contact with the virtual object is detected.

The second point is to define the level of accuracy required for the generation of the controlled forces. Considering the high sensitivity and resolution capability of the mechanoreceptors in the skin and that many of the potential applications of the envisaged visual-haptic VR system, like for example the on line marketing of newly produced textiles/garments, would require the possibility of discerning very fine differences of the rendered features, the device should be able to accurately control global interaction forces that can be of the order of few centi-newtons (1 gf).

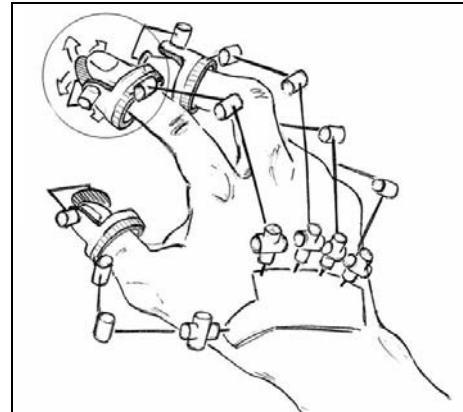


Figure 24: Conceptual scheme of a force feedback device intended for the simulation of the direct interaction with the 5 fingertips of the hand.

Finally, the device should allow as much as possible a natural interaction with the virtual object, posing minor limitations and constraints to the dexterity of the human hand and arm movements. This strongly suggests the adoption of an anthropomorphic configuration for the device structure that could be worn on the user's hand. All these requirements could be attained by the device whose conceptual scheme is sketched in Figure 24.

The device would be composed by 5 independent exoskeletons, each having 5 Degree of Freedoms (DoFs), 3 DoFs for positioning and 2 DoFs for orientating the end plate supporting the tactile actuator. Furthermore each exoskeleton would be equipped with a contactless position sensor for the tracking of the fingertip during the non contact phase and by an accurate explicit force control that makes use of a 3 component force sensor providing the exact measure of the global resultant force delivered to the fingertips through the contact area.

In short the envisaged device would have a total of 25 actuated a position sensorised degrees of freedom, 5 force sensors each having at least 3 component measure capability and 5 contactless position sensors with at least 3 component measure capability. This huge number of components should be integrated in a compact device that can be easily worn by the user, without limiting the natural motion capability of the hand. This sets clearly the terms of the technological challenge.

With respect to the state of art, in order to attain the envisaged device three main technological areas have to be investigated:

1. The development of a compact anthropomorphic hand exoskeleton able to track the fingertip movements;
2. The development of an encountered haptic interface for the fingertip able of changing the position and orientation of a plate end effector with respect to the fingertip skin according to the local geometry of the virtual object and able to render the transition from the non-contact to contact phases;

3. The development of an accurate explicit force control and related force sensor.

All these major technological areas are under investigation at PERCRO. The attained results are described in the following paragraphs.

5.2 Development of the hand exoskeleton

A Hand Exoskeleton (HE) is a device able to exert forces on the phalanges of the fingers. Several works (for example [VT99] and [BPBB02]) have addressed the development of different kinds of HEs.

Basically two types of HE can be identified in literature:

- *Multi-phalanx*: are multipoint devices where the force are exerted on each phalanx of the finger; each exerted force is directed along a fixed direction (normal to the phalanx)
- *Single Phalanx*: the device exerts forces only on the distal phalanx (the fingertip) of the finger; in this case the force has 3DoF and can be exerted in any wanted direction.

In our application, since the attention is focused on the fingertip area, a Single Phalanx 3DoF device has been realized.

The device uses quasi-anthropomorphic kinematics. This solution allows exploiting the benefits of anthropomorphic kinematics like maximum ratio of the available over the needed workspace and minimum encumbrance of the linkages. At the same time it is not perfectly anthropomorphic in order to avoid the singularity that would occur when the finger is all extended. In Figure 25 a CAD model of the HE is shown. It can be noticed that the encumbrance of the device has been located in the dorsal side of the hand (except for the fingertip indeed) with the aim of allowing the complete closing of the hand. This has been achieved through the use of Remote Centre of Rotation mechanisms. These mechanisms are able to implement a rotational joint having an axis located outside the linkages.

The whole mechanism has 4DoF but it is actuated with only three motors thanks to the coupling of the last DoF (end-effector Joint) with the previous one. The coupling is acceptable because also in the human hand the last phalanx can be rarely moved independently from the middle phalanx during natural movements.

The HE is equipped with three electrical motors with low speed reduction ratios (1:14). The actuators are placed at the base of the mechanism in proximity of the dorsal side of the palm. The joints are actuated through in tension steel cable transmissions.

The position sensing is achieved with common encoders located on the axis of the motors and the force sensing is achieved with a purposely developed 3DoF force sensor placed at the end effector.

A purposely developed electronics for the sensor acquisition and the driving of the motors has been located inside the motor box at the base of the mechanism.

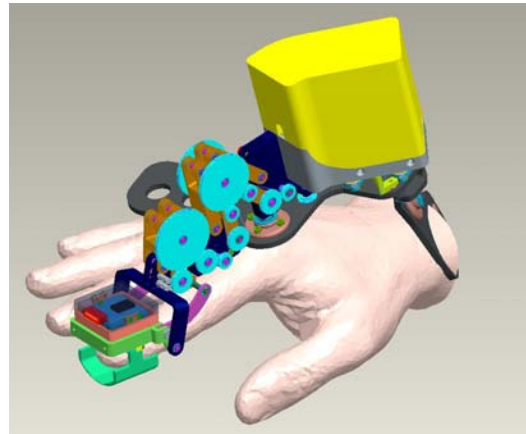


Figure 25: CAD model of the Hand Exoskeleton

This activity has been performed in the framework of the EU funded RTD Project HAPTEX (HAPtic rendering of virtual TEXTiles)

5.3 Development of an encountered haptic interface

Haptic interaction allows to naturally perform both haptic exploration of shapes/surfaces and active manipulation of objects. Nevertheless shape recognition isn't efficient if only kinaesthetic cues are provided. The restriction imposed on the fingertip contact region can blunt the haptic perception of shape and so local haptic cues play an important role in haptic perception of shape [JM06].

In [DH05] it is demonstrated how curvature discrimination can be carried out through a device providing only directional cues at the level of the fingertip, without any kinaesthetic information and moreover with a planar motion of the finger. This concept is also exploited to build robotic systems that can orient mobile surfaces on the tangent planes to the virtual object that is simulated, only at the contact points with the finger [DH05].

The system developed at PERCRO is composed of an encountered haptic interface. The basic concept for this type of devices was introduced in [SYYM04]. Our device is mounted on a kinaesthetic haptic interface, according to the overall configuration shown in Figure 26.

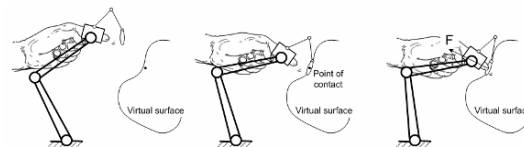


Figure 26: Conceptual scheme of the developed encountered haptic interface

In order to describe the working principle of the device, let's consider the case in which the user is interacting with the virtual object: when the finger is far from the surface of the object, the plate of the fingertip haptic interface is kept far apart from the fingertip. When the finger touches the virtual surface, the plate comes into contact with the fingertip with an orientation determined by the geometric

normal of the explored surface. Simultaneously a reaction force, proportional to the penetration, is exerted by the supporting kinaesthetic haptic interface.

The supporting haptic interface is a pure translational parallel manipulator with three degrees of freedom (DoF).

The end-effector is connected to the fixed base via three serial kinematics chains (legs), consisting of two links connected by an actuated revolute joint and two universal joints at the leg end. The design of the device was optimized in order to minimize the friction forces and the inertia of the moving parts, obtaining the required transparency of the mechanism during the haptic exploration.

The actuation is realized by three brushed DC motor through a in tension steel cable transmission, characterized by low friction and zero backlash.

The encountered haptic interface was devised to bring the final plate into contact with the fingertip with different orientations and positioning with respect to the fingertip skin, according to the local geometry of the surface of the virtual object in correspondence of the contact point. These requirements have been met using a 5 DoFs kinematics, three of them (translational) devoted to the positioning in space of the end plate and the two others (rotational) for its orientation. A hybrid kinematics, consisting of a first parallel translational stage and a second parallel rotational stage, resulted the most suitable solution (Figure 27).

The translational stage has the same kinematics of the supporting haptic interface, with 3-UPU legs. In each leg the cable connected to the motor and a compression spring are mounted aligned to the centres of the universal joints. The spring works in opposition with the motor, in order to generate the required actuation force and to guarantee a pre-load on the cable.

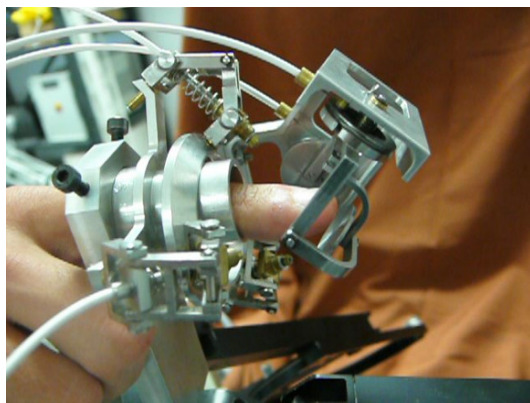


Figure 27: Detail of the fingertip haptic interface

The kinematics of the rotational stage allows the platform to rotate about two axis fixed to the translational stage.

Preliminary psychophysical tests have been carried on using the described system.

The method of constant stimuli for absolute threshold was employed for the discrimination of curvature by means of haptic cues. Four participants were recruited for this experiment. Each participant was informed about the procedure and did not present any dysfunction of the finger.

They were experts on haptic interfaces but novices on this device. The test consisted in exploring a virtual surface with curvature varying from 0 m⁻¹ (plane surface) to 6.67 m⁻¹ (a sphere of radius equal to 150 mm). The observers were asked to answer to the question “is it a curved or a plane surface?”. The curvature values were 0, 1.82, 2.22, 2.86, 3.33, 4, 5, 6.67 m⁻¹, and 100 stimuli were presented to each participant. The stimulus with a percentage of 50% of affirmative responses was considered as the threshold for detection of curvature. In Figure 28 is shown the device and a planar scheme of the haptic cues displayed to the users.

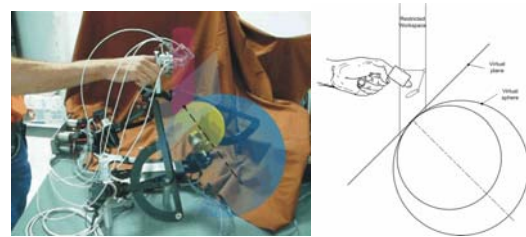


Figure 28: Experimental set-up

The test was carried out in two different modalities, A and B. In condition A both the kinaesthetic and the local geometry haptic cues were provided to the observers: the mobile platform of the fingertip device was kept in contact with the fingertip when the user was in contact with the surface, with an orientation tangent to the displayed virtual surface. In condition B only the kinaesthetic feedback was provided by the supporting haptic interface. In both modalities the haptic exploration was carried out in a restricted workspace, limited by a vertical cylinder with diameter of 25 mm.

The sigmoid curves for the two modalities are represented in Figure 29 for all the subjects, where the thresholds are pointed out, resulted using the fingertip haptic device (red line) and with the only kinaesthetic cues (blue line).

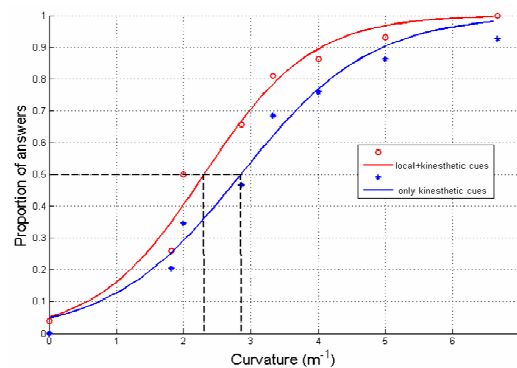


Figure 29: Sigmoid curves for curvature threshold

The average values of curvature threshold for the two conditions resulted respectively in 2.35±0.35 and 3.02±0.58 m⁻¹, with an improvement of 22.2% in the performance for modality A.

Although the ANOVA between the two conditions do not reach a significant level (p=0.069), we can however presume that further investigations employing more accurate schemes and conditions, e.g. method of limits and

TSD, and carried out on a larger sample of subjects/points, may statistically point out this difference. The data are consistent with what has been already found in literature for discrimination of curvature, either using haptic interfaces or real objects.

This activity has been performed in the framework of the EU funded RTD Network of Excellence ENACTIVE.

5.4 Development of accurate explicit force control

The simulation of fine interaction, in which the exchanged forces are of the order of few grams, requires the implementation of highly accurate force feedback devices.

This could be achieved through a careful design of the mechanical component of the device. However, when the required workspace is large and the performances are demanding, it is necessary to introduce advanced control technique in order to compensate the unwanted resistant forces.

In this work we implemented a motion-based explicit force control algorithm using a force sensor located at the end-effector of a haptic device. The general architecture of this algorithm was introduced in [VK93]. The control is implemented on a device that was originally designed for open loop control, i.e. without force sensing.

The haptic device taken as reference for this work is the GRAB system (described in [AMA*03]) developed by PERCRO Laboratory in the framework of the homonymous European RTD Project. GRAB is a 6 DoFs desktop force-feedback device with a serial kinematics. The first 3 DoFs (RRP) are required to track the position of the fingertip in 3D space and the remaining 3 (RRR) are required to track its orientation. The first 3 DoFs are actuated in order to exert forces of arbitrary orientation at the fingertip, whereas the last 3 DoFs are passive and are realized by a gimbal. On the last DoF of the gimbal a thimble is mounted for the connection to the user's fingertip. The 3 motors are remotely located: the first 2 actuators are mounted on the fixed link (base), while the third actuator is mounted on the second moving link. Forces are transmitted from the actuators to the joints by in tension metallic tendons routed on idle pulleys. There are no geared reducers and the motors are sensorised by optical encoders. For implementing explicit force-feedback control, a commercial 6-components force sensor from ATI (NANO17) has been placed at the base of the gimbal.

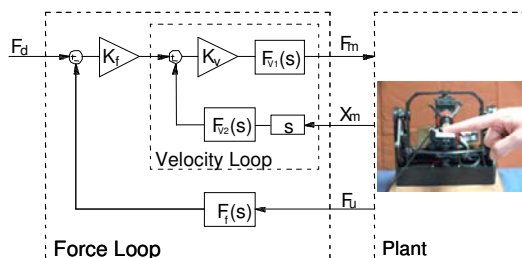


Figure 30: Scheme of the force control

The 1 DoF scheme of the controller is represented in Figure 30: it consists of an inner velocity loop-outer force loop scheme. This one-dimensional scheme can be simply generalized for implementing a three dimensional force control: it's sufficient to transform the velocity reference,

obtained as output of gain K_v , in a joint speed reference by simply multiplying it for the inverse of Jacobian matrix, which describe the inverse differential kinematics of device. This solution allows general control architecture to be safe and simplify the interface procedure with user.

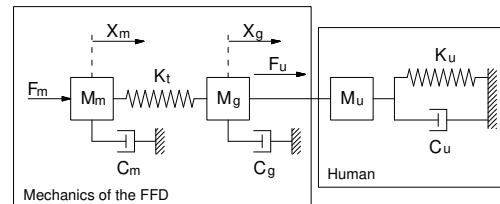


Figure 31: Mechanical model

The proper dimensioning of such controller required the following step:

- Building of a dynamic model of the mechanical system: the 1 DoF model (Figure 31) has been taken as reference; this model was experimentally validated.

- Estimation of the user impedance. The impedance of the finger in a thimble has been experimentally measured since it has a strong influence on the stability of the loop.

The design of the controller was carried on focusing on the inner velocity loop first. Two filters were introduced for the velocity loop:

- 1) a third order Chebyshev Type I for guaranteeing the stability of the velocity loop
- 2) a filter for velocity signal (velocity signal generated by digital encoders)

The outer force loop was then designed.

A Butterworth filter was introduced to guarantee the stability.

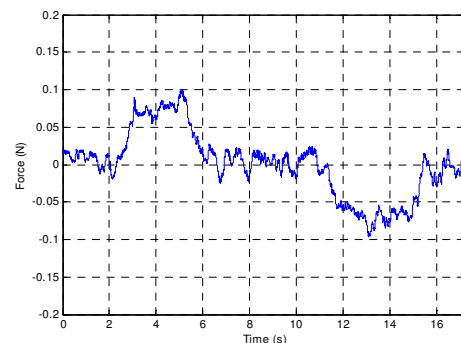


Figure 32: Plot of the resistant force versus time during finger tracking at constant velocity

A field test investigating the system capability to track the finger motion has been carried on. The subjects were asked to move their fingers at constant velocity and the system was set to display no forces. The maximum module of the acquired resistant force was about one tenth of Newton (10 grams force), as it can be observed in Figure 32.

This activity has been carried on in the framework of the EU funded RTD Project HAPTEX (HAPtic rendering of virtual TEXTiles)

5.5 Conclusion and future works

The development of force feedback technologies suitable for the rendering of the direct interaction with deformable objects is a tremendously challenging activity due to the very sophisticated capability of the human haptic sensorial system.

A reference configuration for the device addressing the case of direct interaction with the five fingertips of the hand has been identified. The three main technological areas required for attaining the envisaged device, namely the development of a compact anthropomorphic hand exoskeleton, the encountered haptic interface techniques, the accurate explicit force control and the compact 3 component force sensor, are now under investigation at PERCRO and the preliminary results are encouraging. The next research goal will be the integration of all the developed technologies in a single compact device.

6. Acknowledgments

The project "HAPTEX - HAPtic sensing of virtual TEXTiles" (Contract Nr.: IST-6549) is funded by the Sixth Framework Programme (FP6) of the European Union.

References

- [AMA*03] AVIZZANO C. A., MARCHESCHI S., ANGERILLI M., FONTANA M., BERGAMASCO M.: A Multi-Finger Haptic Interface for Visually Impaired People. In *Proceedings of the 2003 IEEE International Workshop on Robot and Human Interactive Communication*, Vol. 31, (2003), 165- 170.
- [AH98] ASTLEY, O. R., HAYWARD, V., "Multirate haptic simulation achieved by coupling finite element meshes through Norton equivalents", *Robotics and Automation*, 1998. *Proceedings*. IEEE, vol. 2, 16-20 May 1998 989 – 994, 1998.
- [BED89] BERNSTEIN L. E., EBERHARDT S. P., DEMOREST M. E.: Single-channel vibrotactile supplements to visual perception of intonation and stress. *J. Acoust. Soc. Am.* 85, (1989), 397–405.
- [BHA*03] BENALI-KHOUDJA M., HAFEZ M., ALEXANDRE J. M., KHEDDAR A.: Electromagnetically Driven High-Density Tactile Interface Based on a Multi-Layer Approach. *International Symposium on Micro-mechatronics and Human Science (MHS 2003)*, 147–152, 2003-
- [BHW94] BREEN D.E., HOUSE D.H., WOZNY M.J.: Predicting the Drape of Woven Cloth Using Interacting Particles, *Computer Graphics (ACM SIGGRAPH 94 proc.)*, Addison-Wesley, pp 365-372, 1994.
- [BMF03] BRIDSON R., MARINO S., FEDKIW R., 2003, Simulation of Clothing with Folds and Wrinkles, Eurographics-SIGGRAPH Symposium on Computer Animation, pp 28-36.
- [BPBB02] BOUZIT M., POPESCYU G., BURDEA G., BIOAN R.: The Rutgers Master II-nd Force Feedback Glove, In *Proc. IEEE Vr. Haptic Symposium*, Vol.7 (March 2002) 256-263.
- [BW98] BARAFF D., WITKIN A.: Large Steps in Cloth Simulation, *Computer Graphics (SIGGRAPH'98 proceedings)*, Addison-Wesley, 32, pp 106-117, 1998.
- [BWH*06] BERGOU M., WARDETZKY M., HARMON D., ZORIN D., GRINSPUN E.: A Quadratic Bending Model for Inextensible Surfaces, *Proc. of Eurographics Symposium on Geometry Processing*, pp 227-230, 2006.
- [CK02] CHOI K.J., KO H.S.: Stable but Responsive Cloth, *Computer Graphics (SIGGRAPH'02 proceedings)*, Addison Wesley, 2002.
- [CSB*06] CORNES N. L., SULLEY R., BRADY A. C., SUMMERS I. R.: Measurements on pacinian corpuscles in the fingertip. *Proc. Enactive'06*, Montpellier, (2006), 117–118.
- [DH05] DOSTMOHAMED H., HAYWARD V.: Trajectory of contact region on the fingerpad gives the illusion of haptic shape. *Experimental Brain Research* 164, Vol.3 (2005), 387–394.
- [DSB99] DESBRUN M., SCHRÖDER P, BARR A.: Interactive Animation of Structured Deformable Objects, *Proceedings of Graphics Interface*, A K Peters, 1999.
- [EDC96] EISCHEN J.W., DENG S., CLAPP T.G.: Finite-Element Modelling and Control of Flexible Fabric Parts, *Computer Graphics in Textiles and Apparel (IEEE Computer Graphics and Applications)*, IEEE Press, pp 71-80, Sept. 1996.
- [EEH00] EBERHARDT B., ETZMUSS O., HAUTH M.: Implicit-Explicit Schemes for Fast Animation with Particle Systems, *Proceedings of the Eurographics workshop on Computer Animation and Simulation*, Springer-Verlag, pp 137-151, 2000.
- [EGS03] ETZMUSS O., GROSS J., STRASSER W.: Deriving a Particle System from Continuum Mechanics for the Animation of Deformable Objects, *IEEE Transactions on Visualization and Computer Graphics*, IEEE Press, pp 538-550, 2003.
- [ES87] EPPINGER S., SEERING, W.: Understanding bandwidth limitations in robot force control. In *Proc. Robotics and Automation Conference, 1987 IEEE International Conference on*, Vol.4 (1987), 904- 909.
- [EWS96] EBERHARDT B., WEBER A., STRASSER W.: A Fast, Flexible, Particle-System Model for Cloth Draping, *Computer Graphics in Textiles and Apparel (IEEE Computer Graphics and Applications)*, IEEE Press, pp 52-59, Sept. 1996.
- [FMF*92] FORMBY, C., MORGAN, L. N., FORREST, T. G., RANEY, J. J.: The role of frequency selectivity in measures of auditory and vibrotactile temporal resolution. *J. Acoust. Soc. Am.* 91, (1992), 293–305.
- [GBH01] GESCHIEDER G. A., BOLANOWSKI S. J., HARDICK K. R.: The frequency selectivity of information-processing channels in the tactile sensory system. *Somatosensory And Motor Research* 18, (2001) 191–

- 201.
- [GGR*03] GOVINDARAJ M., GARG A., RAHEJA A., HUANG G., METAXAS D.: Haptic simulation of fabric hand. *Proc. Eurohaptics '03*, Dublin, (2003), 253–260.
- [GHDS03] GRINSPUN E., HIRANI A., DESBRUN M., SCHRÖDER P., 2003, Discrete Shells, ACM Symposium on Computer Animation, ACM Press.
- [GPU*03] GOVINDARAJ, M., PASTORE, C., UPADHYAY, A., METAXAS, D., HUANG, G., RAHEJA, A.: Haptic simulation of fabric hand. Technical report, *National Textile Research Annual Report*, 2003.
- [Hap05a] HAPTEX CONSORTIUM, "System Requirements and Architectural Design", HAPTEX Deliverable D1.1, <http://haptex.miralab.unige.ch/>, 2005
- [Hap05b] HAPTEX CONSORTIUM, "First set of measurements", HAPTEX Deliverable D3.1, <http://haptex.miralab.unige.ch/>, 2005
- [Hap06a] HAPTEX CONSORTIUM, "Specification of the Whole Haptic Interface", HAPTEX Deliverable D4.1, <http://haptex.miralab.unige.ch/>, 2005
- [Hap06b] HAPTEX CONSORTIUM, "Separate Haptic and Tactile Interface", HAPTEX Deliverable D4.2, <http://haptex.miralab.unige.ch/>, 2005
- [HB04] HAFEZ M., BENALI-KHOUDJA M.: 3D Tactile Rendering Based on Bi (Multi) stable SMA Monolithic Systems. *International Symposium on Micro-mechatronics and Human Science (MHS 2004)*, (2004), 93–98.
- [HE01] HAUTH M., ETZMUSS O.: A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods, Eurographics 2001 proceedings, Blackwell, 2001.
- [Hua02] HUANG, G.: Feel the fabric via the phantom. *PhD thesis*, University of Pennsylvania, 2002.
- [JYV00] JOHNSON K. O., YOSHIOKE T., VEGA-BERMEDEZ F.: Tactile functions of mechanoreceptive afferents innervating the hand. *J. Clin. Neurophysiol.* 17, (2000), 539–558.
- [JM06] JANSSON G., MONACI L.: Identification of real objects under conditions similar to those in haptic displays: providing spatially distributed information at the contact areas is more important than increasing the number of areas. *Virtual Reality 9*, Vol.4 (2006), 243–249.
- [KAK*06] KYUNG K. U., AHN M., KWON D. S., SRINIVASAN M.A.: A compact planar distributed tactile display and effects of frequency on texture judgement. *Advanced Robotics* 20, (2006), 563–580.
- [Kaw80] KAWABATA, S.: The standardization and analysis of hand evaluation, 2nd ed. The hand evaluation and standardization committee, *The Textile Machinery Society of Japan*, Osaka, 1980.
- [KBD*07] KILLEBREW J. H., BENSMAIA S. J., DAMMANN J. F., DENCHEV P., HSIAO S. S., CRAIG J. C., JOHNSON K. O.: A dense array stimulator to generate arbitrary spatio-temporal tactile stimuli. *Journal Of Neuroscience Methods* 161, (Mar. 2007), 62–74.
- [KC02] KANG Y.M., CHO H.G.: Bilayered Approximate Integration for Rapid and Plausible Animation of Virtual Cloth with Realistic Wrinkles, Computer Animation 2000 proceedings, IEEE Computer Society, pp 203-211, 2002.
- [KK07] KYUNG K. U., KWON D. S.: Perceived roughness and correlation with frequency and amplitude of vibrotactile frequency. *Proc. Eurohaptics '06*, Paris (2006), 277–282.
- [LPH07] LEVESQUE V., PASQUERO J., HAYWARD V.: Braille Display by Lateral Skin Deformation with the STReSS² Tactile Transducer. *Proc. World Haptics '07*, Tsukuba, (2007), 115–120.
- [MDDB00] MEYER M., DEBUNNE G., DESBRUN M., BARR A. H., Interactive Animation of Cloth-like Objects in Virtual Reality, Journal of Visualization and Computer Animation, John Wiley & Sons, 2000.
- [MKE03] METZGER J., KIMMERLE S., ETZMUSS O.: Hierarchical Techniques in Collision Detection for Cloth Animation, Journal of WSCG, 11 (2) pp 322-329, 2003.
- [MMLMT05] MÄKINEN M., MEINANDER H., LUIBLE C., MAGNANAT-THALMANN N.: Influence of physical parameters on fabric hand. In *Proc. HAPTEX'05 Workshop on Haptic and Tactile Perception of Deformable Objects* (2005).
- [MTB06] MAGNENAT-THALMANN N., BONANNI U.: Haptics in Virtual Reality and Multimedia. *IEEE MultiMedia*, vol. 13, no. 3, pp. 6-11, Jul-Sept, 2006.
- [PH03] PASQUERO J., HAYWARD V.: STReSS: A Practical Tactile Display System with One Millimeter Spatial Resolution and 700 Hz Refresh Rate. *Proc. Eurohaptics '03*, Dublin, (2003), 94–110.
- [PVTf92] PRESS W.H., VETTERLING W.T., TEUKOLSKY S.A., FLANNERY B.P., Numerical Recipes in C, Second edition, Cambridge University Press, 1992.
- [RHD*87] RABINOWITZ W. M., HOUTSMA, A. J. M., DURLACH, N. I., DELHORNE, L. A.: Multi-dimensional tactile displays: Identification of vibratory intensity, frequency and contactor area. *J. Acoust. Soc. Am* 82, (1987), 1243–1252.
- [SBS*05] SUMMERS I. R., BRADY A. C., SYED M., CHANTER C. M.: Design of Array Stimulators for Synthetic Tactile Sensations. *Proc. World Haptics'05*, Pisa, (2005), 586–587.
- [SC02] SUMMERS I.R., CHANTER C.M.: A broadband tactile array on the fingertip. *J. Acoust. Soc. Am.* 112, (2002), 2118–2126.
- [SCS*01] SUMMERS I. R., CHANTER C. M., SOUTHALL A. L., BRADY A. C.: Results from a Tactile Array on the Fingertip. *Proc. Eurohaptics '01*, Birmingham, (2001) 26–28.
- [She85] SHERRICK C. E.: A scale rate for tactual vibration. *J. Acoust. Soc. Am.* 78, (1985), 78–83.
- [SFR*05] SALSEDO F., FONTANA M., RUFFALDI E., BERGAMASCO M., MAGNENAT-THALMANN N., VOLINO P., BONANNI U., BRADY A., SUMMERS I., QU J., ALLERKAMP D., BÖTTCHER G., WOLTER F.-E.,

- MÄKINEN M., MEINANDER H.: Architectural design of the haptex system. In *Proc. HAPTEX'05 Workshop on Haptic and Tactile Perception of Deformable Objects* (2005).
- [SWM*05] SUMMERS I. R., WHYBROW J. J., MILNES P., BROWN B. H., STEVENS J. C.: Tactile perception; comparison of two stimulation sites. *J. Acoust. Soc. Am.* 118, (2005), 2527–2534.
- [SYYM04] SATO Y., YOSHIKAWA T., YOKOKOHI Y., MURAMORI N.: Designing an encountered-type haptic display for multiple fingertip contacts based on the observation of human grasping behaviour. In *Proceedings of the Symposium on haptic interfaces for virtual environment and teleoperator systems*, (2004) 66-73.
- [TW06] THOMASZEWSKI B., WACKER M.: Bending Models for Thin Flexible Objects, *WSCG Short Communication Proc.*, 9(1), 2006
- [TWS06] THOMASZEWSKI B., WACKER M., STRASSER W., 2006, A Consistent Bending Model for Cloth Simulation with Corotational Subdivision Finite Elements, *Proc. of ACM-SIGGRAPH Symposium on Computer Animation 2006*, pp 107-116.
- [Ver63] VERILLO R. T.: Effect of contactor area on the vibrotactile threshold. *J. Acoust. Soc. Am.* 35, (1963), 1962–1966.
- [VDB*07] VOLINO P., DAVY P., BONANNI U., LUIBLE C., MAGNENAT-THALMANN N., MÄKINEN M., MEINANDER H.: From Measured Physical Parameters to the Haptic Feeling of Fabric. *The Visual Computer*, Springer Berlin/Heidelberg, vol. 23, no. 2, pp. 133–142. February 2007.
- [VFS69] VERILLO R. T., FRAIOLI A. J., SMITH R. L.: Sensation magnitude of vibrotactile stimuli. *Perception & Psychophysics* 6, (1969), 366–372.
- [VGV90] VAN DOREN C. L., GESCHEIDER G. A., VERRILLO R. T.: Vibrotactile temporal gap detection as a function of age. *J. Acoust. Soc. Am.* 87, (1990), 2201–2205.
- [VK93] VOLPE R., KHOSLA P.: A theoretical and experimental investigation of explicit force control strategies for manipulators. In *IEEE Transactions on Automatic Control*, vol.38, 11 (1993), 1634-1650.
- [VMT97] VOLINO P., MAGNENAT-THALMANN N.: Developing Simulation Techniques for an Interactive Clothing System, *Virtual Systems and Multimedia (VSMM'97 proceedings)*, IEEE Press, Geneva, Switzerland, pp 109-118, 1997.
- [VMT00] VOLINO P., MAGNENAT-THALMANN N.: Implementing fast Cloth Simulation with Collision Response, *Computer Graphics International Proceedings*, IEEE Computer Society, pp 257-266, 2000.
- [VMT01] VOLINO P., MAGNENAT-THALMANN N.: Comparing Efficiency of Integration Methods for Cloth Simulation, *Computer Graphics International Proceedings*, IEEE Computer Society, 2001.
- [VMT05] VOLINO P., MAGNENAT-THALMANN N.: Accurate Garment Prototyping and Simulation, *Computer-Aided Design & Applications*, 2(1-4), CAD Solutions, 2005.
- [VMT06] VOLINO P., MAGNENAT-THALMANN N., 2006, Simple Linear Bending Stiffness in Particle Systems, *Proc. of ACM-SIGGRAPH Symposium on Computer Animation 2006*, pp 101-105.
- [VT99] VIRTUAL TECHNOLOGIES INC.: *Cyber Grasp User Guide*, 1999.
- [WLH02] WAGNER C. R., LEDERMAN S. L., HOWE, R. D.: A Tactile Shape Display Using RC Servomotors. *Proc. 10th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, Orlando, (2002), 354.
- [WLH04] WAGNER C. R., LEDERMAN S. L., HOWE R. D.: Design and Performance of a Tactile Shape Display Using RC Servomotors. *Haptics-e: The Electronic Journal Of Haptics Research* 3, (2004).

Populating Virtual Environments with Crowds

Eurographics 2007 Tutorial T2

Organizers

Daniel Thalmann (EPFL, Lausanne, Switzerland)
Carol O'Sullivan (Trinity College, Dublin, Ireland)

Speakers

Daniel Thalmann (EPFL, Lausanne, Switzerland)
Carol O'Sullivan (Trinity College, Dublin, Ireland)
Barbara Yersin (EPFL, Lausanne, Switzerland)
Jonathan Maim (EPFL, Lausanne, Switzerland)
Rachel McDonnell (Trinity College, Dublin, Ireland)

EG 2007 Course on Populating Virtual Environments with Crowds

Organisers

Daniel Thalmann
EPFL VRlab
Email : Daniel.Thalmann@epfl.ch
URL: <http://vrlab.epfl.ch>

Carol O'Sullivan
Trinity College, Dublin, Ireland
Carol.OSullivan@cs.tcd.ie

Lecturers: Daniel Thalmann (EPFL), Carol O'Sullivan (Trinity College), Barbara Yersin (EPFL), Jonathan Maïm (EPFL), Rachel McDonnell (Trinity College)

Abstract

For many years, it was a challenge to produce realistic virtual crowds for special effects in movies. Now, there is a new challenge: the production of real-time autonomous Virtual Crowds. Real-time crowds are necessary for games, VR systems for training and simulation and crowds in Augmented Reality applications. Autonomy is the only way to create believable crowds reacting to events in real-time. This course will present state-of-the-art techniques and methods.

Keywords: population, crowd simulation, informed virtual environments, autonomous agents, perception

Necessary background and potential target audience for the tutorial: experience with computer animation is recommended but not mandatory. The course is intended for animators, designers, and students in computer science.



Detailed outline of the tutorial

The necessity to model virtual populations occurs in many applications of computer animation and simulation. Such applications encompass several different domains – representative or autonomous agents in virtual environments, perceptual metrics and human factors, training, education, simulation-based design, and entertainment. Realistically reproducing dynamic life in the

real-time simulation of virtual environments is also a great challenge.

In this course, we will first present in detail the different approaches to creating virtual crowds, including particle systems with flocking techniques using attraction and repulsion forces, copy and pasting techniques, and agent-based methods.



We will survey methods for animating the individual members that make up a crowd, encompassing a variety of approaches, with particular focus on how example-based synthesis methods can be adapted for crowds. Agent

EUROGRAPHICS 2007

architectures for scalable crowd simulation will also be presented.

The course will cover the topics of real-time crowd rendering, including image-based/impostor, polygonal and point-based techniques. The topic of Level of Detail (LOD) crowd animation will also be covered, not only for rendering, but also for animation. Perceptual issues with respect to the appearance and movement of crowds of characters will be addressed.

New challenges in the production of real-time crowds for games, VR systems for training and simulation will be presented and analysed, with an emphasis on techniques for highly scalable crowd rendering. The course will be illustrated with many examples from recent movies and real-time applications in Emergency Scenarios and Cultural Heritage (such as adding virtual audiences in Roman or Greek theaters).



Resume of the presenters

Daniel Thalmann is Professor and Director of The Virtual Reality Lab (VRLab) at EPFL, Switzerland. He is a pioneer in research on Virtual Humans. His current research interests include Real-time Virtual Humans in Virtual Reality, Networked Virtual Environments, Artificial Life, and Multimedia. He is coeditor-in-chief of the Journal of Computer Animation and Virtual Worlds and member of the editorial board of the Visual Computer and 4 other journals. Daniel Thalmann was member of numerous Program Committees, Co-chair, and Program Co-chair of several conferences including IEEE VR 2000. He has also organized 5 courses at SIGGRAPH on human animation and crowd simulation. Daniel Thalmann has published numerous papers in Graphics, Animation, and Virtual Reality. He is coeditor of 30 books included the recent "Handbook of Virtual Humans", published by John Wiley and Sons and coauthor of several books. He received his PhD in Computer Science in 1977 from the University of Geneva and an Honorary Doctorate (Honoris Causa) from University Paul-Sabatier in Toulouse, France, in 2003.

Carol O'Sullivan is an Associate Professor and acting head of the Computer Science department in Trinity College Dublin, where she also leads the Graphics, Vision and Visualisation group (GV2). Her research interests include computer graphics, perception, virtual humans, crowds, and physically-based animation. She has managed a range of projects with significant budgets and successfully supervised many researchers. She has been a member of many IPCs, including the SIGGRAPH and Eurographics papers committees, and has published about 90 peer-reviewed papers. She is an associate editor of IEEE CG&A and Graphical models, and has organised and co-chaired several international conferences and workshops, including Eurographics 2005, the SIGGRAPH/EG Symposium on Computer Animation 2006 and the SIGGRAPH/EG Campfire on Perceptually Adaptive Graphics 2001.

Rachel McDonnell is a postdoctoral researcher at the Graphics, Vision and Visualization Group in Trinity College Dublin where she recently finished her PhD entitled "Realistic Crowd Animation: A Perceptual Approach". Her research interests include perceptually adaptive graphics, real-time crowd simulation, virtual human animation and cloth simulation.

Barbara Yersin is a PhD student at the VRLab, EPFL. She has achieved her Master project at the University of Montreal, after which she has received her Master in Computer Science in March 2005 from EPFL (Swiss Federal Institute of Technology in Lausanne). Her main interests are in computer graphics, particularly crowd simulation and animation.

Jonathan Maïm is PhD student at VRLab at the Swiss Federal Institute of Technology in Lausanne (EPFL). In April 2005, he receives a Master Degree in Computer Science from EPFL after achieving his Master Project at the University of Montréal. His research efforts are concentrated on building an architecture for simulating real-time crowds of virtual humans.

Selected Publications

S. Raupp Musse, D.Thalmann, A Behavioral Model for Real Time Simulation of Virtual Human Crowds, IEEE Transactions on Visualization and Computer Graphics, Vol.7, No2, 2001, pp.152-164.

B. Ulicny, P. de Heras Ciechowski, D. Thalmann, Crowdbrush: Interactive Authoring of Real-time Crowd Scenes, Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation '04, 2004, pp.243-252

B. Ulicny, D. Thalmann, Towards Interactive Real-Time Crowd Behavior Simulation, Computer Graphics Forum, 21(4):767-775, December 2002

P. de Heras Ciechowski, S. Schertenleib, J. Maïm, D. Maupu and D. Thalmann, Real-time Shader Rendering for Crowds in Virtual Heritage, VAST '05, 2005

EUROGRAPHICS 2007

N. Magnenat-Thalmann, D. Thalmann (eds), Handbook of Virtual Humans, John Wiley, 2004

C. O'Sullivan, J. Cassell, H. Vilhjalmsson, J. Dingliana, S. Dobbyn, B. McNamee, C. Peters and T. Giang, Levels of Detail for Crowds and Groups, Computer Graphics Forum, 21(4), 2002.

S. Dobbyn, J. Hamill, K. O'Connor and C. O'Sullivan, Geopostors: A Real-Time Geometry/Impostor Crowd Rendering System. ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games 2005, pp. 95-102.

S. Dobbyn, R. McDonnell, L. Kavan, S. Collins, and C. O'Sullivan, Clothing the masses: Real-time clothed crowds with variation. In Eurographics Short Papers 2006, pp. 103–106.

R. McDonnell, S. Dobbyn, and C. O'Sullivan, Crowd Creation Pipeline for Games, In Proceedings of the 9th International Conference on Computer Games, CGames 2006, pp. 183-190.

J. Hamill, R. McDonnell, S. Dobbyn, and C. O'Sullivan, Perceptual Evaluation of Impostor Representations for Virtual Humans and Buildings. Computer Graphics Forum 24(3) (EUROGRAPHICS 2005 Proceedings) 2005.

R. McDonnell, S. Dobbyn, and C. O'Sullivan, LOD Human Representations: A Comparative Study. Proceedings of the First International Workshop on Crowd Simulation (V-CROWDS '05) 2005.

R. McDonnell, S. Dobbyn, S. Collins, and C. O'Sullivan, Perceptual evaluation of LOD clothing for virtual humans. In Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation 2006, pp. 117–126.

J. Pettré, P. de Heras Ciechowski, J. Maïm, B. Yersin, J.-P. Laumond, D. Thalmann. Real-time navigating crowds: scalable simulation and rendering. Computer Animation and Virtual Worlds. Volume 17, Issue 3-4 , Pages 445 – 455. Special Issue: CASA 2006.

B. Yersin, J. Maïm, P. de Heras Ciechowski, S. Schertenleib and D. Thalmann, Steering a Virtual Crowd Based on a Semantically Augmented Navigation Graph, Proceedings of the First International Workshop on Crowd Simulation (V-CROWDS '05) 2005.

Smooth Movers: Perceptually Guided Human Motion Simulation. Rachel McDonnell, Fional Newell and Carol O'Sullivan. Eurographics/ACM SIGGRAPH Symposium on Computer Animation (SCA'07). To appear (2007)

State-of-the-Art: Real-Time Crowd Simulations

B. Ulicny, P. de Heras Ciechowski, S. R. Musse² & D. Thalmann

VRLab, EPFL
CH-1015, Lausanne, Switzerland
branislav.ulicny, pablo.deheras, daniel.thalmann@epfl.ch
<http://vrlab.epfl.ch>

² CROMOS Lab, Universidade do Vale do Rio dos Sinos
Ciências Exatas e Tecnológicas - PIPCA
Av. Unisinos 950
93022-000 - São Leopoldo - RS, Brazil
soraiaarm@exatas.unisinos.br
<http://www.inf.unisinos.br/cromoslab>

Abstract

Crowds are part of our everyday experience; nevertheless, in virtual worlds they are still relatively rare. In the past, main reasons hindering a wider use of virtual crowds in the real-time domain were their high demands on both general and graphics performance coupled with high costs of content production. The situation is, though, changing fast; market forces are pushing performance of the consumer hardware up, reaching and surpassing performance of professional graphics workstations from just few years ago. With current consumer-grade personal computers it is possible to display 3D virtual scenes with thousands of animated individual entities at interactive framerates. In this report, we present the related works on the subject of groups and crowd simulation discussing several areas such as behavioral simulation, crowd motion control, crowd rendering and crowd scenario authoring.

Keywords: Autonomous agents, behavioral animation, computer graphics, crowd simulations, flocking, image-based rendering, multi-agent systems, impostors, virtual reality.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation I.3.3 [Picture/Image Generation]: Display algorithms I.2.11 [Distributed Artificial Intelligence]: Multi-agent systems

1. Introduction to crowd simulations

Although collective behavior has been studied since as early as the end of the nineteenth century [LeB95], attempts to simulate it by computer models are quite recent, with most of the works done only in the mid and late nineties. In the past years researchers from a broad range of fields such as architecture [SOHTG99, PT01, TP02], computer graphics [BG96, HB94, MT01, Rey87, TLC02b, UT02, BMdOB03], physics [HM95, HFV00, FHV02], robotics [MS01], safety science [Sim04, Sti00, TM95a], training systems [Bot95, VSMA98, Wil95], and sociology [JPvdS01, MPT92, TSM99] have been creating simu-

lations involving collections of individuals. Nevertheless, despite apparent breadth of the crowd simulation research basis, it can be noted that interdisciplinary exchange of ideas is rare; researchers in one field are usually not very aware of works done in the other fields.

Most approaches were application-specific, focusing on different aspects of the collective behavior, using different modeling techniques. Employed techniques range from those that do not distinguish individuals such as flow and network models in some of the evacuation simulations [TT92], to those that represent each individual as being controlled by more or less complex rules based on physical laws

[HFV00, HIK96], chaos equations [SKN98] or behavioral models in training systems [Wil95] or sociological simulations [JPvdS01].

We can distinguish two broader areas of crowd simulations. The first one is focusing on a **realism of behavioral aspects** with usually simple 2D visualizations like evacuation simulators, sociological crowd models, or crowd dynamics models. In this area, a simulated behavior is usually from a very narrow, controlled range (for example, people just flying to exit or people forming ring crowd structures) with efforts to quantitatively validate correspondence of results to real world observations of particular situations [TM95b]. Ideally, a simulation's results would be then consistent with data sets collected from field observations or video footage of real crowds either by human observers [SM99] or by some automated image processing method [CYC99, MVCL98]. Visualization is used to help to understand simulation results, but it is not crucial. In most cases, a schematic representation, with crowd members represented by colored dots, or sticky figures, is enough, sometimes even preferable as it allows highlighting important information.

In the second area, a main goal is **high quality visualization** (for example, in movie productions and computer games), but usually the realism of the behavior model is not the priority. What is important is a convincing visual result, which is achieved partly by behavior models, partly by human intervention in the production process. A virtual crowd should both look well and be animated in a believable manner, the emphasis of the research being mostly on rendering and animation methods. Crowd members are visualized as fully animated three dimensional figures that are textured and lit to fit into the environment [TLC02b]. Here, behavior models do not necessarily aim to match quantitatively the real world, their purpose is more in alleviating of human animators work, and to be able to respond to inputs in case of interactive applications.

Nevertheless, a recent trend seems to be a **convergence of both areas**, where visualization oriented systems are trying to incorporate better behaviors models to ease creation of convincing animations [Ant98, Cha04] and behavior oriented models are trying to achieve better visualization, especially in the domain of evacuation simulators [Exo04, STE]. We can expect that the most demanding applications would be training systems, where both valid replication of the behaviors and high quality visualization is necessary for a training to be effective.

1.1. Requirements and constrains for crowd modeling

Real-time crowds bring different challenges compared to the systems either involving small number of interacting characters (for example, the majority of contemporary computer games), or non-real-time applications (as crowds in movies, or visualizations of crowd evacuations after off-line model

computations). In comparison with single-agent simulations, the main conceptual difference is the **need for efficient variety management** at every level, whether it is visualization, motion control, animation or sound rendering. As everyday experiences hint, virtual humans composing a crowd should look different, move different, react different, sound different and so forth. Even if assuming perfect simulation of a single virtual human would be possible, still creating a simulation involving multiple such humans would be a difficult and tedious task. Methods easing control of many characters are needed; however such methods should still preserve ability to control individual agents.

In comparison with non-real-time simulations, the main technical challenge is **increased demand on computational resources** whether it is general processing power, graphics performance or memory space. One of the foremost constraining factors for real-time crowd simulations is crowd rendering. Fast and scalable methods both to compute behavior, able to take into account inputs not known in advance, and to render large and varied crowds, are needed. While non-real-time simulations are able to take advantage of knowing a full run of the simulated scenario (and therefore, for example, can run iteratively over several possible options selecting the globally best solution), real-time simulations have to react to the situation as it unfolds in the moment.

2. Crowd simulation areas

In order to create a full simulation of the crowd in the virtual environment, many issues have to be solved. The areas of relevance for crowd simulation and some associated questions include:

Crowd behavior generation: How should a virtual crowd respond to changes in their surroundings? How should agents respond to behaviors of other agents? What is an appropriate way of modeling perception for many agents? [Rey87, TT94, HB94, BCN97, BH97, Rey99, Mus00] [UT02, NG03]

Crowd motion control: How should virtual entities move around and avoid collisions with both a static environment and dynamic objects? How can a group move in a coordinated manner? [ALA*01, GKM*01, AMC03, LD04]

Integration of crowds in virtual environments:

Which aspects of the environment need to be modeled? Which representation of environmental objects is best suited for fast behavior computation? [FBT99, BLA02, KBT03, LMA03]

Virtual crowd rendering and animation: How to display many animated characters fast? How to display a wide variety of appearances? How to generate varied animations? [ABT00, LCT01, TLC02a, WS02]

Interaction with virtual crowds: How and which information should be exchanged between real and virtual

humans? What is the most efficient metaphor to direct crowds of virtual extras? [FRMS*99, UdHCT04]

Generation of virtual individuals: How to generate a heterogeneous crowd? How to create a population with desired distribution of features? [GKMT01, SYCGMT02]

Authoring of scenarios: How to author complex crowd scenes in an efficient way? How to distribute crowd members in designated areas? How to distribute features over a population? [Che04, UdHCT04]

Many of these aspects are to a greater or lesser extent intertwined. For example, efficiency of rendering constrains the possible variety of behaviors and appearances; higher-level behavior generation controls lower-level motion systems, but the behavior should also respond appropriately to collisions encountered while moving; the behavior model affects interaction possibilities; the environment representation affects possible behaviors; authoring tools allow handling of more complex behavior and environment representations and so on.

3. Overview of crowd simulations

3.1. Crowd evacuation simulators

One of the largest areas where crowd behaviors have been modeled is the domain of safety science and architecture with the dominant application of crowd evacuation simulators. Such systems model movements of a large number of people in usually closed and well-defined spaces like inner areas of buildings [TM95a], subways [Har00], ships [KMKWS00] or airplanes [OGLF98]. Their goal is to help designers to understand the **relation between the organization of space and human behavior** [OM93].

The most common use of evacuation simulators is the modeling of crowd behavior in case of forced evacuation from a confined environment due to some threat like fire or smoke. In such a situation, a number of people have to evacuate the given area, usually through a relatively small number of fixed exits. Simulations are trying to help with answering questions like: Can the area be evacuated within a prescribed time? Where do the hold-ups in the flow of people occur? Where are the likely areas for a crowd surge to produce unacceptable crushing pressure? [Rob99] The most common modeling approach in this area is the use of cellular automata serving both as a representation of individuals and a representation of the environment.

Simulex [TM95a, Sim04] is a computer model simulating the escape movement of persons through large, geometrically complex building spaces defined by 2D floor plans and connecting staircases. Each individual has attributes such as position, body size, angle of orientation and walking speed. Various algorithms as distance mapping, way finding, overtaking, route deviation and adjustment of individual speeds due to proximity of crowd members are used to compute

egress simulation, where individual building occupants walk towards and through the exits.

K. Still developed a collection of programs named *Legion* for simulation and analysis of the crowd dynamics in evacuation from constrained and complex environments like stadiums [Sti00]. Dynamics of crowd motion is modeled by mobile cellular automata. Every person in the crowd is treated as an individual, calculating its position by scanning its local environment and choosing an appropriate action.

3.2. Crowd management training systems

The modeling of crowds has also been essential in police and military simulator systems used for training in how to deal with mass gatherings of people.

CACTUS [Wil95] is a system developed to assist in planning and training for public order incidents such as large demonstrations and marches. The software designs are based on a world model in which crowd groups and police units are placed on a digitized map and have probabilistic rules for their interactive behavior. The simulation model represents small groups of people as discrete objects. The behavioral descriptions are in the form of a directed graph where the nodes describe behavioral states (to which correspond actions and exhibited emotions) and transitions represent plausible changes between these states. The transitions depend on environmental conditions and probability weightings. The simulation runs as a decision making exercise that can include pre-event logistic planning, incident management and debriefing evaluation.

Small Unit Leader Non-Lethal Training System [VSMA98] is a simulator for training US Marines Corps in decision making with respect to the use of non-lethal munitions in peacekeeping and crowd control operations. Trainees learn rules of engagement, the procedures for dealing with crowds and mobs and ability to make decisions about the appropriate level of force needed to control, contain, or disperse crowds and mobs. Crowds move within a simulated urban environment along instructor-predefined pathways and respond both to actions of a trainee and to actions of other simulated crowds. Each crowd is characterized by a crowd profile - series of attributes like fanaticism, arousal state, prior experience with non-lethal munitions, or attitude toward Marines. During an exercise, the crowd behavior computer model operates in real time and responds to trainee actions (and inactions) with appropriate simulated behaviors such as loitering, celebrating, demonstrating, rioting and dispersing according to set of Boolean relationships defined by experts.

3.3. Sociological models of crowds

Despite being a field primarily interested in studying collective behavior, only a relatively small number of works on crowd simulations have been done in sociology.

McPhail et al. [MPT92] studied individual and collective actions in temporary gatherings. Their model of the crowd is based on perception control theory [Pow73] where each separate individual is trying to control his or her experience in order to maintain a particular relationship to others: in this case it is a spatial relationship with others in a group. The simulation program called *GATHERING* graphically shows movement, milling, and structural emergence in crowds. The same simulation system was later used by Schweingruber [Sch95] to study the effects of reference signals common to coordination of collective behavior and by Tucker et al. [TSM99] to study formation of arcs and rings in temporary gatherings.

Jager et al. [JPvdS01] modeled clustering and fighting in two-party crowds. Crowd is modeled by a multi-agent simulation using cellular automata with rules defining approach-avoidance conflict. The simulation consists of two groups of agents of three different kinds: hardcore, hangers-on and bystanders, the difference between them consisting in the frequency with which they scan their surroundings. The goal of the simulation was to study effects of group size, size symmetry and group composition on clustering, and 'fights'.

3.4. Group behavior in robotics and artificial life

Researchers working in the field of artificial life are interested in exploring how group behavior emerges from local behavioral rules [Gil95]. Software models and groups of robots were designed and experimented with in order to understand how complex behaviors can arise in systems guided by simple rules. The main source of inspiration is nature, where, for example, social insects efficiently solve problems such as finding food, building nests, or division of labor among nestmates by simple interacting individuals without an overseeing global controller. One of the important mechanisms contributing to a distributed control of the behavior is *stigmergy*, indirect interactions among individuals through modifications of the environment [BDT99].

Dorigo introduced *ant systems* inspired by behaviors of real ant colonies [Dor92]. Ant algorithms have been successfully used to solve a variety of discrete optimization problems including travelling salesman problem, sequential ordering, graph colouring or network routing [BDT00]. Besides insects, also groups of more complex organisms such as flocks of birds, herds of animals and schools of fish have been studied in order to understand principles of their organization. Recently, Couzin et al. presented a model how animals that forage or travel in groups can make decisions even with a small number of informed individuals [CKFL05].

Principles from biological systems were also used to design behavior controllers for autonomous groups of robots. Mataric studied behavior-based control for a group of robots, experimenting with a herd of 20 robots whose behavior repertoire included safe wandering, following, aggregation,

dispersion and homing [Mat97]. Molnar and Starke have been working on assignment of robotic units to targets in a manufacturing environments using a pattern formation inspired by pedestrian behavior [MS01]. Martinoli applied swarm intelligence principles to autonomous collective robotics, performing experiments with robots that were gathering scattered objects and cooperating to pull sticks out of the ground [Mar99]. Holland and Melhuish experimented with a group of robots doing sorting of objects based on ant behaviors where ants sort larvae and cocoons [HM99]. In an interesting work a robot was used to control animal behavior, Vaughan et al. developed a mobile robot that gathers a flock of real ducks and manoeuvres them safely to a specified goal position [VSH*00].

3.5. Crowds in virtual worlds

In order to have a persuasive application using crowds in virtual environments, various aspects of the simulation have to be addressed, including behavioral animation, environment modeling and crowd rendering. If there is no satisfactory rendering, even the best behavior model will not be very convincing. If there is no good model of a behavior, even a simulation using the best rendering method will look dumb after only few seconds. If there is no appropriate model of the environment, characters will not behave believably, as they will perform actions at wrong places, or not perform at all.

The goal of behavioral animation is to **ease the work of designers** by letting virtual characters perform autonomously or semi-autonomously complicated motions which otherwise would require large amounts of human animators' work; or, in case of interactive applications, the behavioral models allow characters to **respond to user initiated actions**.

In order for a behavior to make sense, besides characters, also their surrounding environment has to be modeled, not just graphically but also semantically. Indeed, a repertoire of possible behaviors is very dependent on what is and what is not included in a model of the environment. It happens very often that the environment is visually rich, but the interaction of characters with it is minimal.

Finally, for interactive applications, it is necessary to display a varied ensemble of virtual characters in an efficient manner. Rendered characters should visually 'fit' into the environment, they should be affected by light and other effects in the same manner as their surroundings.

Next, we will present representative works for each of these topics grouped according to their main focus.

Behavioral animation of groups and crowds

Human beings are arguably the most complex known creatures, therefore they are also the most complex creatures

to simulate. A behavioral animation of human (and humanoid) crowds is based on foundations of group simulations of much more simple entities, notably flocks of birds [Rey87, GA90] and schools of fish [TT94]. The first procedural animation of flocks of virtual birds was shown in the movie by Amkraut, Girard and Karl called *Eurhythm*, for which the first concept [AGK85] was presented at The Electronic Theater at SIGGRAPH in 1985 (final version was later presented at Ars Electronica in 1989). The flock motion was achieved by a global vector force field guiding a flow of flocks [GA90].

In his pioneer work, Reynolds [Rey87] described a distributed behavioral model for simulating aggregate motion of a flock of birds. The technical paper was accompanied by an animated short movie called “Stanley and Stella in: Breaking the Ice” shown at the Electronic Theater at SIGGRAPH '87. The revolutionary idea was that a **complex behavior** of the group of actors can be obtained by **simple local rules** for members of the group instead of some enforced global condition. The flock is simulated as a complex particle system, with the simulated birds (called *boids*) being the particles. Each boid is implemented as an independent agent that navigates according to its local perception of the environment, the laws of simulated physics, and the set of behaviors. The boids try to avoid collisions with one another and with other objects in their environment, match velocities with nearby flock mates and move towards a center of the flock. The aggregate motion of the simulated flock is the result of the interaction of these relatively simple behaviors of the individual simulated birds. Reynolds later extended his work by including various steering behaviors as goal seeking, obstacle avoidance, path following or fleeing [Rey99], and introduced a simple finite-state machines behavior controller and spatial queries optimizations for real-time interaction with groups of characters [Rey00].

Tu and Terzopoulos proposed a framework for animation of artificial fishes [TT94]. Besides complex individual behaviors based on perception of the environment, virtual fishes have been exhibiting unscripted collective motions as schooling and predator evading behaviors analogous to flocking of boids.

An approach similar to boids was used by Bouvier et al. [BG96, BCN97] to simulate human crowds. They used a combination of particle systems and transition networks to model crowds for the visualization of urban spaces. At the lower level, attraction and repulsion forces, analogous to physical electric forces, enable people to move around the environment. Goals generate attraction forces, obstacles generate repulsive force fields. Higher level behavior is modeled by transition networks with transitions depending on time, visiting of certain points, changes of local population densities and global events.

Brogan and Hodgins [BH97, HB94] simulated group behaviors for systems with **significant dynamics**. Compared

to the boids, a more realistic motion is achieved by taking into account physical properties of the motion, such as momentum or balance. Their algorithm for controlling the movements of creatures proceeds in two steps: first, a perception model determines the creatures and obstacles visible to each individual, and then a placement algorithm determines the desired position for each individual given the locations and velocities of perceived creatures and obstacles. Simulated systems included groups of one-legged robots, bicycle riders and point-mass systems.

Musse and Thalmann [Mus00, MT01] presented a **hierarchical model** for real-time simulation of virtual human crowds. Their model is based on groups, instead of individuals: groups are more intelligent structures, individuals follow the groups specification. Groups can be controlled with different levels of autonomy: guided crowds follow orders (as go to certain place or play a particular animation) given by the user in run-time; programmed crowds follow a scripted behavior; and autonomous crowds use events and reactions to create more complex behaviors. The environment comprises a set of interest points, which signify goals and waypoints; and a set of action points, which are goals that have associated some actions. Agents move between waypoints following Bezier curves.

Recently, another work was exploring group modeling based on hierarchies. Niederberger and Gross [NG03] proposed an architecture of hierarchical and heterogeneous agents for real-time applications. Behaviors are defined through specialization of existing behavior types and weighted multiple inheritance for creation of new types. Groups are defined through recursive and modulo based patterns. The behavior engine allows for the specification of a maximal amount of time per run in order to guarantee a minimal and constant framerates.

Ulicny and Thalmann [UT01, UT02] presented a crowd behavior simulation with a modular architecture for multi-agent system allowing autonomous and scripted behavior of agents supporting variety. In their system, the behavior is computed in layers, where decisions are made by behavioral rules and execution is handled by hierarchical finite-state machines.

Perceived complexity of the crowd simulation can be increased by using **level of details** (LOD). O’Sullivan et al. [OCV*02] described a simulation of crowds and groups with level of details for geometry, motion and behavior. At the geometrical level, subdivision techniques are used to achieve smooth rendering LOD changes. At the motion level, the movements are simulated using adaptive levels of detail. Animation subsystems with different complexities, as a keyframe player or a real-time reaching module, are activated and deactivated based on heuristics. For the behavior, LOD is employed to reduce the computational costs of updating the behavior of characters that are less important. More complex characters behave according to their

motivations and roles, less complex ones just play random keyframes.

Environment modeling for crowds

Environment modeling is closely related to the behavioral animation. The purpose of the models of the environment is to facilitate simulation of entities dwelling in their surrounding environments. Believability of virtual creatures can be greatly enhanced if they behave in accordance with their surroundings. On the contrary, the suspense of disbelief can be immediately destroyed if they perform something not expected or not permitted in the real world, such as passing through the wall or walking on the water. The greatest efforts have been therefore directed to representations and algorithms preventing 'forbidden' behaviors to occur: till quite recently the two major artificial intelligence issues concerning game development industry were collision avoidance and path-planning [Woo99, DeL00].

Majority of the population in the developed world lives in cities; it is there where most of the human activities take place nowadays. Accordingly, most of the research have been done for **modelling of virtual cities**. Farenc et al. [FBT99] introduced an **informed environment** dedicated to the simulation of virtual humans in the urban context. The informed environment is a database integrating semantic and geometrical information about a virtual city. It is based on a hierarchical decomposition of a urban scene into environment entities, like quarters, blocks, junctions, streets and so on. Entities can contain a description of the behaviors that are appropriate for agents located on them; for example, sidewalk tells that it should be walked on, or bench tells that it should be sit on. Furthermore, the environment database can be used for a path-finding that is customized according to the type of the client requesting the path, so that, for example, a pedestrian will get paths using sidewalks, but a car will get paths going through roads.

Another model of a virtual city for a behavioral animation was presented by Thomas and Donikian [TD00]. Their model is designed with the main emphasis on a traffic simulation of vehicles and pedestrians. The environment database is split into two parts - a hierarchical structure containing a tree of polygonal regions, similar to the informed environment database; and a topological structure with a graph of a road network. Regions contain information on directions of circulation, including possible route changes at intersections. The agents then use the database to navigate through the city.

In a recent work, Sung et al. [SGC04] presented a new approach to control the behavior of a crowd by storing behavioral information into the environment using structures called **situations**. Compared to previous approaches, environmental structures (situations) can overlap; behaviors corresponding to such overlapping situations are then composed using probability distributions. Behavior functions define

probabilities of state transitions (triggering motion clips) depending on the state of the environment features or on the past state of the agent.

On the side focused on more generic **path-planning** issues, several works have been done. Kallmann et al. [KBT03] proposed a fast path-planning algorithm based on a fully dynamic constrained Delaunay triangulation. Bayazit et al. [BLA02] used global roadmaps to improve group behaviors in geometrically complex environments. Groups of creatures exhibited behaviors such as homing, goal searching, covering or shepherding, by using rules embedded both in individual flock members and in roadmaps. Tang et al. [TWP03] used a modified A* algorithm working on grid overlayed over a height-map generated terrain. Recently, Lamarche and Donikian [LD04] presented a topological structure of the geometric environment for a fast hierarchical path-planning and a reactive navigation algorithm for virtual crowds.

Crowd rendering

Real-time rendering of a large number of 3D characters is a considerable challenge; it is able to exhaust system resources quickly even for state of the art systems with extensive memory resources, fast processors and powerful graphic cards. 'Brute-force' approaches that are feasible for a few characters do not scale up for hundreds, thousands or more of them. Several works have been trying to circumvent such limitations by clever use of graphics accelerator capabilities, and by employing methods profiting from the fact that our perception of the scene as a whole is limited.

We can perceive in full details only a relatively small part of a large collection of characters. A simple calculation shows that to treat every crowd member as equal is rather wasteful. Modern screens can display around two millions of pixels at the same time, where fairly complex character can contain approximately ten thousand triangles. Even if assuming that every triangle would be projected to a single pixel, and that there would be no overlap of characters, the screen fully covered by a crowd would contain only around two hundred simultaneously visible characters. Of course, in reality the number would be much smaller, a more reasonable estimate is around a few dozen of fully visible characters, with the rest of the crowd being either hidden behind these prominent characters or taking significantly less screen space. Therefore it makes sense to take full care only of the foremost agents, and to replace the others with some less complex approximations. Level of details techniques then switch visualizations according to position and orientation of the observer.

Billboarded impostors are one of the methods used to speed up crowd rendering. Impostors are partially transparent textured polygons that contain a snapshot of a full 3D character and are always facing the camera. Aubel et al. [ABT00] introduced dynamically generated impostors to

render animated virtual humans. In their approach, an impostor creating process is running in parallel to full 3D simulations, taking snapshots of rendered 3D characters. These cached snapshots are then used over several frames instead of the full geometry until a sufficient movement of either camera or a character will trigger another snapshot, refreshing the impostor texture.

In another major work using impostors, Tecchia et al. [TLC02a] proposed a method for real-time rendering of animated crowd in a virtual city. Compared to the previous method, impostors are not computed dynamically, but are created in a preprocessing step. Snapshots are sampled from viewpoints distributed in the sphere around the character. This process is repeated for every frame of the animation. In run-time, images taken from viewpoints closest to the actual camera position are then used for texturing of the billboard. Additionally, the silhouettes of the impostors are used as shadows projected to a ground surface. Multitexturing is used to add variety by modulating colors of the impostors. In a later work they added lighting using normal maps [TLC02b]. Their method using precomputed impostors is faster than dynamical impostors, however it is very demanding on texture memory, which leads to lower image quality as size of textures per character and per animation frame have to be kept small.

A different possibility for a fast crowd display is to use **point-based rendering techniques**. Wand and Strasser [WS02] presented a multi-resolution rendering approach which unifies image based and polygonal rendering. They create a view dependant octree representations of every keyframe of animation, where nodes store either a polygon or a point. These representations are also able to interpolate linearly from one tree to another so that in-between frames can be calculated. When the viewer is at a long distance, the human is rendered using point rendering; when zoomed in, using polygonal techniques; and when in between, a mix of the two.

An approach that has been getting new life is the one of **geometry baking**. By taking snapshots of vertex positions and normals, complete mesh descriptions are stored for each frame of animation. Since current desktop PCs have large memories many such frames can be stored and re-played. A hybrid approach of both baked geometry and billboarding is to be presented at I3d, where only a few actors are fully geometrical while the vast number of actors are made up of billboards [DH005].

What is common to all approaches is instancing of template humans, by changing the texture or color, size, orientation, animation, animation style and position. This is carefully taken care of to smoothly transition from one representation to another so as not to create pops in representation styles. In the billboarding scenario this is done by applying different colors on entire zones such as torso, head, legs and arms. This way the texture memory is used more efficiently

as the templates are more flexible. For the geometrical approaches these kind of differences are usually represented using entirely different textures as the humans are too close just to change basic colour for an entire zone [UdHCT04].

Crowds in non-real time productions

One of the domains with a fastest growth of crowd simulations in recent years are special effects. While only ten years ago, there were no digital crowds at all, nowadays almost every blockbuster has some, with music videos, television series and advertisements starting to follow. In comparison with crowds of real extras, virtual crowds allow to significantly reduce costs of production of massively populated scenes and allow for bigger creative freedom because of their flexibility. Different techniques, as replications of real crowd video footage, particle systems or behavioral animation, have been employed to add crowds of virtual extras to shots in a broad range of movies, from historical dramas [Tit97, Gla00, Tro04], through fantasy and science fiction stories [Sta02, The03, Mat03], to animated cartoons [The94, Ant98, A b98, Shr04].

The main factors determining the choice of techniques are the required visual quality and the production costs allowed for the project [Leh02]. It is common to use different techniques even in a single shot in order to achieve the best visuals - for example, characters in the front plane are usually real actors, with 3D characters taking secondary roles in the background.

Although a considerable amount of work was done on crowds in movies, only relatively little information is available, especially concerning more technical details. Most knowledge comes from disparate sources, for example, from "making-of" documentary features, interviews with special effect crew or industry journalist accounts. For big budget productions, the most common approach is **in-house development of custom tools** or suites of tools which are used for a particular movie. As the quality of the animation is paramount, large libraries of motion clips are usually used, produced mainly by motion capture of live performers. All production is centered around shots, most of the times only few seconds long. In contrast to real-time simulations, there is little need for continuity of the simulation over longer periods of the time. It is common that different teams of people work on parts of the shots which are then composited in post-processing.

The most advanced crowd animation system for non real-time productions is *Massive*; used to create battle scenes for *The Lord of the Rings* movie trilogy [Mas04]. In *Massive*, every agent makes decisions about its actions depending on its sensory inputs using a brain composed of thousands of logic nodes [Koe02]. According to brain's decision, the motion is selected from extensive library of motion captured clips with precomputed transitions. For example, in the second part of the trilogy over 12 millions of motion captured frames

(equivalent to 55 hours of animation) was used. Massive also uses rigid body dynamics, a physics-based approach to facilitating realistic stunt motion such as falling, or animation of accessories. For example, a combination of physics-based simulation and custom motion capture clips was used to create the scene of “The Flooding of Isengard” where orcs are fleeing from a wall of water and falling down the precipice [Sco03].

In comparison with real-time application, the quality of motion and visuals in non real-time productions is far superior, however it comes at a great cost. For example for *The Lord of the Rings: The Two Towers*, rendering of all digital characters took ten months of computations on thousands computer strong render farm [Doy03].

Crowds in games

In current computer games virtual crowds are still relatively rare. The main reason is that crowds are inherently costly, both in terms of real-time resources requirements and for costs of a production. Nevertheless, the situations is starting to change, with real-time strategy genre leading the way as increase of sizes of involved armies has direct effect on a gameplay [Rom04, The04a].

The main concern for games is the **speed of both rendering and behavior computation**. In comparison with non real-time productions, the quality of both motion and rendering is often sacrificed in a trade-off for fluidity. Similarly to movies production, computer games often inject into virtual world realism coming from real world by using large libraries of animation, which are mostly motion captured. The rendering uses level-of-details techniques, with some titles employing animated impostors [Med02].

To improve costs of behavior computations for games that involve a large number of simulated entities, simulation level-of-detail techniques have been employed [Bro02, Rep03]. In such techniques, behavior is computed only for a characters that are visible or near to be visible to a player. Characters are created in a space around the player with parameters set according to some expected statistical distributions, the player lives in a “simulation bubble”. However, handling simulation LOD is much more complex as handling of rendering LOD. It is perfectly correct not to compute visualization for agents that are not visible, but not computing behaviors for hidden agents can lead to an incoherent world. In some games it is common that the player causes some significant situation (for example, traffic jam), looks away, and then after looking back, the situation is changed in an unexpected way (a traffic jam is “magically” resolved).

In case the scenario deals with hundreds or thousands of entities, many times the selectable unit with distinct behavior is a formation of troops, not individual soldiers. What appears to be many entities on the screen is indeed only

one unit being rendered as several visually separated parts [Sho00, Med02, Pra03].

A special case are sport titles such as various football, basketball or hockey simulations, where there is a large spectator crowd, however only of very low details. In the most cases there is not even a single polygon for every crowd member (compared to individual impostors in strategy games). Majority of the crowd is just texture with transparency applied to stadium rows, or to a collection of rows, and only few crowd members, close to the camera can be very low polygon count 3D models.

Crowd scenario authoring

No matter how good is a crowd rendering or a behavior model, a virtual crowd simulation is not very useful, if it is too difficult to produce a content for it. The authoring possibilities are an important factor influencing the usability of crowd simulation system, especially when going beyond a limited number of “proof-of-concept” scenarios. When increasing the number of involved individuals it becomes more difficult to create unique and varied content of scenarios with large numbers of entities. Solving one set of problems for crowd simulation (such as fast rendering and behavior computation for large crowds) creates a new problem of how to create content for crowd scenarios in an efficient manner.

Only recently, researchers started to explore the ways how to author crowd scenes. Anderson et al. [AMC03] achieved interesting results for a particular case of flocking animation following constrains. Their method can be used, for instance, to create and animate flocks moving in shapes. Their algorithm generates a constrained flocking motion by iterating simulation forwards and backwards. Nevertheless, their algorithm can get very costly when increasing the number of entities and simulation time.

Ulicny et al. [UdHCT04] proposed a method to create complex crowd scenes in an intuitive way using a Crowd-Brush tool. By employing a brush metaphor, analogous to the tools used in image manipulation programs, the user can distribute, modify and control crowd members in real-time with immediate visual feedback. This approach works well for creation and modification of spatial features, however the authoring of temporal aspects of the scenario is limited.

Sung et al. [SGC04] used a situation-based distributed control mechanism that gives each agent in a crowd specific details about how to react at any given moment based on its local environment. A painting interface allows to specify situations easily by drawing their regions on the environment directly like drawing a picture on the canvas. Compared to previous work where the user adds, modifies and deletes crowd members, here the interface operates on the environment.

Chenney [Che04] presented a novel technique for representing and designing velocity fields using flow tiles. He applied his method on a city model with tiles defining the flow

of people through the city streets. Flow tiles drive the crowd using the velocity to define the direction of travel for each member. The use of divergence free flows to define crowd motion ensures that, under reasonable conditions, the agents do not require any form of collision detection.

4. Discussion

We presented an overview of the works on crowd simulations done in different fields such as sociology, safety science, training systems, computer graphics or entertainment industry. Based on the analysis of published research works and data available on industry applications, we made following observations.

Domain specificity: While some of the know-how is transferable across the fields, each of the domains dealing with crowds poses unique challenges and requires different solutions. It is indeed the targeted application that drives most of the design choices while creating a simulation of the crowd. There is no "silver bullet" solution, the ultimate crowd simulation that would be fitting all purposes. Features that are advantageous for one purpose are disadvantages in the other and trade-offs have to be resolved in a different manner. For example, most of the crowd evacuation simulators use discrete 2D grid representation of the world as it is easier to handle, to analyze and to validate. However, such representation is too coarse for crowd simulations with 3D articulated bodies. The controller that drives a virtual humanoid in a movie or a computer game has to be more complex than the behavior model that drives 2D dots. It is not enough to decide global position and orientation of the entity; features like type of the motion, its dynamics and transition, or biomechanical constraints have to be taken into account. A simple re-application of evacuation models to 3D visualizations leads to awkward, unrealistic looking animations. Humans can get easily enchanted by seeing artificial objects performing behaviors that are not expected from them (such as geometrical primitives fleeing in 2D labyrinth), but are very critical at evaluating of (what are expected to be) the other people. Motions that look reasonable for 2D dots can look very artificial when applied to virtual humans. Even a relatively straightforward transition from segmented skeletons to fully skinned bodies in many cases reveals disturbing imperfections in the motion. For applications where the visual quality is the most important (as in movies or games), the behavior has to be constrained by availability of motions and transitions among the motions (for example, when using physically based simulation [HB94] or motion graphs [SGC04]).

Application focus: The consequence of the crowd models being domain specific is that in the majority of cases the applications are focusing either on the realism of behavioral aspects, or on the quality of the visualization. The most representative examples of the former category are evacuation simulations, which are usually validated on a large scale sta-

tistical parameters such as the number of the people passing through a particular exit in a defined time interval. Behaviors of individuals are not detailed and not defined beyond the narrow scope of the simulation; for example, before or after the incident people are either static or have random Brownian motion. The examples of the latter category are crowds in movies and games, where the goal of the behavior model is to alleviate designers from the tedious tasks of orchestration of animation for large number of entities or to respond to the actions of the user. The repertoire of behaviors is larger; for example, as the most common use of virtual crowds are battles scenes, virtual armies have to be able to navigate around the environment, to attack using different weapons and to defend themselves against various enemies. The most challenging area for crowd simulation are training simulations as there is a need for both behavior realism and persuasive visualization. Present crowd management training systems have been focusing on training strategical skills therefore giving more emphasis on behavioral simulation with visualization being only schematic. Tactical on-site training of crowd management with the trainee immersed in the virtual world is not yet explored.

Crowd models: It is difficult to transfer current knowledge about real crowds from social sciences into crowd simulations. Most of the sociology work on crowds is about macroscopic behavior, not directly dealing with actions of a particular person in particular situation in particular time instance. Methodological observations about microscopic behaviors are sparse: sociological models based on collected real data have a limited scope. The quality of the crowd behavior model is prominent in safety science applications; however, despite calls for including more knowledge about psychology into evacuation models [Sim95], most of the current applications still model behavior of the crowd based more on physical than on psychological principles. Demands on crowd models are different for entertainment industry applications. For production purposes it is preferable to be able to control the crowd instead of just observing the results of the model. Emergent behavior has sense as far as it alleviates designers from tedious tasks. The crowd can be controlled "top-down" where the group behavior is imposed by design, or "bottom-up", where the collective behavior emerges from the behavior of individuals. Group based approaches have the advantage of easier handling when group membership does not need to change, however they bring the disadvantage of overhead when group membership changes often.

Trends: Virtual crowds are a relatively new topic, with majority of the research and commercial applications done in the past few years, especially concerning real-time crowds. The most visible trend is the increase of the number of simulated entities; new techniques together with rapidly evolving hardware allows to handle bigger crowds. Another recently appearing trend is about going beyond simple quantitative improvements towards increasing of complexity of entities at all levels - whether it is visualization, anima-

tion, or behaviors. Both quantitative and qualitative improvements require novel methods, as in the most cases it is not straightforward to apply the method that work for small number of entities to a large crowd. Similarly to other areas in computer graphics and virtual reality simulations, the major driving force of the innovation starts to be the entertainment industry resulting from the large investments due to increasing revenues from entertainment applications. For example, many movies with virtual crowds were blockbusters with revenues in order of hundreds of millions of the dollars and more [Sta02, Mat03, The03, Shr04] allowing to finance large internal research and development (R&D) teams. Even the military, which used to be one of the largest traditional sponsors of the simulation research, starts to use in some cases commercial entertainment technologies for its training instead of costly own R&D [Mac01, ZHM*03].

5. Future challenges and conclusions

We see several possible directions for future research in the area of interactive crowd simulations:

Heterogeneity: In current crowd simulation systems, the whole crowd is constituted by the same type of agents. Even while creating the individuality of the agents by varying parameters, the principle of the behavior computation is the same for every entity. It is possible to create a heterogeneous crowd simulation, where different agents can have completely different behavior computation engines. Such architecture could, for example, lead to an increase of the behavioral variety, while keeping individuals simpler compared to a homogeneous simulation with the same variety.

Scalability: In order to increase the number of simulated entities, the crowd simulation should be scalable [SGC04]. This can be achieved, for example, by using behavior and animation level-of-details [ACF01, AW04], where there are a different computational demands for agents, depending on their relative position to the observer. The behavior model should then allow to work with different fidelity, for example, by using iterative algorithms, or also heterogeneous crowds could be employed.

Variety: The variety of the virtual crowd can be enhanced by adapting methods, capable of producing higher levels of the variety, for the crowd simulations. The natural candidates are methods, which deal with variety sources in the real people, such as parametric generation of bodies [Seo03] or faces [BV99].

Parallelization: The computation of the crowd simulation can be speeded up by using parallelization [QMHZ03]. However, the parallelization of the agents becomes practical only for the hardware that supports a parallel execution of more threads than there are potentially parallelizable application components. For example, recently US military experimented with a combat simulation running on

128 node Linux cluster handling 100.000 entities (which means that each sequential node took care of on average 780 entities) [The04b].

The rapid adoption of the crowd simulation in movies and other non real-time productions in recent years shows that there is a great demand for virtual crowds. It is not so difficult to imagine why - humans are social creatures and real world reflects this fact, most of the people are surrounded by other people. It is therefore expected to see crowds in the works of both fact and fiction.

A similar reasoning holds also for interactive virtual environments such as computer games, training systems or educational applications - we expect to see them populated. However, while in movies it can be still possible, even if not practical, to use crowd of real extras, interactive applications have to rely fully on the virtual crowds. As already current generation personal computers are capable of handling thousands of real-time virtual characters, we believe that in coming years there will be more and more interactive virtual crowds.

We can expect to see a convergence between non real-time and real-time domains, in a manner similar to other areas in computer graphics. The convergence will be fueled both by increases in the power of both general purpose and graphics processors and by the development of novel methods and algorithms. In non real-time applications, the real-time methods can be used to improve the productivity for creating crowd scenes because of shorter production cycles and immediacy of the changes allowing new ways of authoring. On the other hand, in real-time applications, there will be improvements in quality of both rendering and behaviors moving towards the results possible before only by lengthy computations in non-real time productions.

References

- [A b98] A bug's life, 1998. movie homepage, <http://www.pixar.com/featurefilms/abl>. 7
- [ABT00] AUBEL A., BOULIC R., THALMANN D.: Real-time display of virtual humans: Levels of detail and impostors. *IEEE Transactions on Circuits and Systems for Video Technology* 10, 2 (2000), 207–217. 2, 6
- [ACF01] ARIKAN O., CHENNEY S., FORSYTH D. A.: Efficient multi-agent path planning. In *Computer Animation and Simulation '01* (2001), Magnenat-Thalmann N., Thalmann D., (Eds.), SpringerComputerScience, Springer-Verlag Wien New York, pp. 151–162. Proc. of the Eurographics Workshop in Manchester, UK, September 2–3, 2001. 10
- [AGK85] AMKRAUT S., GIRARD M., KARL G.: Motion studies for a work in progress entitled "Eurythmy". *SIGGRAPH Video Review*, 21 (1985). (second item, time code 3:58 to 7:35). 4

- [ALA*01] ASHIDA K., LEE S.-J., ALLBECK J. M., SUN H., BADLER N. I., METAXAS D.: Pedestrians: Creating agent behaviors through statistical analysis of observation data. In *Proc. Computer Animation '01* (2001), IEEE Press. 2
- [AMC03] ANDERSON M., MCDANIEL E., CHENNEY S.: Constrained animation of flocks. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'03)* (2003), pp. 286–297. 2, 8
- [Ant98] AntZ, 1998. movie homepage, <http://www.pdi.com/feature/antz.htm>. 2, 7
- [AW04] AHN J., WOHN K.: Motion level-of-detail: A simplification method on crowd scene. In *Proc. Computer Animation and Social Agents '04* (2004), pp. 129–137. 10
- [BCN97] BOUVIER E., COHEN E., NAJMAN L.: From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. *Journal of Electrical Imaging* 6, 1 (January 1997), 94–107. 2, 5
- [BDT99] BONABEAU E., DORIGO M., THERAULAZ G.: *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999. 4
- [BDT00] BONABEAU E., DORIGO M., THERAULAZ G.: Inspiration for optimization from social insect behaviour. *Nature* 406 (2000), 39–42. 4
- [BG96] BOUVIER E., GUILLOTEAU P.: Crowd simulation in immersive space management. In *Proc. Eurographics Workshop on Virtual Environments and Scientific Visualization '96* (1996), Springer-Verlag, pp. 104–110. 1, 5
- [BH97] BROGAN D., HODGINS J.: Group behaviors for systems with significant dynamics. *Autonomous Robots* 4 (1997), 137–153. 2, 5
- [BLA02] BAYAZIT O. B., LIEN J.-M., AMATO N. M.: Better group behaviors in complex environments using global roadmaps. In *Proc. Artificial Life '02* (2002). 2, 6
- [BMdOB03] BRAUN A., MUSSE S. R., DE OLIVEIRA L. P. L., BODMANN B. E. J.: Modeling individual behaviors in crowd simulation. In *Proc. Computer Animation and Social Agents '03* (2003). 1
- [Bot95] BOTTACI L.: A direct manipulation interface for a user enhanceable crowd simulator. *Journal Of Intelligent Systems* 5, 2-4 (1995), 249–272. 1
- [Bro02] BROCKINGTON M.: Level-of-detail AI for a large role-playing game. In *AI Game Programming Wisdom* (2002), Rabin S., (Ed.), Charles River Media. 8
- [BV99] BLANZ B., VETTER T.: A morphable model for the synthesis of 3d faces. In *Proc. Siggraph '99* (1999), pp. 187–194. 10
- [Cha04] Character Studio, 2004. <http://www.discreet.com/products/cs>. 2
- [Che04] CHENNEY S.: Flow tiles. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'04)* (2004), pp. 233–245. 2, 8
- [CKFL05] COUZIN I. D., KRAUSE J., FRANKS N. R., LEVIN S. A.: Effective leadership and decision-making in animal groups on the move. *Nature* 433 (2005), 513–516. 4
- [CYC99] CHOW T. W. S., YAM J. Y.-F., CHO S.-Y.: Fast training algorithm for feedforward neural networks: application to crowd estimation at underground stations. *Artificial Intelligence in Engineering* 13 (1999), 301–307. 2
- [DeL00] DELOURA M. (Ed.): *Game Programming Gems*. Charles River Media, 2000. 6
- [DHOO05] DOBBYN S., HAMILL J., O'CONNOR K., O'SULLIVAN C.: Geopostors: A real-time geometry/impostor crowd rendering system. In *Proc. ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games* (2005). 7
- [Dor92] DORIGO M.: *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992. 4
- [Doy03] DOYLE A.: The two towers. *Computer Graphics World* (February 2003). 7
- [Exo04] Exodus, 2004. the evacuation model for the safety industry, homepage, <http://fseg.gre.ac.uk/exodus>. 2
- [FBT99] FARENC N., BOULIC R., THALMANN D.: An informed environment dedicated to the simulation of virtual humans in urban context. In *Proc. Eurographics'99* (1999), Blackwell, pp. 309–318. 2, 6
- [FHV02] FARKAS I., HELBING D., VICSEK T.: Mexican waves in an excitable medium. *Nature* 419 (2002), 131–132. 1
- [FRMS*99] FARENC N., RAUPP MUSSE S., SCHWEISS E., KALLMANN M., AUNE O., BOULIC R., THALMANN D.: A paradigm for controlling virtual humans in urban environment simulations. *Applied Artificial Intelligence Journal - Special Issue on Intelligent Virtual Environments* 14, 1 (1999), 69–91. 2
- [GA90] GIRARD M., AMKRAUT S.: Eurhythmy: Concept and process. *The Journal of Visualization and Computer Animation* 1, 1 (1990), 15–17. Presented at The Electronic Theater at SIGGRAPH '85. 4
- [Gil95] GILBERT N.: Simulation: an emergent perspective. In *New technologies in the social sciences* (Bournemouth, UK, 27-29th October 1995). 4
- [GKM*01] GOLDENSTEIN S., KARAVELAS M., METAXAS D., GUIBAS L., AARON E., GOSWAMI A.: Scalable nonlinear dynamical systems for agent steering and crowd simulation. *Computers & graphics* 25, 6 (2001), 983–998. 2

B. Ulicny, P. de Heras Ciechowski, S. R. Musse & D. Thalmann / State-of-the-Art: Real-time crowd simulations

- [GKMT01] GOTO T., KSHIRSAGAR S., MAGNENAT-THALMANN N.: Automatic face cloning and animation. *IEEE Signal Processing Magazine* 18, 3 (2001). 2
- [Gla00] Gladiator, 2000. movie homepage, <http://www.dreamworks.com>. 7
- [Har00] HAREESH P.: Evacuation simulation: Visualisation using virtual humans in a distributed multi-user immersive VR system. In *Proc. VSMM '00* (2000). 3
- [HB94] HODGINS J., BROGAN D.: Robot herds: Group behaviors for systems with significant dynamics. In *Proc. Artificial Life IV* (1994), pp. 319–324. 1, 2, 5, 9
- [HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature* 407 (2000), 487–490. 1
- [HIK96] HOSOI M., ISHIJIMA S., KOJIMA A.: Dynamical model of a pedestrian in a crowd. In *Proc. IEEE International Workshop on Robot and Human Communication '96* (1996). 1
- [HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Phys. Rev. E* 51 (1995), 4282–4286. 1
- [HM99] HOLLAND O. E., MELHUISH C.: Stigmergy, self-organisation, and sorting in collective robotics. *Artificial Life* 5 (1999), 173–202. 4
- [JPvdS01] JAGER W., POPPING R., VAN DE SANDE H.: Clustering and fighting in two-party crowds: Simulating the approach-avoidance conflict. *Journal of Artificial Societies and Social Simulation* 4, 3 (2001). 1, 4
- [KBT03] KALLMANN M., BIERI H., THALMANN D.: Fully dynamic constrained delaunay triangulations. In *Geometric Modelling for Scientific Visualization* (2003), Brunnert G., Hamann B., Mueller H., Linsen L., (Eds.), Springer-Verlag, pp. 241–257. 2, 6
- [KMKWS00] KLÜPFEL H., MEYER-KÖNIG M., WAHLE J., SCHRECKENBERG M.: Microscopic simulation of evacuation processes on passenger ships. In *Theoretical and Practical Issues on Cellular Automata* (2000), Bandini S., Worsch T., (Eds.), Springer, London, pp. 63–71. 3
- [Koe02] KOEPEL D.: Massive attack. *Popular Science* (November 2002). 7
- [LCT01] LOSCOS C., CHRYSANTHOU Y., TECCHIA F.: Real-time shadows for animated crowds in virtual cities. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST'01)* (New York, Nov. 15–17 2001), Shaw C., Wang W., (Eds.), ACM Press, pp. 85–92. 2
- [LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum* 23, 3 (2004). (Proc. Eurographics '04). 2, 6
- [LeB95] LEBON G.: *Psychologie des Foules*. Alcan, Paris, 1895. 1
- [Leh02] LEHANE S.: Digital extras. *Film and Video Magazine* (July 2002). 7
- [LMA03] LOSCOS C., MARCHAL D., A.MEYER: Intuitive crowd behavior in dense urban environments using local laws. In *Proc. Theory and Practice of Computer Graphics (TPCG'03)* (2003). 2
- [Mac01] MACEDONIA M.: Games, simulation, and the military education dilemma. In *The Internet and the University: 2001 Forum* (2001), Devlin M., Larson R., Meyerson J., (Eds.), Educause. 9
- [Mar99] MARTINOLI A.: *Swarm Intelligence in Autonomous Collective Robotics: From Tools to the Analysis and Synthesis of Distributed Collective Strategies*. PhD thesis, EPFL, Lausanne, 1999. 4
- [Mas04] MASSIVE, 2004. Crowd animation software for visual effects, <http://www.massivesoftware.com>. 7
- [Mat97] MATARIC M.: Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence* (1997), 323–336. 4
- [Mat03] Matrix, 2003. movie homepage, <http://whatisthematrix.warnerbros.com>. 7, 9
- [Med02] Medieval: Total War, 2002. game homepage, <http://www.totalwar.com>. 8
- [MPT92] MCPHAIL C., POWERS W., TUCKER C.: Simulating individual and collective actions in temporary gatherings. *Social Science Computer Review* 10, 1 (Spring 1992), 1–28. 1, 3
- [MS01] MOLNAR P., STARKE J.: Control of distributed autonomous robotic systems using principles of pattern formation in nature and pedestrian behavior. *IEEE Trans. Syst. Man Cyb. B* 31, 3 (June 2001), 433–436. 1, 4
- [MT01] MUSSE S. R., THALMANN D.: A hierarchical model for real time simulation of virtual human crowds. *IEEE Transactions on Visualization and Computer Graphics* 7, 2 (April-June 2001), 152–164. 1, 5
- [Mus00] MUSSE S. R.: *Human Crowd Modelling with Various Levels of Behaviour Control*. PhD thesis, EPFL, Lausanne, 2000. 2, 5
- [MVCL98] MARANA A. N., VELASTIN S. A., COSTA L. F., LOTUFO R. A.: Automatic estimation of crowd density using texture. *Safety Science* 28, 3 (1998), 165–175. 2
- [NG03] NIEDERBERGER C., GROSS M.: Hierarchical and heterogenous reactive agents for real-time applications. *Computer Graphics Forum* 22, 3 (2003). (Proc. Eurographics'03). 2, 5

- [OCV*02] O'SULLIVAN C., CASSELL J., VILHJÁLMS-SON H., DINGLIANA J., DOBBYN S., MCNAMEE B., PETERS C., GIANG T.: Levels of detail for crowds and groups. *Computer Graphics Forum* 21, 4 (Nov. 2002), 733–741. 5
- [OGLF98] OWEN M., GALEA E. R., LAWRENCE P. J., FILIPPIDIS L.: The numerical simulation of aircraft evacuation and its application to aircraft design and certification. *The Aeronautical Journal* 102, 1016 (1998), 301–312. 3
- [OM93] OKAZAKI S., MATSUSHITA S.: A study of simulation model for pedestrian movement with evacuation and queuing. In *Proc. International Conference on Engineering for Crowd Safety '93* (1993). 3
- [Pow73] POWERS W. T.: *The Control of Perception*. Aldine, Chicago, 1973. 3
- [Pra03] Praetorians, 2003. game homepage, <http://praetorians.pyrostudios.com>. 8
- [PT01] PENN A., TURNER A.: Space syntax based agent simulation. In *Pedestrian and Evacuation Dynamics*, Schreckenberg M., Sharma S., (Eds.). Springer-Verlag, Berlin, 2001. 1
- [QMHZ03] QUINN M. J., METOYER R. A., HUNTER-ZAWORSKI K.: Parallel implementation of the social forces model. In *Proc. Pedestrian and Evacuation Dynamics'03* (2003), Galea E., (Ed.). 10
- [Rep03] Republic: the Revolution, 2003. game homepage, <http://www.elixir-studios.co.uk/nonflash/republic/republic.htm>. 8
- [Rey87] REYNOLDS C. W.: Flocks, herds, and schools: A distributed behavioral model. In *Proc. SIGGRAPH '87* (1987), pp. 25–34. 1, 2, 4
- [Rey99] REYNOLDS C. W.: Steering behaviors for autonomous characters. In *Proc. Game Developers Conference '99* (1999), pp. 763–782. 2, 5
- [Rey00] REYNOLDS C. W.: Interaction with groups of autonomous characters. In *Proc. Game Developers Conference '00* (2000), pp. 449–460. 5
- [Rob99] ROBBINS C.: Computer simulation of crowd behaviour and evacuation. *ECMI Newsletter*, 25 (March 1999). 3
- [Rom04] Rome: Total War, 2004. game homepage, <http://www.totalwar.com>. 8
- [Sch95] SCHWEINGRUBER D.: A computer simulation of a sociological experiment. *Social Science Computer Review* 13, 3 (1995), 351–359. 3
- [Sco03] SCOTT R.: Sparking life: Notes on the performance capture sessions for 'The Lord of the Rings: The Two Towers'. *ACM SIGGRAPH Computer Graphics* 37, 4 (2003), 17–21. 7
- [Seo03] SEO H.: *Parameterized Human Body Modeling*. PhD thesis, University of Geneva, 2003. 10
- [SGC04] SUNG M., GLEICHER M., CHENNEY S.: Scalable behaviors for crowd simulation. *Computer Graphics Forum* 23, 3 (2004). Proc. Eurographics '04. 6, 8, 9, 10
- [Sho00] Shogun: Total War, 2000. game homepage, <http://www.totalwar.com>. 8
- [Shr04] Shrek 2, 2004. movie homepage, <http://www.shrek2.com>. 7, 9
- [Sime95] SIME J.: Crowd psychology and engineering. *Safety Science* 21 (1995), 1–14. 9
- [Sim04] Simulex, 2004. evacuation modeling software, product information, <http://www.ies4d.com>. 1, 3
- [SKN98] SAIWAKI N., KOMATSU T., NISHIDA S.: Automatic generation of moving crowds in the virtual environments. In *Proc. AMCP '98* (1998). 1
- [SM99] SCHWEINGRUBER D., MCPHAIL C.: A method for systematically observing and recording collective action. *Sociological Methods & Research* 27, 4 (May 1999), 451–498. 2
- [SOHTG99] SCHELHORN T., O'SULLIVAN D., HAKLAY M., THURSTAIN-GOODWIN M.: Streets: an agent-based pedestrian model. In *Proc. Computers in Urban Planning and Urban Management* (1999). 1
- [Sta02] Star Wars, 2002. movie homepage, <http://www.starwars.com/>. 7, 9
- [STE] STEPS, Simulation of Transient Evacuation and Pedestrian movements. <http://www.fusion2.mottmac.com/html/06/software.cfm>. 2
- [Sti00] STILL G.: *Crowd Dynamics*. PhD thesis, Warwick University, 2000. 1, 3
- [SYCGMT02] SEO H., YAHIA-CHERIF L., GOTO T., MAGNENAT-THALMANN N.: Genesis : Generation of e-population based on statistical information. In *Proc. Computer Animation '02* (2002), IEEE Press. 2
- [TD00] THOMAS G., DONIKIAN S.: Modelling virtual cities dedicated to behavioural animation. In *Proc. Eurographics '00* (2000), vol. 19, pp. 71–80. 6
- [The94] The Lion King, 1994. movie homepage, <http://disney.go.com/disneyvideos/animatedfilms/lionking>. 7
- [The03] The Lord of the Rings, 2003. movie homepage, <http://www.lordoftherings.net>. 7, 9
- [The04a] The Lord of the Rings, The Battle for Middle Earth, 2004. game homepage, http://www.eagames.com/pccd/lotr_bfme. 8
- [The04b] The wars of the virtual worlds,, 2004. University of Southern California, <http://www.isi.edu/stories/96.html>. 10

B. Ulicny, P. de Heras Ciechowski, S. R. Musse & D. Thalmann / State-of-the-Art: Real-time crowd simulations

- [Tit97] Titanic, 1997. movie homepage, <http://www.titanicmovie.com>. 7
- [TLC02a] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Image-based crowd rendering. *IEEE Computer Graphics and Applications* 22, 2 (March-April 2002), 36–43. 2, 6
- [TLC02b] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Visualizing crowds in real-time. *Computer Graphics Forum* 21, 4 (November 2002), 753–765. 1, 2, 7
- [TM95a] THOMPSON P., MARCHANT E.: A computer-model for the evacuation of large building population. *Fire Safety Journal* 24, 2 (1995), 131–148. 1, 3
- [TM95b] THOMPSON P., MARCHANT E.: Testing and application of the computer model 'simulex'. *Fire Safety Journal* 24, 2 (1995), 149–166. 2
- [TP02] TURNER A., PENN. A.: Encoding natural movement as an agent-based system: An investigation into human pedestrian behaviour in the built environment. *Environment and Planning B: Planning and Design* 29 (2002), 473–490. 1
- [Tro04] Troy, 2004. movie homepage, <http://troymovie.warnerbros.com>. 7
- [TSM99] TUCKER C., SCHWEINGRUBER D., MCPHAIL C.: Simulating arcs and rings in temporary gatherings. *International Journal of Human-Computer Systems* 50 (1999), 581–588. 1, 4
- [TT92] TAKAHASHI T. S. H.: Behavior simulation by network model. *Memoirs of Kougakuin University*, 73 (1992), 213–220. 1
- [TT94] TU X., TERZOPOULOS D.: Artificial fishes: Physics, locomotion, perception, behavior. In *Proc. SIGGRAPH '94* (1994), pp. 43–50. 2, 4, 5
- [TWP03] TANG W., WAN T. R., PATEL S.: Real-time crowd movement on large scale terrains. In *Proc. Theory and Practice of Computer Graphics* (2003), IEEE. 6
- [UdHCT04] ULICNY B., DE HERAS CIECHOWSKI P., THALMANN D.: Crowdbrush: Interactive authoring of real-time crowd scenes. In *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA'03)* (2004), pp. 243–252. 2, 7, 8
- [UT01] ULICNY B., THALMANN D.: Crowd simulation for interactive virtual environments and vr training systems. In *Proc. Eurographics Workshop on Animation and Simulation* (2001), Springer-Verlag, pp. 163–170. 5
- [UT02] ULICNY B., THALMANN D.: Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum* 21, 4 (Dec. 2002), 767–775. 1, 2, 5
- [VSH*00] VAUGHAN R. T., SUMPTER N., HENDERSON J., FROST A., CAMERON S.: Experiments in automatic flock control. *Robotics and Autonomous Systems* 31 (2000), 109–177. 4
- [VSMA98] VARNER D., SCOTT D., MICHELETTI J., AICELLA G.: UMSC Small Unit Leader Non-Lethal Trainer. In *Proc. ITEC'98* (1998). 1, 3
- [Wil95] WILLIAMS J.: *A Simulation Environment to Support Training for Large Scale Command and Control Tasks*. PhD thesis, University of Leeds, 1995. 1, 3
- [Woo99] WOODCOCK S.: Game AI: The state of the industry. *Game Developer Magazine* (August 1999). 6
- [WS02] WAND M., STRASSER W.: Multi-resolution rendering of complex animated scenes. *Computer Graphics Forum* 21, 3 (2002). (Proc. Eurographics'02). 2, 7
- [ZHM*03] ZYDA M., HILES J., MAYBERRY A., WARDYNSKI C., CAPPS M., OSBORN B., SHILLING R., ROBASZEWSKI M., DAVIS M.: Entertainment R&D for defense. *IEEE Computer Graphics and Applications* (January / February 2003), 2–10. 9

Computerized Models for Virtual Humans and Crowds

Daniel Thalmann,

EPFL VRlab, Switzerland

Abstract

Interactive systems, games, VR and multimedia systems require more and more flexible Virtual Humans with individualities. There are mainly two approaches:

1) Recording the motion using motion capture systems, then to try to alterate such a motion to create this individuality. This process is tedious and there is no reliable method at this stage.

2) Creating computational models which are controlled by a few parameters. One of the major problem is to find such models and to compose them to create complex motion. Such models can be created for walking, grasping, but also for groups and crowds.

1. Introduction

Virtual humans simulations are becoming each time more popular. Nowadays many systems are available to animate virtual humans. Such systems encompass several different domains, as: autonomous agents in virtual environments, human factors analysis, training, education, virtual prototyping, simulation-based design, and entertainment. In the context of Virtual Humans, a Motion Control Method (MCM) specifies how the Virtual Human is animated and may be characterized according to the type of information it privileged in animating this Virtual Human. For example, in a keyframe system for an articulated body, the privileged information to be The problem is basically to be able to generate variety among a finite set of motion requests and then to apply it to either an individual or a member of a crowd. A single autonomous agent and a member of the crowd present the same kind of 'individuality'. The only difference is at the level of the modules that control the main set of actions. With this formulation, one can also see that the personality of an agent (i.e. the set of noisy actions) can be preserved whenever it is in a crowd, alone. Figure 1 shows a group of Virtual Humans in a room and Figure 2 Virtual Humans in city.

The problem is basically to be able to generate variety among a finite set of motion requests and then to apply it to either an individual or a member of a crowd. A single autonomous agent and a member of the crowd present the same kind of 'individuality'. The only difference is at the level of the modules that control the main set of actions. With this formulation, one can also see that the personality of an agent (i.e. the set of noisy actions) can be preserved whenever it is in a crowd, alone.



Figure 1. A group of Virtual Humans



Figure 2. Virtual Humans in a city

To create this flexible Virtual Humans with individualities, there are mainly two approaches:

- Recording the motion using motion capture systems (magnetic or optical), then to try to alterate such a motion to create this individuality. This process is tedious and there is no reliable method at this stage.

- Creating computational models which are controlled by a few parameters. One of the major problem is to find such models and to compose them to create complex motion. Such models can be created for walking, running, grasping, but also for interaction, groups, and crowds.

2. Motion Capture and Retargeting

The first approach consists in recording the motion (Fig. 3) using motion capture systems (magnetic or optical), then to try to alterate such a motion to create this individuality. This process is tedious and there is no reliable method at this stage. Even if it is fairly easy to correct one posture by modifying its angular parameters (with an Inverse Kinematics engine, for instance), it becomes a difficult task to perform this over the whole motion sequence while ensuring that some spatial constraints are respected over a certain time range, and that no discontinuities arise. When one tries to adapt a captured motion to a different character, the constraints are usually violated, leading to problems such as the feet going into the ground or a hand unable to reach an object that the character should grab. The problem of adaptation and adjustment is usually referred to as the Motion Retargeting Problem.



Figure 3. Motion capture

Witkin and Popovic [WP95] proposed a technique for editing motions, by modifying the motion curves through warping functions and produced some of the first interesting results. In a more recent paper [PW99], they have extended their method to handle physical elements, such as mass and gravity, and also described how to use characters with different numbers of degrees of freedom. Their algorithm is based on the reduction of the character to an abstract character which is much simpler and only contains the degrees of freedom that are useful for a particular animation. The edition and modification are then computed on this simplified character and mapped again onto the end user skeleton. Bruderlin and Williams [BW95] have described some basic facilities to change the animation, by modifying the motion parameter curves. The user can define a particular posture at time t , and the system is then responsible for smoothly blending the motion around t . They also introduced the notion of motion displacement map, which is an offset added to each motion curve. The Motion Retargeting Problem term

was brought up by Michael Gleicher [G98]. He designed a space-time constraints solver, into which every constraint is added, leading to a big optimisation problem. He mainly focused on optimising his solver, to avoid enormous computation time, and achieved very good results. Bindiganavale and Badler [BB98] also addressed the motion retargeting problem, introducing new elements: using the zero-crossing of the second derivative to detect significant changes in the motion, visual attention tracking (and the way to handle the gaze direction) and applying Inverse Kinematics to enforce constraints, by defining six sub-chains (the two arms and legs, the spine and the neck). Finally, Lee and Shin [JS99] used in their system a coarse-to-fine hierarchy of B-splines to interpolate the solutions computed by their Inverse Kinematics solver. They also reduced the complexity of the IK problem by analytically handling the degrees of freedom for the four human limbs

Lim and Thalmann [LT00] have addressed an issue of solving customers' problems when applying evolutionary computation. Rather than the seemingly more impressive approach of wow-it-all-evolved- from-nothing, tinkering with existing models can be a more pragmatic approach in doing so. Using interactive evolution, they experimentally validate this point on setting parameters of a human walk model for computer animation while previous applications are mostly about evolving motion controllers of far simpler creatures from scratch.

Given a captured motion associated to its Performer Skeleton, we decompose the problem of retargeting the motion to the End User Skeleton into two steps

- First, computing the Intermediate Skeleton matrices by orienting the Intermediate Skeleton bones to reflect the Performer Skeleton posture (Motion Converter).
- Second, setting the End User Skeleton matrices to the local values of the corresponding Intermediate Skeleton matrices.

The first task is to convert the motion from one hierarchy to a completely different one. We introduce the Intermediate Skeleton model to solve this, implying three more subtasks: manually set at the beginning the correspondences between the two hierarchies, create the Intermediate Skeleton and convert the movement. We are then able to correct the resulting motion and make it enforce Cartesian constraints by using Inverse Kinematics. When considering motion conversion between different skeletons, one quickly notices that it is very difficult to directly map the Performer Skeleton values onto the End User Skeleton, due to their different proportions, hierarchies and axis systems. This raised the idea of having an Intermediate Skeleton: depending on the Performer Skeleton posture, we reorient its bones to match the same directions. We have then an easy mapping of the Intermediate Skeleton values onto the End User Skeleton. The first step is to compute the Intermediate Skeleton (Anatomic Binding module). During the animation, motion conversion takes two passes, through the Motion Converter and the Motion Composer (which has a graphical user interface).

3. Creating Computational Models

The second approach consists in creating computational models which are controlled by a few parameters. One of the major problem is to find such models and to compose them to create complex motion. Such models can be created for example for walking.

Walking has global and specific characteristics. From a global point of view, every human-walking has comparable joint angle variations. However, at a close-up, we notice that individual walk characteristics are overlaid to the global walking gait.

We use the walking engine described in [BMT90] which has been extended in the context of a european project on virtual human modeling [BCH95]. Our contribution consists in integrating the walking engine as a specialized action in the animation framework. Walking is defined as a motion where the center of gravity alternatively balances from the right to the left side. It has the following characteristics

- at any time, at least one foot is in contact with the floor, the ‘single support’ duration (ds).
- there exists a short instant during the walk cycle, where both feet are in contact with the floor, the ‘double support’ duration (dds).
- it is a periodic motion which has to be normalized in order to adapt to different anatomies.

The joint angle variations are synthesized by a set of periodic motions which we briefly mention here:

- sinus functions with varying amplitudes and frequencies for the humanoid’s global translations (vertical, lateral and frontal) and the humanoid’s pelvic motions (forward/backward, left/right and torsion)
- periodic functions based on control points and interpolating hermite splines. They are applied to the hip flexion, knee flexion, ankle flexion, chest torsion, shoulder flexion and elbow flexion.

The parameters of the joint angle functions can be modified in a configuration file in order to generate personalized walking gaits, ranging from tired to energetic, sad to happy, smart to silly. The algorithm also integrates an automatic speed tuning mechanism which prevents sliding on the supporting surface. Many high level parameters can be adjusted dynamically, such as linear and angular velocity, foot step locations and the global walk trajectory. The walk engine has been augmented by a specialized action interface and its full capacity is therefore available within the animation framework. The specialized action directly exports most common high level parameter adjustment functions. For fine-tuning, it is still possible to explicitly access the underlying motion generator. The walk animation engine has been developed in the early nineties. However it suffered from not being easily combined with other motions, for example a walking human giving a phone call with a wireless phone was hardly possible. Now, that the walking engine is integrated as a specialized action, a walking and phoning human is easily done, simply by performing the walk together with a ‘phone’-keyframe for

example. In Figure 4, we show an example of parameterized.



Figure 4. Individualized walking

More recently, Glardon et al. [GBT04] have proposed a novel approach to generate new generic human walking patterns using motion-captured data, leading to a real-time engine intended for virtual humans animation. The method applies the PCA (Principal Component Analysis) technique on motion data acquired by an optical system to yield a reduced dimension space where not only interpolation, but also extrapolation are possible, controlled by quantitative speed parameter values. Moreover, with proper normalization and time warping methods, the generic presented engine can produce walking motions with continuously varying human height and speed with real-time reactivity. Figure 5 shows examples.



Figure 5. Examples of PCA-based walking humans

4. Crowds and Groups

Animating crowds [MT01] is challenging both in character animation and a virtual city modeling. Though different textures and colors may be used, the similarity of the virtual people would be soon detected by even non-experts, say, “everybody walks the same in this virtual city!”. It is, hence, useful to have a fast and intuitive way of generating motions with different personalities depending on gender, age, emotions, etc., from an example motion, say, a genuine walking motion. The problem is basically to be able to generate variety among a finite set of motion requests and then to apply it to either an individual or a member of a crowd. It also needs very good tools to tune the motion [EBM00].

The proposed solution addresses two main issues: i) crowd structure and ii) crowd behavior. Considering crowd structure, our approach deals with a hierarchy composed of crowd, groups and agents, where the groups are the most complex structure containing the information to be distributed among the individuals. Concerning crowd behavior, our virtual agents are endowed with different levels of autonomy. They can either act according to an innate and scripted crowd behavior (programmed behavior), react as a function of triggered events (reactive or autonomous behavior) or be guided by an interactive process during simulation (guided behavior). We introduced the term <guided crowds> to define the groups of virtual agents that can be externally controlled in real time [MBC98]. Figure 6 shows a crowd guided by a leader.



Figure 6. Crowd guided by a leader

In our case, the intelligence, memory, intention and perception are focalized in the group structure. Also, each group can obtain one leader. This leader can be chosen randomly by the crowd system, defined by the user or can emerge from the sociological rules. Concerning the crowd control features, The crowd aims at providing autonomous, guided and programmed crowds. Varying degrees of autonomy can be applied depending on the complexity of the problem. Externally controlled groups, <guided groups>, no longer obey their scripted behavior, but act according to the external specification. At a lower level, the individuals have a repertoire of basic behaviors that we call innate behaviors. An innate behavior is defined as an “inborn” way to behave. Examples of individual innate behaviors are goal seeking behavior, the ability to follow scripted or guided events/reactions, the way trajectories are processed and collision avoided. While the innate behaviors are included in the model, the specification of scripted behaviors is done by means of a script language. The groups of virtual agents whom we call <programmed groups> apply the scripted behaviors and do not need user intervention during simulation. Using the script language, the user can directly specify the crowd or group behaviors. In the first case, the system automatically distributes the crowd behaviors among the existing groups. Events and reactions have been used to represent behavioral rules. This reactive character of the simulation can be programmed in the script language (scripted control) or directly given by an external controller. We call the groups of virtual agents who apply the behavioral rules <autonomous groups>.

The train station simulation (Figure 7) includes many different actions and places, where several people are

present and doing different things. Possible actions include “buying a ticket”, “going to shop”, “meeting someone”, “waiting for someone”, “making a telephone call”, “checking the timetable”, etc. This simulation uses external control to guide some crowd behaviors in real time.



Figure 7. Train station simulation.

More recently, we developed a new crowd engine allowing to display up to 50'000 thousands virtual humans in real-time. This makes Computational models even more important. Figure 8 shows two examples.



Figure 8. Examples of large crowds.

5. Perception

Let's now consider the simulation of a referee during a tennis match. He has to decide if the ball is out or in. One solution is to calculate the intersection between the impact point of the ball and the court lines. Such an analytical

calculation will lead to the decision that the ball is out for 0.01 millimeters. Ridiculous, nobody in reality could take such an objective decision, this is not believable. The decision should be based on the evaluation of the visual aspect of the scene as perceived by the referee.

In a more general context, it is tempting to simulate perception by directly retrieving the location of each perceived object straight from the environment. This is of course the fastest solution (and has been extensively used in video-games until the mid-nineties) but no one can ever pretend that it is realistic at all (although it can be useful, as we will see later on). Consequently, various ways of simulating visual perception have been proposed, depending on whether geometric or semantic information (or both) are considered. Renault et al. introduced first the concept of synthetic vision [RMT90] then extended by Noser et al. [NRT95]. Tu and Terzopoulos [TT94] implemented a realistic simulation of artificial fishes. Other authors [KL99] [BG95] [PO02] also provided synthetic vision approaches. In the next section, we are going to compare now rendering-based vision, geometric vision and database access.

5.1 Synthetic Vision

Rendering-based vision from Noser and Renault et al. [NRT95] is achieved by rendering of-screen the scene as viewed by the agent. During the process, each individual object in the scene is assigned a different colour, so that once the 2D image has been computed, objects can still be identified: it is then easy to know which object is in sight by maintaining a table of correspondences between colours and objects' IDs. Furthermore, highly detailed depth information is retrieved from the view z-buffer, giving a precise location for each object. An other application of synthetic vision is real-time collision avoidance for multiple agents: in this case, each agent is perceiving the others, and dynamically creates local goals so that it avoids others while trying to reach its original global goal.

Rendering-based vision is the most elegant method, because it is the more realistic simulation of vision and addresses correctly vision issues such as occlusion for instance. However, rendering the whole scene for each agent is very costly and for real-time applications, one tend to favour geometric vision.

One problem is how to decide that an object is in the field of view of the Virtual Human and that he/she can identify it. We can imagine for example that the Virtual Human's wife is in front of the VH but hidden by a wardrobe and on the computed 2D image contains only one pixel for the wife, can he recognize his wife based on such a detail ?

Bordeux et al. [BBT99] has proposed a perception pipeline architecture into which filters can be combined to extract the required information. The perception filter represents the basic entity of the perception mechanism. Such a filter receives a perceptible entity from the scene as input, extracts specific information about it, and finally decides to let it pass through or not.

The criteria used in the decision process depends on the perception requirements. For virtual objects, they

usually involve considerations about the distance and the relative direction of the object, but can also be based on shape, size, colour, or generic semantic aspects, and more generally on whatever the agent might need to distinguish objects. Filters are built with an object oriented approach: the very basic filter for virtual objects only considers the distance to the object, and its descendants refine further the selection.

Actually, the structure transmitted to a filter contains, along with the object to perceive, a reference to the agent itself and previously computed data about the object. The filter can extend the structure with the results of its own computation, for example the relative position and speed of the object, a probable time to impact or the angular extension of the object from the agent's point of view. Since a perception filter does not store data concerning the objects that passed through it, it is fully reentrant and can be used by several agents at the same time. This allows the creation of a common pool of filters at the application, each agent then referencing the filters it needs, thus avoiding useless duplication.

However, the major problem with Geometric vision is to find the proper formulas when intersecting volumes (for instance, intersecting the view frustum of the agent with a volume in the scene). One can use bounding boxes to reduce the computation time, but it will always be less accurate than Synthetic vision. Nevertheless, it can be sufficient for many applications and, as opposed to rendering-based vision, the computation time can be adjusted precisely by refining the bounding volumes of objects.

Database access makes maximum use of the scene data available in the application, which can be distributed in several modules. For instance, the objects position, dimensions and shape are maintained by the rendering engine whereas semantic data about objects can be maintained by a completely separate part of the application. Due to scalability constraints as well as plausibility considerations, the agents generally restrain their perception to a local area around them instead of the whole scene. This method is generally chosen when the number of agents is high. In Musse's [MT01] crowd simulation, human agents directly know the position of their neighbours and compute coherent collision avoidance trajectory. As said before, the main problem with the method is the lack of realism, which can only be alleviated by using one of the other methods.

These various approaches to visual perception have their advantages and disadvantages dependent essentially of the complexity and the context of the scenes. But, finally no approach can solve common problematics as the following one: What makes a little girl to be lost in a crowd ? The child will be lost if she just does not know where is her family. Now imagine a virtual crowd where each individual is indexed. It will be extremely easy to find where is the girl (index 345) and the parents (index 748). At this stage, we could just activate a function making the girl walking towards his parents. This is completely unrealistic from a behavioural point of view.

5.2 Memory

Noser et al. [NRT95] made a few years ago a character trying to find the exit from a maze. To simulate the memory process, they used an octree structure to store the information see by the character. The results were that the second time, it was straightforward for the character to find the exit. Again, this is not so convincing as never somebody could remember all the paths inside a maze. This kind of memory can then easily be linked to the synthetic vision: the 2D rendering and the corresponding z-buffer data are combined in order to determine whether the corresponding voxel of the scene is occupied by an object or not. By navigating through the environment, the agent will progressively construct a voxel-based representation of it. Of course, a rough implementation of this method would suffer from dramatic memory cost, because of the high volume required to store all voxels. Noser proposed to use octrees instead which successfully reduces the amount of data. Once enough information has been gathered through exploration, the agent is then able to locate things and find its way.

Peters and O’Sullivan [PO02] propose a system of memory based on what is referred to a “stage theory” by Atkinson and Shiffrin [AS68]. They propose a model where information is processed and stored in 3 stages: sensory memory, short-term memory, and long-term memory.

Although these approaches are quite interesting, they do not solve the following simple problematic. Imagine now a Virtual Human inside a room containing 100 different objects. Which objects can we consider as memorized by the Virtual Human ? Can we decide that when an object is seen by the actor, it should be stored in his memory. To answer this question, we have just to consider the popular family game consisting in showing 20 objects during 2 minutes to people and asking them to list the objects. Generally nobody is able to list the 20 objects. Now, how to model this inability to remember all objects ?

5.3 Integration of Virtual Sensors

The modelling of an AVA gaining its independence with regard to its virtual representation remains an important theme in research and is very close to autonomous robotics. It helps also to understand and model human behaviour.

The AVA collects information only through the virtual sensors described earlier (Figure 9). We assume that vision is the main canal of information between the AVA and its environment as indicated by the standard theory in neuroscience for multi-sensorial integration [E98].

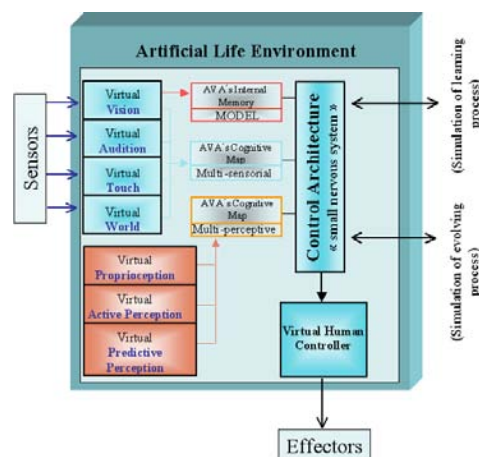


Figure 9: A schematic representation of our *ALifeE*. Virtual Vision discovers the VE, constructs the different types of Perception and updates the AVA’s Cognitive Map to obtain a multi-perceptive mapping. Then the Control Architecture uses both the “cognitive maps” and the “memory model” to interact with the learning, development, and control processes of the AVA (Virtual Human Controller).

The sensorial modalities update the AVA’s cognitive map to obtain a multi-sensorial mapping. For example, visual memory in the AVA’s internal memory is used for a global move from point A to point B. Should obstacles be present, it would have to be replaced for a local move by direct vision of the environment.

In our approach, we tried to integrate all the multi-sensorial information from the AVA’s virtual sensors. In fact, an AVA in a VE may have different degrees of autonomy and different sensorial canals depending on the environment. For instance, an AVA moving in a VE represented by a well-lit room will use primarily the sensorial information of vision. However if the light is turned off, the AVA will appeal to the acoustic or tactile sensorial information in the same way a human would move around in a dark room [SKA02].

From this observation we derive the hypotheses underlying our *ALifeE* framework approach. They are backed up by the latest research in neuroscience [P02], which describes a partial re-mapping at the behavioural level of the human including:

- *Assignment*: the prediction of the acoustic position of an object from its visual positions requires a transformation from its *eye-centred* (vision sensor) coordinates to its *head-centred* ones (auditory sensor). The comparison of these two types of results can be used to determine whether the acoustic and visual signals are directly connected to the same object.
- *Recoding*: the choice of the reference frame to integrate the sensorial signals.

6. Conclusion

In order to develop truly interactive multimedia systems with Virtual Humans, games, and interactive movies, we need a flexible way of animating these Virtual Humans. Altering motion obtained from a motion capture system is not the best solution. Only computational models can offer this flexibility unless powerful motion retargeting methods are developed, but in this case they will look similar to computational models.

Acknowledgments

The author would like to thank all people who have contributed to these projects especially Luc Emering, Soraia Musse, Ik Soo Lim, Pascal Glardon, Mireille Clavien, Toni Conde, and Pablo de Heras. Research has been partly funded by the Swiss National Foundation for Research and the Federal Office for Education and Science in the framework of the CROSSES project.

References

- [AS68] ATKINSON R., SHIFFRIN R., Human Memory : a Proposed System and its Control Processes, in: *K.Spence and J.Spence, the Psychology of Learning and Motivation: Advances in Research and Theory*, Vol.2, NY, Academic Press, 1968.
- [BB98] BINDIGANAVALA R., BADLER N.I. Motion abstraction and mapping with spatial constraints. In N. Magnenat-Thalmann and D. Thalmann, editors, *Modeling and Motion Capture Techniques for Virtual Environments, Lecture Notes in Artificial Intelligence*, pages 70–82. Springer, November 1998. held in Geneva, Switzerland, November 1998.
- [BBT99] BORDEUX C., BOULIC R., THALMANN D., An Efficient and Flexible Perception Pipeline for Autonomous Agents, *Proc. Eurographics '99*, Milano, Italy, pp.23-30.
- [BCH95] BOULIC R., CAPIN T., HUANG Z., MOCCOZET L., MOLET T., KALRA P., LINTERMANN B., MAGNENAT-THALMANN N., PANDZIC I., SAAR K., SCHMITT A., SHEN J., THALMANN D., The HUMANOID Environment for Interactive Animation of Multiple Deformable Human Characters, *Proc. Eurographics '95*, Maastricht, August 1995, pp.337-348.
- [BG95] BLUMBERG B.M., GALYEAN T.A., Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments, *Proc. SIGGRAPH 95*, 1995, pp.47-54.
- [BMT90] BOULIC R., MAGNENAT-THALMANN N., THALMANN D., A Global Human Walking Model with Real-time Kinematics Personification, *The Visual Computer*, Vol.6, No6, December 1990, pp.344-358.
- [BW95] BRUDERLIN A., WILLIAMS L. Motion signal processing. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 97–104. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06- 11 August 1995.
- [E98] Elfes G. Occupancy Grid: A Stochastic Spatial Representation for Active Robot Perception. In *6th Conference on Uncertainty in AI*, 1990
- [EBM00] L.EMERING, R.BOULIC, T.MOLET, D.THALMANN, Versatile Tuning of Humanoid Agent Activity, *Computer Graphics Forum*, 2000
- [G98] GLEICHER G. Retargeting motion to new characters. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 33–42. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.
- [GBT04] GLARDON P., R. BOULIC R., THALMANN D., PCA-based Walking Engine using Motion Capture Data, *Computer Graphics International, June 2004*, pp.292-298.
- [JS99] JEHEE L., SHIN S.Y.. A hierarchical approach *Proceedings of SIGGRAPH 99*, pages 39–48, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.
- [KL99] KUFFNER J., LATOMBE J.C., Fast Synthetic Vision, Memory, and Learning Models for Virtual Humans, *Proc. Computer Animation 1999*, IEEE CS Press, pp.118-127.
- [LT00] LIM I.S., THALMANN D., Solve Customers' Problems: Interactive Evolution for Tinkering with Computer Animation, *Proc. 2000 ACM Symposium on Applied Computing (SAC2000)*, pp. 404-407
- [MBC98] MUSSE, S.R., BABSKI, C., CAPIN, T. AND THALMANN, D. Crowd, Modelling in Collaborative Virtual Environments. *ACM VRST '98*, Taiwan
- [MT01] MUSSE S.R., THALMANN D., A Behavioral Model for Real-Time Simulation of Virtual Human Crowds, *IEEE Transactions on Visualization and Computer Graphics*, Vol.7, No2, 2001, pp.152-164.
- [NRT95] NOSER H., O. RENAULT O., D. THALMANN, N. MAGNENAT THALMANN, Navigation for Digital Actors based on Synthetic Vision, Memory and Learning, *Computers and Graphics*, Pergamon Press, Vol.19, No1, 1995, pp.7-19.
- [P02] POUGET A., A computational perspective on the neural basis of multi-sensory spatial representations. *Nature Reviews/Neuroscience*, 2002; 3:741-747.
- [PO02] PETERS C., O'SULLIVAN C., A Memory Model for Autonomous Virtual Humans, *Proc. Third Irish Eurographics Workshop on Computer Graphics*, Dublin, pp. 21-26.
- [PW99] POPOVIC Z., WITKIN A.. Physically based motion transformation. *Proceedings of SIGGRAPH 99*, pages 11–20, August 1999. ISBN 0-20148-560-5. Held in Los Angeles, California.
- [RMT90] RENAULT O., MAGNENAT-THALMANN N., THALMANN D., A Vision-based Approach to Behavioural Animation, *Journal of Visualization and Computer Animation*, Vol.1, No1, 1990, pp.18-21.
- [SKA02] Strösslin Th, Krebsler Ch, Arleo A, Gerstner W. Combining Multimodal Sensory Input for Spatial Learning. In *Proceedings of ICANN, 2002*; 87-92. *LNCS 2415*.Springer-Verlag.
- [TT94] TU X., TERZOPOULOS D., Artificial Fishes, Physics, Locomotion, Perception, Behaviour, *Proc. SIGGRAPH '94*, pp.43-50.
- [WP95] WITKIN A., POPOVIC Z.. Motion warping. *Proceedings of SIGGRAPH 95*, pages 105–108, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California

Real-Time Crowds: Architecture, Variety, and Motion Planning

Jonathan Maïm¹ Barbara Yersin¹ Daniel Thalmann¹

¹ Virtual Reality Laboratory, EPFL, Switzerland

Abstract

To simulate realistic virtual crowds in real time, three main requirements need satisfaction. First of all, quantity, i.e., the ability to simulate thousands of characters in real time. Secondly, quality, because each virtual human composing a crowd needs to look unique in its appearance and animation. Finally, realism in terms of crowd motion and navigation. In this tutorial, we explain how all objectives can be reached together. We first detail an efficient and versatile architecture able to simulate thousands of characters in real time. Then, state-of-the-art techniques to transform similar instances of a crowd into unique individuals are introduced. Finally, a hybrid motion planning approach, able to manage navigation and obstacle avoidance in real time, is presented. Overall, we show that it is possible to combine these three aspects to simulate large, realistic, and visually appealing crowds in real time.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism - Animation

1. Introduction

In these tutorial notes, we focus on technical aspects for creating an architecture sustaining real-time crowd simulation composed of several thousand of varied individuals planning for their path and avoiding collision. We begin the tutorial with a description of the different virtual human representations commonly used in crowd simulation in Section 2 and how each of them can cast shadows. Then, in Section 3, we introduce our crowd architecture, and the pipeline developed to process crowds in real time. In Section 4, we detail several techniques that can be efficiently used to vary the appearance of similar characters, instantiated from a same human template. Finally, we also present in Section 5 a hybrid and scalable motion planning architecture able to manage thousands of characters in complex environments in real time. Our conclusion is presented in Section 6.

2. Virtual Human Representations

In an ideal world, graphic cards would be able, at each frame, to render an infinite number of triangles with an arbitrary complex shading on them. To visualize crowds of virtual humans, we would simply use thousands of very detailed

meshes, e.g., capable of hand and facial animation. Unfortunately, in spite of the recent programmable graphics hardware advances, we are still compelled to stick to a limited triangle budget per frame. This budget is spent wisely to be able to display dense crowds without too much perceptible degradations. The concept of levels of detail (LOD), extensively treated in the literature (see Luebke *et al.* [LWC*02]) is exploited to meet our real-time constraints. For a crowd of virtual humans specifically, and depending on the location of the camera, a character is rendered with a particular representation, resulting from the compromise of rendering cost and quality. In this Section, we first introduce the data structure we use to create and simulate virtual humans: the human template. Then, we describe the three levels of detail a human template uses: the deformable mesh, the rigid mesh, and finally the impostor.

2.1. Human Template

A type of human such as a woman, man, or child is described as a human template, which consists of :

- A skeleton, composed of joints, representing articulations,

- A set of meshes, all representing the same virtual human, but with a decreasing number of triangles,
- Several appearance sets, used to vary its appearance,
- A set of animation sequences which it can play.

Each rendered virtual human is derived from a human template, *i.e.*, it is an instance of a human template. In order for all the instances of a same human template to look different, we use several appearance sets, that allow to vary the texture applied to the instances, as well as modulate the colors of the texture (see Section 4).

2.2. Deformable Mesh

A deformable mesh is a representation of a human template composed of triangles. It is enveloping a skeleton of 78 joints, used for animation: when the skeleton moves, the vertices of the mesh follow smoothly its joint movements, similarly to our skin. We call such an animation a skeletal animation. Each vertex of the mesh is influenced by one or a few joints. Thus, at every keyframe of an animation sequence, a vertex is deformed by the weighted transformation of the joints influencing it. The corresponding equation is:

$$v(t) = \sum_{i=1}^n \chi_i^t \chi_i^{-ref} v^{ref} \quad (1)$$

where $v(t)$ is the deformed vertex at time t , χ_i^t is the global transform of joint i at time t , χ_i^{-ref} is the inverse global transform of the joint in the reference position, and v^{ref} is the vertex in its reference position. This technique is known as skeletal subspace deformation, or skinning.

The skinning can be efficiently performed by the GPU: the deformable mesh sends the joint transformations of its skeleton to the GPU, that takes care of moving each vertex according to its joint influences. However, it is important to take into account the limitations of graphic cards (Shader Model 2 & 3 [nvi06]), that can store only up to 256 atomic values, *i.e.*, 256 vectors of four floating points. The joint transformations of a skeleton can be sent to the GPU as 4x4 matrices, *i.e.*, four atomic values. This way, the maximum number of joints a skeleton can have reaches:

$$\frac{256}{4} = 64 \quad (2)$$

When wishing to perform hand and facial animation, 64 joints are not sufficient. Our solution is to send each joint transformation to the GPU as a unit quaternion and a translation, *i.e.*, two atomic values. This allows to double the number of joints possible to send. Note that one usually does not wish to use all the atomic structures of a GPU exclusively for the joints of a skeleton, since it is usually exploited to process other data.

Rendering deformable meshes is very costly, due primarily to a pipeline flush occurring each time a new virtual human is rendered, and also to the expensive vertex skinning

and joint transmission. Nevertheless, it would be a great quality drop to do without them, indeed :

- They are the most flexible representation to animate, allowing even for facial and hand animation (if using a sufficiently detailed skeleton),
- Such animation sequences, called skeletal animations, are cheap to store: for each keyframe, only the transformation of deforming joints, *i.e.*, those moved in the animation, need to be kept. Thus, a tremendous quantity of those animations can be exploited in the simulation, increasing crowd movement variety,
- Procedural and composited animations are suited for this representation, *e.g.*, idle motions can be generated on-the-fly (see for example Egges *et al.* [EGMT06]),
- Blending is also possible for smooth transitions between different skeletal animations.

Unfortunately, the cost of using deformable meshes as the sole representation of virtual humans in a crowd is too prohibitive. We therefore use them in a limited number and only at the fore-front of the camera. Note that before switching to rigid meshes, we use several deformable meshes, keeping the same animation algorithm, but with a mesh of a decreasing number of triangles.

Skinned and textured deformable meshes require skilled designers. But once finished, they are automatically used as the raw material to derive all subsequent representations: the rigid meshes and the impostors.

2.3. Rigid Meshes

A rigid mesh is a precomputed geometric posture of a deformable mesh, thus sharing the very same appearance. A rigid animation sequence is always inspired from an original skeletal animation, and from an external point of view, both look alike. However, the process to create them is different. To compute a keyframe of a rigid animation, the corresponding keyframe for the skeletal animation is retrieved. It provides a skeleton posture (or joint transformations). Then, as a preprocess, each vertex is deformed on the CPU, in opposition to a skeletal animation, where the vertex deformation is achieved online, and on the GPU. Once the rigid mesh is deformed, it is stored as a keyframe, in a table of vertices, normals (3D points), and texture coordinates (2D points). This process is repeated for each keyframe of a rigid animation. At runtime, a rigid animation is simply played as the succession of several postures or keyframes. There are several advantages in using such a rigid mesh representation:

- It is much faster to display, because the skeleton deformation and vertex skinning stages are already done and stored in keyframes. The communication between the CPU and the GPU is kept to a minimum, since no joint transformation needs to be sent, and pipeline flushing is significantly reduced.

- It looks exactly the same as the skeletal animation used to generate it.

The gain in speed brought by this new representation is considerable. It is possible to display about 10 times more rigid meshes than deformable meshes (see Section 3.5 for detailed results). However, the rigid meshes need to be displayed farther from the camera than deformable meshes, because they allow for neither procedural animations, nor blending, and no composited, facial, or hand animation is possible.

2.4. Impostor

An impostor is the less detailed representation, and extensively exploited in the domain of crowd rendering [TLC02, DH005, MR06]. An impostor represents a virtual human with only two textured triangles, forming a quad, which is enough to give the wanted illusion at long range from the camera. Similarly to a rigid animation, an impostor animation is a succession of postures, or keyframes, inspired from an original skeletal animation. The main difference with a rigid animation is that it is only a 2D image of the posture that is kept for each keyframe, instead of the whole geometry. Creating an impostor animation is complex and time consuming. Thus, its construction is achieved in a pre-process, and the result is then stored into a database in a binary format (see Section 3.4), similarly to a rigid animation. We detail here how each keyframe of an impostor animation is developed. The first step when generating such a keyframe for a human template is to create two textures, or atlas:

- A normal map, storing in its texels the 3D normals as RGB components. This normal map is necessary to apply the correct shading to the virtual humans rendered as impostors. Indeed, if the normals were not saved, a terrible shading would be applied to the virtual human, since it is represented with only two triangles. Switching from a rigid mesh to an impostor would thus lead to awful popping artefacts.
- A UV map, storing in its texels the 2D texture coordinates as RG components. This information is also very important, because it allows to apply correctly a texture to each texel of an impostor. Otherwise, we would need to generate an atlas for every texture of a human template.

Since impostors are only 2D quads, we need to store normals and texture coordinates from several points of view, so that, at runtime, when the camera moves, we can display the correct keyframe from the correct camera view point. In summary, each texture described above holds a single mesh posture for several points of view. This is why we also call such textures atlas. We illustrate in Figure 1 a 1024x1024 atlas for a particular keyframe. The top of the atlas is used to store the UV map, and its bottom the normal map. The main advantage of impostors is that they are very efficient, since only two triangles per virtual human are displayed. Thus, they constitute the biggest part of the crowd. However, their

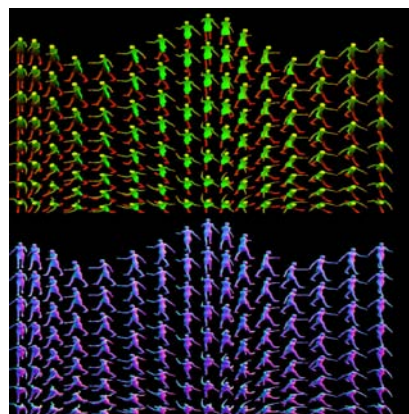


Figure 1: A 1024x1024 atlas storing the UV map (above) and the normal map (below) of a virtual human performing a keyframe of an animation from several points of view.

rendering quality is poor, and thus they cannot be exploited close to the camera. Moreover, the storage of an impostor animation is very costly, due to the high number of textures that need to be saved. We summarize in Table 2 the perfor-

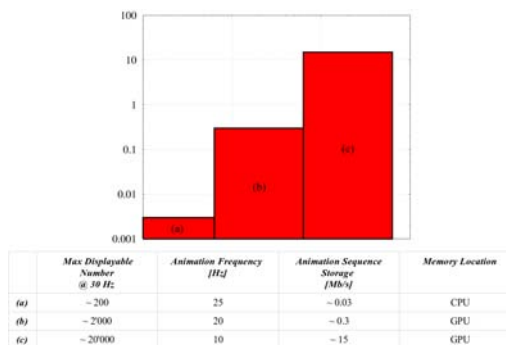


Figure 2: Storage space in [Mb] for one second of an animation clip of (a) a deformable mesh, (b) a rigid mesh, and (c) an impostor.

mance and animation storage for each virtual human representation. We observe that each step down the representation hierarchy allows to increase by an order of magnitude the number of displayable characters. We also note that the faster the display of a representation the bigger the animation storage. Finally, rigid meshes and impostors are stored in GPU memory, which is usually much smaller than CPU memory. Figure 3 summarizes the shared resources inside a human template.

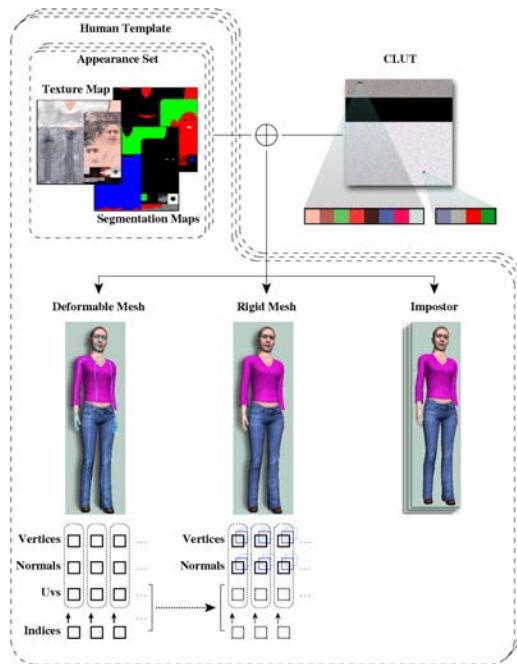


Figure 3: shared resources between representations inside a human template.

2.5. Shadows

In our architecture, illumination ambiances are set from four directional lights, whose direction and diffuse and ambient colors are preably (or interactively) defined by the designer. The light coming from the sun is the only one provoking shadows. As we lack a real-time global illumination system, the three other lights are present to provide enough freedom for the designer to give a realistic look to the scene. This configuration has given us satisfaction as we mainly work on outdoor scenes. See Figure 6 (left) and 7 for results.

Virtual humans cast shadows on the environment and, reciprocally, the environment casts shadows on them. This is achieved using a shadow mapping algorithm [Wil78] implemented on the GPU. At each frame, virtual humans are rendered twice:

- The first pass is from the directional light view perspective, *i.e.*, the sun. The resulting z -buffer values are stored in the shadow map.
- The second pass is from the camera view perspective. Each pixel is transformed into light perspective space and its z value is compared with the one stored in the shadow map. Thus, it is possible to know if the current pixel is in shadow or not.

So, we need to render twice the number of virtual humans

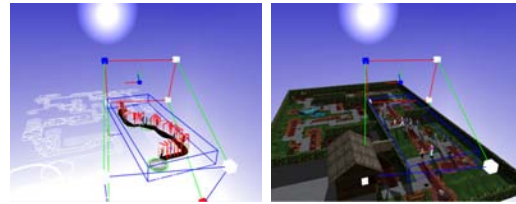


Figure 4: Shadowed scene with apparent directional light frustum.

really present. Though with modern graphics hardware, rendering to a z -only framebuffer is twice as fast as rendering to a complete framebuffer, one expects a certain drop in the frame rate. Moreover, standard shadow mapping suffers from important aliasing artefacts located at shadow borders. Indeed, the resolution of the shadow map is finite, and the bigger the scene, the more aliasing artefacts appear. To alleviate this limitation, several strategies are used:

- Dynamically constrain the shadow map resolution to visible characters, and
- Combine percentage closer filtering [RSC87] with stochastic sampling [Coo86], to obtain fake soft shadows [Ura05].

We now further describe how to dynamically constrain the shadow map resolution to visible characters. A directional light, as its name indicates, is defined only by a direction. Rendering from a directional light implies using an orthographic projection, *i.e.*, its frustum is a box, as depicted in Figure 4. An axis-aligned bounding box (AABB) is a box whose faces have normals that coincide with the basis axes [MH99]. They are very compact to store; only its two extreme points are necessary to determine the whole box. AABB are often used as bounding volumes, *e.g.*, in a first pass of a collision detection algorithm, to efficiently eliminate simple cases.

A directional light necessarily has an orthographic frustum aligned along its own axes. So, we can consider this frustum as an AABB. The idea is, at each frame, to compute the box englobing all the visible virtual humans, so that it is as tight as possible. Indeed, using an AABB as small as possible allows to have a less stretched shadow map. At each frame, we compute this AABB in a four-step algorithm:

1. The crowd AABB is computed in world coordinates, using visible navigation graph vertices. By default, the AABB height is set to two meters, in order to bound the characters at their full height.
2. The light space axes are defined, based on the light normalized direction L_z :

$$L_x = \text{normalize}((0, 1, 0)^T \wedge L_z).$$

$$L_y = \text{normalize}(L_z \wedge L_x).$$

3. The directional light coordinate system is defined as the 3x3 matrix $M_l = [L_x, L_y, L_z]$.
4. The eight points composing the AABB (in world coordinates) are multiplied by M_l^{-1} , *i.e.*, the transpose of M_l . This operation expresses these points in our light coordinate system.

Note that remultiplying the obtained points by M_l would express the crowd AABB back into world coordinates. In Figure 4 are illustrated the shadows obtained with this algorithm. Practically, to be able to choose an adequate resolution given the situation, *e.g.*, detailed shadows for characters close to the camera, we use three different shadow maps: one for the shadows cast by the environment, one for the people (deformable and rigid meshes) near the camera, and one for people far from it (impostors).

3. Architecture

The main problem when dealing with thousands of characters is the quantity of information that needs to be processed for each one of them. Such a task is very demanding, even for modern processors. Simple approaches, where virtual humans are processed one after another, in no specific order, provokes costly state switches for both the CPU and GPU. For an efficient use of the available computing power, and to approach hardware peak performance, data flowing through the same path need to be grouped. In this Section, we present an architecture able to handle, early in its pipeline, the sorting of virtual human related data into grouped slots, allowing the simulation of thousands of characters. Moreover, it is versatile enough to be stressed in very different scenarii, *e.g.*, in confined environments like an auditorium or a classroom, as well as in large-scale environments like a crowded fun fair or city.

The Section is divided as follows: first, in Section 3.1, we briefly introduce the human data structure our architecture employs. Then, we delve into each of the pipeline stages in Section 3.2. In Section 3.3, motion kits, a data structure specifically developed for managing the different levels of detail at the animation stage are described. Concerning efficiency of storage and data management, we mainly employ a database to store all the virtual human related data, as detailed in Section 3.4. Finally, in Section 3.5, we show the overall performance of our architecture.

3.1. Human Data Structures

Virtual human instances are shared in several data structures, and a unique identifier is associated to each one of them. Our crowd data structure is mainly composed of two arrays. An array of body entities, and an array of brain entities. The unique identifier of each virtual human is used to index these arrays and retrieve specific data, which is distributed in a

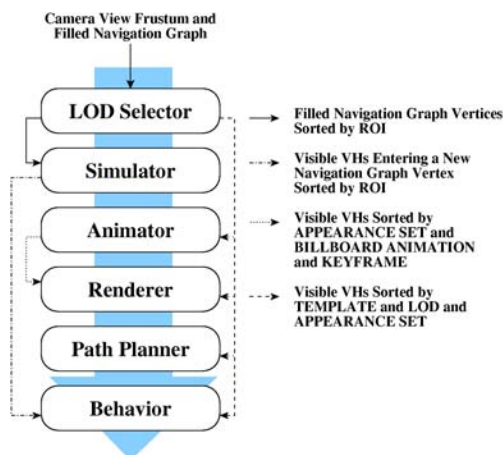


Figure 5: Crowd architecture pipeline.

body and brain entity. Body data consists in all the parameters used at every frame, like the position and orientation of the virtual human. Brain data is more related to behavior parameters, and is less regularly exploited. By separating these parameters from the body entity, we tighten the storage of very often used data. Indeed, such a regrouping improves performance: in a recent work [PdHCM*06], while experimenting different steering methods, we observed that with a varying number of characters in a very large scale (tens of thousands), the performance of the different methods remained about the same. Memory latency to jump from an instance to the other was the bottleneck when dealing with big crowds.

3.2. Pipeline Stages

In this Section, we first provide a short reminder on the navigation graph structure, which is used for crowd motion planning (see Section 5), but also provides a very convenient structure to process the virtual humans hierarchically instead of individually. Then, we detail the stages of the pipeline illustrated in Figure 5.

For a given scene, a navigation graph is provided and used to steer virtual humans along predefined paths. The graph is composed of a set of vertices, represented in the scene as vertical cylinders where no collision with the environment can occur. Two vertices can be connected by an edge, represented as a gate between two overlapping cylinders (see Figure 8). When several cylinders overlap, their consecutive gates delimit a corridor. In a scene, a path to follow is defined as a sequence of gates to reach one after the other, *i.e.*, simple sub-goals for the chosen steering method (See Section 5 for more details). During simulation, each vertex keeps a list of the ids of virtual human currently travelling



Figure 6: (left) Virtual humans navigating in a complex environment. (right) Similar image with apparent levels of detail; in red: the rigid meshes, in green: the impostors.



Figure 7: Dense crowd in a large environment.

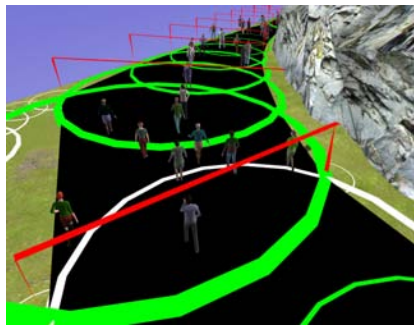


Figure 8: Virtual humans steering along a path sustained by a navigation graph structure (in green and white). Overlapping vertices form gates (in red). Consecutive gates on the path form corridors (in black).

through it. Pettre *et al.* [PdHCM*06,PGT07] fully detail the necessary steps to construct navigation graph from an arbitrary 3D scene. Here follows the detailed description of each pipeline stage.

The **LOD Selector** is the first stage of the pipeline. It receives as input a navigation graph filled with virtual hu-

man ids and the camera view frustum. The role of the LOD Selector entity is to categorize graph vertices, *i.e.*, to score each one of them for further processing. We have two different scores to attribute to each vertex. Firstly, a level of detail (LOD), determined by finding the distance from the vertex to the camera and its eccentricity from the middle of the screen. This LOD score is then used to choose the appropriate virtual human representation inside the vertex. Secondly, the LOD Selector associates with each vertex a score of interest, resulting in an environment divided into regions of different interest (ROI). For each region, we choose a different motion planning algorithm. Regions of high interest use accurate, but more costly techniques, while regions of lower interest may exploit simpler methods (See Section 5 for more details).

The LOD Selector uses the navigation graph as a hierarchical structure to avoid testing individually each character. The processing of data is achieved as follows: firstly, each vertex of the graph is tested against the camera view frustum, *i.e.*, frustum culled. Empty vertices are not even scored, nor further held in the process for the current frame; indeed, there is no interest to keep them in the subsequent stages of the pipeline. On the other hand, vertices filled with at least one character and outside the camera view are kept, but they are not assigned any LOD score, since they are outside the view frustum, and thus, their virtual humans are not displayed. As for their ROI score, they get the lowest one: no dynamic collision avoidance between pedestrians need be achieved. However, even if they are not in the camera field, virtual humans contained in these vertices need a minimal simulation to sporadically move along their path. Without care, when they quit the camera field, they immediately stop moving, and thus, when the camera changes its point of view, packages of stagnant characters suddenly move again, causing a disturbing effect for the user. Finally, the vertices that are filled and visible are assigned a higher ROI score, and then are further investigated to sort their embedded virtual humans by human template, LOD, and appearance set.

At the end of this first stage, we obtain two lists. The first one contains all virtual human ids, sorted by human template, by LOD, and finally by appearance set. The second list contains occupied vertices, sorted by ROI. Obtaining such lists takes some time. However, it is very useful to group data and process through the next stages of the pipeline. We illustrate in the following pseudo-code how the first list is typically used in the next stages of the pipeline:

```

For each human template:
  apply human template common data
  operations, e.g., get its skeleton,
For each LOD:
  apply LOD common data operations,
  e.g., enable LOD specific shader program,
For each appearance set:
  apply appearance set common data
  operations, e.g., bind textures,
For each virtual human id:
  get body or brain structure from the id,
  apply specific operations on it.

```

The second stage is the **Simulator**, which uses the second

list to iterate through all ROI slots and obtain the corresponding filled vertices. At this stage, virtual humans are considered as individual 3D points, and depending on the ROI, the proper motion planning method is applied. Please, refer to Section 5 for more details on the techniques used for each ROI.

The **Animator** is responsible for the animation of the characters, whichever the representation they are using. The slots of visible virtual humans, sorted by human template, LOD, and appearance set in the LOD Selection phase, are the main data structure used in this stage. Below is described the specific tasks that are achieved for the deformable meshes:

```
For each human template:
  get its skeleton,
  For each deformable mesh LOD:
    For each appearance set:
      For each virtual human id:
        get the corresponding body,
        update the animation time (between 0.0 and 1.0),
        perform general skeletal animation,
        perform facial skeletal animation,
        perform hand skeletal animation.
```

Since the virtual humans are also sorted by LOD, we can iterate over the deformable meshes without having to check that they actually are deformable. Performing a skeletal animation, whether it is for the face, the hands or all the joints of a virtual human, can be summarized in four steps. First, the correct keyframe, depending on the animation time, is retrieved. Note that at this step, it is possible to perform a blending operation between two animations. The final keyframe used is then the interpolation of the ones retrieved from each animation. The second step is to duplicate the original skeleton relative joint matrices in a cache. Then, in the cache, the matrices of the joints modified by the keyframe are overwritten. Finally, all the relative matrices (including those not overwritten) are multiplied to obtain global matrices, and each of them is post-multiplied by the inversed global matrices of the skeleton. Note that optional animations, like facial animation, are usually performed only for the best deformable mesh LOD, *i.e.*, the most detailed mesh, at the fore-front.

For the rigid meshes, the role of the Animator is much reduced, since all the deformations are pre-computed (see Section 2):

```
For each human template:
  For each rigid mesh LOD:
    For each appearance set:
      For each virtual human id:
        get the corresponding body,
        update the animation time (between 0.0 and 1.0).
```

Note that we do not iterate over all LOD slots, since we are only concerned with the rigid meshes. Once again, the sorting achieved in the LOD Selection stage ensures that we are exclusively iterating over rigid meshes, without costly tests.

Finally, for the impostors, since a keyframe of an impostor animation is only represented by two texture atlases, no

specific deformation needs to be achieved. However, we assign the animator a special job: to update a new list of virtual human ids, specifically sorted to allow a fast rendering of impostors. Indeed, at initialization, and for each human template, a special list of virtual human ids is created, sorted by appearance set, impostor animation, and keyframe. The first task achieved by the Animator is to reset the impostor specific list in order to refill it accordingly to the current state of the simulation. Then, to refill this list, an iteration is performed over the current up-to-date list, the one sorted by human template, LOD, and appearance set (updated in the LOD Selection stage):

```
For each human template:
  get its impostor animations,
  For the only impostor LOD:
    For each appearance set AS:
      For each virtual human id:
        get the corresponding body,
        update the animation time (between 0.0 and 1.0),
        get body's current impostor animation id a,
        get body's current impostor keyframe id k,
        put virtual human id in special list[AS][a][k].
```

This way, the impostor specific list is updated every time the data passes through the Animator stage, and is thus ready to be exploited at the next stage, the Renderer.

The **Renderer** represents the phase where draw calls are issued to the GPU to display the crowd. As detailed in Section 2.5, rendering shadows is a two-pass algorithm, and achieved in this stage: first, deformable and rigid meshes, and impostors are sequentially rendered from the point of view of the sun, *i.e.*, the main directional light. Then, they are consecutively rendered from the point of view of the camera. To diminish state change overhead, the number of draw calls are minimized, thanks to our slots of visible humans sorted by human template, LOD and appearance set. In the following pseudo-code, we show the second pass in the deformable mesh rendering process:

```
For each human template:
  For each deformable mesh LOD:
    bind vertex, normal, index, and texture buffer,
    send to the GPU the joint ids influencing each vertex,
    send to the GPU their corresponding weights,
    For each appearance set:
      send to the GPU texture specular parameters,
      bind texture and segmentation maps,
      For each virtual human id:
        get the corresponding body,
        send the joint orientations from cache,
        send the joint translations from cache.
```

This second pass is preceded by another pass, used to compute the shadows. Note that in this first pass, the process is quite similar, although data useless for shadow computation is not sent, *e.g.*, normal and texture parameters. In this rendering phase, one can see the full power of the sorted lists: all the instances of a same deformable mesh have the same vertices, normals and texture coordinates. Thus, these coordinates need to be binded only once per deformable mesh LOD. The same applies for the appearance sets: even though they are used by several virtual humans, each needs to be sent only once to the GPU. Note that each joint transformation is sent to the GPU as two vectors of four floating points

(see Section 2.2), retrieved from the cache filled in the Animation phase.

For the rigid meshes, the process is quite different, since all the vertex deformations have been achieved in a pre-process. We develop here the second pass in pseudo-code:

```

For each human template:
  For each rigid mesh LOD:
    bind texture coordinate buffer,
    bind indices buffer,
    For each appearance set:
      send to the GPU texture specular parameters,
      bind texture and segmentation maps,
    For each virtual human id:
      get the corresponding body,
      get the correct rigid animation keyframe,
      bind its vertex and normal buffer.

```

In the rendering phase of the rigid meshes, only the texture coordinates and indices can be binded at the LOD level, in opposition to the deformable meshes, where all mesh data is binded at this level. The reason is obvious: for a deformable mesh, all the components representing its mesh information (vertices, normals, *etc.*) are the same for all instances. It is only later, on the GPU, that the mesh is deformed to fit the skeleton posture of each individual. For a rigid mesh, its texture coordinates, along with its indices (to access the buffers), remain the same for all of their instances. However, since the vertices and normals are displaced in a pre-process and stored in the keyframes of a rigid animation, it is only at the individual level, where we know the animation played, that their binding can be achieved. Note that since the vertices sent to the GPU are already deformed, there is no specific work to be achieved in the vertex shader. Concerning the shadow computation phase, *i.e.*, the first pass, the pseudo-code is the same, but without sending useless data, like normal and texture information.

Rendering impostors is fast, thanks to the virtual human id list sorted by human template, appearance set, animation, and keyframe, that is updated at the Animation phase. Here follows the corresponding pseudo-code:

```

For each human template:
  get its impostor animations,
  For each appearance set:
    bind texture and segmentation maps,
  For each impostor animation:
    For each keyframe:
      bind normal map,
      bind UV map,
    For each virtual human id:
      get the corresponding body,
      get the correct point of view,
      send to GPU texture coordinates where
      to get the correct virtual human posture
      and point of view.

```

The **Path Planner** is performing the collision avoidance between virtual humans. It is at the Simulator stage that sub-goals are set several frames ahead, and that the followed directions are interpolated by steering methods. The Path Planner cares only for collision avoidance, and runs at a lower frequency than the other presented stages. Note that we put this stage and the next one, the Behavior, after the Renderer, because the GPU is parallelly rendering. So, instead of waiting for the frame to finish being rendered, we concurrently

use the CPU. The different algorithms used by the Path Planner are detailed in Section 5.

The **Behavior** is the phase exploiting the slots of virtual humans reaching new navigation graph vertices. All along the entire pipeline, virtual humans cannot change their current animation or steering, because it would invalidate our various sorted slots. This last stage is thus the only one which is allowed to change the steering and current animation sequence of virtual humans. It is always achieved at the end of the pipeline, one frame ahead. Basically, each time a character is entering a new graph vertex (detected at the Simulator phase), we apply a probability to change the steering and / or animation. For instance, a character entering a new vertex with a walk animation clip has a probability to start playing another animation sequence, *e.g.*, an idle one.

3.3. Motion Kits

We have developed three levels of representation for the virtual humans: the deformable meshes, the rigid meshes, and the impostors (see Section 2). When playing an animation sequence, a virtual human is treated differently depending on its current distance and eccentricity to the camera, *i.e.*, the current level of detail it uses. For clarity purpose, we recall giving an animation clip a different name depending on which level of detail it applies to. An animation clip intended for a deformable mesh is a skeletal animation, one for a rigid mesh is a rigid animation, and finally, an animation clip for an impostor is an impostor animation.

We have already shown that the main advantage of using less detailed representations is the speed of rendering. However, for the memory, the cost of storing an animation sequence for a deformable, a rigid mesh, or an impostor is impressively growing (see Figure 2). From this, it is obvious that the number of animation sequences stored must be limited for the less detailed representations. It is also true that we want to keep as many skeletal animation clips as possible for the deformable meshes, firstly, because their storage requirement is cheap, and secondly, for variety purpose. Indeed, deformable meshes are at the forefront, close to the camera, and several virtual humans playing the same animation clip are immediately noticed.

The issue arising is then the switching from a level of representation to another. For instance, what should happen if a deformable mesh performing a walk cycle reaches the limit at which it switches to the rigid mesh representation? If a rigid animation with the same walk cycle (same speed) has been precomputed, switching is done smoothly. However, if the only rigid animation available is a fast run cycle, the virtual human will “pop” from a representation to the other, greatly disturbing the user. We therefore need each skeletal animation to be linked to a resembling rigid animation, and similarly to an impostor animation. For this reason, we have

developed the motion kit data structure. We first describe the motion kit data structure in Section 3.3.1 and then its implementation in Section 3.3.2

3.3.1. Data Structure

A motion kit holds several items:

- A name, identifying what sort of animation it represents, e.g., walk_1.5ms,
- Its type, determined by four identifiers: action, subaction, left arm action, and right arm action,
- A link to a skeletal animation,
- A link to a rigid animation,
- A link to an impostor animation.

Each virtual human knows only the current motion kit it uses. Then, at the Animator stage, depending on the distance of the virtual human to the camera, the correct animation clip is used. Note that there is always a 1:1 relation between the motion kits and the skeletal animations, i.e., a motion kit is useless if there is no corresponding skeletal animation. As for the rigid and impostor animations, their number is much smaller than for skeletal animations, and thus, several motion kits may point to the same rigid or impostor animation. For instance, imagine a virtual human using a motion kit representing a walk cycle at 1.7 m/s. The motion kit has the exact skeletal animation needed for a deformable mesh (same speed). If the virtual human is a rigid mesh, the motion kit may point to a rigid animation at 1.5 m/s, which is the closest one available. And finally, the motion kit also points to the impostor animation with the closest speed. The presented data structure is very useful to easily switch from a representation to another. In Figure 9, we show a schema representing a motion kit and its links to different animation clips. All the motion kits and the animations are stored in a database, along with the links joining them (see Section 3.4). One may wonder what the four identifiers are for. They are used as categories to sort the motion kits. With such a classification, it is easy to randomly choose a motion kit for a virtual human, given certain constraints. Firstly, the action type describes the general kind of movement represented by the motion kit. It is defined as either:

- stand: for all animations where the virtual human is standing on its feet,
- sit: for all animations where the virtual human is sitting,
- walk: for all walk cycles, or
- run: for all run cycles.

The second identifier is the subaction type, which more restrains the kind of activity of the motion kit. Its list is non-exhaustive, but it contains descriptors such as talk, dance, idle, etc. We have also added a special subaction called none, which is used when a motion kit does not fit in any of the other subaction types. Let us note that some action / subaction couples are likely to contain no motion kit at all. For instance, a motion kit categorized as a sit action and a dance

subaction is not likely to exist. The third and fourth identifiers: left and right arm actions are used to add some specific animation to the arms of the virtual humans. For instance, a virtual human can walk with the left hand in its pocket and the right hand holding a cellphone. For now, we have three categories that are common to both identifiers: none, pocket, and cellphone. However, this list can be extended to other possible arm actions. For instance, holding an umbrella, pull a caster suitcase, or scratch one's head.

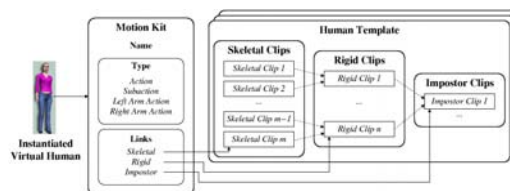


Figure 9: Example of motion kit structure. On the left, a virtual human instantiated from a human template point to the motion kit it currently uses. In the center, a motion kit with its links identifying the corresponding animations to use for all human templates.

When one creates a varied crowd, it is simple for each virtual human to ask randomly for one of all the motion kits available. If the need is more specific, like a crowd following a path, it is easy to choose only the adequate walk / run motion kits, thanks to the identifiers.

3.3.2. Implementation

In our architecture, the motion kits are stored in a four-dimensional table:

```
Table[ action id][ subaction id]
      [left arm action id][right arm action id].
```

For each combination of the four identifiers, a list of motion kits corresponding to the given criteria is stored. As previously mentioned, not all combinations are possible, and thus, some lists are empty. In Figure 10, a virtual human is playing a skeletal animation, linked to a motion kit with the following identifiers: walk, none, cellphone, pocket. In our architecture, an animation (whatever its level of detail) is dependent on the human template playing it: for a deformable mesh, a skeletal animation sequence specifies how its skeleton is moved, which causes the vertices of the mesh to get deformed on the GPU. Since each human template has its own skeleton, it is impossible to share such an animation with other human templates. Indeed, it is easy to imagine the difference there is between a child and an adult skeleton. For a rigid animation, it is the already deformed vertices and normals that are sent to the GPU, thus such an animation is specific to a mesh, and can only be performed by a virtual human having this particular set of vertices, i.e., issued from



Figure 10: A virtual human using a motion kit with identifiers: walk, none, cellphone, pocket.

the same human template. Finally, an impostor animation clip is stored as a sequence of pictures of the virtual human. It is possible to modify the texture and color used for the instances of the same human template, but it seems obvious that such animation pictures cannot be shared by different human templates. This specificity is reflected in our implementation, where three lists of skeletal, rigid, and impostor animations are stored for each human template.

It follows that each motion kit should also be human template-dependent, since it has a physical link to the corresponding animation triplet. However, this way of managing the data is far from optimal, because usually, an animation (whatever its level of detail) is always available for all the existing human templates. It means that, for instance, if a template possesses an animation imitating a monkey, all other human templates are likely to have it. Thus, making the information contained in a motion kit human template-dependent would be redundant. We introduce 2 simple rules that allow us to keep a motion kit independent from a human template:

1. For any motion kit, all human templates have the corresponding animations.
2. For all animations of all human templates, there is a corresponding motion kit.

We now explain how, thanks to these assertions, we can keep a motion kit independent from the human templates and still know to which animation triplet it should link. First, note that each human template contains amongst other things:

- A list of skeletal animations,
- A list of rigid animations,
- A list of impostor animations.

Now, following the two rules mentioned above, all human templates contain the same number of skeletal animations, the same number of rigid animations, and the same number of impostor animations. If we manage to sort similarly these

animation lists for all human templates, we can link the motion kits with them by using their index in the lists. We show a simple example in Figure 9, where a structure representing the human templates is depicted: each human template contains a list of skeletal, rigid, and impostor animations. On the left of the image, a motion kit is represented, with all its parameters. Particularly, it possesses three links that indicate where the corresponding animations can be found for all human templates. These links are represented with arrows in the figure, but in reality, they are simply indices that can be used to index each of the three animation lists for all human templates.

With this technique, we are able to treat all motion kits independently from the human templates using them. The only constraint is to respect the rules (1) and (2).

3.4. Database Management

We use the locomotion engine of Glardon *et al.* [GBT04b, GBT04a] to generate various locomotion cycles. Although this engine is fast enough to generate a walk or run cycle in real-time, it cannot keep up that rhythm with thousands of virtual humans. When this problem first occurred, the idea of precomputing a series of locomotion cycles and store them in a database came up. Since then, this system has proved very useful for storing other unchanging data. The main tables that can be found in the database are the following:

- Skeletal animations,
- Rigid animations,
- Impostor animations,
- Motion kits,
- Human templates, and
- Accessories.

In this Section, we detail what advantages and drawbacks we meet by using such a database, and what kind of information we can safely store there.

As previously mentioned, all the skeletal, rigid and impostor animations can neither be generated online, nor at the initialization phase of the application, because the user would have to wait during an important amount of time before the simulation launch. This is why the database is used. With it, the only work that needs to be done at initialization is to load the animation sequences, so that they are ready when needed at runtime. Although this loading phase may look time consuming, it is quite fast, since all the animation data is serialized into a binary format. Within the database, the animation tables have four important fields [†]: unique id, motion kit id, template id and serialized data. For each animation entry A , its motion kit id is later used to create the

[†] By field, understand a column in the database that allows for queries.

necessary links (see previous Section), while its template id is needed to find to which human template A belongs. It also allows to restrain the number of animations to load to the strict minimum, *i.e.*, only those needed for the human templates used in the application. It is mainly the serialized data that allows to distinguish a skeletal from a rigid or a impostor animation. For a skeletal animation, we mainly serialize all the information concerning the orientation of each joint for each keyframe. With a rigid animation, for each keyframe, a set of already deformed vertices and normals are saved. Finally, for a impostor animation, two series of images of the human template are kept (the normal and the UV map) for several keyframes and points of view.

Another table in the database is used to store the motion kits. It is important to note that since they are mainly composed of simple data, like integers and strings (see previous Section), they are not serialized in the database. Instead, each of their elements is introduced as a specific field: unique id, name, speed, four identifiers (action id, subaction id, left arm action id, right arm action id), and two special motion kit ids (rigid motion kit id, impostor motion kit id). When loading a motion kit M from the database, its basic information, *i.e.*, speed, name, *etc.*, are directly extracted to be saved in our application. Each of the two special motion kit ids is an index referring to another motion kit. This reference is necessary to complete the linking between M and its corresponding rigid and impostor animations.

We have introduced in the database a table in order to store the unchanging data of the human templates. Indeed, we have some human templates already designed and ready to be used in the crowd simulation. This table has the following fields: unique id, name, skeleton hierarchy, and skeleton posture. The skeleton hierarchy is a string summarizing the skeleton features, *i.e.*, all the joint names, ids, and parent. When loading a human template, this string is used to create its skeleton hierarchy. The skeleton posture is a string giving the default posture of a skeleton : with the previous field, the joints and their parents are identified, but they are not placed. In this specific field, we get for each joint its default position and orientation, relatively to its parent. As one can notice, for now the human template table is incomplete, *e.g.*, the appearance sets are missing, and no information is serialized, similarly to the motion kits. This is mainly due to a lack of time (indeed, as of today, our crowd simulator is still being developed). But it certainly is an advantage to further fill this table with more data in a binary format, so that the loading of human templates is faster at initialization.

Finally, the database possesses two tables dedicated to accessories. An accessory is a mesh used to add variety and believability to the appearance of the virtual humans. For instance, it can be a hat, a pair of glasses, a bag, *etc.* (see Section 4.3 for more details). In a first table, we store the elements specific to an accessory, independently from the human template wearing it : unique id, name, type, serialized

data. In the serialized data is stored all the vertex, normal and texture information to make an accessory displayable. The second table is necessary to share information between the accessories and the human templates. As specified in Section 4.3, the displacement of a specific accessory relatively to a joint is different for each human template. This displacement is stored as a matrix. So, in this second table, we employ a field template id and a field accessory id to know exactly where the field matrix must be used. Thus, for each accessory / human template couple, corresponds an entry within this table. Note that we also store there the joint to which the accessory needs to be attached. This is because in some special cases, they may differ from a skeleton to another. For instance, when we attach a back pack to a child template, the joint used is a vertebra that is different from the one for an adult template.

Using a database to store serialized information has proven to be very useful, because it greatly accelerates the initialization time of the application. The main problem is its size, which increases each time a new element is introduced into it. However, with real-time constraints, we allow ourselves to have a sufficiently large database within reasonable limits to obtain varied crowds.

3.5. Performance

We have just detailed the different necessary steps to create and exploit a fast architecture for simulating crowds. We first showed the interest of using several representations, *i.e.*, deformable meshes, rigid meshes, and impostors. Then, we fully detailed each step of our pipeline for fast animation and rendering of thousands of virtual humans. Through the use of motion kits, we allowed for switching smoothly from a representation to another, limiting animation popping artefacts. We exposed how a database can be exploited to store all unchanging data, and finally, we introduced a shadow map algorithm adapted to crowds.

We now expose the performance obtained with this architecture. In Figure 2, the various storage requirements, depending on the animation types are exposed. In Figure 11, we compare the frame rates obtained in two cases. Firstly, when sorted virtual human lists are exploited, as detailed in Section 3.1. Secondly, when the Animator and Renderer stages do not use sorted lists, but directly each virtual human, one after another, in no specific order. With such a process, all the information needed by the GPU has to be sent for each virtual human, independently from the data that may be shared by several of them. As one can observe in Figure 11, when using highly detailed deformable meshes, the results obtained with or without sorted lists are almost similar. This can be explained by the communications sent from the CPU to the GPU (joint transmission): such transmissions imply a pipeline flush for each rendered virtual human, thus becom-

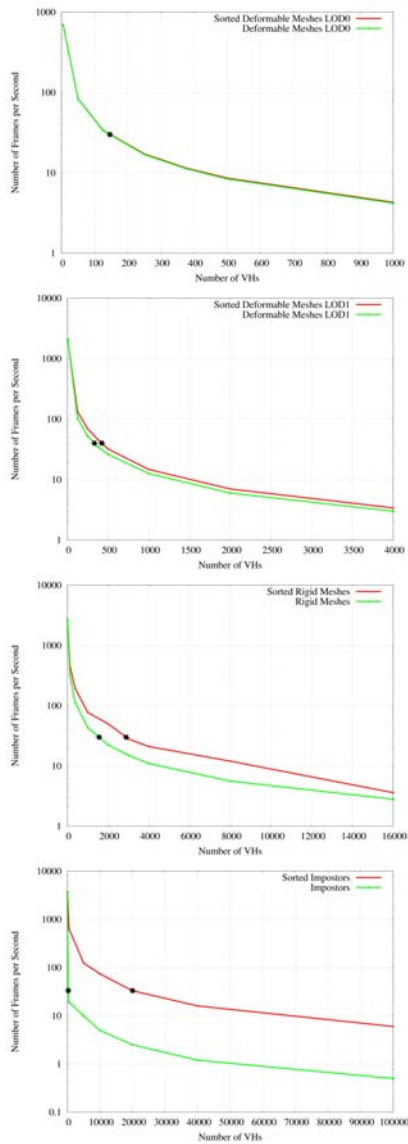


Figure 11: Frames per second obtained for (a) highly detailed deformable meshes, (b) simple deformable meshes, (c) rigid meshes, and (d) impostors. The red lines show the results obtained when working with sorted lists, the green ones with a naive approach. The stars indicate the results for 30 frames per second. (e) Conditions in which the tests have been achieved: five human templates, steering and animation enabled, no shadows, no accessories, no collision avoidance.

ing the bottleneck of the application. However, when less detailed representations are exploited, the advantage of sorting the lists becomes clear. A few images directly obtained from our running architecture are shown in Figure 6(left) and 7(c). In Figure 6(right), one can observe the distance at which the virtual humans switch to lower representations: in red are the rigid meshes, and in green the impostors.

4. Crowd Variety

When simulating a small group of virtual humans, it is easy to make them look singularly different: one can use several human templates and textures for each virtual human present in the scene, and assign them different animations. However, when the group extends to a crowd of thousands of people, this solution becomes unfeasible. First, in terms of design, it is unimaginable to create one mesh and series of animations per individual. Moreover, the memory space required to store all the data would be far too demanding. There is no direct solution to this problem, but it is however possible to achieve good results by multiplying the levels where variety can be introduced. First of all, several human templates can be used. Secondly, for each template, several textures can be designed. Thirdly, the color of each part of a texture can be varied so that two virtual humans issued from the same template and sharing the same texture have not the same clothes / skin / hair color. Finally, we also develop the idea of accessories, which allows a human mesh to be "augmented" with various objects such as a hat, a watch, a backpack, glasses, etc. Variety can also be achieved through animation. We mainly concentrate on the locomotion domain, where we vary the movements of the virtual humans in two ways. Firstly, by generating in a preprocess several locomotion cycles (walk and run) at different speeds, that are then played by the virtual humans online. Secondly, we use off-line inverse kinematics to enhance the animation sequences with particular movements, like having a hand in the pocket, or at the ear as if making a phone call. In the following Section, we further develop each necessary step to vary a crowd in appearance: in Section 4.1, we show the three levels where variety can be achieved. Then, in Section 4.2, we detail how we segment the texture of a virtual human in order to apply varied colors to each identified body part. Moreover, accessories are fully explained in Section 4.3. We also describe animation variety in Section 4.4.

4.1. Variety at Three Levels

When referring to appearance variety, we mean how we modulate the rendering aspect of each individual of a crowd. This term is completely independent from the animation sequences played, the motion planning or the behavior of the virtual humans. First of all, let us remind that a human template is a data structure containing:

- A skeleton, defining what and where are its joints,

- A set of meshes, representing its different levels of detail,
- Several appearance sets, *i.e.*, textures and their corresponding segmentation maps,
- A set of animation sequences that can only be played by this human template.

For further indications on the human template structure, the reader is invited to refer to Section 2. We apply appearance variety at three different levels. The first, coarsest level is simply the number of human templates used. It seems obvious that the more human templates, the more variety. In Figure 12, we show five different human templates to illustrate this. The main issue when designing many human templates is the time required to design them and the memory requirements to store them. Their number needs thus to be limited. In order to mitigate this problem, we further vary the human templates by creating several textures and segmentation map sets for each one of them. For simplification, we designate a texture and its associated segmentation maps as an appearance set. The second level of variety is represented by the texture of an appearance set. Indeed, once an instance of a human template is provided with an appearance set, it automatically assumes the appearance of the corresponding texture. Of course, changing appearance set, and thus, texture, does not change the shape of the human template. For instance, if its mesh contains a pony tail, it will remain whatever the texture applied. However, it can impressively modify the appearance of the human template. In Figure 13, we show five different textures applied to the same human template. Finally, at the third level, we can play with color variety on each body part of the texture, thanks to the segmentation maps of the appearance set. We fully dedicate the next Section to this particular level. In Figure 14, we show several color modulated instances of a single mesh and appearance set.



Figure 12: Five different human templates.

4.2. Color Variety

Human templates possess several textures, improving the sense of variety. But too often, characters sharing the same



Figure 13: Five different textures of a single human template.



Figure 14: Several color varied instances of a single mesh and texture.

texture, *i.e.*, looking exactly the same, appear in the vicinity of the camera, breaking the feeling of uniqueness of the spectator. Differentiating character body parts and then applying a unique combination of colors to each of them is a way to obtain variation inside a single texture.

4.2.1. Principles of the Method

Previous work on increasing the variety in color appearance for the characters composing a crowd share the common idea of storing the segmentation of body parts in a single alpha layer, *i.e.*, each body part is represented by a defined level of intensity of the alpha channel. Tecchia *et al.* [TLC02] use multi-pass rendering and the alpha channel to select parts to render for impostors. Dobbyn *et al.* [DH0005] and De Heras *et al.* [dHCSM*05] avoid multi-pass rendering by using programmable graphics hardware. They also extend the method for being usable by 3D virtual humans too. Figure 15 depicts a typical texture and its associated alpha zone map. The method is based on texture color modulation: the final color C_b of each body part is a modulation of its texture color C_t by a random color C_r :

$$C_b = C_t C_r. \quad (3)$$

Colors C_b , C_t , and C_r can take values between 0.0 and 1.0. In order to have a large panel of reachable colors, C_t should be as light as possible, *i.e.*, near to 1.0. Indeed, if C_t is too dark, the modulation by C_r will give only dark colors. On the other hand, if C_t is a light color, the modulation by C_r will provide not only light colors, but also dark ones. This explains why part of the texture has to be reduced to a light luminance, *i.e.*, the shading information and the roughness of the material. The drawback of passing the main parts of the texture to luminance is that funky colors can be generated, *i.e.*, characters are dressed in colors that do not match. Some constraints have to be added when modulating colors randomly.

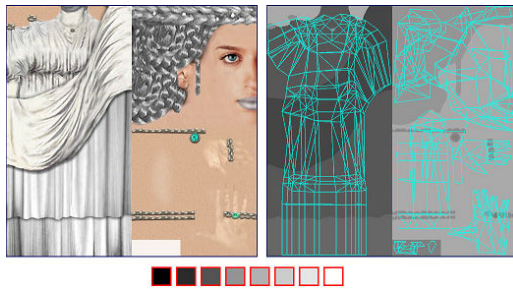


Figure 15: Typical RGBA image used for color variety. The RGB part composes the texture and the alpha the segmentation map.

4.2.2. HSB Color Spaces

The standard RGB color model representing additive color primaries of red, green, and blue is mainly used for specifying color on computer screens. With this system, it is hard to constrain colors effectively (see Figure 16). In order to quantify and control the color parameters applied to the crowd, a user-friendly color is used. Smith [Smi78] proposed a model that deals with everyday life color concepts, *i.e.*, hue, saturation and brightness, which are more linked to the human color perception than the RGB system. This system is called the HSB (or HSV) color model (see Figure 17):

- the hue defines the specific shade of color, as a value between 0 and 360 degrees,
- the saturation denotes the purity of the color, *i.e.*, highly saturated colors are vivid while low saturated colors are washed-out, like pastels. Saturation can take values between 0 and 100, and
- the brightness measures how light or dark a color is, as a value between 0 and 100.

In the process of designing virtual human color variety, localized constraints are dealt with: some body parts need very specific colors. For instance, skin colors are taken from a specific range of unsaturated shades with red and yellow

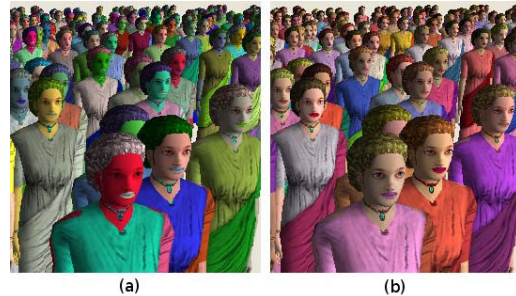


Figure 16: Random color system (a) versus HSB control (b).

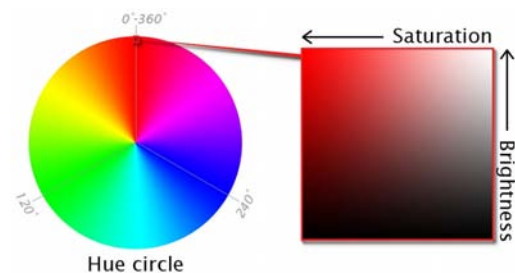


Figure 17: HSB color space. Hue is represented by a circular region. A separate square region may be used to represent saturation and brightness, *i.e.*, the vertical axis of the square indicates brightness, while the horizontal axis corresponds to saturation.

dominance, almost deprived of blue and green. Eyes are described as a range from brown to green and blue with different levels of brightness. These simple examples show that one cannot use a random color generator as is. The HSB color model enables control of color variety in an intuitive manner. Indeed, as shown in Figure 18, by specifying a range for each of the three parameters, it is possible to define a 3D color space, called the HSB map.



Figure 18: The HSB space is constrained to a three dimensional color space with the following parameters (a): hue from 20 to 250, saturation from 30 to 80 and brightness from 40 to 100. Colors are then randomly chosen inside this space to add variety on the eyes texture of a character (b).

4.2.3. Segmentation Maps

The method presented in Section 4.2.1 is perfectly adequate when viewing crowds at far distances. However, when some individuals are close to the camera, the method tends to have too sharp transitions between body parts. There is no smooth blending between different parts, *e.g.*, the transition between skin and hair, as depicted in Figure 19. Also, character closeups bring the need for a new method capable of handling detailed color variety, for instance, subtle make-up effects for female characters. Moreover, at short distances, materials should be illuminated differently to obtain realistic characters at the forefront. To obtain a detailed color variety method, we propose, for each appearance set, to use segmentation maps.

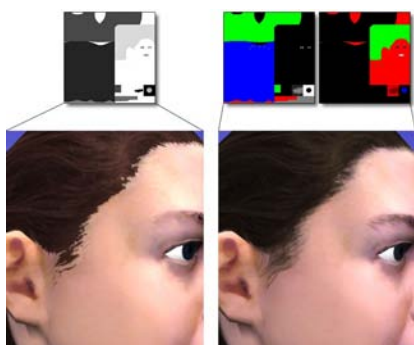


Figure 19: Closeup of the transition between skin and hair: artifacts in previous methods when segmenting body parts in a single alpha layer (left), smooth transitions between parts with our method (right).

A segmentation map is a four channel image, delimiting four body parts (one per channel) and sharing the same parameterization as the texture of the appearance set. The intensity of each body part is thus defined throughout the whole body of each character, *i.e.*, 256 levels of intensity are possible for each part, 0 meaning it is not present at this location, and 255 meaning it is fully present. For our virtual humans, we have made experiments with eight body parts, *i.e.*, two *RGBA* segmentation maps per appearance set. The results are satisfying for our specific needs, but the method can be used with more segmentation maps if more parts are needed. For instance, it would be possible to use the method for adding color variety to a city by creating segmentation maps for buildings. Using segmentation maps to efficiently distinguish body parts also provides two advantages over previous methods:

- Possibility to apply different illumination models to each body part. With previous methods, achieving such effects requires costly fragment shader branching.

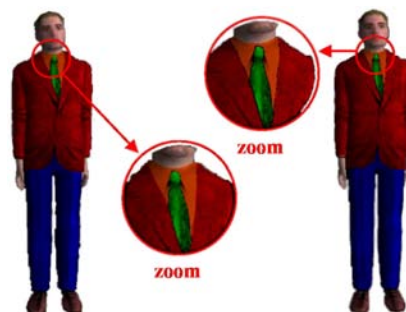


Figure 20: Bilinear filtering artifacts in the alpha layer can be seen in the right zoomed-in version, near the borders of the orange shirt, the green tie and the red vest [Mau05].

- Possible mipmapping activation and use of linear filtering, which greatly reduce aliasing. Since previous methods use the alpha channel of the texture to segment their body parts, they cannot benefit from this algorithm, which causes the appearance of artefacts at body part seams (see Figure 20).

Figure 21 depicts the different effects achievable with our color variety method: make-up, cloth patterns, freckles, etc. and localised specular parameters. The segmentation maps are designed manually. Ideally, for a given pixel, we wish the sum of the intensity of each body part to reach 255. When designing the segmentation maps with a software like Adobe Photoshop, unwanted artefacts may later appear within the smooth transitions between body parts. Indeed, some pixel sums of intensity levels may not reach 255. For instance, imagine the transition between the hair and the skin of a virtual human. A pixel of the segmentation map may reach a contribution of 100 for the skin part, while the hair part contribution is of 120. Their sum amounts to 220. Although this is not an issue while designing the segmented body parts in Photoshop, it leads to problems when trying to normalize the contributions in the application. Indeed, with simple normalization, such pixels compensate the uncomplete sum with a black contribution, thus producing a final color much darker than expected. This is illustrated in Figure 22. The proposed solution is to compensate this lack with white instead of black, to get a real smooth transition without unwanted dark zones. The results obtained with our three levels of appearance variety are illustrated in Figure 23, where several instances of a single human template are displayed, taking full advantage of all available appearance sets and color variety.

4.2.4. Color Variety Storage

Each segmentation map of a human template is divided into four different body parts. Each of these parts has a specific color range, and specular parameters. The

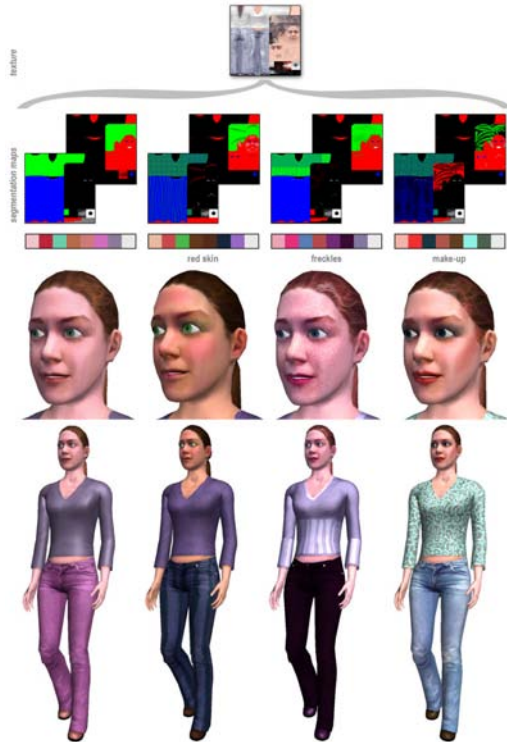


Figure 21: Examples of achievable effects through appearance sets (make-up, freckles, clothes design, etc), and per body part specular parameters (shiny shoes, glossy lips, etc).

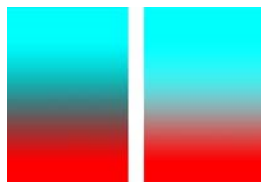


Figure 22: A blue to red gradient. (a) The sum of the red and blue contributions does not reach 255 in some pixels, causing the gradient to suffer from an unwanted black contribution, (b) A white contribution is added so that the sum of contributions is always 255.



Figure 23: Several instances of a single human template, exploiting all its appearance sets and color variety.

eight body parts we need are designed in two different segmentation maps, *i.e.*, two *RGBA* images, each containing four channels and thus four body parts. At its birth, each character is assigned a unique set of eight random colors from the constrained color spaces, similarly to De Heras *et al* [dHCSM*05]. These eight colors are stored in eight contiguous *RGB* texels, starting at the top-left of a 1024×1024 image, called Color Look Up Table (CLUT). We show an illustration of a CLUT in Figure 24. Therefore, if a 1024×1024 image is used for storing the CLUT, it is possible to store a set of up to:

$$\frac{1024 \cdot 1024}{8} = 131,072 \quad (4)$$

unique combinations of colors. Note that illumination parameters are set per body part and thus not saved within the CLUT, but directly sent to the GPU.

4.3. Accessories

We have already described how to obtain varied clothes and skin colors by using several appearance sets. Unfortunately, even with these techniques, the feeling of watching the same person is not completely overcome. The main reason is the lack of variety in the human templates used. Indeed, it is very often the same human template (or a small number of them) that is used for the whole crowd, resulting in large groups of similarly shaped humans. We cannot increase too much the number of human templates, because it requires a lot of work for a designer: create the human template, its textures, its skinning, its different levels of detail, *etc*. Note that the number of human templates is also limited by The storage. However, in real life, people have different haircuts, they wear hats or glasses, carry bags, *etc*. These particularities may look like details, but it is with the sum of those details that we are able to distinguish anyone. In this Section, we first explain what exactly are accessories. Then, we show from a technical point of view the different kinds of accessories we have identified, and how to develop each of them



Figure 24: A CLUT image used to store the color of each virtual human body parts and accessories.

in a crowd application. An accessory is a simple mesh representing any element that can be added to the original mesh of a virtual human. It can be a hat as well as a handbag, or glasses, a clown nose, a wig, an umbrella, a cellphone, etc. Accessories have two main purposes: firstly, they allow to easily add appearance variety to virtual humans. Secondly, they make characters look more believable: even without intelligent behavior, a virtual human walking around with a shopping bag or a cellphone looks more realistic than the one just walking around. The addition of accessories allows a spectator to identify himself to a virtual human, because it performs actions that the spectator himself does everyday. We basically distinguish two different kinds of accessories that are incrementally complex to develop. The first group is composed of accessories that do not influence the movements of a virtual human. For instance, whether someone wears a hat or not will not influence the way he walks. The second group gathers the accessories requiring a small variation in the animation clip played, *e.g.*, a virtual human moving with an umbrella or with a bag still walks the same way, but the arm in contact with the accessory needs an adapted animation sequence.

4.3.1. Simple Accessories

The first group of accessories does not necessitate any particular modification of the animation clips played. They simply need to be correctly "placed" on a virtual human. Each accessory can be represented as a simple mesh, independent from any virtual human. First, let us lay the problem for a single character. The issue is to render the accessory at the

correct position and orientation, accordingly to the movements of the character. To achieve this, we can "attach" the accessory to a specific joint of the virtual human. Let us take a real example to illustrate our idea : imagine a walking person wearing a hat. Supposing that the hat has the correct size and does not slide, it basically has the same movement as the head of the person as he walks. Technically, this means that the series of matrices representing the head movement are the same for the hat movement. However, the hat is not placed at the exact position of the head. It usually is on top of the head and can be oriented in different ways, as shown in Figure 25. Thus, we also need the correct displacement between the head joint position and the ideal hat position on top of it. In summary, to create a simple accessory, our needs are the following:

- For each accessory:
 - A mesh (vertices, normals, texture coordinates),
 - A texture,
- For each human template / accessory couple:
 - The joint to which it must be attached,
 - A matrix representing the displacement of the accessory, relatively to the joint.

Note that the matrix representing the displacement of the accessory is not only specific to one accessory, but specific to each human template / accessory couple. This allows us to vary the position, the size, and the orientation of the hat depending on which virtual human mesh we are working with. This is depicted in Figure 25, where the same hat is worn differently by two human templates. It is also important to note that the joint to which the accessory is attached is also dependent on the human template. This was not the case at first : a single joint was specified for each accessory, independently from the human templates. However, we have noticed that depending on the size of a virtual human, some accessories may have to be attached to different joints. For instance, a backpack is not attached to the same vertebra if it is for a child or a grown up template. Finally, with this information, we are able to assign each human template a different set of accessories, increasing greatly the feeling of variety.

4.3.2. Complex Accessories

The second group of accessories we have identified is the one that requires slight modifications of the animation sequences played. Concerning the rendering of the accessory, we still keep the idea of attaching it to a specific joint of the virtual human. The additional difficulty is the modification of the animation clips to make the action realistic. For instance, if we want to add a cellphone accessory, we also need the animation clips allowing the virtual human to make a phone call. We focus only on locomotion animation sequences. Our raw material is a database of motion captured walk and run cycles that can be applied to the virtual humans. From each animation clip, an adjustment of the arm



Figure 25: Two human templates wearing the same hat, in their default posture. The pink, yellow and blue points represent the position and orientation of the root, the head joint ($m1$), and the hat accessory ($m2$), respectively.

motion is performed in order to obtain a new animation clip integrating the wanted movement, *e.g.*, hand close to the ear. These animation modifications can be generalized to other movements that are independent from any accessory, for instance, hands in the pockets. This is why we fully detail the animation adaptation process in Section 4.4.2.

4.3.3. Loading and Initialization

In this Section, we focus on the architectural aspect of accessories, and how to assign them to all virtual humans. First of all, each accessory has a type, *e.g.*, "hat" or "back pack". We differentiate seven different types, but this number is arbitrary. In order to avoid the attribution of, for instance, a cowboy hat and a cap on the same head, we never allow a character to wear more than one accessory of each type. To distribute accessories to the whole crowd, we need to extend the following data structures (introduced in Section 2):

- **Human template:** each human template is provided with a list of accessory ids, sorted by type. This way, we know which template can wear which accessory. This process is necessary, since all human templates cannot wear all accessories. For instance, a school bag would suit the template of a child, but for an adult template, it would look much less believable,
- **Body entity:** each body entity possesses one accessory slot per existing type. This allows to later add up to seven accessories (one of each type) to the same virtual human.

We also create two data structures to make the accessory distribution process efficient:

- **Accessory entity:** each accessory itself possesses a list of body ids, representing the virtual humans wearing it. They are sorted by human template.
- **Accessory repository:** an empty repository is created to receive all accessories loaded from the database. They are sorted by type.

At initialization, the above data structures are filled. We detail this process in the following pseudo-code:

```

For each accessory in database:
  load its data contained in the database,
  create its vertex buffer (for later rendering),
  insert it into the accessory repository (sorted by type).
For each human template h:
  For each accessory a suitable to h :
    insert a's id into h's list l (sorted by type) .
For each body b:
  get human template h of b,
  get accessory id list l of h,
  For each accessory type t in l:
    choose randomly an accessory a of type t,
    assign a to the correct accessory slot of b,
    push b's id in a's body id list (sorted by human template) .

```

The process of filling these data structures is done only once at initialization, because we assume that once specific accessories have been assigned to a virtual human, they never change. However, it would be easy to change online the accessories worn, through a call to the last loop. Note that a single vertex buffer is created for each loaded accessory, independently from the number virtual humans wearing it.

4.3.4. Rendering

Since the lists introduced in the previous Section are all sorted accordingly to our needs, the rendering of accessories is much facilitated. We show in the following pseudo-code our pipeline:

```

1 For each accessory type t of the repository:
2   For each accessory a of type t:
3     bind vertex buffer of a,
4     send a's appearance parameters to the GPU,
5     get a's list l of body ids (sorted by human template) .
6 For each human template h in l:
7   get the joint j of h to which a is attached,
8   get the original position matrix m1 of j,
9   get the displacement matrix m2 of couple [a,h],
10  For each body b of h:
11   get matrix m3 of b's current position,
12   get matrix m4 of j's current deformation for b,
13   multiply current modelview matrix by mi (i=1..4),
14   call to vertex buffer rendering.

```

Although this pseudo-code may seem complex at first sight, it is quite simple and well optimized to minimize state switches. First of all, at line (3), each accessory has its vertex buffer binded. We can process this way, independently from the bodies, because an accessory never changes its shape or texture. Then, we process through each accessory's body id list (5). This list is sorted by human template (6), allowing us to retrieve information common to all its instances, *i.e.*, the joint j to which is attached the accessory (7), along with its original position matrix $m1$ in the skeleton (8), and

the original displacement matrix $m2$ between $m1$ and the desired position of the accessory (9). An example with a hat attached to the head joint of two human templates is illustrated in Figure 25. Once the human template data is retrieved, we iterate over each body wearing the accessory (10). A body entity also has specific data that is required: its position for the current frame (11), and the displacement of its joint, relatively to its original position, depending on the animation played (12). Figure 26 illustrates the transformation represented by these matrices. Finally, by multiplying the matrices extracted from the human template and body data, we are able to define the exact position and orientation of the accessory (13). The rendering of the vertex buffer is then called and the accessory is displayed correctly (14).

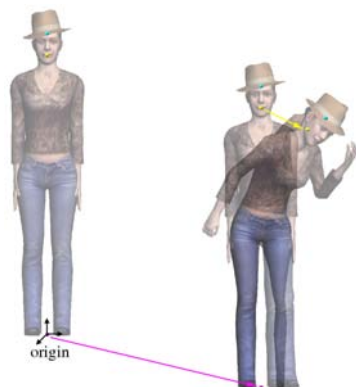


Figure 26: Left: a human template in default posture. Right: the same human template playing an animation clip. The displacement of the body, relatively to the origin ($m3$) is depicted in pink, the displacement of the head joint due to the animation clip ($m4$) in yellow.

4.3.5. Empty Accessories

We have identified seven different accessory types. And, through the accessory attribution pipeline, we assign seven accessories per virtual human. This number is important and the results obtained can be unsatisfying: indeed, if all characters wear a hat, glasses, jewelry, a back pack, etc. they look more like christmas trees than believable people. We need the possibility to have people without accessories too. To allow for this, we could simply randomly choose for each body accessory slot, whether it is used or not. This solution works, but a more efficient one can be considered. Indeed, at the rendering phase of a large crowd, testing each slot to know whether it is used or not implies useless code branching, *i.e.*, precious computation time. We therefore propose a faster solution to this problem by creating empty accessories. An empty accessory is a fake one, possessing no geometry nor vertex buffer. It only possesses a

unique id, similarly to all other accessories. At initialization, before loading the real accessories from the database, the following pseudo-code is executed:

```

For each accessory type $t$:
  create one empty accessory e of type $t$,
  put $e$ in the accessory repository (sorted by type),
For each human template $h$:
  put $e$'s id in $h$'s accessory id list.

```

The second loop over the human templates is necessary in order to make all empty accessories compatible with all human templates. Once this preprocess done, the loading and attribution of accessories is achieved as detailed in Section 4.3.3. This fore introduction of empty accessories causes later their possible insertion in some of the accessory slots of the bodies. Note that if, for instance, a body entity gets an empty accessory for hat, reciprocally, the id of this body will be added to the empty accessory's body id list. This is illustrated with an example in Figure 27. One may wonder how the rendering is achieved. If keeping the same pipeline as detailed in Section 4.3.4, we meet troubles when attempting to render an empty accessory. Moreover, some useless matrix computation would be done. Our solution is simple. Since the empty accessories are the first ones to be inserted into the accessory repository (sorted by type), we only need to skip the first element of each type to avoid their computation and rendering. The pseudo code given in Section 3.4 only needs a supplementary line, which is:

```

1b skip first element of t.

```

With this solution, we take full advantage of accessories, obtaining varied people, not only through the vast choice of accessories, but also through the possibility of not wearing them. And there is no need for expensive tests within the rendering loop. In Figure 28, we show the results obtained when using accessories in addition to the appearance variety detailed in Section 2.

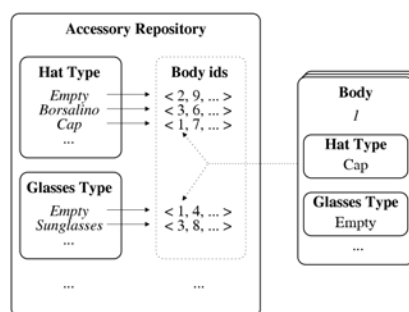


Figure 27: Left: a representation of the accessory repository, sorted by type. Each accessory possesses its own list of body ids. Reciprocally, all bodies possess slots filled with their assigned accessories. Right: illustrated example of the accessory slots for body with id 1.



Figure 28: Several instances of a single human template, varied through the appearance sets, color variety, and accessories.

4.3.6. Color Variety Storage

In Section 4.2.4, we detail how to apply color variety to the different body parts of a texture. The same method can be applied to the accessories. A human texture is segmented in eight body parts, each having its specific color range. At initialization, for each instantiated virtual human and each body part, a color is randomly chosen in a range to modulate the original color of the texture. Since accessories are smaller and less complex than virtual humans, we only use four different parts, *i.e.*, one segmentation map per appearance set. Then, similarly to the characters, each instance of each accessory is randomly assigned four colors within the *HSB* ranges defined for each part. These four random colors have also to be stored. We reemploy the CLUT used for storing the virtual humans color variety to save the colors of the accessories. In order not to confuse the color variety of the body parts and those of the accessories, we store the latter contiguously from the bottom-right of the CLUT (see Figure 24). Each character thus needs eight texels for its own color variety and $7 * 4$ other texels for all its potential accessories. This sums up to 36 texels per character. A $1024 * 1024$ CLUT is therefore able to roughly store more than 29000 unique color variety sets.

4.3.7. scalability

We can simulate a high number of virtual humans, thanks to our different representations. It is important to note that the above description of accessories solves only the case of dynamically animated virtual characters, *i.e.*, deformable meshes. However, if we want to ensure continuity when switching from a representation to another, it is important to also find a solution for the other LOD : a hat on the head of a virtual human walking away from the camera cannot suddenly disappear when the virtual human is switching to a lower representation. We develop here how to make accessories scalable. First, let us detail how accessories can be scaled to fit rigid meshes. An accessory has an animation clip of its own, similar to the animation of a particular joint of a

virtual human. If we wanted to simply apply the rigid mesh principle to accessories, we would have to store an important quantity of information:

```
For each rigid animation:
  For each keyframe:
    For each vertex of the accessory:
      save its new position which is found through
      the animation matrices,
      save its corresponding normal, which is found
      through the animation matrices.
```

As one can see, this pipeline corresponds to the one used to store the vertices and normals of a rigid mesh at each keyframe of a defined animation clip. If we analyze this pipeline, we can observe that there is a clear redundancy in the information stored: firstly, an accessory is never deformed, which means that its vertices do not move, relatively to each other. They can be considered as a single group transformed by the animation matrices. The same applies to the normals of the accessory. Secondly, as detailed in Section 3.3, it is impossible to store in a database a rigid and an impostor animation clip for each existing skeletal animation. It follows that creating all the rigid / impostor versions of an animation clip for each possible accessory cannot be considered. In order to drastically diminish the information to store for an accessory in a rigid animation, we propose a solution in two steps: Firstly, as previously detailed, there is no need to store all the vertices and all the normals at each keyframe of an animation sequence, since the mesh is not deformed. It is sufficient to keep a single animation matrix per keyframe, valid for all vertices. Then, at runtime, the original mesh representing the accessory is transformed by the stored animation matrices. Secondly, we can regroup all accessories depending on the joint they are attached to. For instance, all hats and all glasses are attached to the head. So, basically, they all have the same animation. The only difference between a pair of glasses and a hat is the position where they are rendered, relatively to the head position (the hat is above the head, the glasses in front of it). So, we only need to keep this specific displacement for each accessory relatively to its joint. This corresponds to a single matrix per human template / accessory couple, which is completely independent from the animation clip played (see Section 4.3.1 and 4.3.4). In summary, with this solution, we only need:

```
For each rigid animation:
  For each keyframe:
    For each joint using an accessory:
      a single matrix representing
      the transformation of the joint at this keyframe,
```

and

```
For each human template / accessory couple (independent
of the animation):
  a matrix representing the accessory's displacement,
  relatively to the joint.
```

Scaling the accessory principle to impostors proves to be complicated. Once again, a naive approach would be as follows:

```
For each original impostor animation (without accessories):
  For all possible combinations of accessories:
    create a similar impostor animation directly
    containing these accessories.
```

One can quickly imagine the explosion the memory would endure, even when starting with only a few original impostor animations. We cannot afford to generate one impostor animation for each possible combination of accessories. The first possible simplification is to let the unnoticeable accessories disappear. Indeed, impostors are usually employed when the virtual humans are far from the camera, and thus, small details, taking only a few pixels can be ignored. Such accessories would be watches, jewelry, and others. Of course, it is also dependent on the distance from the camera where the impostors are used, and whether such disappearances are noticeable or not. As for larger accessories, like hats or bags, we are still working to find the best solution, but this work is in progress, and as of today, we have no finite solution to expose.

4.4. Animation Variety

As explained in previous Sections, it is possible to vary the appearance of individuals, even when issued from the same human template. However, we introduced in Section 3.3 the necessity to also provide a large variety of animation clips to the simulation. Virtual humans can be visually as different as possible, if they all perform the same animation, the result is not realistic at all. In this Section, we detail two techniques we employ to vary the animation of characters, while remaining in the domain of navigating crowds, *i.e.*, working with locomotion animations.

4.4.1. Locomotion

First of all, in order to obtain variety in animation, there is a great need for a huge set of raw animation cycles that can then be further varied. We recall here the locomotion engine of Glardon *et al.* that we have used to generate our original set of walk and run cycles. Glardon *et al.* have introduced a PCA-based walk engine capable of animating on the fly human-like characters of any size and proportions by generating complete locomotion cycles [GBT04b,GBT04a]. They have captured walk and run motions from several people, from which they have created a normalized model. There are mainly three high-level parameters which allow to modulate these motions:

- Personification weights: five people, different in height and gait have been captured while walking and running. This variable allows the user to choose how he wishes to parametrize these different styles.
- Speed: the five subjects have been captured at many different speeds. This parameter allows to choose at which velocity the walk/run cycle should be generated.
- Locomotion weights: this parameter defines whether the cycle is a walk or a run animation.

Thus, the engine is able to generate a whole range of varied locomotion cycles for a given character. To efficiently animate the locomotion of each individual, we generate in a pre-

process a certain number of locomotion cycles for each human template. We have used this engine to generate over 100 different locomotion cycles per human template: for each one of them, we sample walk cycles at speeds varying from 0.5 *m/s* up to 2 *m/s* and similarly for the run cycles between 1.5 *m/s* and 3 *m/s*. Each human template is also assigned a particular personification weight so that it has its own style. With such a high number of animations, we are already able to perceive a sense of variety in the way the crowd is moving. Virtual humans walking together with different locomotion styles and speeds add to the realism of the simulation. Once provided with a large set of animation clips, the issue becomes to store and use them in an efficient way. In Section 3.4, we fully detail how the whole data is managed.

4.4.2. Accessory Movements

Variety in movement is one necessary condition for achieving believable synthetic crowds as individuals are seldom unrolling the sole locomotion cycle while moving from one place to another. The upper limb movements being not compulsory in locomotion, hands are most of the time exploited for accessory activities such as holding an object (cell phone, bag, umbrella, *etc.*) or are simply protected by remaining in the pocket of some cloth (see Figure 29). These activities constitute alternate coordinated movements that have to match the continuously changing constraints issued from the primary locomotion movement. Indeed, constantly reusing the same arm posture through the locomotion cycle leads to a loss a believability; for example a hand "in-the-pocket" should follow the pelvis forward-backward movement when large steps are performed. For these reasons, a specific animation cycle has to be defined also for an accessory movement that is to be exploited with locomotion. We achieve the accessory movement design stage after the design of the individual locomotion cycles for a set of discretized speeds. We exploit a Prioritized Inverse Kinematics solver [BB04] that allows combining various constraints with a priority level if necessary. The required input is:

- The set of locomotion cycles,
- One "first guess" posture of the hand and arm, possibly with the clavicle, designed with the skinned target character,
- The set of "effector" points to be constrained on the hand or arm, (see Figure 30, the three coloured cubes on the hand),
- For each effector, its corresponding target goal location expressed in other local frames of the body; for example relative to the head for a cell-phone conversation, or to the pelvis and thigh for a hand in a trousers' pocket (see Figure 30, the three corresponding coloured cubes attached to the pelvis),
- If an effector is more important than the others, the user can associate it with a greater priority level. Our solver ensures that the achievement of other effectors goals does not perturb the high priority one.

All the additional elements to the original locomotion cycles can be specified by an animator by locating them on the target character mesh in a standard animation software. The resulting set of parameters can be saved in a configuration file for the second stage of running the Inverse Kinematics adjustment of the posture for all frames of the locomotion cycles (Figure 31). The resulting accessorized locomotion cycles are saved in files for a further storage optimization stage. Figure 32 shows successive postures from such a movement.

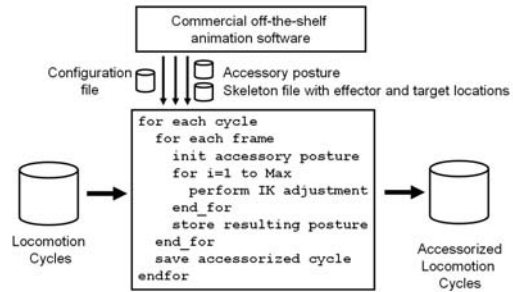


Figure 31: Overview of the two-stage process for producing accessorized locomotion cycles.



Figure 29: Examples of accessory movements (hands in the pocket, phone call, hand on hip, ...).



Figure 32: Example of posture from an accessorized locomotion cycle.

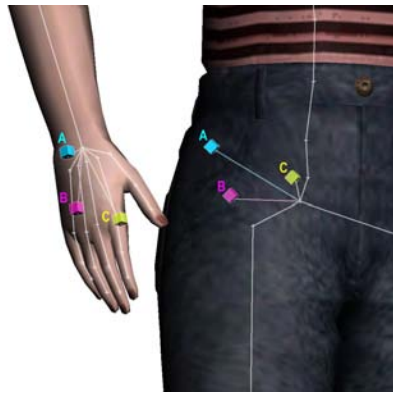


Figure 30: Set of controlled effectors attached to the hand and corresponding goal positions attached to the pelvis.

5. Motion Planning

Realistic real-time motion planning for crowds has become a fundamental research field in the Computer Graphics community. The simulation of urban scenes, epic battles, or other

environments that show thousands of people in real time require fast and realistic crowd motion. Domains of application are vast: video games, psychological studies, architecture, and many others. We present a novel architecture offering a hybrid, scalable solution for real-time motion planning of thousands of characters in complex environments.

Real crowds are formed by thousands of individuals that move in a bounded environment. Each pedestrian has individual goals in space that he wants to reach, avoiding obstacles. People perceive their environment, and use this information to choose the shortest path in time and space that leads to their goal. Emergent behaviors can also be observed in crowds. For example, in places where the space is small and very crowded, people form lanes to maximize their speed. Also, when dangerous events such as fires occur, pedestrians tend to react in very chaotic ways to escape.

Planning crowd motion in real time is a very expensive task, which is often decoupled into two distinct parts: path planning and obstacle avoidance. Path planning consists in finding the best way to reach a goal. Obstacles can either be other pedestrians or objects that compose the environment. The path selection criteria are the avoidance of congested zones, and minimization of distance and travel time. Path

planning must also offer a variety of paths to spread pedestrians in the whole scene. Avoidance, on the other hand, must inhibit collisions of pedestrians with obstacles. For real-time simulations, such methods need to be efficient as well as believable.



Figure 33: Pedestrians using our hybrid motion planning architecture to reach their goal and avoid each other.

Multiple motion planning approaches for crowds have been introduced. As of today, several fast path planning solutions exist. Avoidance however, remains a very expensive task. Agent-based methods offer realistic pedestrian motion planning, especially when coupled with global navigation. This approach gives the possibility to add individual and cognitive behaviors for each agent, but becomes too expensive for a large number of pedestrians. Potential field approaches handle long and short-term avoidance. Long term avoidance predicts possible collisions and inhibits them. Short term avoidance intervenes when long-term avoidance alone cannot prevent collisions. These methods offer less believable results than agent-based approaches, because they do not provide the possibility to individualize each pedestrian. However, this characteristic also implies much lower computational costs.

We present a hybrid architecture to handle realistic crowd motion planning in real time. In order to obtain high performance, our approach is scalable. As briefly introduced in Section 3.2, we divide the scene into multiple regions of varying interest, defined at initialization and modifiable at runtime. According to its level of interest, each region is ruled by a different motion planning algorithm. Zones that attract the attention of the user exploit accurate methods, while computation time is saved by applying less expensive algorithms in other regions. Our architecture also ensures that no visible disturbance is generated when switching from an algorithm to another.

Our results shows that it is possible to simulate up to ten thousand pedestrians in real time with a large variety of goals. Moreover, the possibility to introduce and interactively modify the regions of interest in a scene offers a

way for the user to select the desired performance and to distribute the computation time accordingly. A simulation of pedestrians taking advantage of our architecture to plan their motion in a city environment is illustrated in Figure 33.

The remainder of this Section is organized as follows: first, in Section 5.1, we introduce previous work in crowd motion planning. Then, in Section 5.2, we describe at a high-level our motion planning architecture, and how we exploit it to distribute regions of three different levels of interest. In Section 5.3, the integration of the various approaches employed and the optimizations applied to keep high frame rates are detailed. Finally, in Section 5.4, we run several tests in different conditions and environments to assess our architecture. Finally, limitations are discussed in Section 5.5.

5.1. Crowd Motion Planning Background

Crowd behavior and motion planning are two topics that have long been studied in fields such as Robotics and Sociology. More recently however, and due to the technology improvements, these domains have aroused the interest of the Computer Graphics community as well.

The first studied approach, *i.e.*, agent-based, represents a natural way to simulate crowds as independent individuals interacting with each other. Such algorithms usually handle short distance avoidance, and navigation remains local. Reynolds [Rey99] proposed to use simple rules to model crowds of interacting agents. Heřgeas *et al.* [HLTC03] introduced a model based on cellular automata and the physical properties of the environment, while Kirchner and Shadschneider [KS01] used static potential fields to rule a cellular automaton. Metoyer and Hodgins [MH03] proposed an avoidance algorithm based on a bayesian decision process. Nevertheless, the main problem with agent-based algorithms is their low performance. With these methods, simulating thousands of pedestrians in real time requires the use of particular machines supporting heavy parallelizations [Rey06]. Moreover, such approaches forbid the construction of autonomous adaptable behaviors, and can only manage crowds of pedestrians with local objectives.

To solve the problems inherent in local navigation, some behavioral approaches have been extended with global navigation. Bayazit *et al.* [BLA03] stored global information in nodes of a probabilistic roadmap to handle navigation. Sung *et al.* [SKG05] combined probabilistic roadmaps with motion graphs to find paths and animations to steer characters to a goal, while Lau and Kuffner [LK06] used precomputed search trees of motion clips to accelerate the search for the best paths and motion sequences to reach an objective. Lamarche and Donikian [LD04] used automatic topological model extraction of the environment for navigation. Another method, introduced by Kamphuis and Overmars [KO04], allows a group of agents to stay together while trying to reach

a goal. Although these approaches offer appealing results, they are not fast enough to simulate thousands of pedestrians in real time. Loscos *et al.* [LMM03] presented a behavioral model based on a 2D map of the environment. Their method is suited for simulating wandering crowds, but does not provide high level control on pedestrian goals. As introduced in Section 3.2 Pettré *et al.* [PdHCM*06, PGT07] presented a novel approach to automatically extract a topology from a scene geometry and handle path planning using a *navigation graph* (see Figure 34). The main advantage of this technique is that it handles uneven and multi-layered terrains. Nevertheless, it does not treat inter-pedestrian collision avoidance. Finally, Helbing *et al.* [HMS94, HFV00] used agent-based approaches to handle motion planning, but mainly focused on emergent crowd behaviors in particular scenarii.

Another approach for motion planning is inspired from fluid dynamics. Such techniques use a grid to discretize the environment into cells. Hughes [Hug02, Hug03] interpreted crowds as density fields to rule the motion planning of pedestrians. The resulting potential fields are dynamic, guiding pedestrians to their objective, while avoiding obstacles. Chenney [Che04] developed a model of flow tiles that ensures, under reasonable conditions, that agents do not require any form of collision detection at the expense of precluding any interaction between them. More recently, Treuille *et al.* [TCP06] proposed realistic motion planning for crowds. Their method produces a potential field that provides, for each pedestrian, the next suitable position in space (a *waypoint*) to avoid all obstacles. Compared to agent-based approaches, these techniques allow to simulate thousands of pedestrians in real time, and are also able to show emergent behaviors. However, they produce less believable results, because they require assumptions that prevent treating each pedestrian with individual characteristics. For instance, only a limited number of goals can be defined and assigned to groups of pedestrians. The resulting performance depends on the size of the grid cells and the number of groups.

The work presented in this Section introduces a new hybrid architecture offering a realistic and scalable solution for real-time crowd motion planning. Based on a navigation graph, we divide the environment into regions of varying interest. In regions of high interest, we exploit a potential field-based approach. Since we only use it locally, we can plan motion for many more groups and with finer grid cells than with an algorithm purely based on it. In other regions, motion planning is ruled by the navigation graph and short-term collision avoidance algorithms. Our local use of potential field-based approach allows us to plan motion for many more groups and with finer grid cells than with a purely potential field algorithm.

5.2. Motion Planning Architecture

The foundation of our motion planning architecture is

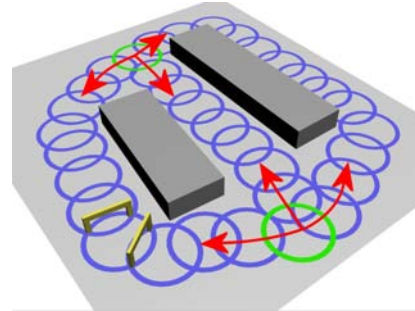


Figure 34: A navigation graph composed of a single navigation flow (in blue) connecting two distant vertices (in green). The navigation flow is composed of three different paths that can be followed in either direction (red arrows). Two edges are also represented as gates (in yellow).

based on navigation graphs, automatically extracted from the mesh of an arbitrary environment. This approach has the advantage of robustly handling path planning. Vertices represent cylindrical zones of the walkable space, while edges are the gates where pedestrians can cross the space from one vertex to another. To connect two distant vertices, it is possible to generate a *navigation flow*, composed of a set of varied paths. We show an example of such a flow in Figure 34. Thanks to this approach, pedestrian spreading is ensured. During simulation, pedestrians are assigned one navigation flow, and one direction. When they reach an extremity of the flow, they reverse their direction, and choose a new path, minimizing their travel time, *e.g.*, avoiding congested areas. Vertices offer a suitable structure of the walkable space. Indeed, they can be exploited to classify different regions of the scene. For instance, Pettré *et al.* [PdHCM*06, PGT07] used them to define several levels of simulation, each updated at different frequencies.

The goal of our architecture is to handle thousands of pedestrians in real time. To achieve this result, we exploit the above mentioned vertex structure to divide the environment into regions ruled by different motion planning techniques. We classify these regions with a level of interest. The most interesting zones are ruled by realistic but expensive techniques, while others use simpler and faster solutions. Regions of interest (ROI) can be defined in any number and anywhere in the walkable space with high-level parameters. Moreover, it is possible to dynamically modify these parameters at runtime. Such flexibility is indeed desirable, because it allows the user to first choose the desired performance, and then distribute ROI, *i.e.*, computation time, as wished.

We observe that by defining only three different ROI, we obtain a simple and flexible architecture for realistic results:

- ROI 0 is composed of vertices of **high** interest.

- ROI 1 regroups vertices of **low** interest.
- ROI 2 contains all other vertices, of **no** interest.

Practically, we position the ROI with respect to the camera position and field of view. ROI 0 is directly in front of it, and/or in zones where important visible events occur. ROI 1 covers the remaining visible space, while ROI 2 includes all vertices outside the view frustum. Note that this choice is arbitrary, and our architecture is versatile enough to satisfy any other environment decomposition.

For regions of no interest (ROI 2), path planning is ruled by the navigation graph. Pedestrians use linear steering to follow the list of waypoints on their path edges. To use the minimal computation resources, obstacle avoidance is not handled.

Path planning in regions of low interest (ROI 1) is also ruled by the navigation graph. To steer pedestrians to their waypoints, an approach similar to Reynolds' is used [Rey99]. In these regions, for obstacle avoidance, an agent-based short-term algorithm (detailed in Section 5.3.4) is exploited. Although agent-based, this algorithm works at a low level, and thus stays simple and efficient.

In the regions of high interest (ROI 0), path planning and obstacle avoidance are both ruled by a potential field-based algorithm, similarly to Treuille *et al.* [TCP06]. Compared to agent-based approaches, potential fields are less expensive, and still offer results more realistic than the ones of ROI 1 and 2, because collision avoidance is planned in the long-term. Nevertheless, in certain situations, this approach fails to avoid collisions. To overcome this problem, the same short-term algorithm as in ROI 1 is also activated in ROI 0.

An important concern when dealing with regions ruled by different motion planning algorithms is to keep smooth and unnoticeable transitions at their borders. The way we place ROI implicitly solves this issue. Firstly, ROI 2 is always outside the view frustum, and thus does not require any specific attention. Secondly, passing the borders between ROI 0 and ROI 1 is always smooth, because they both use the same short-term avoidance algorithm.

5.3. Implementation

In this Section, the details of our hybrid architecture implementation are presented. We mainly focus on the initialization and runtime operations to construct and manage the scalable crowd motion planning. Firstly, in Section 5.3.1, the initialization phase is detailed, *i.e.*, the grid construction over the graph space, the initialization of the structure of neighbor cells and of the ROI. Then, we describe each step of the runtime pipeline, composed of five stages:

- Classification of graph vertices in correct ROI (Section 5.3.2).
- Potential field computation (Section 5.3.3).

- Short-term avoidance algorithm computation (Section 5.3.4).
- Pedestrian steering (Section 5.3.5).
- Continuity maintenance between grid and navigation graph (Section 5.3.6).

5.3.1. Initialization

First of all, for the given environment, a navigation graph is created, and navigation flows generated. We maintain a list of all active vertices, *i.e.*, of all vertices belonging to at least one path. The others are simply discarded, since no pedestrian will ever pass on them during simulation. Then, a grid is disposed on the scene, its size limited by the bounding rectangle containing all graph vertices. This grid is composed of an array of cells, each containing the link to its neighbor cells, and intrinsic parameters used to compute the potential.

Many of the cells that compose the grid are not needed in the simulation, because they represent zones that are not covered by graph vertices, and thus indicate static obstacles. Moreover, some vertices are not used by any navigation flow, and thus are not exploited by pedestrians, as illustrated in Figure 35. Thus, we test whether each cell center is inside a vertex that composes a path. If not, the cell is deactivated. The main advantage of this preprocess is the reduction of the number of cells in which the potential field computation is necessary. Finally, each cell is linked only to its active neighbors.

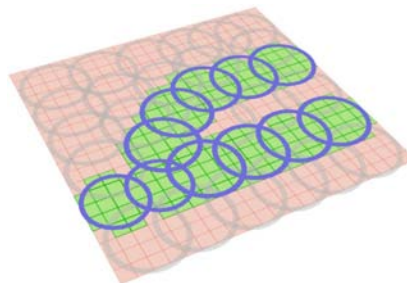


Figure 35: The grid is placed on top of the graph, and only cells within a vertex that is part of a path stay active (in green).

5.3.2. Classification of Graph Vertices in ROI

To define a ROI, the user specifies three parameters: a position, a radius, and a level of interest. All vertices whose center is contained within this region are assigned the specified level. These parameters can be modified at any moment, implying a re-classification of vertices.

In our practical use of ROI, we create three lists corresponding to our three levels of interest. At runtime, we first automatically detect vertices that are outside the view frustum, and insert them into the list with the lowest level of

interest (ROI 2). We then iterate over the remaining vertices, testing whether they are inside a ROI 0. If it is the case, the vertex is classified as of high interest and put in the corresponding list. Otherwise, it is put in the remaining list, of low interest (ROI 1). In the next two sections, we detail how pedestrian motions are planned in ROI 0 and 1.

5.3.3. Potential Field Computation

To accelerate the potential field computation, it is possible to group pedestrians, as suggested by Treuille *et al.* [TCP06]. In our case, pedestrians in ROI 0 having the same navigation flow and direction, *i.e.*, having the same goal, are grouped together. Thus, for each of these navigation flows, there are two groups. Groups are recomputed at each time step, to correctly classify pedestrians that change ROI.

Once this is achieved, for each group, a potential field is computed. At the goal, the potential is set to 0, and increased while spreading over the grid. Given the potential gradient, each pedestrian is assigned a new waypoint, corresponding to the center of a neighbor cell. The potential field computation itself is not further discussed (for details, see [TCP06]), but, taking advantage of our architecture, we introduce two techniques to reduce its computation time. First of all, we have observed no visual alteration when lowering the potential computation frequency to a reasonable value, as opposed to every time step. We thus have empirically set it to 5 Hz. Secondly, with our approach, the potential computation is only required in regions of high interest (ROI 0). These regions only cover part of the scene, and thus part of the grid. By computing the potential only for the cells located inside ROI 0, it is possible to drastically decrease computation time. However, goals are often outside these regions, and thus, it is impossible to initiate the potential computation. For each group, we therefore create subgoals, situated just outside ROI 0, as illustrated in Figure 36. We use the navigation flow structure to identify them: for every path of every flow leaving ROI 0, the first vertex met in the direction of the goal, is a subgoal. The potential computation is initiated in the central cell of every subgoal, and spread over all cells inside ROI 0 vertices. To obtain the same behavior as if the potential was computed all over the grid, we do not initiate the potential of the subgoal cells to 0, but approximate it. For each subgoal cell c inside vertex v_c , the potential ϕ_c is computed as:

$$\phi_c = C \cdot \sum_{v \in P(v_c)} (v.density + 1) \cdot v.radius \quad (5)$$

Where v is a vertex of path $P(v_c)$, starting at v_c and leading to the final goal. The density of v is given by the number of pedestrians inside it per square meter. With Equation 5, the contribution to the potential of each vertex v is defined as its radius, weighted by its degree of occupation. To avoid having a null contribution from an empty vertex, we always add 1 to the computed density. Constant C is used to weight the sum so that values for ϕ_c are in the same range as if the

potential was computed from the goal. Note that vertex v_c may be part of several paths at the same time. In this case, we compute Equation 5 for each path, and assign the lowest result to ϕ_c .

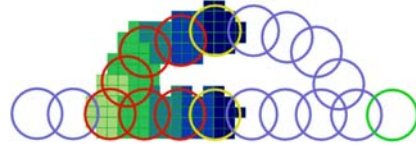


Figure 36: Potential is computed for vertices in ROI 0 (in red) and vertices that have been identified as subgoals (in yellow). The final goal is displayed in green. Potential starts in the central cells of the subgoals with an approximated value.

5.3.4. Short-Term Avoidance Algorithm

In this Section, we detail our short-term avoidance algorithm, which is a simplified low-level agent-based approach. It is used to efficiently avoid local inter-pedestrian collisions in both ROI 0 and 1. Particularly, in ROI 0, it complements the potential field approach, which may fail when the available space is too small and too crowded.

Algorithm 1 details step by step how we manage short-term avoidance. First of all, we need to find pedestrians that can potentially collide. To avoid an exhaustive search, we take advantage of the grid structure covering the whole environment: at runtime, every pedestrian in ROI 0 or 1 is registered in its current grid cell (line 3). This way, we can reduce the search for possible collisions to a small set of neighbor cells. Although this simplification does not cut down the order of complexity in $O(n^2)$, it significantly decreases n , as compared to a brute force approach [Rey87]. To keep the algorithm fast, the two steps mentioned above are alternated during simulation (line 1): we first register the pedestrians to their cell at one time step, while the search for potential collisions and their avoidance is achieved at the next step (line 4). Given the low distance covered by a pedestrian in such a short time lapse, the algorithm robustness is guaranteed.

The avoidance itself is based on two values: a distance of *security* α , fixed at $2 m$, and a distance of *emergency* β , at $0.5 m$. For each pedestrian p in ROI 0 or 1, we start by searching for its neighbor cells in an area of radius α (line 6). Then, for each pedestrian $p_{neighbor}$ contained in a neighbor cell, we test the angle between the heading direction of p and its distance vector to $p_{neighbor}$. If this angle is too small, the current waypoint of p is rotated away from its neighbor (line 11). An illustration of this situation is shown in Figure 37. To make sure the pedestrian still reaches its goal, note that the waypoint is set back to its original position at

the next time step. There are still some cases when this approach fails, e.g., in overcrowded places. If p and $p_{neighbor}$ are at an emergency distance (line 8), p is gently slid aside its neighbor.

```

Data: set of pedestrians in  $\{ROI\ 0 \cup ROI\ 1\}$ , set of grid cells, security distance  $\alpha$ , emergency distance  $\beta$ 
Result: updated set of pedestrians in  $\{ROI\ 0 \cup ROI\ 1\}$ 
if isEven(frameNumber) then
  for each pedestrian  $p \in \{ROI\ 0 \cup ROI\ 1\}$  do
    register  $p$  in its current cell  $c_p$ 
  end
end
else
  for each pedestrian  $p$  in cell  $c_p$  do
     $Set_{neighbors} = findNeighbors(c_p, \alpha)$ 
    for each pedestrian  $p_{neighbor}$  in  $Set_{neighbors}$  do
      if distance( $p, p_{neighbor}$ ) <  $\beta$  then
        slide  $p$  away from  $p_{neighbor}$ 
      end
      else if angle( $p, p_{neighbor}$ )  $\in [-\frac{\pi}{4}, \frac{\pi}{4}]$  then
        rotateWaypoint( $p, p_{neighbor}$ )
      end
    end
  end
end

```

Algorithm 1: Short-term avoidance algorithm.

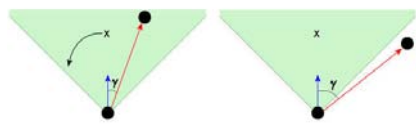


Figure 37: Two pedestrians are closer than the security distance. An angle γ is computed between the first pedestrian's heading vector (in blue) and the two characters distance vector (in red). (left) The angle γ is in the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$ (green zone) and a collision avoidance is attempted by rotating the first pedestrian's waypoint. (right) The second character is outside the green zone, and no avoidance procedure is yet required.

5.3.5. Steering

Both navigation graph and potential field approaches provide waypoints toward which pedestrians have to move. A smooth steering algorithm is necessary to obtain a fluid movement toward these points. The seek behavior of Reynolds [Rey99] has the advantage of producing a believable steering toward a target point in space. We use this steering model for pedestrians of both ROI 0 and 1. For ROI 2, a linear steering is employed.

5.3.6. Continuity Maintenance

In our motion planning architecture, we work with two approaches based on different spaces: a navigation graph defined by its vertices and edges, and a grid composed of cells. This duality brings up two issues when switching from one space to the other. More precisely, when a pedestrian passes from ROI 0 to ROI 1.

The first issue arises when a pedestrian enters the active grid space (ROI 0). Its position is then only updated in the grid, but no longer in the graph. It implies that this character stays registered in the same vertex while progressing in the grid. Thus, its next waypoint on the graph also remains the same. When the pedestrian eventually exits ROI 0, it turns back to meet the graph waypoint it has long since passed. To avoid this problem, we keep updating the pedestrian position in the graph, even in ROI 0: if a pedestrian enters this region, we keep track of its distance to its next graph waypoint. When the distance is under a given threshold, the pedestrian is registered in the next vertex.

The second issue occurs when two or more paths of the same navigation flow are present in ROI 0. Since path planning in that area is ruled by the potential field, a pedestrian chooses the path where the potential is the lowest, as in Figure 38 (right). However, this path does not necessarily correspond to the one it is registered to in the graph (Figure 38 (left)). In the worst case, the pedestrian becomes completely lost when exiting ROI 0: it is within a vertex that does not belong to the path it should follow. To solve this problem, when any pedestrian exits ROI 0, we test whether it still is on the same graph path. If not, we look for a new path using this vertex and register the pedestrian to it.

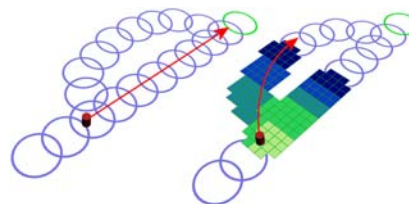


Figure 38: (left) In graph space, the path followed by the pedestrian is the right one. (right) In grid space, the potential field is lower on the left path. High potential is represented in light green and low potential in dark blue.

5.4. Performance Tests

We have run several tests in different crowded environments with an Athlon64 4000+, with 2 GB of memory and two NVidia 6800 ultra in SLI mode. For all tests, pedestrians are represented with two human templates using several

textures, and exploiting color variety techniques. They are rendered as impostors, and use a walk animation, sampled at a frequency of 20 Hz. Note that in all the following tests, we observe interesting emergent behaviors, *e.g.*, lane formations or panic effects, that make the crowd motion planning more realistic.

We use a first environment, representing a city pedestrian area, to test the performance of our motion planning architecture, compared with our implementation of the purely potential field-based approach of Treuille *et al.* [TCP06]. In this scene, the camera position is fixed at a predefined position. For our tests, we define three regions in the environment. The one with the highest level of interest (ROI 0) has a radius of 15 m, and is static, positioned at the center of the scene. Note that we have voluntarily set this region in the center of the scene, where the eye is naturally attracted, rather than in front of the camera. The remaining space inside the view frustum is of low interest (ROI 1), and the other zones are classified in ROI 2. We have tested the efficiency of both approaches with cells of $3 \times 3 m^2$, and an increasing number of pedestrians and groups, starting from 2 groups and 200 pedestrians up to 12 groups, totaling 1,200 characters. Figure 39 shows the results of this comparison. We can see that the performance of our approach logically decreases with the increasing number of groups, but much more slowly than with the purely potential field-based approach. There are two reasons. Firstly, our technique only computes the potential field in a limited region of high interest (ROI 0). Secondly, only a subset of the total number of groups passes in this region, minimizing the number of potential fields to compute. This test has also been performed with the ROI 0 dynamically moving on the city place. Even so, the obtained results remain similar to those illustrated in Figure 39.

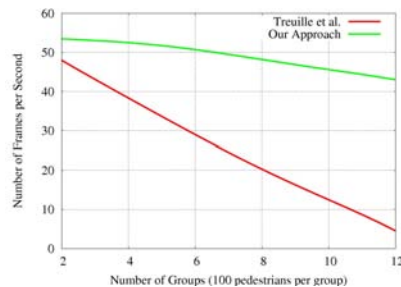


Figure 39: Comparison between our approach and our implementation of the purely potential field-based approach of Treuille *et al.* [TCP06] for a varying number of groups. Each group is composed of a hundred pedestrians.

Our second test is achieved with a crowd of 10,000 pedestrians in a large scene with 12 navigation flows, *i.e.*, 24 groups, spread over the whole environment, as demonstrated

in Figure 40. For this scenario, the different regions of interest are placed according to the camera position. If the camera moves, the regions are also displaced. The cell size is set to $4 \times 4 m^2$, and the obtained performance is of about 20 fps.

For our third scenario, we use the same city pedestrian area as in the first test, but extend it with several surrounding streets and buildings. There are 5,000 pedestrians and some cars navigate on the roads. We illustrate this scenario in Figure 41. Each cell of the grid covers a $3 \times 3 m^2$ area. Since the user attention is mainly drawn by the cars, which threaten to hit pedestrians at every moment, a region of high interest (ROI 0) is set around each of them. Moreover, to make pedestrians flee the potential collision, a high discomfort and speed increase are set in front of the cars, as in [TCP06]. As a result, pedestrians close to a car are always in a region of high interest, and thus ruled by a potential field. In front of cars particularly, the pedestrians flee the zone of danger, demonstrating an emergent panic behavior. The remaining visible environment is classified as a region of low interest (ROI 1), so that pedestrians still take care to avoid each other. Finally, the zone outside the view frustum is set as of no interest (ROI 2). The resulting fps varies between 15 and 30, depending on the number of visible cars (1 to 3), and the size of their surrounding ROI 0, (10 to 15 m radius).



Figure 40: 10,000 pedestrians planning their motion in a large landscape of fields. There are 12 navigation flows and the cell size is set to $4 \times 4 m^2$.

Finally, we have tested the evolution of the frame rate with a fixed number of groups and an increasing number of pedestrians. The test has been achieved in a large scene with 24 groups, a cell size of $3 \times 3 m^2$, and 1 to 5 regions of high interest, distributed over the scene. Each of them has a fixed radius of 15 m. For the remaining of the scene, ROI 2 is not exploited; all vertices are classified as ROI 1. During the test, the rendering of the scene and pedestrians was deactivated to analyze the sole motion planning cost. The results, in Figure 42, show that even with 5 different regions of interest, our architecture still manages the motion planning of 10,000 pedestrians at interactive frame-rates (between 10 and 15 fps). Note that the increasing number of pedestrians

does not as much influence the potential computation, which is more sensible to the number of groups, than the short-term avoidance, which has a complexity in $O(n^2)$.



Figure 41: A city scene where pedestrians avoid a car surrounded by a ROI 0.

All tests show that our motion planning architecture offers high performance for large crowd motion planning. The possibility to select and distribute regions of interest as wished gives the opportunity to easily tune the simulation for the desired frame rate, and to define many groups, *i.e.*, many different goals. Additionally, compared to a purely potential-field based technique, much smaller cells can be used for obtaining better visual results in long-term avoidance cases.

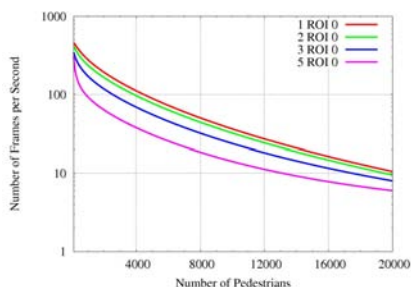


Figure 42: Performance obtained for a number of groups fixed to 24 and an increasing number of pedestrians, without rendering. 1 to 5 ROI 0 with a radius of 15 m each are placed in the scene, while the remaining space is entirely in ROI 1.

5.5. Limitations

There are some limitations to our motion planning architecture. Firstly, in too crowded narrow environments, severe bottlenecks may appear, making the use of our potential field-based approach a waste of computational time. However, it is possible to force such regions to always keep a

predefined lower level of interest, *e.g.*, ruled by a short-term avoidance algorithm. Another limitation is the use of a group-based approach. Indeed, we are constrained to assign general goals for groups of pedestrians. Assigning one different goal to each pedestrian would be too prohibitive for real-time applications. Yet, we note that our architecture is able to handle many more groups than previous potential field-based methods. This is mainly due to our massive reduction of the number of cells in which the potential actually needs to be computed, and implies the possibility to refine the grid for more accurate results.

6. Conclusion

In this tutorial, we have detailed the numerous aspects that need to be taken into account when simulating crowds in real time. We have shown how to efficiently exploit the computational time with a complete description of our architecture and pipeline. In order to obtain a large variety of characters, we have described several techniques, fast and robust, based on the use of a limited number of human templates. Means to obtain variety in animation have also been introduced, while our hybrid scalable motion planning algorithm has been thoroughly detailed. Tests and results have also been presented to estimate the achieved performance for different parts of the complete architecture.

7. Acknowledgements

We would like to thank Mireille Clavien for her great job on designing virtual humans and creating several figures of this tutorial. We acknowledge Antoine Schmid and Ronan Boulic for their work on accessory movements, Fiorenzo Morini for his work on motion planning. This research was sponsored by the Swiss National Research Foundation.

References

- [BB04] BAERLOCHER P., BOULIC R.: An inverse kinematics architecture enforcing an arbitrary number of strict priority levels. *Vis. Comput.* 20, 6 (2004), 402–417.
- [BLA03] BAYAZIT O. B., LIEN J.-M., AMATO N. M.: Better group behaviors in complex environments using global roadmaps. In *ICAL 2003: Proceedings of the eighth international conference on Artificial life* (Cambridge, MA, USA, 2003), MIT Press, pp. 362–370.
- [Che04] CHENNEY S.: Flow tiles. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, 2004), Eurographics Association, pp. 233–242.
- [Coo86] COOK R. L.: Stochastic sampling in computer graphics. *ACM Trans. Graph.* 5, 1 (1986), 51–72.

- [dHCSM*05] DE HERAS CIECHOMSKI P., SCHERTENLEIB S., MAÏM J., MAUPU D., THALMANN D.: Real-time Shader Rendering for Crowds in Virtual Heritage. In *VAST '05, 2005* (2005).
- [DHOO05] DOBBYN S., HAMILL J., O'CONNOR K., O'SULLIVAN C.: Geopostors: a real-time geometry / impostor crowd rendering system. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (New York, NY, USA, 2005), ACM Press, pp. 95–102.
- [EGMT06] EGGES A., GIACOMO T. D., MAGNENAT-THALMANN N.: Synthesis of realistic idle motion for interactive characters. In *Game Programming Gems 6* (2006).
- [GBT04a] GLARDON P., BOULIC R., THALMANN D.: A coherent locomotion engine extrapolating beyond experimental data. In *Proc. of Computer Animation and Social Agent* (2004).
- [GBT04b] GLARDON P., BOULIC R., THALMANN D.: Pca-based walking engine using motion capture data. In *Proc. of Computer Graphics International* (2004).
- [HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature* 407, 6803 (September 2000), 487–490.
- [HLTC03] HEIGEAS L., LUCIANI A., THOLLOT J., CASTAGNÉ N.: A physically-based particle model of emergent crowd behaviors. In *Graphicon* (2003).
- [HMS94] HELBING D., MOLNÁR P., SCHWEITZER F.: Computer simulations of pedestrian dynamics and trail formation. *Evolution of Natural Structures* (1994), 229–234.
- [Hug02] HUGHES R.: A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological* 36 (July 2002), 507–535(29).
- [Hug03] HUGHES R. L.: The flow of human crowds. *Annual Review of Fluid Mechanics* 35, 1 (2003), 169–182.
- [KO04] KAMPHUIS A., OVERMARS M. H.: Finding paths for coherent groups using clearance. In *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation* (Aire-la-Ville, Switzerland, Switzerland, 2004), Eurographics Association, pp. 19–28.
- [KS01] KIRCHNER A., SHADSCHNEIDER A.: Simulation of evacuation processes using a bionics-inspired cellular automaton model for pedestrian dynamics. In *Physica A* (2001), pp. 237–244.
- [LD04] LAMARCHE F., DONIKIAN S.: Crowd of virtual humans: a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum* 23, 3 (2004), 509–518.
- [LK06] LAU M., KUFFNER J. J.: Precomputed search trees: Planning for interactive goal-driven animation. In *2006 ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (Sept. 2006), pp. 299–308.
- [LMM03] LOSCOS C., MARCHAL D., MEYER A.: Intuitive crowd behaviour in dense urban environments using local laws. In *TPCG '03: Proceedings of the Theory and Practice of Computer Graphics 2003* (Washington, DC, USA, 2003), IEEE Computer Society, p. 122.
- [LWC*02] LUEBKE D., WATSON B., COHEN J. D., REDDY M., VARSHNEY A.: *Level of Detail for 3D Graphics*. Elsevier Science Inc., New York, NY, USA, 2002.
- [Mau05] MAUPU D.: Creating variety - a crowd creator tool, 2005.
- [MH99] MÖLLER T., HAINES E.: *Real-time rendering*. A. K. Peters, Ltd., Natick, MA, USA, 1999.
- [MH03] METOYER R. A., HODGINS J. K.: Reactive pedestrian path following from examples. In *CASA '03: Proceedings of the 16th International Conference on Computer Animation and Social Agents (CASA 2003)* (Washington, DC, USA, 2003), IEEE Computer Society, p. 149.
- [MR06] MILLAN E., RUDOMIN I.: Impostors and pseudo-instancing for gpu crowd rendering. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (New York, NY, USA, 2006), ACM Press, pp. 49–55.
- [nvi06] NVIDIA: Nvidia geforce 8800 gpu architecture overview, 2006.
- [PdHCM*06] PETTRÉ J., DE HERAS CIECHOMSKI P., MAÏM J., YERSIN B., LAUMOND J.-P., THALMANN D.: Real-time navigating crowds: scalable simulation and rendering. *Journal of Visualization and Computer Animation* 17, 3-4 (2006), 445–455.
- [PGT07] PETTRÉ J., GRILLON H., THALMANN D.: Crowds of moving objects: Navigation planning and simulation. In *Proceedings of IEEE International Conference on Robotics and Automation* (2007).
- [Rey87] REYNOLDS C. W.: Flocks, herds and schools: A distributed behavioral model. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 25–34.
- [Rey99] REYNOLDS C.: Steering behaviors for autonomous characters, 1999.
- [Rey06] REYNOLDS C.: Big fast crowds on ps3. In *sandbox '06: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (New York, NY, USA, 2006), ACM Press, pp. 113–121.
- [RSC87] REEVES W. T., SALESIN D. H., COOK R. L.:

J. Mäim, B. Yersin and D. Thalmann / Real-Time Crowds: Architecture, Variety and Motion Planning

- Rendering antialiased shadows with depth maps. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1987), ACM Press, pp. 283–291.
- [SKG05] SUNG M., KOVAR L., GLEICHER M.: Fast and accurate goal-directed motion synthesis for crowds. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 291–300.
- [Smi78] SMITH A. R.: Color gamut transform pairs. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), ACM Press, pp. 12–19.
- [TCP06] TREUILLE A., COOPER S., POPOVIC; Z.: Continuum crowds. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM Press, pp. 1160–1168.
- [TLC02] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Image-based crowd rendering. *IEEE Computer Graphics and Applications* 22, 2 (2002), 36–43.
- [Ura05] URALSKY Y.: Efficient soft-edged shadows using pixel shader branching. In *GPU Gems 2* (2005).
- [Wil78] WILLIAMS L.: Casting curved shadows on curved surfaces. In *SIGGRAPH '78: Proceedings of the 5th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1978), ACM Press, pp. 270–274.

Populating Virtual Environments with Crowds: Level of Detail for Real-Time Crowds

S. Dobbyn and C. O'Sullivan

Graphics, Vision and Visualisation (GV2) Lab, Trinity College Dublin, Ireland

Abstract

Computer generated crowds have become increasingly popular in films. However, their presence in the real-time domain, such as computer games, is still quite rare. Even though there has been extensive research conducted on human modelling and rendering, the majority of it is concerned with realistic approximations using complex and expensive geometric representations. When dealing with the visualisation of large-scale crowds, these approaches are too computationally expensive, and different approaches are needed in order to achieve an interactive frame rate.

1. Introduction

This part of the tutorial describes the main research related to the real-time visualisation and animation of virtual crowds in the following manner:

- We first introduce general **character visualisation** techniques using the fixed function graphics pipeline, and show how recent improvements in graphics hardware has greatly improving the realism of characters in computer games. Furthermore, we describe **acceleration techniques** for the rendering of large crowds which can be subdivided into three categories: **visibility culling** methods, **geometrical level of detail (LOD)** and sample-based rendering techniques such as using **image-based** and **point-based** representations.
- Then, we describe **character animation techniques**, including how a character's model is animated using the **layered approach**, and the various techniques for generating character animations such as **kinematics**, **physically-based animation** and **procedural animation**. We also describe how **animation and simulation level of detail** provides a computationally efficient solution for the simulation of crowds.

2. Character Visualisation

2.1. Character Model

The most common model used for representing characters in 3-D computer graphics is the mesh model. A mesh is de-

finied as a collection of polygons, where each polygon's surface is made up of three or more connected vertices, and is typically used to represent an object's surface such as a character's skin. Since 3-D graphics hardware is optimised to handle triangles, meshes are typically made up of this type of polygon in 3-D applications. A simple model, consisting of a low number of triangles (i.e., several hundred), can be used to model a character's general shape. However, as the need for realism increases, more detailed models are necessary and require a high number of triangles (i.e., several thousand) to model the character's hands, eyes and other body-parts. This extra detail comes at a greater rendering cost and a balance between realism and interactivity is necessary, especially when rendering large crowds of characters. While current graphics cards can render over several hundred million unlit triangles per second (e.g. ATI's and NVIDIA's current cards), a static scene such as an urban environment populated with multiple characters could require rendering several hundred thousand triangles. Therefore, depending on the scene complexity, the number of triangles in the character's mesh or any other scene object is limited in order to maintain a real-time frame rate.

Real-time lighting of these meshes is necessary to provide depth cues and thus enhance the scene's realism. Otherwise, the triangles are rendered with a single colour creating a flat unrealistic look. Typically, the lighting of the character's mesh in games is implemented with basic Gouraud shading [Gou71]. Gouraud shading is a method for linearly interpolating a colour across a polygon's surface and is used

to achieve smooth lighting, giving a mesh a more realistic appearance. As a result of its smooth visual quality and its modest computational demands, since lighting calculations are performed per-vertex and not per-pixel, it is by far the predominant shading method used in 3-D graphics hardware. Additionally, texture-mapping [Cat74], which allows the attaching of a two-dimensional image onto the polygon's surface, can greatly improve the realism of a human's mesh. These textures are usually artist-drawn or scanned photographs and are typically used to capture the detail of areas such as human's hair, clothes and skin (as shown in Figure 1). The image is loaded into memory as a rectangular array of data where each piece of data is called a *texel* and each of the polygon's vertices are assigned texture coordinates to specify which texels are mapped to the surface.



Figure 1: Simple Texturing-Mapping: (a) Mesh without texture-mapping, (b) Texture Map (c) Texture-mapped mesh.

2.2. Character Rendering

Until a few years ago, the only option for hardware-accelerated graphics was to use the fixed function pipeline. This is where texture addressing, texture blending and final fragment colouring are fixed to perform in set ways. The introduction of the *multitexture* extension [Ope04], allowed lighting effects involving several different types of texture maps to be performed in a single rendering pass. This extension provides the capability to specify multiple sets of texture coordinates that address multiple textures, which means that the previous and slower method of multi-pass rendering can be avoided. More recently, hardware vendors have exposed general programmable pipeline functionality, allowing for more versatile ways of performing these operations through programmable customisation of vertex and fragment operations [Ope04]. With the introduction of multi-texturing and programmable graphics hardware, coupled with the improvements in hardware capability such as the increase in triangle fill-rates, texture memory size and memory bandwidth, we are seeing an exciting era of realistic character rendering and animation techniques which were previously unfeasible to employ at interactive rates.

There has been extensive research on enhancing the realism of a character's mesh by applying various per-pixel lighting effects (see Figure 2). Environment mapping [BN76] can be used to simulate an object reflecting its environment. For characters such as soldiers wearing shiny

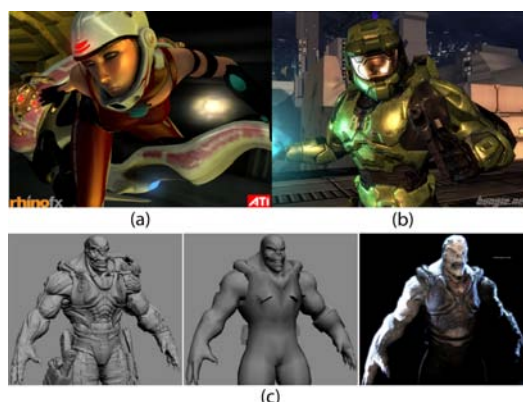


Figure 2: Per-pixel lighting effects such as environment mapping in (a) Ruby Demo ((© ATI Technologies) and (b) Halo 2 (© 2004 Microsoft Corporation), and (c) Normal mapping in Unreal Engine 2003 (© 2005 Epic Games Inc).

armour, environment mapping can greatly improve their realism. Per-pixel bump mapping [Kil00] can be used to perturb the surface's normal vector in the lighting equation to simulate wrinkles or bumps. This is used to increase the visual detail of the character's clothing and appearance without increasing geometry. More recently, this approach has been extended by using a normal map image, generated from a highly detailed character's mesh, in conjunction with a low detailed mesh to improve its visual detail [COM98, Map]. Displacement mapping is another method which adds surface detail to a model by using a height map to translate vertices along their normals [Don05]. In order to speed up the lighting calculations for a static object, the lighting can be pre-computed and stored for each polygon in a texture called a light map [SKvW*92] and this method was made famous by iD Software's "Quake" games. In addition to the speed increase, this method allows complex and more realistic illumination models to be used in generating the map. With dynamic objects, the light map needs to be calculated on a per-frame basis, as otherwise shading artefacts will manifest. Sander et al. [SGM04] recalculate the light map using graphics hardware for each frame in order to correctly shade the character's skin as it moves within its environment. However, generating real-time light maps for a large number of characters is unfeasible at interactive frame-rates.

More recently, more realistic character effects borrowed from the film industry have been implemented in real-time. Based on the technique used to light the face of digital characters in the film *The Matrix Reloaded*, Sander et al. [SGM04] produced realistic looking skin in real-time. Scheuermann et al. [Sch04] improved the rendering of real-time hair using a polygonal model, where the hair shading is based on the work on light scattering of human hair fibers by Marschner et al. [MJC*03] and on Kayiya et al.'s fur ren-

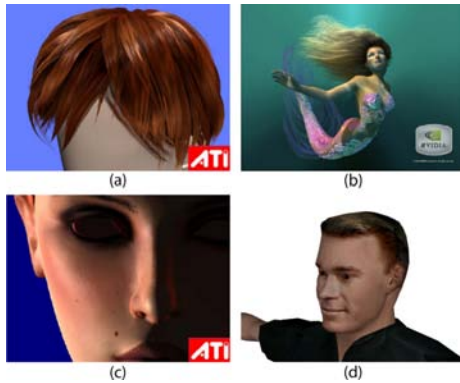


Figure 3: Real-time hair rendering based on the light scattering of human hair fibres [MJC*03] and a fur rendering model [KK89] using a (a) polygonal model [Sch04] (b) a particle system [Wlo04]. (c) Real-time skin rendering based on subsurface scattering [SGM04]. (d) Hair and skin rendered with simple texturing.

dering model [KK89]. While this technique has greatly improved the realism of real-time hair, in addition to using low geometric complexity, it assumes little or no hair animation and is not suitable for all hair styles. Wloka [Wlo04] uses a similar rendering approach for underwater hair which is animated by treating it as a particle system. Unfortunately, these techniques can only be used for a limited number of characters, since they are computationally intensive, and therefore simple texture-mapped triangles are typically used for an individual's skin and hair detail within large crowds (Figure 3).

2.3. Acceleration Techniques for Rendering Large-Scale Crowds

The requirement in interactive systems for real-time frame rates means that only a limited number of polygons can be displayed by the graphics engine in each frame of a simulation. Visibility culling techniques provide the first step to avoid rendering off-screen characters, and therefore reducing the number of triangles displayed per frame. However, other rendering techniques are needed since a large portion of the crowd could potentially be on-screen.

2.3.1. Visibility Culling

Culling provides a mechanism to reduce the number of triangles rendered per frame by not drawing what the viewer cannot see. The basic idea behind culling is to discard as many triangles as possible that are not visible in the final rendered image. The two main types are visibility and occlusion culling.

Visibility culling discards any triangles that are not within the camera's view-frustum. In the case of a large scenes

containing several thousand characters, it would be computationally expensive to view-frustum cull each character's triangles. However, it can be used to avoid rendering potentially off-screen characters by testing their *bounding-volumes* with respect to the view-frustum. For further details on various optimized view-frustum culling techniques utilizing bounding-volumes see [AM00].

The aim of occlusion culling is to quickly discard any objects that are hidden by other parts of the scene. Various research has been conducted on effective ways of establishing occluding objects utilizing software methods or 3-D graphics hardware. For a detailed survey of these techniques see [COCSD03]. For crowds populating a virtual city environment, occlusion culling is a method that can greatly improve the frame rate, since a large portion of the crowd will be occluded by buildings, especially when the viewpoint is at ground level.

2.3.2. Geometric-Based Rendering and Level of Detail

Level of detail (LOD) is an area of research that has grown out of the long-standing trade-off between complexity and performance. LOD stems from the work done by James Clark where the basic principles are defined [Cla76]. The fundamental idea behind LOD, is that when a scene is being simulated, it uses an approximate simulation model for small, distant, or important objects in the scene. The main area of LOD research has focussed on geometric LOD, which attempts to reduce the number of rendered polygons by using several representations of decreasing complexity of an object. For each frame, the appropriate model or resolution is selected, usually based on the object's distance to the camera. In addition to distance, other LOD selection factors that can be used are screen space size, priority, hysteresis, and perceptual factors. Since the work done by Clark [Cla76], the literature on geometric LOD has become quite extensive. Geometric LOD has been used since the early days of flight simulators, and has more recently been incorporated in walkthrough systems for complex environments by Funkhouser et al. [FST92, FS93], and Maciel et al. [Mac93].



Figure 4: Five discreet mesh models containing (a) 2,170 (b) 1,258 (c) 937 (d) 612 and (e) 298 triangles.

One approach for managing the geometric LOD of virtual

humans is using a discrete LOD framework. A discrete LOD framework involves creating multiple versions of an object's mesh, each at a different LOD, during an offline process (see Figure 4). Typically, a highly detailed (also known as a high resolution) mesh, is simplified by hand or using automatic tools to create multiple low resolution meshes varying in detail. At run-time, depending on the LOD selection criteria, the appropriate resolution mesh is chosen in order to maintain an interactive frame rate.

Another good solution for altering the geometric detail of a character in games is through the use of subdivision surfaces [Lee02]. In the beginning, one of the main problems with geometric LOD was the generation of the different levels of detail for each object, which was a time-consuming process as it was all done by hand. Since then, several LOD algorithms have been published in order to automatically generate the different levels of detail for an object [EDD⁹⁵, Hop96]. Subdivision surfaces is one method, based on a continuous LOD framework, where a desired level of detail is extracted at run-time by performing a series of edge collapsing/vertex splitting on the model. Starting with a low-resolution mesh, a subdivision scheme can be used to produce a more detailed version of the surface by using masks to define a set of vertices and corresponding weights, which are in turn used to create new vertices or modify existing ones. By applying these masks to the mesh's vertices, a new mesh can be generated. An advantage of using masks is that different type of masks can be used in order to deal with boundary vertices and crease generation. In [OCV⁰²], O'Sullivan et al. describe a framework that uses subdivision surfaces as a means to increase or decrease the appearance of a human's mesh within groups and crowds depending on their importance to the viewer.

In order to solve the problem of rendering large numbers of humans, De Heras Ciechowski et al. [dHCUCT04] avoid computing the deformation of a character's mesh by storing pre-computed deformed meshes for each key-frame of animation, and then carefully sorting these meshes to take cache coherency into account. Ulicny et al. [UdHCT04] improve on their performance by using 4 LOD meshes consisting of 1038, 662, 151 and 76 triangles and disabling lighting for the lowest LOD, thereby achieving a frame rate several times higher. To introduce crowd variety, they use several template meshes for the humans, and clone and modify these meshes at run-time by applying different textures, colors, and scaling factors to create the illusion of variety. They succeed in simulating several hundred humans populating an ancient Roman theatre and a virtual city at interactive frame-rates.

Gosselin et al. [GSM05] present an efficient technique for rendering large crowds while taking variety into account. Their approach involves reducing the number of API calls need to draw a character's geometry by rendering multiple characters per draw call, each with their own unique animation. This is achieved by packing a number of instances of



Figure 5: Rendering crowds using a discrete LOD approach [dHCUCT04].

character vertex data into a single vertex buffer and implementing the skinning of these instances in a vertex shader. As vertex shading is generally the bottleneck of such scenes containing a large number of deformable meshes, they minimize the number of vertex shader operations that need to be performed.

In their simulation, they use one directional light to simulate the sun, and three local diffuse lights. The shading of each character's mesh is performed by per-pixel shading and a normal map generated from a high resolution model is used. Specular lighting is calculated for the sun and is attenuated using a gloss map to allow for parts of the character to have differing shininess. Realism is further increased by using an ambient occlusion map generated from the high resolution model. This map approximates the amount of light that could reach the model from the external lighting environment and provides a realistic soft look to the character's illumination. Finally, using a ground occlusion texture which represents the amount of light a character should receive from the sun based on their position in the world, the illusion that the terrain is shading the characters as they move within the environment is created. So that the characters are not a carbon copy of each other, they use a colour lookup texture, which specifies 16 pairs of colours that can be used to modulate the character, with a mask texture to specify which portions should be modulated. In addition to this, decal textures to add other various details to the character's model, such as badges, are applied (see Figure 6).

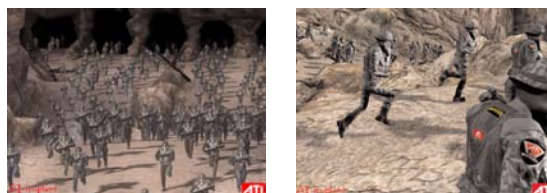


Figure 6: Geometric-based representations rendered with various per-pixel shading effects [GSM05] (© 2005 ATI Technologies).

While Gosselin et al. provide techniques to improve the rendering performance of multiple character meshes, the

crowd is homogeneous in nature since it is made of individuals that are using the same template model and are animated with the same running motion. Recently, Dudash et al. [Dud07] has extended this work and provided an efficient way of adding variation to the crowd's animation and appearance. Their approach involves using instancing through the DirectX 10 API in an attempt to reduce the number of draw calls, state changes and buffer updates.

They achieve mesh variation by breaking a character into a collection of mesh sub-sections. For each character, each sub-section is initialised with an alternate mesh randomly selected from multiple template meshes at load time. At run-time, they make a list for each sub-mesh containing the per-instance data of the characters that are using that particular piece and then draw each instance. To provide for a more heterogeneous crowd animation, they avoid the usual technique of using the limited number of shader constants for the animation data. Instead, they encode all the animations' frame data into a texture, from which the vertex shader looks up the bone matrices for each character's current frame of animation. In this way, their characters can perform any frame of any animation defined in the texture. Additionally, they implement a discrete LOD system where characters in the distance use mesh sub-sections of lower resolution.

2.3.3. Image-Based Crowd Rendering

Image-based rendering (IBR), stems from the research by Maciel et al. [MS95] on using texture mapped quadrilaterals, referred to as planar impostors, to represent objects in order to maintain an interactive frame rate for the visual navigation of large environments. Consequently, due to this planar impostor providing a good visual approximation to complex objects at a fraction of the rendering cost, a large amount of research has introduced different types of impostors such as layered impostors [DSSD99], billboard clouds [DDSD03], and texture depth images [JW02] for rendering acceleration of various applications. A survey of these different types, including their application and their advantages and disadvantages, can be found in [JWP05]. To represent a virtual human, Tecchia et al. [TC00] and Aubel et al. [ABT00] both use planar impostors. However, they differ in how the impostor image is generated. The two main approaches to the generation of the impostor images are: dynamic generation and static generation (also referred to as pre-generated impostors).

Aubel et al. use a dynamically generated impostor approach to render a crowd of 200 humans performing a 'Mexican wave' [ABT00]. With dynamically generated impostors, the impostor image is updated at run-time by rendering the object's mesh model to an off-screen buffer and storing this data in the image. This image is displayed on a quadrilateral, which is dynamically orientated towards the viewpoint. This uses less memory, since no storage space is devoted to any impostor image that is not actively in use. Unlike dynamically generated impostors for static objects, where the



Figure 7: Image-based crowds: (a) Dynamically generated image-based crowds [ABT00] (b) Pre-generated image-based crowds [TC00].

generation of a new object impostor image depends solely on the camera motion, animated objects such as a virtual human's mesh also have to take self-deformation into account. Aubel et al.'s solution to this problem is based on the sub-sampling of motion. By simply testing distance variations between some pre-selected joints in the virtual human's skeleton, the virtual human is re-rendered if the posture has significantly changed.

The planar nature of the impostor can cause visibility problems as a result of it interpenetrating other objects in the environment. To solve this problem, Aubel et al. propose using a multi-plane impostor which involves splitting the virtual human's mesh into separate body parts, where each body part has its own impostor representation. However, this approach can cause problems similar to those mentioned in Section 3, resulting in gaps appearing. Unfortunately, dynamically generated impostors rely heavily on reusing the current impostor image over several frames in order to be efficient, as animating and rendering the human's mesh off-screen is too costly to perform regularly. Therefore, this approach does not lend itself well to scenes containing large dynamic crowds, as this would require a coarse discretization of time, resulting in jerky motion.

Tecchia et al. [TC00] use pre-generated impostors for rendering several thousand virtual humans walking around a virtual city at an interactive frame rate. Pre-generated impostors involve the pre-rendering of an impostor image of an object for a collection of viewpoints (called reference viewpoints) around the object. Unfortunately, since virtual humans are animated objects, they present a trickier problem in comparison to static objects. As well as rendering the virtual human from multiple viewpoints, multiple key-frames of animation for each viewpoint need to be rendered, which greatly increases the amount of texture memory used. In order to reduce the amount of texture memory consumed, Tecchia et al. reduce the number of reference viewpoints needed for each frame by using a symmetrical mesh representation animated with a symmetrical walk animation, so that already generated reference viewpoints can be mirrored to generate new viewpoints. At run-time, depending on the viewpoint with respect to the human, the most appropriate refer-

ence viewpoint is selected and displayed on a quadrilateral, which is dynamically orientated towards the viewer. To allow for the dynamic lighting of the impostor representation, Techia et al. [TLC02] pre-generate normal map images for each viewpoint by encoding the surface normals of the human's mesh as a RGB colour value. By using a per-pixel dot product between the light vector and a normal map image, they compute the final value of a pixel through multi-pass rendering and require a minimum of five rendering passes.

The main advantage of this approach is that it is possible to deal with the geometric complexity of an object in a pre-processing step. However, with pre-generated impostors, since the object's representation is fixed, 'popping' artefacts are introduced as a result of being forced to approximate the representation for the current viewpoint with the reference viewpoint. To avoid these artefacts, the number of viewpoints around the object for the pre-generation of the impostor images can be increased. However this can later cause problems with the consumption of texture memory. Image warping is another technique of reducing the popping effect, but this method can also introduce its own artefacts. Since a pre-generated approach requires a large number of reference viewpoints for several frames of animation, this makes it unsuitable for scenes containing a variety of human models that each needs to perform a range of different motions.

Dobbyn et al. [DH005] developed the Geopostor system, which provides for a hybrid combination of pre-generated impostor and detailed geometric rendering techniques for virtual humans. By switching between the two representations, based on a "pixel to texel" ratio, their system allows visual quality and performance to be balanced. They improved on existing impostor rendering techniques and developed a programmable hardware based method for adjusting the lighting and colouring of the virtual humans' skin and clothes (see Figure 8).



Figure 8: Geopostor system.

Recently, Millán et al. [MR06b] described a LOD system which takes advantage of existing programmable graphics hardware in order to improve the simulation and rendering performance of their crowd system. They simulate several hundred thousand characters in real-time by storing each character's position and orientation in a pixel buffer which is updated by a fragment program. Once the pixel buffer is updated, this data is subsequently used by graphics hardware

to render the characters using a particular representation selected based on a LOD map. The LOD map is a 2D grid rendered on a per-frame basis, where each pixel represents a position in the world and stores a specific encoded value representing its distance from the camera. Once the LOD map is generated, it is stored in a pixel buffer which is looked up by the vertex processor to select a LOD representation for each character instance. Similar to Dobbyn et al. [DH005], they use a geometry/impostor based LOD system. However, they use the vertex processor to rotate each impostor's billboard towards the camera view and calculate the texture coordinates of the most suitable viewpoint to be mapped.

2.3.4. Point Sample Rendering

Another sampled-based approach for the visualisation of virtual humans is point sample rendering, which involves replacing a mesh with a cloud of points, approximately pixel-sized [LW85]. Wand et al. [WS02] use a pre-computed hierarchy of triangles and sample points to represent a scene. This involves converting key-frame animations of meshes into a hierarchy of point samples and triangles at different resolutions. They partition the scene's triangles using an octree structure and choose sample points which are distributed uniformly on the surface area of the triangles in each node. Using this multi-resolution data structure, they are able to render large crowds of animated characters.



Figure 9: Point-based crowds: (a) Stadium populated with animated 16,000 fans and (b) Crowd of 90,000 humans walking on the spot.

For smaller crowds, consisting of several thousands of objects, each object is represented by a separate point sample and its behaviour is individually simulated. Larger crowds are handled differently, with a hierarchical instantiation scheme, which involves constructing multi-resolution hierarchies (e.g., a crowd of objects) out of a set of multi-resolution sub-hierarchies (e.g., different animated models of single objects). While this allows them to render arbitrarily complex scenes, such as 90,000 humans walking on the spot and a football stadium inhabited by 16,000 fans (see Figure 9), less flexibility is provided for the motion of the objects, since the hierarchies are pre-computed and therefore cannot be used in simulating a large crowd moving within its environment. For a comparison between point-based models and impostors see [MR06a].

3. Character Animation

The problem with using a mesh to represent a dynamic object, such as human character, is that a way of animating the mesh is needed to reflect the motion of the character. In older generation games, the character consisted of a hierarchy of meshes, where each mesh represented a particular body part and was animated in some way (e.g., Lara Croft in Tomb Raider). However, the main problem with this approach is that holes can appear where two or more meshes meet. These gaps can be hidden either by clever modelling using clothing or armour, at the cost of requiring extra polygonal detail, or by constraining the movement of the bones. However, depending on the type of character being modelled, this is not always possible. Nowadays, a character's mesh is typically animated by using a layered animation approach.

3.1. Layered Animation

The layered animation approach works by layering a character's mesh on top of a skeleton structure and deforming the mesh based on the animation of the underlying skeletal layer. The skeleton consists of a hierarchy of joints interconnected by bones, where each joint defines where a bone begins and is used as its pivot point. Except for the bone at the root of the hierarchy (known as the *root bone*), each bone is linked to a parent bone and has either one, multiple, or no child bones. To easily transform a bone from one coordinate space to another, each bone's position and rotation is stored in a transformation matrix. The global transformation matrix of each bone is dependent on the matrices of all of its parents, and can be calculated as a function of both its local and parent's global transformation matrices.

In order to deform the mesh, the mesh and the skeleton first need to be setup in a reference pose, typically using DaVinci's Vitruvian man pose, to facilitate their respective alignment. Each vertex in the mesh is assigned either one or more influencing bones with a corresponding weight to specify the amount of influence each bone has on it. Linear blend skinning (LBS) is used for deforming the mesh [Lan98, Lan99], where the deformation of each vertex's position (V') and normal (N') is calculated as a function of the vertex's original position relative to each deforming bone (V_i), its normal (N), each deforming bone's global transformation matrix (TM_i) and its influencing weight (w_i) (Equation 1). When calculating the deformation of the normals, only the rotational component is used by getting the inverse transpose of the global transformation matrix ($(TM_i^{-1})^T$).

$$\begin{aligned} V' &= \sum w_i \times TM_i \times V_i \\ N' &= \sum w_i \times (TM_i^{-1})^T \times N \end{aligned} \quad (1)$$

Linear blend skinning can be implemented through programmable graphics hardware by using a vertex program

and this greatly improves its performance [Dom, GSM05]. This technique is fast to compute and therefore has become widespread in recent games. While problems can arise for large bone rotations, causing the mesh to collapse to a single point, this can be solved by adding extra bones [Web00], or using spherical blend skinning [KZ05].

3.2. Animation of a Character's Skeleton

Traditionally, an articulated structure, such as a skeleton, is animated using computer animation data stored as key-frames. A key-frame allows the transformation of a bone (i.e., its position and rotation) to be specified as a function of time. This allows complicated animations to be simply stored as a set of key-frames for each bone. While the most simple method of generating key-frame animations for articulated structures is through kinematics, extensive research on providing other ways of generating animation data has been carried out, focusing on physical simulation and procedural animation.

3.2.1. Kinematics

A common method for animating an articulated structure in real-time is with kinematics, which is based on properties of motion such as position and velocity over time. A character's key-frame animation is typically generated from data that has been created manually through kinematics by an animation artist using a key-frame editor.

Forward kinematics specifies joint rotations as a function of time and is useful in pre-generating character animations in modeling/animation packages, such as 3D Studio Max. Once the animation has been created, it can be subsequently exported as key-frame data to be used within an application. Motion capture systems allow the movements of a real actor to be captured or stored as animation data by using different types of capture hardware and this was the predominant method for animating characters in *The Lord of the Rings Trilogy* [Sco03]. While the quality and realism of manually created animations depends on the skill of the artist, motion captured animations are extremely realistic as a result of using a real human actor. With regards to animating crowds, the main limitation of forward kinematics is that a large database of pre-generated or pre-captured motions is necessary in order to achieve some type of variation amongst the crowd. Otherwise, a crowd consisting of individuals performing the same animation can significantly reduce realism.

Inverse kinematics can resolve the skeleton's joint angles and the corresponding key-frame data so that an end-effector (e.g., the hand bone) is animated towards a target position. The main advantage of this is that it can be used for the real-time generation of various character animations (e.g., pointing in a particular direction, looking at an object and opening a door). Several algorithms exist to resolve the joint

angles with varying computational accuracy of the results, the majority of which can be used with groups of characters in real-time. The main limitation of this technique is that, even though it generates a correct solution, it might not be a high-fidelity human motion.

3.2.2. Physically-Based Animation

Physically-based animation provides a good approach to generating unique and context-sensitive motion and in theory can produce an unlimited number of motion types. However, the problem with using the approach is that it is computationally intensive algorithms and the generated animation is somewhat dependent on various character properties. Therefore, this type of animation is not easily reusable and thus not well-suited for the real-time animation of a large number of characters of various shapes and sizes. Dynamic simulations use Newtonian force-based methods to generate animations utilizing forces that occur in articulated structures (e.g., velocities, mass, collision), in addition to kinematic properties. Physically-based animations have been used for animating virtual athletes in realistic sport simulations [HWBO95], generating physically correct swimming motion for fish [TT94], and characters walking on an uneven terrain [SM01].

3.2.3. Procedural Animation

Procedural algorithms reuse animation data from a library of motions to generate new animations. The two main approaches are combining, and altering animation data. Combining animations involves reusing animations with various techniques such as fading functions, overlapping and blending techniques. Various research has been conducted on providing smooth transitions between motions, such as the simple use of fade-in and fade-out functions [PG96, RCB98] and the more complex weighting and summing techniques [SBMTT99]. Perlin et al. [PG96] reuse and overlap animations by considering human motions as a “combination of temporarily overlapping gestures and stances”. In general, combining animation data provides a good and fast approach for animating characters in real-time applications. However, to allow for some variation, it is important that there is a large library of pre-generated motions that can produce plausible combinations. Motion graphs can be compiled, which are directed graphs that describe how motion may be recombined, to automatically generate transitions to connect motions. The motion graph is generated from the library by identifying similar frames between each pair of motions and using these to form the nodes of the graph. These nodes provide plausible transitions between motions and allow the character to perform more complicated performances [KGP02].

The second approach to procedural animation involves altering the style of animation data based on various techniques such as noise functions [PG96], and emotional transforms based on character-based properties [ABC96]. Even

though more realistic and less repetitious animations are produced by altering the data, these techniques can be computationally intensive and should only be considered for the real-time animation of a limited number of characters.

3.3. Animation Level of Detail

LOD research has recently extended from the area of geometry into areas such as motion and simulation, thus providing a computationally efficient solution for the simulation of crowds. In [GMPO00], Giang et al. propose a LOD framework for animating and rendering virtual humans in real-time. In order to achieve a scalable system, they use a LOD resolver that controls the switching between levels of detail and specifies parameters for controlling the geometric and motion controller. Through these parameters, the LOD resolver has the ability to request different animation levels of detail. The different levels of detail used relate to how the motion is simulated (e.g., pre-defined forward kinematics, inverse kinematics, or dynamics), and its update frequency. This results in smooth realistic animations being applied to virtual humans rated with high importance, while lower level animation techniques are applied to virtual humans in the background, taking minimal perceptual degradation into account.

In [dHCUCT04], the deformation of a character's mesh was pre-computed and stored to avoid these computations at run-time. However, these characters were limited to the number of animations they could perform due to the size limit of memory. To improve on their previous system, in [dHCSMT05] they propose rendering crowds animated using the layered animation approach (see Section 2.3.2) to reduce the consumption of memory and accelerate the animation of the skeleton and the subsequent mesh deformation using a level of detail caching scheme for animations and geometry. They update a character's animation at a specific frequency dependent on its level of detail instead of on a per-frame basis. For example, characters are updated at a minimum of 4Hz at the lowest LOD and at a maximum of 50Hz at the highest LOD, where the LOD selection criteria is based on the character's distance from the camera. The animation of the skeleton and the subsequent mesh deformation are done in software so that they can be reused in a caching scheme.

Ahn et al. [AOW06] detail a motion simplification framework for articulated characters which attempts to speed up animation performance through posture simplification whilst conserving the features of the original motion. The framework involves extracting key postures from a motion and then automatically generating the priority of joint reductions in order to guide the posture simplification process. The experimental results shows that the proposed motion simplification can be successfully applied to a crowd of a thousand articulated characters in real-time.

3.4. Simulation Level of Detail

In [CH97], Carlson and Hodgins use less accurate animation models for selected one-legged creatures in order to reduce the computational cost of simulating groups of these creatures. Three simulation LODs are used for the motion of these creatures: rigid-body dynamics, point mass simulation with kinematic joints and point mass simulation with no kinematic motion of the leg. Their selection of an individual's simulation LOD is based on a individual's importance to the viewer or action in the virtual world.

Ulicny et al. [UT02] discuss the challenges of real-time crowd simulations, focussing on the need to efficiently manage variety, and propose the idea of *levels of variety*. They define a system's variety based on the following levels: level of variety zero (LV0) if a task uses a single solution, level of variety one (LV1) if it has a choice from a finite number of solutions, and level of variety two (LV2) if it is able to use an infinite number of possible solutions. For example, a crowd composed of a single human model would be LV0, several pre-defined model types would be LV1, and finally an infinite number of automatically generated model types would be LV2. Using this concept, they define a modular behavioural architecture based on rules and finite state machines, to provide simple yet sufficiently variable behaviours for individuals in a crowd.

References

- [ABC96] AMAYA K., BRUDERLIN A., CALVERT T.: Emotion from motion. *GI '96: Proceedings of the Conference on Graphics Interface* (1996), 222–229.
- [ABT00] AUBEL A., BOULIC R., THALMANN D.: Real-time display of virtual humans: Levels of details and impostors. *IEEE Transactions on Circuits and Systems for Video Technology* 10, 2 (2000), 207–217.
- [AM00] ASSARSSON U., MÖLLER T.: Optimized view frustum culling algorithms for bounding boxes. *Journal of Graphics Tools: JGT* 5, 1 (2000), 9–22.
- [AOW06] AHN J., OH S., WOHN K.: Optimized motion simplification for crowd animation. *Computer Animation and Virtual Worlds* 17, 3-4 (2006), 155–165.
- [BN76] BLINN J. F., NEWELL M. E.: Texture and reflection in computer generated images. *Communications of the ACM* 19, 10 (1976), 542–546.
- [Cat74] CATMULL E. E.: *A subdivision algorithm for computer display of curved surfaces*. PhD thesis, Dept. of Computer Science, University of Utah, December 1974.
- [CH97] CARLSON D. A., HODGINS J. K.: Simulation levels of detail for real-time animation. *GI '97: Proceedings of the Conference on Graphics Interface* (1997), 1–8.
- [Cla76] CLARK J. H.: Hierarchical geometric models for visible surface algorithms. *Communications of the ACM* 19, 10 (1976), 547–554.
- [COCS03] COHEN-OR D., CHRYSANTHOU Y., SILVA C. T., DURAND F.: A survey of visibility for walk-through applications. *IEEE Transactions on Visualization and Computer Graphics* 9, 3 (2003), 412–431.
- [COM98] COHEN J., OLANO M., MANOCHA D.: Appearance-preserving simplification. *SIGGRAPH '98: Proceedings of the 25th Annual Conference on Computer Graphics and Interactive techniques* (1998), 115–122.
- [DDSD03] DÉCORET X., DURAND F., SILLION F., DORSEY J.: Billboard clouds for extreme model simplification. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 689–696.
- [dHCSMT05] DE HERAS CIECHOMSKI P., SCHERTENLEIB S., MAÏM J., THALMANN D.: Reviving the roman odeon of aphrodisias: Dynamic animation and variety control of crowds in virtual heritage. *VSM '05: Proceeding of the 11th International Conference on Virtual Systems and Multimedia* (2005), 601–610.
- [dHCUCT04] DE HERAS CIECHOMSKI P., ULICNY B., CETRE R., THALMANN D.: A case study of a virtual audience in a reconstruction of an ancient roman odeon in aphrodisias. *VAST '04: The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heirtage* (2004), 9–17.
- [DHOO05] DOBBYN S., HAMIL J., O'CONNOR K., O'SULLIVAN C.: Geopostors: A real-time geometry / impostor crowd rendering system. *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (April 2005), 95–102.
- [Dom] DOMINÉ S.: Mesh skinning. <http://developer.nvidia.com/object/skinning.html>.
- [Don05] DONNELLY W.: *GPU Gems 2 - Per-Pixel Displacement Mapping with Distance Functions*. Addison-Wesley, 2005, pp. 123–136.
- [DSSD99] DÉCORET X., SCHAUFLE G., SILLION F., DORSEY J.: Multi-layered impostors for accelerated rendering. *Computer Graphics Forum (Proceedings of Eurographics '99)* 18, 3 (1999), 61–73.
- [Dud07] DUDASH B.: Skinned instancing, NVIDIA Corporation. <http://developer.download.nvidia.com/SDK/direct3d/samples.html> (2007).
- [EDD*95] ECK M., DEROSE T., DUCHAMP T., HOPPE H., LOUNSBURG M., STUETZLE W.: Multiresolution analysis for arbitrary meshes. *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), 173–182.
- [FS93] FUNKHOUSER T. A., SÉQUIN C. H.: Adaptive display algorithm for interactive frame rate during visualisation of complex virtual environments. *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), 247–254.
- [FST92] FUNKHOUSER T. A., SÉQUIN C. H., TELLER

- S.: Management of large amounts of data in interactive building walkthroughs. *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics* (1992), 11–20.
- [GMPO00] GIANG T., MOONEY R., PETERS C., O'SULLIVAN C.: Aloha: Adaptive level of detail for human animation towards a new framework. *Eurographics 2000 Short Paper Programme* (2000), 71–77.
- [Gou71] GOURAUD H.: Continuous shading of curved surfaces. *IEEE Transactions on Computers* 20, 6 (1971), 623–628.
- [GSM05] GOSSELIN D., SANDER P., MITCHELL J.: *ShaderX3 - Drawing a Crowd*. Charles River Media, 2005, pp. 505–517.
- [Hop96] HOPPE H.: Progressive meshes. *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), 99–108.
- [HWBO95] HODGINS J. K., WOOTEN W. L., BROGAN D. C., O'BRIEN J. F.: Animating human athletes. *SIGGRAPH '95: Proceedings of the 22nd Annual Conference on Computer Graphics and Interactive Techniques* (1995), 71–78.
- [JW02] JESCHKE S., WIMMER M.: Textured depth meshes for real time rendering of arbitrary scenes. *EGRW '02: Proceedings of the 13th Eurographics Workshop on Rendering* (2002), 181–190.
- [JWP05] JESCHKE S., WIMMER M., PURGATHOFER W.: Image-based representations for accelerated rendering of complex scenes. In *Eurographics 2005 STAR Reports* (2005), 1–20.
- [KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 473–482.
- [Kil00] KILGARD M. J.: *A Practical and Robust Bump-mapping Technique for Today's GPUs*. Tech. rep., NVIDIA Corporation, 2000.
- [KK89] KAJIYA J. T., KAY T. L.: Rendering fur with three dimensional textures. *SIGGRAPH '89: Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques* (1989), 271–280.
- [KZ05] KAVAN L., ZARA J.: Spherical blend skinning: A real-time deformation of articulated models. *SI3D '05: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games* (2005), 9–16.
- [Lan98] LANDER J.: Skin them bones. *Game Developer Magazine* (May 1998), 11Ü–16.
- [Lan99] LANDER J.: Over my dead, polygonal body. *Game Developer Magazine* (October 1999), 17Ü–22.
- [Lee02] LEESON W.: *Games Programming Gems III - Subdivision Surfaces for Character Animation*. Charles River Media, 2002, pp. 372Ü–383.
- [LW85] LEVOY M., WHITTED T.: *The Use of Points as a Display Primitive*. Tech. rep., University of North Carolina at Chapel Hill, 1985.
- [Mac93] MACIEL P.: *Visual Navigation of largely unoccluded environments using textured clusters*. PhD thesis, Indian University, Bloomington, January 1993.
- [Map] MAPPER A. N.: <http://www.ati.com>.
- [MJC*03] MARSCHNER S. R., JENSEN H. W., CAMMARANO M., WORLEY S., HANRAHAN P.: Light scattering from human hair fibers. *ACM Transactions on Graphics (TOG)* 22, 3 (2003), 780–791.
- [MR06a] MILLÁN E., RUDOMÍN I.: A comparison between impostors and point-based models for interactive rendering of animated models. In *In Proceedings International Conference in Computer Animation and Social Agents (CASA) 2006* (2006).
- [MR06b] MILLÁN E., RUDOMÍN I.: Impostors and pseudo-instancing for gpu crowd rendering. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (2006), pp. 49–55.
- [MS95] MACIEL P., SHIRLEY P.: Visual navigation of large environments using textured cluster. *SI3D '95: Proceedings of the 1995 Symposium on Interactive 3D Graphics* (1995), 95–102.
- [OCV*02] O'SULLIVAN C., CASSELL J., VILHJÁLMS-SON H., DINGLIANA J., DOBBYN S., MCNAMEE B., PETERS C., GIANG T.: Levels of detail for crowds and groups. *Computer Graphics Forum* 21, 4 (2002), 733–742.
- [Ope04] OPENGL S. G. I.: The OpenGL Graphics System: A Specification. <http://www.opengl.org/documentation/specs/version2.0/glspec20.pdf> (October 2004).
- [PG96] PERLIN K., GOLDBERG A.: Improv: a system for scripting interactive actors in virtual worlds. *SIGGRAPH '96: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (1996), 205–216.
- [RCB98] ROSE C., COHEN M. F., BODENHEIMER B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics Applications* 18, 5 (1998), 32–40.
- [SBMTT99] SANNIER G., BALCISOY S., MAGNENAT-THALMANN N., THALMANN D.: Vhd: a system for directing real-time virtual actors. *The Visual Computer* 15, 7/8 (1999), 320–329.
- [Sch04] SCHEUERMANN T.: Practical real-time hair rendering and shading. *SIGGRAPH 2004 Sketch* (2004).
- [Sco03] SCOTT R.: Sparking life: notes on the performance capture sessions for the *Lord of the Rings: the Two*

- Towers. *SIGGRAPH Computer Graphics* 37, 4 (2003), 17–21.
- [SGM04] SANDER P., GOSSELIN D., MITCHEL J.: Real-time skin rendering on graphics hardware. *SIGGRAPH 2004 Sketch* (2004).
- [SKvW*92] SEGAL M., KOROBKIN C., VAN WIDENFELT R., FORAN J., HAEBERLI P.: Fast shadows and lighting effects using texture mapping. *SIGGRAPH '92: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* 26, 2 (1992), 249–252.
- [SM01] SUN H. C., METAXAS D. N.: Automating gait generation. *SIGGRAPH '01: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques* (2001), 261–270.
- [TC00] TECCHIA F., CHRYSANTHOU Y.: Real-time rendering of densely populated urban environments. *Proceedings of the Eurographics Workshop on Rendering Techniques* (2000), 83–88.
- [TLC02] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Visualizing crowds in real-time. *Computer Graphics Forum* 21, 4 (2002), 753–765.
- [TT94] TU X., TERZOPOULOS D.: Artificial fishes: Physics, locomotion, perception, behavior. *SIGGRAPH '94: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (1994), 43–50.
- [UdHCT04] ULICNY B., DE HERAS CIECHOMSKI P., THALMANN D.: Crowdbush: Interactive authoring of real-time crowd scenes. *SCA '04: Proceedings of the 2004 ACM SIGGRAPH/EUROGRAPHICS Symposium on Computer Animation* (2004), 243–252.
- [UT02] ULICNY B., THALMANN D.: Towards interactive real-time crowd behaviour simulation. *Computer Graphics Forum* 21, 4 (2002), 767–775.
- [Web00] WEBER J.: Run-time skin deformation. In *Proceedings of Game Developers Conference* (2000).
- [Wlo04] WLOKA M.: Advanced rendering techniques. *EUROGRAPHICS 2004 Tutorial* (2004). http://developer.nvidia.com/object/eg_2004_presentations.html.
- [WS02] WAND M., STRASSER W.: Multi-resolution rendering of complex animated scenes. *Computer Graphics Forum* 21, 3 (2002), 483–491.

Optimising and Evaluating the Realism of Virtual Crowds: Perceptual Experiments and Metrics

R. McDonnell, S. Dobbyn, and C. O'Sullivan

Graphics, Vision & Visualisation Lab (GV2), Trinity College Dublin, Ireland

Abstract

Usually developers of real-time crowd systems decide on the virtual human representation they will use based on three factors: the size of the crowd being rendered, each representation's rendering cost and its visual appeal. While there has been extensive research on the numerous ways of graphically representing virtual humans (including their associated rendering cost), only recently have researchers become interested in perceptually evaluating them. Evaluating these representations based on the plausibility of visual appearance and motion would provide a useful metric to help developers of LOD-based crowd systems to improve visual realism while maintaining real-time frame rates. With regards to improving our crowd system, we carried out perceptual evaluation experiments on various virtual human representations using experimental procedures from the area of psychophysics.

Introduction

While there has been little previous work related to the perception of virtual human representations [HMDO05, MDO05], research has been conducted on perception of human motion in the context of computer graphics and has mainly been focused on the effect of animation quality on user perception. Wang et al. [WB03] conducted a set of experiments to evaluate a cost function proposed by Lee et al. [LCR02] for determining the transition quality between motion clips. Other recent work by Harrison et al. [HRvdP04] examined the perceptual impact of dynamic anomalies in human animation. Reitsma and Pollard [RP03] conducted a study, developing a metric to evaluate the perceived error introduced during motion editing. Harrison et al. [HBF02] focused on higher-level techniques for specifying and modifying human motions. Oesker et al. [OHJ00] investigated the extent to which observers perceptually process the LOD in naturalistic character animation. The study most related to our work is by Hodgins et al. [HOT98]. They performed a series of perceptual experiments, the results of which indicated that a viewer's perception of motion characteristics is affected by the geometric model used for rendering. Participants were shown a series of paired motion sequences and asked if the two motions in each pair were the "same" or "different". The motion sequences in each pair were rendered using the same geometric model. For the three types of motion variation tested, sensitivity scores indicated

that subjects were better able to observe changes when viewing the polygonal model than they were with a stick figure model.

With the goal of improving the realism of our crowd system, we carried out the following four sets of perceptual experiments:

1. Perception of Human Appearance

Experiment 1: Impostor Vs. Mesh Detection

At what distance can experiment participants detect that a virtual human is using an impostor or mesh representation?

Experiment 2: Low Vs. High-Resolution Mesh Discrimination

At what distance and at what resolution can experiment participants discriminate between a high resolution and low resolution mesh representation?

Experiment 3: Impostor/Mesh Switching Discrimination

At what distance can experiment participants detect an impostor switching to a mesh?

2. Perception of Motion

Experiment 4: Perception of Human Motion

How well do different virtual human representations replicate motion?

Experiment 5: Perception of Cloth Motion

How well do different representations replicate deformable clothing?

Experiment 6: Applying Motion Capture
Is the sex of a motion captured actor important when applying his/her motion to virtual characters?

3. Perceptual Metrics for Smooth Animation

Experiment 7: Impostor Update Frequency
What is the optimal sampling rate (i.e, the number of viewpoints) for impostors?

Experiment 8: Animation Update Frequency
How many pose changes per second are needed for smooth animation?

Experiment 9: Simulation Level of Detail
Can we save memory by displaying foreground characters at higher update rates than background characters, with no loss of visual fidelity?

4. Evaluation of Metrics

Experiment 10: Impostor Update and Mesh Detection Metric Evaluation

To evaluate the effectiveness of the metrics discovered in Experiment 1, 3 and 7 by placing the characters in crowds of different sizes.

Psychophysics

We will begin with a broad overview of some psychophysical techniques that were used for gathering information for the experiments that we performed.

Psychophysics is the science of human sensory perception and is used to explore two general perceptual problems involving the measurement of sensory thresholds: discrimination and detection [LHEJ01]. Discrimination is the ability to tell two stimuli apart, where each differ by a small amount, usually along a single dimension. Detection is a special case of the discrimination problem, where the reference stimulus is a null stimulus. Typically, both perceptual problems can be investigated using either a classical *yes-no* or a *forced choice* experiment design [Tre95]. A *yes-no* design involves experiment participants deciding on whether the stimuli are the "same" (no response) or "different" (yes response) while forced choice designs consist of the participant identifying a specific target stimulus given a number of choices. Using these designs, the participants responses for each stimulus level can be collected and analysed to estimate discrimination or detection thresholds. In order to measure these thresholds, the participant's cumulative responses are plotted as a graph of *percentage yes* responses (using a *yes-no* design) or *percentage correct* responses (using a forced choice design) for each stimulus level. An S-shaped curve termed a *Psychometric Function* is fitted to the cumulative responses, where the percentage yes or percentage correct is plotted as a function of stimulus.

For a *yes-no* design, the sensitivity threshold is specified by the stimulus intensity required for a person to reach a 50% *yes* point i.e., the point where same and different responses are equally likely. This threshold is known as the *Point of*

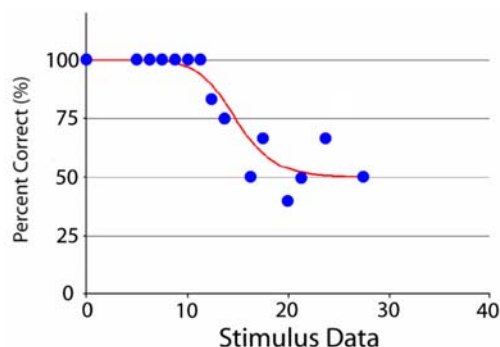


Figure 1: An Ogive function fitted to a participant's data for a *yes-no* design.

Subjective Equality (PSE). For this design, a simple Ogive inverse normal distribution function (see Equation 1) can be used to plot a curve that fits the participant's data (shown in Figure 1) and, from this curve, the PSE can be estimated as the 50% point and calculated using Equation 2. The inverse normal distribution function computes the stimulus intensity (x) for a given probability (P).

$$P_{Ogive}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \quad (1)$$

where : σ is the mean,
 μ is the standard deviation, and
 μ^2 is the variance.

$$PSE_{Ogive} = P_{Ogive}(0.5) \quad (2)$$

For forced choice designs, the threshold is often chosen as a halfway point between chance and 100% correct [Tre95]. For example, for a two alternative forced choice (2AFC) paradigm, the target stimulus is one of two choices. Therefore, the sensitivity threshold is the midpoint between chance (50% point in the case of 2 choices) and 100% correct, which is the 75% point. For experimental data using a 2AFC paradigm, a logistical function is normally used to fit a suitable curve to the participant's data and estimate the PSE using the 75% point. In our experiments we use a slightly modified version of the logistical function (given in Equation 3). The PSE for an experiment using a 2AFC design can be calculated using Equation 4.

$$P_{Logistic}(x) = 1 - \gamma\left(\frac{1}{1 + (\frac{x}{\alpha}) - \beta}\right) \quad (3)$$

where : α is the stimulus at the halfway point,
 β is the steepness of the curve, and
 γ is the probability of being correct by chance.

$$PSE_{Logistic} = P_{Logistic}(0.75) \quad (4)$$

Another interesting threshold that can be estimated from these curves is the *difference threshold* or the *just noticeable difference* (JND). The JND is the smallest difference in intensity required for a person to distinguish two stimuli and this can be estimated as the amount of additional stimulus needed to increase a participant's detection rate from 50% to 75% (for a yes-no design) or from 75% to 87.5% (for a 2AFC design) on the fitted psychometric function. Equation 5 and Equation 6 are used to calculate the JND for an experiment using a yes-no and 2AFC experiment, respectively. Finally, ANalysis of Variance (ANOVA) is used to test the null hypothesis that two means are equal. The null hypothesis is rejected if there are significant differences between the means.

$$JND_{Ogive} = P_{Ogive}(0.75) - P_{Ogive}(0.5) \quad (5)$$

$$JND_{Logistic} = P_{Logistic}(0.875) - P_{Logistic}(0.75) \quad (6)$$

The main problem with measuring thresholds of perception is that participants do not always respond in the same way when presented with identical stimuli in an ideal, noise-free experimental setup. This is mainly due to the fact that the neurosensory system is somewhat noisy, but other reasons such as attentional differences, learning, and adaptation to the experimental setup can also have an effect. To reduce some of these problems, many psychophysical techniques for collecting data have been developed [Tre95]. With regards to our experiments, we use a staircase experimental procedure.

A simple *up-down staircase* procedure involves setting the stimulus level to a pre-defined intensity and presenting the stimulus to the participant [Cor62, Lev71]. Depending on the participant's response, the stimulus level is decreased (for a positive response) or increased (for a negative response) by a fixed amount or *step-size* and the altered stimulus is presented to the participant again. The experiment is terminated once the participant's response changes from positive to negative and vice versa (called a *reversal*) a certain number of times. Figure 2 illustrates the stepping procedure for an up-down staircase terminated after four reversals. It should be noted that care is needed when selecting the step-size. Too large a step-size results in inaccurate threshold estimates and the possibility of *outliers* in the data. Alternatively, too small a step-size may result in an accurate threshold estimate but the risk of participants becoming bored, tired or losing their attention is high. Normally, the appropriate step-size is selected based on the results from preliminary experiments testing several different step-sizes.

To eliminate response bias caused by participants learning how the experimental procedure works, a pair of randomly interleaved staircases can be used [ODGK03]. This involves setting up *ascending* and *descending* staircases, where their respective stimulus level is initialised to a maximum and minimum intensity. These two staircases are then presented

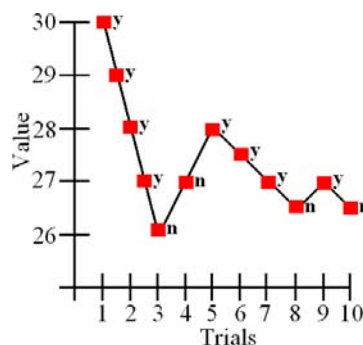


Figure 2: Example of the stepping procedure for an up-down staircase terminated after four reversals.

to the participant in a randomly interleaved manner to eliminate the participant guessing the direction of change of the stimulus intensity. To avoid data being sampled at too high or too low stimulus levels, adaptive procedures can be used to specify how to adapt the stimulus level depending on the participant's response. As a result of this, data sampling is concentrated around the participant's threshold on the psychometric function. Levitt provides an overview of adaptive staircase procedures [Lev71] such as the *transformed up-down* method and the *weighted up-down* method. With transformed up-down methods (used in [MAEH04]), the stimulus is altered depending on the outcome of two or more preceding trials. For example, a three-up one-down (3U-1D) stepping procedure involving the stimulus level is increased only after three successive incorrect responses and decreased with each correct response. With weighted up-down methods, different step-sizes for upward and downward steps are used.

1. Perception of Human Appearance

The main aim of this experiment is to establish if and when various virtual human representations are perceptually equivalent. This is especially important in LOD crowd systems displaying different representations simultaneously. Using a psychophysical approach, we attempt to derive a perceptual metric to aid in deciding when to use a particular representation based on the virtual human's appearance. Hamill et al. [HMDO05] and McDonnell et al. [MDO05] detail these experiments in full.

1.1. Experiment 1: Impostor Vs. Mesh Detection

This experiment aimed to establish the distance at which participants were able to discriminate between a virtual human's high resolution mesh and an impostor. In [DHOO05], it was hypothesised that humans should be able to detect the impostor once the size of a texel is bigger than the size of the impostor image's pixel, as aliasing artifacts then start to occur as a result of stretching the impostor's image onto the quadrilateral. Based on this hypothesis, the system switched between a virtual human's impostor and mesh representation when the impostor image pixel size to impostor texel

size was a ratio of 1:1. In this experiment, we try to find the exact pixel to texel ratio at which the participants are able to discriminate between these representations.

1.1.1. Visual Content and Procedure

Participants were shown the virtual human's geometric and impostor representations at different distances for 5 seconds (Figure 3). Each representation was animated with a 1-second cyclical walk animation. The participants were asked to indicate on which side the virtual human "looked better" by pressing the corresponding trigger button on a USB gamepad. Because applications containing virtual humans would typically involve displaying them from multiple viewpoints, both representations were rotated at 5.625 degrees every 100 milliseconds in a randomised direction around the y-axis, so that the participant was not comparing the representations based on a single viewpoint.

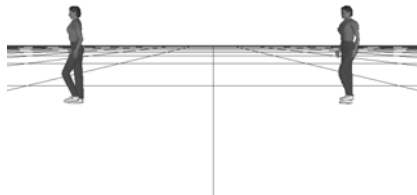


Figure 3: Test application environment with mesh on left and impostor on right.

1.1.2. Results

We recorded the participant's response at each distance and calculated the percentage of correct responses for each ratio at which the representations were displayed. We then plotted this as a function of the ratio and fitted a psychometric curve to the data.

Subsequently the PSE was calculated for each participant. The mean PSE value for all participants was 1.164 ± 0.064 . At this PSE level, the participant will judge the representations with equal likelihood as "looking better". The mean PSE is close to the hypothesised value of 1. This means that when an impostor is directly beside its mesh counterpart, users should not notice the difference between them when the impostor is displayed at a pixel to texel ratio of 1 to 1.

$$Pixel : TexelRatio(distance) = \frac{distance \times PixelSize}{TexelSize} \quad (7)$$

1.2. Experiment 2: Low Vs. High-Resolution Mesh Discrimination

A common LOD approach for reducing the computational cost associated with rendering a high detailed mesh is to



Figure 4: (left) high resolution model, (right) blocky shaped low resolution version.

replace it with a simpler, lower resolution mesh containing fewer triangles, where the loss of detail should be imperceptible to the viewer of the system (see [dHCUCT04]). However, care has to be taken when generating these low resolution meshes, as removing too much detail can produce blocky shaped results, along with animation artifacts due to the loss of joint vertices, and the overall visual realism of the virtual human is reduced (Figure 4). The second experiment was aimed at establishing the resolution, in terms of percentage of vertices, at which participants were able to discriminate between a virtual human's high resolution mesh and a selection of simplified low resolution meshes for three different distances.

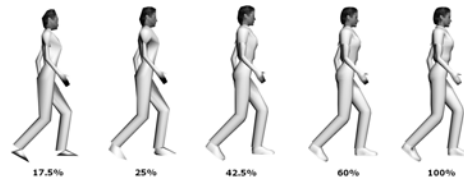


Figure 5: Illustration of some of the simplified models used for Experiment 2.

1.2.1. Visual Content and Procedure

In these experiments, a female model of 2170 polygons was used. Using the 3D Studio Max *Multires* modifier, a selection of low resolution meshes were generated in this manner, ranging from a reduced vertex percentage of 60% to 15% (2170 polygons to 262 polygons) at intervals of 2.5% (see Figure 5).

A high resolution mesh and a low resolution mesh were displayed alongside each other for comparison. The participants were asked to indicate whether the representations looked the "same" or "different" by pressing the respective left or right trigger button on a USB gamepad (Figure 6). Each time the participant indicated a "same" response, the resolution of the low LOD mesh was decreased, otherwise a "different" response increased the resolution.

As mentioned previously, three distances at which to display the representations from the viewer were chosen. The first distance was 5 metres, the second distance was 8 metres and the third distance was 16 metres, which corresponded to



Figure 6: Participant taking part in Experiment 2.

66.6% and 33.3% of the representation's initial screen space size.

1.2.2. Results

We recorded the participant's response for each mesh resolution displayed and calculated the percentage of correct responses for each resolution at which the representations were displayed, and plotted this as a function of the resolution.

It was found that distance affected perception of the low resolution mesh's visual appearance, with participants being able to discriminate better between different resolution meshes at closer distances.

The mean PSE values give us a good indication as to the percentage of vertices necessary for a low resolution mesh to be considered the same as its higher counterpart 50% of the time. However, this value is not very practical for developers, so we also included a further analysis. After much consideration, we chose the 80% probability of acceptance as the level to explore further as a practical level for developers. The 80% probability of acceptance is the percentage of vertices necessary for an observer to say that they found the low resolution mesh equivalent to the high resolution 80% of the time. We felt that this level was high enough to ensure that the difference between the low and high resolution meshes would not be noticed in most situations, such as in a game. Also, this level was considered low enough to be of practical use in real-time, as a 90% probability would result in a mesh almost as detailed as the high resolution mesh, and would not represent a significant saving.

The mean 80% probability of acceptance values are presented in Figure 7, along with the mean PSE values. The corresponding number of vertices and polygons for the 80% probability of acceptance are shown in Table 1.

From these results, a low resolution mesh generated at 42.2% is acceptable to use as a replacement for the high resolution, at the closest distance, as observers will consider them to be same 80% of the time. This suggests that we are using a mesh that is too detailed for the highest LOD in our crowd system, and a less detailed model could be used without the user noticing, while improving the system's performance.

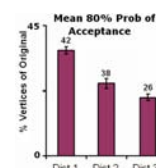


Figure 7: (left) Mean PSE values, (right) Mean 80% probability of acceptance.

Distance	5.0	8.0	16.0
Vertex %	42.2±1.0	37.6±1.0	26.2±1.0
Vertices	487	434	302
Triangles	871	770	524

Table 1: Developer guidelines from 80% probability of acceptance results.

LOD _{Geometry}	Distance (metres)	Cost _{LOD} (ms)	Crowd Size @ 30FPS
High Res 100%	< 5.0	0.0645	517
Low Res 42.2%	> 5.0	0.0241	1,385
Low Res 37.6%	> 8.0	0.0221	1,507
Low Res 31.9%	> 12.416	0.0198	1,686
Low Res 26.2%	> 16.0	0.0135	1,961
Impostor	12.416	0.00697	4,777

Table 2: The distance at which LOD_{Geometry} models are perceptually equivalent (using 80% probability of acceptance values) and their associated rendering cost.

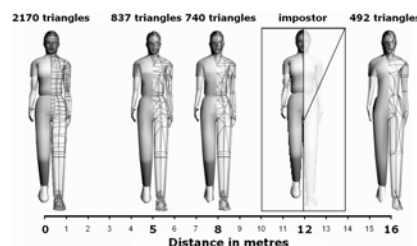


Figure 8: Summary of distances at which to display representations. In this example 12 metres is equivalent to the distance at which an impostor achieves the acceptable pixel to texel ratio.

1.2.3. Developer Guidelines for Experiment 1 and 2

The results from Experiment 1 showed that participants could not discriminate between impostors and their high resolution model at a pixel to texel ratio of approximately 1.16, which corresponds to a distance of 12.416 metres. However, low resolution meshes can be perceptually equivalent to their high resolution mesh at a closer distance. By using the results from Experiment 1, we can estimate the percentage of vertices at which to generate a low resolution mesh that is indistinguishable from the high resolution model at the same

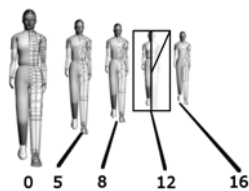


Figure 9: Summary of distances at which to display representations, where representations are shown at the actual distance used.

distance as the impostor. This corresponds to a low resolution mesh of approximately 31.9%, or 621 triangles. Due to the rendering cost of each model (Table 2), we suggest that it would be advantageous to use the impostor instead of a low resolution mesh for virtual humans being displayed at a ratio greater than 1.16. To summarise, Figures 8 and 9 illustrate the distances at which low resolution meshes and impostors are perceptually equivalent to a high resolution mesh.

1.3. Experiment 3: Impostor/Mesh Switching Discrimination

Typically, developers use the LOD approach of switching between a detailed mesh representation and a lower detailed model based on some selection criteria, to help maintain the interactivity of their system. It is important that the switching between models is imperceptible to the viewer, otherwise the overall believability of the system is reduced. While the selection of the model’s resolution can be based on several switching criteria, usually this is based on some distance threshold from the viewer of the system. With respect to our system, we achieve interactive frame rates by using a high resolution mesh only for humans in the front, and impostor representation that can be displayed at a fraction of the rendering cost of the mesh. We switch between these representations in order to maintain the realism of the crowd. While having thresholds for the believability of an impostor is useful when it is displayed beside its equivalent mesh representation, popping artifacts often manifest during the transition from impostor to geometry. These sudden popping artefacts during this transition may be caused either by differences in aliasing, depth information, or using a fixed number of pre-generated viewpoint images which can also cause shading differences.

In Experiment 3, we aimed to establish the distance at which the transition from a pre-generated impostor to a mesh is noticeable. The goal in establishing such a threshold was to provide us with a guide to the distance at which the switching between our impostor and mesh representation should occur in order to reduce any noticeable popping artefacts and therefore maintain the realism of our crowd. This distance can be calculated in terms of a pixel to texel ratio (see Experiment 2), and it was hypothesised that beyond the point of one-to-one pixel to texel ratio, the participants would be unable to detect the transition.

1.3.1. Visual Content and Procedure

For each trial, the same model used in the first experiment was displayed, starting at a specific distance from the viewer, then moving at a constant speed towards the camera, and finally stopping at a specific distance. At some point during the interval the model switched from an initial impostor representation to a mesh representation.

The participants were asked to indicate whether they noticed a “definite change” in the model, by pressing the left or right trigger buttons of the gamepad to indicate their respective yes/no response. Each time the participant indicated a “yes” response, the distance at which the switch occurred was increased, otherwise a “no” response decreased the distance.

Two separate experiments were carried out, with the model either facing the user or spinning on the spot. The virtual human switched from its impostor to its geometric representation at a switching distance ranging from 6 to 31 units.

It was found that, when the virtual human approached the camera too quickly, the resulting rate of change in the texture detail of the geometric representation (since mipmapping was not employed for its texture), caused the participants to perceive a switch where there was none. While the effect of popping artifacts may be reduced by blending, such as in Ebbesmeyer [Ebb98], we aimed to establish baseline thresholds where this would not be necessary. For urban simulations (which generally are constrained to the ground plane), transitions typically occur at the distance where the change in depth information is small due to perspective, and for virtual humans the overall change of depth information is similarly small. A further investigation of the effect of blending on transition detection is desirable.

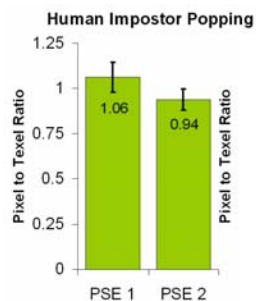


Figure 10: Results of the popping detection experiments (showing the PSE in terms of pixel to texel ratio) for humans facing viewer (1) and spinning (2).

1.3.2. Results

We recorded the participants’ responses for each trial’s switching distance. A psychometric curve was fitted to each participant’s experimental data. The mean PSE calculated (shown as PSE1 in Figure 10), was approximately the predicted one-to-one value. The mean PSE calculated for the

second experiment (shown as PSE2), was less than for PSE1, suggesting that the spinning was a distracting factor.

It should be noted that the results from this experiment are predicated on the texel size the impostor was pre-generated at. The texel size of the impostor used in this experiment was selected to ensure that all 17×8 pre-generated view-points fitted into a 1024×1024 image which is an image size commonly used in these types of applications. While the switching was not detected at a ratio of one-to-one for this texel size, it is hypothesised that this ratio will no longer be valid for impostors generated at a larger texel size due to aliasing artefacts being more noticeable. In order to establish at what texel size the switching is detectable at a one-to-one ratio, this would involve pre-generating impostors at various texel sizes, presenting a virtual human switching from each impostor to the mesh at the one-to-one distance, and evaluating at what texel size the participant is capable of detecting any popping artefacts.

2. Perception of Motion

In a LOD crowd system that simultaneously displays different model representations, as described in [DHOO05], it is important that the quality of the motion of the lower LODs is not significantly different from that of the high resolution. Hodgins et al. [HOT98] showed that model type affected user perception of human motion, when a stick figure model's motion was compared to a polygonal model. Here, we test whether or not the lower detail representations replicate the motion of the high resolution mesh, and we also test their ability to replicate deformable cloth motion. Finally, we look at how motion capture data is perceived when applied to different models. More detailed descriptions of these experiments can be found in [MDO05, MDCO06, MJH*07].

2.1. Experiment 4: Perception of Human Motion

The first experiment in this section aims to analyse the ability of low resolution geometry and impostors to replicate the motion of the higher resolution human mesh that they were generated from.

2.1.1. Visual Content and Procedure

Three different representations of a male model were used. Two of the models were polygonal models with deformable meshes which were manipulated by an underlying skeleton; the high resolution polygon model had a deformable mesh of 2022 polygons, while the low resolution polygon model had only 808 polygons for a deformable mesh. We automatically simplified the mesh as much as possible, without making the simplified version look different from the original. Impostors were the third type evaluated and the same pre-generated approach was used as in the other experiments.

A reference motion R was created which consisted of 10 frames of a key-framed walk motion. The 10 frames of R were modified a number of times to create the arm, leg, and torso motion variation sequences.

Firstly, the performance of the participants in distinguishing

smaller and larger dynamic arm motions was examined. Assessing the arm motion variation involved comparing R to a set of motions which altered the distance of the arm from the body at certain keyframes. The modifications were made by rotating the upper left arm joint in kA at the shoulder along the positive horizontal axis by a fixed number of degrees (Figure 12 (left)). The right arm was altered by the same amount in the reverse direction.

A similar test was conducted to test the ability of the participants in distinguishing larger and smaller leg motions for all representations. The leg was altered by iterative translations along the longitudinal and vertical axes (Figure 12 (middle)). Finally, the ability of the participants to distinguish modifications to the torso was tested. In this instance, the alterations were made by iteratively rotating the lower spine of the skeleton by a fixed number of degrees around the longitudinal axis (Figure 12 (right)).



Figure 12: (left) Top view of min and max arm variation, (middle) Perspective view of min and max leg variation, (right) Top view of min and max torso variation.

2.1.2. Experiment Procedure

Participants viewed pairs of movies, and were asked to specify whether they thought that the motion of the character in the movies was the "same" or "different" (Figure 13). After the first 4-second movie was viewed, the participant pressed a "view next" button on the screen using the mouse. The next movie was then presented for 4 seconds and the participant had to decide whether they thought that the motions were the same or different and press the corresponding on-screen button.



Figure 13: Participant taking part in the Experiment 4.

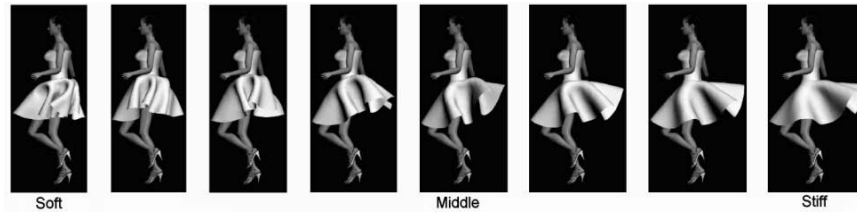


Figure 11: Eight different materials used in order of increasing stiffness

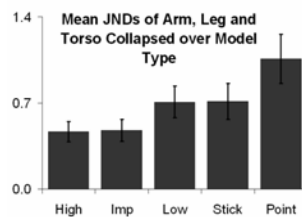


Figure 14: Mean JND values for all motion variations collapsed over model type.

2.1.3. Results

For each participant, the number of times that they viewed a pair of motions at each stimulus level was recorded, along with the number of correct responses that they gave at that level. The percentages of correct responses were then plotted against the stimulus level values. As we are interested in sensitivities to differences in motion for the different model representations, and not actual threshold values we examined the JND values rather than the PSE values in this experiment. Psychometric curves were then fitted to the datasets and, for each participant, the JND were calculated from these curves. The JND was then found by calculating the difference between the PSE and the stimulus level value that corresponded to 75% correct responses on the psychometric curve.

An ANOVA was used to compare mean JND values across all of the participants and showed that there was a significant difference in their sensitivities with respect to the changes viewed. The significances for the differences between model types indicate that the motion of the impostor was closer to that of the high resolution polygon model than that of the low resolution model (Figure 14).

We suggest that this is due to the fact that, even though the impostor appears perceptually different to the high resolution model at the distance shown in the experiments, it replicated the motion of the high resolution model accurately. The low resolution model may not replicate this motion as effectively because there are fewer vertices on the mesh, and even though it is the same skeleton used to deform this mesh, the deformation loses subtle motion information.

2.2. Experiment 5: Perception of Cloth Motion

Displaying large crowds of high quality virtual characters with deformable clothing is not possible in real-time at present because of the cost of rendering the thousands of polygons necessary to depict the subtle motion of the cloth. Current real-time crowd systems are capable of displaying thousands of skinned characters by using lower quality representations instead of high quality to achieve their goal. Hybrid systems that switch between high and lower quality models depending on the distance from the camera, are also a solution to this problem.

Our aim was to extend the hybrid crowd system described in [DHOO05] to include clothed characters. However, we were not certain at the outset which representation (i.e., low detail geometry or impostor) would be most suited to displaying the deformations of simulated cloth, although we hypothesised that low geometry would not be sufficient to reproduce the required deformations. We address the questions: Can impostors and low resolution geometry display a range of different cloth materials? How well can they reproduce individual material types? If compared directly, which representation would resemble a higher quality representation more closely?

2.2.1. Stiffness Sorting Condition

This condition aimed to establish whether the real-time lower detail cloth representations could produce distinguishable levels of cloth stiffness. After some experimentation, we found eight cloth motions which ranged from very soft to very stiff (Figure 11). Eight movies, each showing one of the representations, were placed randomly on a 21-inch widescreen monitor, each playing in a loop. Participants were asked to place the 8 movies in order of stiffness, with the least stiff on the left and the most stiff on the right. The order in which the participants placed the movies was recorded and compared to the actual sequence.

Figure 15 shows us that overall the participants found the low detail geometry cloth more difficult to sort than the high detail, and the impostor representation. There was no statistical significance between sorting the impostor and sorting the high resolution movies. These results indicate that the perception of subtle differences in cloth motions using the impostor is closer to that of the high detail geometry cloth

R. McDonnell, S. Dobbyn, & C. O'Sullivan / Perceptual Experiments and Metrics

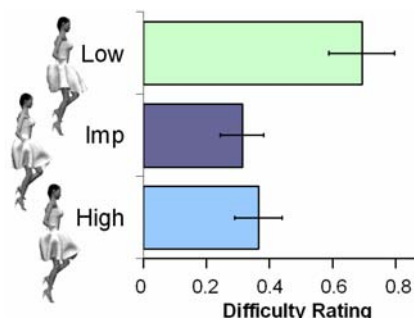


Figure 15: Results of Stiffness Sorting Condition.

simulation than the low geometry simulation. This also supports the findings in Experiment 4 that impostors are better at depicting small differences in human motion.

2.2.2. Stiffness Forced Choice Condition

The next condition we tested was to determine how well the high and low detail geometry and impostor reproduced the stiffness levels of a gold standard cloth (rendered offline with non-realtime rendering and lighting). Participants were shown 2 gold standard movies (that of the stiff and that of the soft skirts) beside each other. They were asked after every trial to indicate which of the two gold standard animations (stiff or soft) was more similar to the current target animation, playing on an adjacent monitor.

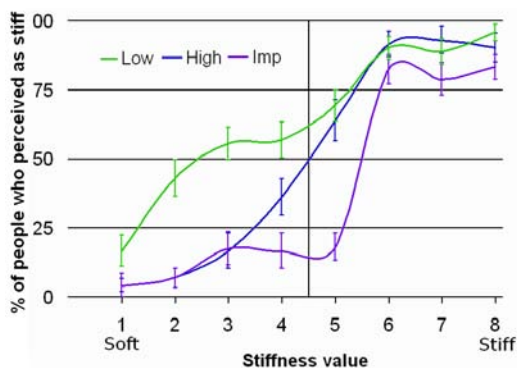


Figure 16: Perceived stiffness for different LOD cloths. Standard error bars are shown.

The interpretation of these results is evident from Figure 16. Participants found that the perceived stiffness of the cloth motion for the impostor was closer to that of the high resolution than the low resolution for low stiffness levels (i.e., soft materials). This suggests that the impostor better matches the high detail geometry motion at low stiffness levels. There is a divergence at the middle stiffness levels, where most participants rated the impostors to be soft, and few found the

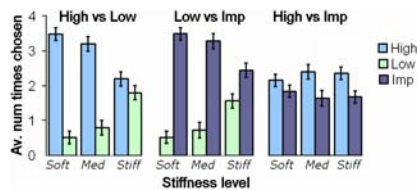


Figure 17: Results for LOD Comparison Condition.

low detail to be soft. At the high stiffness levels, more participants' perception of the low resolution cloth motion was closer to the high detail geometry than the impostor. Overall, there seems to be a tendency for the participants to perceive the low detail geometry cloth motion to be stiffer than the high geometry, and the impostor to be less stiff than the high geometry.

2.2.3. LOD Comparison Condition

A third condition was tested in order to see how well each representation matched the gold standard. Participants were first shown pairs of the most rigid cloth and were asked which cloth animation was most similar to the gold standard rigid cloth. Participants pressed left or right buttons on the gamepad to choose the most similar simulation. They were then shown pairs of the cloth with stiffness approximately halfway on our estimated scale (i.e., the fifth image in Figure 11), and were asked to compare them with the corresponding gold standard cloth. Finally, the participants viewed pairs of the most soft cloth and were asked to compare them to the gold standard as before. The number of times that a participant preferred each LOD representation over each of the others was recorded.

Our results (Figure 17) suggested that, when viewing these representations in a hybrid system that simultaneously displays virtual humans using two types of representation, switching intermittently between them, the low resolution will not resemble the high resolution as closely as the impostor does, thus resulting in significant artifacts.

2.3. Experiment 6: Applying Motion Capture

Crowds simulated with synthetic walking motions can lack personality, so motion captured data can be used to add realism. In this experiment, we investigate some factors that affect the perceived sex of walking virtual humans, with a view to increasing the realism of pedestrians in real-time crowd simulations. We cannot simulate everyone in a crowd with their own personal motion captured walk, as the more motions we use, the greater the demands on potentially limited computational and memory resources (e.g., a games console or hand-held device). Therefore, the challenge is to optimise quality and variety with the resources available. Specifically, we ask the question whether, if there is a clear visual indicator of sex (i.e., a highly realistic, unambiguously female or male model, as shown in Figure 18), will motion or form information dominate our perception of the sex of the character? If motion information alone always determines per-

ceived sex, then we would always need to create templates of every different motion for both males and females. However, if we find that form dominates, or that simulated neutral motions are as good as captured natural motions under some circumstances, then such duplication may not always be necessary. Perhaps some actors' walks can be equally effectively applied to both male and female models. Any of these results would allow us to create "canonical" motions to which variety could later be added, irrespective of sex.

2.3.1. Visual Content and Procedure

Six undergraduate students (3M, 3F) volunteered to be motion captured, each in a separate session per actor. We captured some of the walks without their knowledge to ensure they were walking naturally, then applied the motion capture data to characters in 3D Studio Max and kept one natural walk per actor. We also generated three different neutral walk motions, with neither male nor female characteristics (such as hip sway or shoulder movement).

Four different models were used to display the different motions (Figure 18): highly detailed woman and man models of approximately 35000 polygons each, an androgynous character, and a point light walker. The woman and man were chosen as typical characters that would be used in a computer simulation of natural crowds. The androgynous figure was chosen as it did not appear particularly male or female and so could serve as a control. The point light walker was generated from a generic neutral skeleton and so contained minimal shape information.



Figure 18: Four model representations were animated with real female, real male or synthetic neutral motions. From left to right: Woman model, Man model, Androgynous figure and Point light walker.

Each of the different motion types (3) from each of the actors (3) were applied to each of the model types (4), with two repetitions for each condition, resulting in a total of 72 movies. Participants viewed the movies and were told to take both motion and form/shape into account and they marked their selections on an answer sheet. Participants categorised the character they just saw on a five-point scale of 1: *very male*, 2: *male*, 3: *ambiguous*, 4: *female* or 5: *very female*.

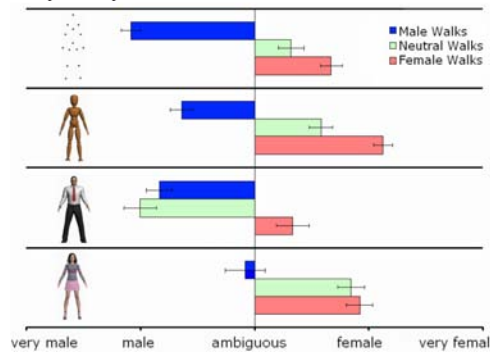


Figure 19: The interacting effects of model and walk type.

2.3.2. Results

Results show that male walks on the woman are rated as ambiguous, as are female walks on the man (Figure 19). This implies that applying motion captured from actors of the opposite sex to the character will produce confusing or unsatisfactory results in general. Interestingly, neutral walks were considered male when viewed on the man and female when viewed on the woman. This implies that for neutral walks, the appearance of the character takes precedence over the motion in determining the sex of the character. This result has implications for computer graphics applications where resources are limited, as re-using the same neutral walks on male and female characters would appear to produce the desired effect.

Also, for a character with androgynous appearance, the motion information is most important when determining the sex (as without motion, the androgynous figure was consistently rated to be ambiguous).

3. Perceptual metrics for smooth animation

In this section, we describe a series of experiments designed to provide metrics to developers for smooth animations. The first experiment in this section finds the optimal number of viewpoint images necessary for smooth impostor animations. The next experiment aims to find the optimal pose update rate for characters performing different animations. Finally, we look at simulation LOD and establish whether different pose updates would be acceptable if displayed together in one scene. These experiments are described in full in [MDCO06, MNO07].

3.1. Experiment 7: Impostor Update Frequency

From the results recorded in previous experiments, we can see that impostors are good at representing the deforming folds of cloth and are a good substitute for high resolution geometry clothed models at a 1:1 pixel to texel ratio distance from the camera. As mentioned above, impostors are generated by rendering multiple images of the human from different viewpoints for every frame of animation. The appropriate viewpoint is selected with respect to the camera in the real-time system. Typically, these viewpoints are generated at regular intervals around a sphere, so the sampling



Figure 20: Generating impostor images from a camera positioned on the circumference of a circle every 45° .

density can be described by the number of degrees difference between each segment of the sampled sphere (e.g., Figure 20 shows impostor images generated every 45°).

Ideally, impostors would be generated at very small intervals around a sphere, the same number of times that a polygonal model would be updated, which would allow seamless transitions between the images. However, as we are using pre-generated textures, texture memory consumption prevents choosing such a dense sampling, so there is a need to pick an optimal number of viewpoint images to generate. Dobbyn et al. [DHOO05] and Tecchia et al. [TLC02] report rendering 17 and 16 viewpoints of their impostors from one side of the human and mirroring the impostors for the reverse angle. This corresponds to an update rate of 10.58° (180° divided by 17) and 11.25° respectively. However, they do not specify their reasons for choosing these numbers of viewpoints. With the addition of cloth to the impostors, mirroring is no longer possible due to the non-symmetric nature of cloth (the cloth on one side is usually not identical to the other side, due to the folds occurring in different places). Also, it was not clear that this directional sampling density would be appropriate when clothing was added to the impostors. As in Tecchia et al. and Dobbyn et al., interpolation was not used between different views, as it would be computationally intensive and could introduce visual artifacts.



Figure 21: Six different characters used in impostor update frequency test

3.1.1. Visual Content and Procedure

We chose 6 characters as stimuli for this experiment (Figure 21). We pre-generated 256 impostors for every frame of the 10-frame animation, which corresponded to an optimal sampling density of 1.4° (i.e., 360° divided by 256). The

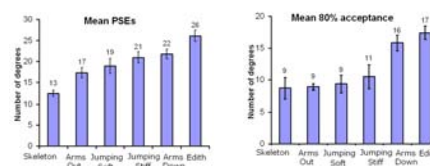


Figure 22: (a) Mean PSE values for all models, (b) Mean values for 80% probability of acceptance

characters moved on a circular path at a normal walk pace, with the closest point to the viewer on the circle being the pixel to texel ratio distance reported in Experiment 1, as impostors would not be viewed any closer than this in a crowd system. The character was placed at a random point on this path, and walked for 3 seconds, in either a clockwise or an anticlockwise direction. Participants were asked to specify, using the gamepad, whether they perceived the change in orientation of the character as jerky or smooth. The next trial they saw depended on their previous response.

For each participant, a psychometric curve was fitted to their dataset as described previously. This allowed us to find the Point of Subjective Equality (PSE). The PSE is the point at which participants were equally likely to find an animation smooth or jerky, i.e., where they have a 50% chance of considering that motion smooth (Figure 22). A psychometric probability curve for each of the characters, derived from all of the data, was then created, using the average of all participants' PSE and standard deviations (Figure 23).

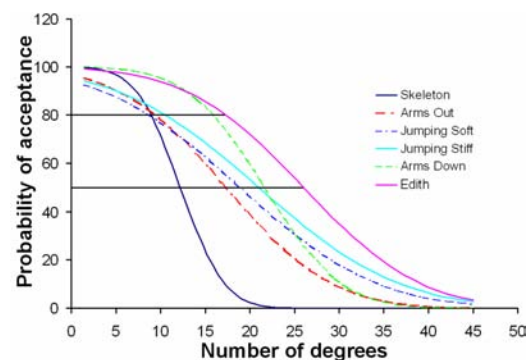


Figure 23: Probability of acceptance curve derived from psychometric data. This shows the number of degrees necessary for each of the characters to be considered smooth for a range of different probability of acceptance values. For example, at 10 degrees, 80% of the time participants found Edith, a typical pedestrian, to be smooth.

3.1.2. Developer Guidelines

For normal walking characters, with either stiff or soft clothing, a viewpoint update rate of 17° is necessary to guarantee with 80% likelihood that users will not notice viewpoint



Figure 24: Character 1, character 2 and character 2 with deformable clothing.

changes of the impostors. This corresponds to 21 images that need to be generated at equal spacing around the character. We suggest rounding to the nearest even number of images (22) in order to include the direct front and the direct back view images, particularly in applications where a front-on view would be most noticeable. For other characters whose width to depth ratios are large, a viewpoint update rate of 9° is advised. This corresponds to 40 images around the character. In [DHOO05] and [TLC02], updates of 10.58° and 11.25° were used. We can now see that these rates were underestimates for complex characters but overestimates for normal walking characters.

3.2. Experiment 8: Animation Update Frequency

Pose update rate can be defined as the frequency of individual simulation steps displayed when animating a character. For most real-time crowd simulation, the pose update rate is largely constrained by the available hardware and overall scene complexity. Individual mesh or image-based (impostor) keyframes can be “pre-baked” for background characters, which reduces rendering and simulation costs but increases memory consumption (thus the pps must remain low).

In this experiment, we identify some factors and thresholds for the perceived smoothness of animated virtual characters. In a Baseline Condition, we first determined whether different pose update rates are in fact needed for human motions with different character and motion properties. Our detailed Movement Condition examined the impact and interactions of various motion properties.

3.2.1. Baseline Condition

We first tried to identify baseline factors that affected the perceived smoothness of animated human motion. The goal was to find the threshold among 15 update rates (ranging from 4pps to 63pps) at which the participants found the different animations smooth, for each of the conditions tested. The conditions were *character type* (a male ‘character 1’ and a female ‘character 2’ shown in Figure 24) and *motion type* (motion captured kungfu kick, jumping jack, walking and jogging), and *character cloth type* (i.e., character 2 as depicted in Block 1 with simple skinned clothing, and character 2 with physically simulated deformable clothing). Each condition was shown at each update rate a number of times

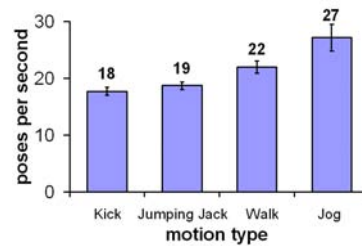


Figure 25: Mean 50% threshold values for different animation types. Error bars show the standard error of the mean.

and the participant pressed the left or right mouse button on the laptop to indicate “smooth” or “jerky” for each movie.

3.2.2. Analysis

We found that character type did not affect update rate in our experiments. Surprisingly, deformable clothing did not have an effect. However, motion type did have a significant effect, with motions moving further across the screen needing more updates than other motions (Figure 25).

We felt that the distance that the character moved across the screen must have been a factor, as the walk and jog motion moved much more across the screen than the other two motions. Furthermore, the amount of activity in the motion clip seemed to have an effect. Therefore we designed our next experiment condition to focus on these two factors in particular.

3.2.3. Movement Condition

The second condition examined more formally the effect of different motion types and their interactions.

3.2.4. Visual Content and Procedure

Two motion complexities were chosen: *Normal walk* with arms by the side, and *Complex walk*, the same walk motion with added activity in the arms, torso and head, each moving in time with the legs of the walk cycle. Three different cycle rates were chosen: *Lo* (1.5 cycles/sec), *Med* (2.72 cycles/sec) and *Hi* (3.75 cycles/sec). Four different linear velocities were chosen: *V0* (walking on the spot), *V1* (walking 1/3rd of the distance across the screen, i.e., 7.75screen centimetres/sec), *V2* (walking 4/6ths of the distance across the screen, 15.5cm/sec) and *V3* (walking the full distance across the screen, 23cm/sec).

We split the 4 linear velocities into four separate experiment blocks. Participants viewed all four blocks, with a one minute break in between each block. As before, they were asked to indicate whether the animation looked “smooth” or “jerky” at each trial.

3.2.5. Analysis

As in the baseline experiment, we fitted psychometric curves to participants’ data for each of the conditions, and were thus able to calculate their 50% threshold values.

To summarize our findings for the Movement experiment, in Figure 27 we show a chart of the 80% acceptance thresholds



Figure 26: System Setup for Movement Condition.

(i.e., the level at which observers will say 'smooth' 80% of the time). We felt that these values could be of practical use to developers since the thresholds at this level were reasonable for real-time performance. Figure 28 depicts these values in a more useful way for developers to choose the correct update rates for their particular animations.

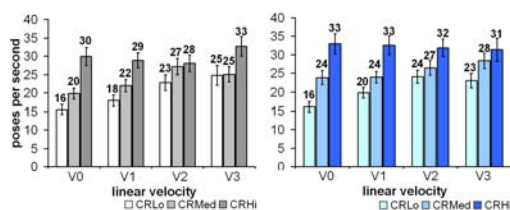


Figure 27: (l) Thresholds for 80% probability of acceptance for normal walk. (r) Thresholds for 80% probability of acceptance for complex walk. Error bars show the standard error of the mean. CR=Cycle Rate.

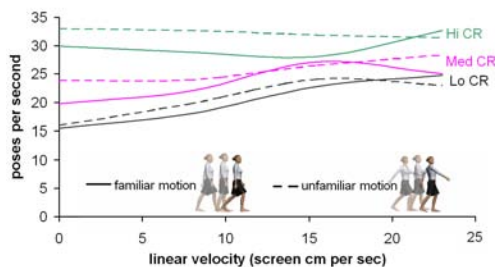


Figure 28: Summary of Movement Experiment results. For the familiar motion, the legs were the fastest moving body parts. We calculated that the fastest pixel was moving at 7 screen cm/sec (Lo CR), 12cm/sec (Med CR), and 15cm/sec (Hi CR). For the unfamiliar motion, the arms were the fastest and we calculated that the fastest moving pixel was moving at 13cm/sec, 23cm/sec and 32cm/sec for Lo, Med and Hi CR.

These results will perhaps be of most use (and immediately applicable) for real-time character simulation, particularly when the characters have cyclical motions. In a pre-

processing step, each motion could be labeled with an optimal update rate, based on the cyclical update rate and complexity of the motion. At run-time, this rate for all characters could change priority when the camera is moving fast, to account for the added jerkiness which occurs with fast camera motion.

3.3. Experiment 9: Simulation Level of Detail

For memory critical systems such as real-time crowds using impostors, the fewer poses required to make an animation appear smooth the better. In Experiment 8, we have provided thresholds of acceptability for different characters, depending on their cycle rate, complexity and the amount of camera motion in the scene. However, we have not yet established whether different pose updates would be acceptable if displayed together in one scene, as our previous experiments showed one image at a time. Therefore, in this experiment, we consider simulation level of detail by examining the effect of its implementation on perceived motion smoothness. We give participants a discrimination task, in which they can view two pose update rates simultaneously and make their decisions based on a comparison of the two.

3.3.1. Visual Content and Procedure

We used the same character as in the previous experiments (character 2). Each movie displayed one character in the front and a group of characters in the back (Figure 29). The characters in the background were updated at 5 different rates, ranging from 5 to 30pps, and this was the stimulus level - again, we wished to determine the thresholds among the 5 update rates at which all characters appeared smooth, for each set of conditions.



Figure 29: Crowd in experiment setup.



Figure 30: Crowd in game setting used in SLOD experiment 2. Characters have different form and colour from one another and background is not white.

This experiment had a 3-way design. The first condition was walk type: *in step* (i.e., all characters started at the same pose, and moved like an army) or *out of step* (i.e., all characters started at different poses which represented a more natural setting for a crowd of pedestrians.); the second condition was background group size: *small* (1 character), *medium* (6 characters) or *large* (12 characters); and the third condition was foreground character update rate: *update1* (30pps, which from our previous experiment we know to be the threshold at which 99% of participants perceived the motion presented in this experiment to be smooth) or *update2* (20pps, the threshold with a 75% probability of being perceived smooth).

We also wished to examine how a more natural scenario, such as that found in games, would affect our results. Therefore, we compared the 'out of step' large group to a new 'out of step' large crowd where all characters were different and appeared in a more complex background scene (Figure 30).

3.3.2. Analysis

In order for our SLOD experiment results to be of use to developers, we analysed the 80% probability of acceptance values, estimated from participants' psychometric curves. We found that 16pps was considered sufficient for all background characters that we tested (at 80% probability of acceptance there was no significant main effect of crowd size, walk type, or foreground character update rate). This result could be of great benefit to LOD crowd systems in particular. At present, in hybrid geometry/impostor crowds (e.g., [DHOO05]), 10pps are used for both foreground and background characters due to the memory consumption of impostors. However, this rate results in jerky looking animation. Using 30pps for the geometry and 10pps for the impostors resulted in noticeable differences, and using 30pps for the impostors is too costly in terms of memory. If, as our results suggest, it is possible to display geometry at 30pps and impostors at 16pps without observers noticing the difference, this will result in the ability to store double the number of characters in memory than would be possible if the impostors were being displayed at 30pps.

In order to evaluate our SLOD metrics, we plugged the value of 16pps for background characters into a simple geometry/impostor crowd scenario and found the differences in SLOD to be imperceptible. Although the differences were imperceptible in the example we tested, all of the characters were walking on the spot, so switching between update rates as the characters moved from foreground to background was not present.

4. Evaluation of Metrics

In this section, we evaluate the effectiveness of the previous perceptual metrics. This experiment is described in detail in [MDCO06].

4.1. Experiment 10: Impostor Update and Mesh Detection Metric Evaluation

This experiment aimed to test the validity of the results of the previous experiments in a real crowd scenario. The crowd scene was populated with the female jumping model used in the previous cloth experiments. The characters in the scene were either all wearing the most stiff skirt from the experiments, or the most soft skirt.

The experiment included three typical crowd systems: full geometry, hybrid high polygon/impostor and hybrid high polygon/low polygon. In the full geometry crowd system, all characters were high resolution polygonal models of 8983 polygons each (6172 for the human model and 2811 for the skirt). The hybrid high polygon/impostor system contained the high resolution polygon models nearest to the camera, and the impostor representations at the back (Figure 31 bottom). The latter were displayed at the pixel-to-textel ratio at which they are perceptually equivalent to high resolution meshes. This pixel-to-textel ratio was found for individual characters, but was never tested on a large crowd. We used the results of the previous experiment to choose the number of viewpoint images necessary for the two models.

The hybrid high/low resolution polygon system contained high resolution characters at the front, and low resolution at the back (Figure 31 middle). Five hundred and thirty polygons were chosen for the low resolution skirt.

In a typical hybrid crowd system, the LOD choice depends on the distance of a character from the camera. As the camera moves through the scene, switching between representations will occur, due to the camera distance changing. To examine this effect in our experiments, the camera zoomed up and down through a corridor between the characters at a speed of 4m/s, in order to ensure that LOD switching occurred. Switching between impostor viewpoints also occurred in this case.

The effect of switching between impostor viewpoints was then examined independently by allowing the camera to only pan from left to right at a speed of 2m/s, where the impostor distance was fixed. In this case, the impostor viewpoints were changing but no switching back and forth to high resolution geometry occurred.

One hundred and eight 4-second movies were created in total: Three types of system: *All Hi, Hi/Lo, Hi/Imp* \times 2 skirt types: most stiff and least stiff \times 3 crowd sizes: small (50 humans), medium (100 humans) and large (1000 humans) \times 2 conditions: camera panning from left to right, camera moving up and down a fixed corridor \times 3 random placings: 3 different random placings of characters in the scene.

Each participant viewed the sequence, and was asked for every trial, whether all of the characters in the scene were the same or if they noticed that some of the characters looked different in any way.

We first analysed the effect of camera panning or zooming to determine how effective or metrics for impostor viewpoint switching (Experiment 7) and LOD switching (Experiment 3) were in a crowd scenario. An ANOVA was performed



Figure 31: (top) Small crowd of All Hi with stiff cloth. (middle) Medium crowd of Hi/Lo with stiff cloth. (bottom) Large crowd of Hi/Imp with soft cloth.

on the dataset and it was found that there was no significant main effect of camera motion on the ability of participants to tell the differences between the representations. This implied that participants were unaware of the switching between representations or switching between impostor viewpoints, which is very good news for hybrid systems.

We then analysed the effect of stiffness (Figure 32). For the stiff cloth, participants noticed the difference between the hybrid *Hi/Lo* and *All Hi* movies significantly more times. Also, they noticed the difference between *Hi/Lo* more times than the *Hi/Imp*. There was no statistical significance between the number of times that they noticed a difference in *All Hi* compared to *Hi/Imp*.

For the soft cloth, there was a difference between *All Hi* and *Hi/Lo*, and between *Hi/Lo* and *Hi/Imp*. Again, there was no difference between *All Hi* and *Hi/Imp*. This implied that having a hybrid crowd using impostors and high resolution geometry will introduce less artifacts than a hybrid crowd with low resolution geometry.

As expected, there was a statistically significant difference

between *Hi/Lo* stiff and soft, with the soft cloth low geometry being noticed more times than the stiff cloth. There was also a difference between impostor stiff and soft - with differences in the soft being noticed fewer times than differences in the stiff cloth. Similar differences in stiffness were present for *All Hi*.

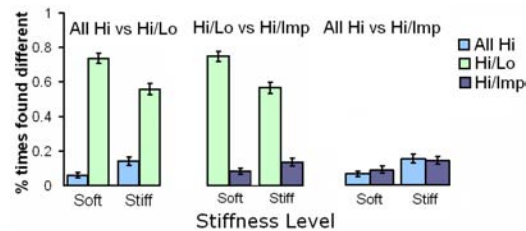


Figure 32: Experiment 10: LOD vs Stiffness

The previous experiments all depicted scenes with only 1 or 2 characters. This represents the worst-case scenario, as the character was being analysed directly, with no surrounding distractions. Surprisingly, it was found that there was no overall effect of crowd size, implying that differences could be noticed just as easily in small crowds as large crowds.

5. Conclusions and Future Work

We have gained new insights into the effects of different level of detail representations with respect to human motion, appearance and secondary motion. Although these results are useful in terms of helping to improve real-time crowd systems, this is still an area of research that would benefit from more perceptual studies. The ultimate goal is to create a framework for highly detailed crowds, driven by perceptual metrics.

In Experiment 10 we found that the size of the crowd did not affect perception of background character LOD. While this was a surprising result, the participants only had the task of trying to perceive differences between the foreground and background characters. We feel that if the viewer was asked to perform a different task, these background differences might not be perceived as often, and the crowd size may have an effect in this case. We base this assumption on the fact that previous research by Cater et al. [CCW03] has shown that humans fail to notice degradations in image quality in parts of the scene unrelated to their assigned task. Also, Harrison et al. [HRvdP04] noticed that expectation about the task affected perception of motion. Varying the task is certainly something we would like to examine in the future, as it is likely that viewers of crowd scenes will be involved in game-play in the foreground of the scene, or navigating through a city.

We presented guidelines for developers on the number of impostor viewpoints needed in order to produce imperceptible switching, for one elevation of impostors (i.e., those on the ground plane). Using the same number for higher elevations

might be wasteful, so it would be interesting to determine the number of updates needed for all elevations, using similar psychophysical methods.

References

- [CCW03] CATER K., CHALMERS A., WARD G.: Detail to attention: Exploiting visual tasks for selective rendering. In *Proceedings of the Eurographics Symposium on Rendering* (2003), pp. 270–280.
- [Cor62] CORNSWEET T.: The staircase method in psychophysics. *American Journal of Psychology* 75, 3 (1962), 485–491.
- [dHCUCT04] DE HERAS CIECHOMSKI P., ULICNY B., CETRE R., THALMANN D.: A case study of a virtual audience in a reconstruction of an ancient roman odeon in aphrodisias. *The 5th International Symposium on Virtual Reality, Archaeology and Cultural Heiritage (VAST)* (2004), 9–17.
- [DHOO05] DOBBYN S., HAMILL J., O'CONNOR K., O'SULLIVAN C.: Geopostors: a real-time geometry / impostor crowd rendering system. In *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (2005), pp. 95–102.
- [Ebb98] EBBESMEYER P.: Textured virtual walls - achieving interactive frame rates during walkthroughs of complex indoor environments. In *Proceedings of the Virtual Reality Annual International Symposium* (1998), pp. 220–228.
- [HBF02] HARRISON J., BOOTH K. S., FISHER B. D.: Experimental investigation of linguistic and parametric descriptions of human motion for animation. *Computer Graphics International* (2002), 154–155.
- [HMDO05] HAMILL J., MCDONNELL R., DOBBYN S., O'SULLIVAN C.: Perceptual evaluation of impostor representations for virtual humans and buildings. *Computer Graphics Forum* 24, 3 (2005), 623–633.
- [HOT98] HODGINS J., O'BRIEN J., TUMBLIN J.: Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics* 4, 4 (1998), 307–316.
- [HRvdP04] HARRISON J., RENSINK R. A., VAN DE PANNE M.: Obscuring length changes during animated motion. *ACM Transactions on Graphics* 23, 3 (2004), 569–573.
- [LCR02] LEE J., CHAI J., REITSMA P.: Interactive control of avatars animated with human motion data. *International Conference on Computer Graphics and Interactive Techniques* (2002), 491–500.
- [Lev71] LEVITT H.: Transformed up-down methods in psychoacoustics. *Journal of the Acoustical Society of America* 49 (1971), 467–477.
- [LHEJ01] LINSCHOTEN M., HARVEY L., ELLER P., JAFEK W.: Fast and accurate measurement of taste and smell thresholds using a maximum-likelihood adaptive staircase procedure. *Perception and Psychophysics* 63, 8 (2001), 1330–1347.
- [MAEH04] MANIA K., ADELSTEIN B. D., ELLIS S. R., HILL M. I.: Perceptual sensitivity to head tracking latency in virtual environments with varying degrees of scene complexity. In *APGV '04: Proceedings of the 1st Symposium on Applied perception in graphics and visualization* (2004), pp. 39–47.
- [MDCO06] MCDONNELL R., DOBBYN S., COLLINS S., O'SULLIVAN C.: Perceptual evaluation of LOD clothing for virtual humans. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 117–126.
- [MDO05] MCDONNELL R., DOBBYN S., O'SULLIVAN C.: LOD human representations: A comparative study. *Proceedings of the First International Workshop on Crowd Simulation* (2005), 101–115.
- [MJH*07] MCDONNELL R., JÖRG S., HODGINS J. K., NEWELL F., O'SULLIVAN C.: Virtual shapers & movers: Form and motion affect sex perception. In *Proceedings of the ACM Siggraph/Eurographics Symposium on Applied Perception in Graphics and Visualisation (to appear)* (2007).
- [MNO07] MCDONNELL R., NEWELL F., O'SULLIVAN C.: Smooth movers: Perceptually guided human motion simulation. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (to appear)* (2007).
- [ODGK03] O'SULLIVAN C., DINGLIANA J., GIANG T., KAISER M. K.: Evaluating the visual fidelity of physically based animations. *ACM Transactions on Graphics* 22, 3 (2003), 527–536.
- [OHJ00] OESKER M., HECHT H., JUNG B.: Psychological evidence for unconscious processing of detail in real-time animation of multiple characters. *Journal of Visualization and Computer Animation* 11, 2 (2000), 105–112.
- [RP03] REITSMA P., POLLARD N.: Perceptual metrics for character animation: Sensitivity to errors in ballistic motion. *ACM Transactions on Graphics* 22, 3 (2003), 537–542.
- [TLC02] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Visualizing crowds in real-time. *Computer Graphics Forum (Eurographics 2002)* 21, 4 (2002), 753–765.
- [Tre95] TREUTWEIN B.: Minireview: Adaptive psychophysical procedures. *Vision Research* 35, 17 (1995), 2503–2522.
- [WB03] WANG J., BODENHEIMER B.: An evaluation of a cost metric for selecting transitions between motion segments. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 232–238.

Populating Virtual Environments with Crowds: Real-Time Crowd Rendering with Pre-Generated Impostors

S. Dobbyn, R. McDonnell, and C. O'Sullivan

Graphics, Vision and Visualisation (GV2) Lab, Trinity College Dublin, Ireland

Abstract

Although many new games are released each year, it is very unusual to find large-scale crowds populating the environments depicted. Such applications need to deal with having limited resources available at each frame. With many hundreds or thousands of potential virtual humans in a crowd, traditional techniques rapidly become overwhelmed and are not able to sustain an interactive frame-rate. Therefore, simpler approaches to the rendering of the crowds are needed. Additionally, these new approaches must provide for variety, as environments inhabited by carbon-copy clones can be disconcerting and unrealistic. This part of the tutorial describes the impostor representation used in our crowd rendering system, detailing our programmable hardware based method for lighting and adding variation.

1. Introduction

This part of the tutorial describes the impostor representation used in our Geopostor system, a real-time geometry/impostor crowd rendering system (Figure 1). The Geopostor system has been developed to solve the challenging problem of large-scale crowds by simulating virtual humans as scene extras, equivalent to those found in films. Since these agents are in the background, they are not the focus of the user's attention and therefore simpler animation, rendering and behavioural techniques can be applied to them in order to reduce the computational load of crowded scenes.

Our main contribution is that our system provides for a hybrid combination of image-based (i.e., impostor) and detailed geometric rendering techniques for virtual humans. By switching between the two representations, based on a pixel to texel ratio [DHOO05], our system allows visual quality and performance to be balanced. We improve on existing impostor rendering techniques and present a programmable hardware based method for the lighting of impostors. Furthermore, we improve the realism of the crowd by adding variation to an individual's motion and appearance.

2. Real-Time Crowd Rendering with Pre-Generated Impostors

While a deformable mesh was the obvious choice for the virtual human's highest LOD in our crowd system, there are

a number of reasons why we chose an impostor approach for the lowest LOD over a continuous and a discrete LOD framework. Firstly, impostors involve replacing a 3D object with an image of the object mapped onto a quadrilateral. This is advantageous mainly because it avoids the cost associated with rendering the object's full geometry. Secondly, automatic tools used to pre-generate low-resolution meshes required for a discrete LOD framework sometimes do not give the required results, thus necessitating a lot of time-consuming editing by hand. Finally, switching between two meshes of different resolutions can be quite noticeable as a result of the silhouettes not matching. A continuous LOD framework utilizing subdivision surfaces offers a good solution to this problem, since the detail of a character can be increased and reduced at run-time, as demonstrated recently by Leeson [Lee02]. While subdivision surfaces provide a means of improving the appearance of virtual humans [OCV*02], they are not suitable for a crowd's lowest geometric LOD representation, since the surface's original polygonal model, used as its starting point, consists of several hundred polygons.

With regards to our impostor model, we decided on a pre-generated approach, since dynamically generating impostors would involve reusing the current dynamically generated image over several frames in order to be efficient. For dynamically generated impostors, the generation of a new impostor image for a virtual human depends on both camera motion



Figure 1: Screenshots of the Geopostor system.

and the amount the virtual human's posture has changed. This method works well with small groups of humans but as the number of virtual humans dramatically increases, numerous new impostor images need to be generated and this produces a bottleneck. Therefore, this method is not well suited for rendering large crowds of dynamic humans.

3. Generation of the Impostor Images

For our virtual human's lowest LOD representation, we use pre-generated impostors based on the work of Tecchia et al. [TC00]. A set of template mesh models were used in the pre-generation of the necessary impostor images in 3D Studio MAX. To facilitate the introduction of colour and animation variation and to ensure that the pre-generated impostor matches its mesh counterpart, these models required additional setup steps to be implemented in 3D Studio MAX. The mesh's triangles were organised into groups where each group represented a particular body part (as shown in Figure 2(b) and (c)) and was assigned a specific pre-defined material. It should be noted that the diffuse colour of each material is set to white (as shown in Figure 2(a)) to allow colour modulation of the pre-generated impostors, which will be discussed later. The meshes in our system use a single image for the detail of the character and this was grey-scaled in 3D Studio MAX to allow colour modulation without losing detail.

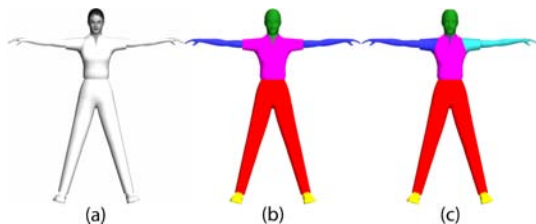


Figure 2: (a) High LOD mesh representation in 3D Studio MAX. (b) The grouping of triangles based on material used (shown by the different colours). (c) The grouping of triangles based on the body part it represents (shown by the different colours).

Once these additional steps were carried out, the mesh was skinned and a walk animation was created for the underlying skeleton. This key-framed animation was created using Character Studio's footstep creation tool and consisted of a one second, cyclical animation with a key-frame occurring every 100 milliseconds (10Hz). While animations are typically sampled at a minimum of 30Hz, 10Hz was used in the system to reduce the virtual human's memory footprint. With regards to the default walk animation, it is important that both the mesh model and the motion are symmetrical in order to minimize the amount of texture memory the impostor images consume. This halves the number of viewpoints from which the model needs to be rendered, since a viewpoint image for a particular key-frame can be mirrored to obtain the opposite viewpoint for the corresponding symmetrical key-frame. Figure 3 illustrates a walk animation, where there is a difference of five key-frames between each pair of symmetrical key-frames. In the case of asymmetric animation, such as a side-step left or right motion, impostor images need to be generated around both sides of the model, doubling the amount of memory consumed. However, the impostor images only need to be generated for a side-step left motion since it can be mirrored to obtain a side-step right motion. Additionally, a side-step motion is typically short in duration (e.g., 0.5 seconds) and therefore less key-frames are needed.

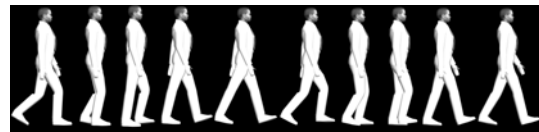


Figure 3: Precalculating and storing the deformation of a mesh performing a walk animation for 10 key-frames.

A MaxScript plug-in was written to render the images needed by the impostor representation in 3D Studio Max. The process used is illustrated in Figure 4. The plug-in positions the virtual human mesh model at the center of a sphere consisting of 32 segments and a radius equal to the distance from which we wish to render the impostor images. For 10 frames of animation, a detail map image and a normal map

image are rendered from 17 viewpoints around one side of the model and from 8 elevations:

- **Impostor detail map**

This image is used to store the detail of the mesh's decal texture for each viewpoint. It is generated by rendering the mesh, with shading and anti-aliasing disabled, into an image of 256×256 pixels. To allow for variation, each pixel in the image's alpha channel needs to be encoded with a specific alpha value associated with the material at that particular pixel. In order to do this, the plug-in utilizes 3D Studio Max's Graphics buffer or *G-buffer* which allows data such as object ID, material ID, and UV coordinates to be stored in a number of separate channels. The plug-in stores the material ID at each pixel in the G-buffer and these values are used to lookup and store the associated alpha value in the alpha-channel. Background pixels are assigned an alpha value of 255 to distinguish which pixels need to be transparent when displaying the impostor at run-time.

- **Impostor normal map**

This image is used to store the mesh's surface normals in eye-space for each pixel in the detail map. The normal maps in [DHO05] took a considerable time to generate, as per-pixel look ups and operations were needed, so we improved the algorithm by using a less computationally intensive technique. A copy of the character's mesh at the current frame was first needed. Each vertex normal was first converted into eye-space coordinates, to find the normal with respect to the camera, and then converted into an RGB colour (using Equation 1). Per-vertex colouring was then used to paint the RGB colours onto the vertices of the copied meshes (3D Studio Max's VertexPaint modifier was used to do this). These vertex colours were interpolated over the polygons, creating a character mesh with normal map colours. The normal map image was then generated by rendering an image of this mesh, from the current viewpoint. Per-vertex colouring and interpolation are operations that are performed very quickly, as they are supported by graphics hardware. This meant that the image could be produced almost immediately, without the need for slow per-pixel operations.

$$\begin{aligned} Pixel_R &= ((0.5 * N_x) + 0.5) * 255 \\ Pixel_G &= ((0.5 * N_y) + 0.5) * 255 \\ Pixel_B &= ((0.5 * N_z) + 0.5) * 255 \end{aligned} \quad (1)$$

Once these images have been generated, the plug-in removes any unused space and combines them into a single detail and normal map image of 1024×1024 pixels for a particular frame of animation. For each frame of animation impostor image, the data needed to render each viewpoint at run-time is stored in a text-based Impostor Data File (IDF). This file includes each viewpoint's row and column ID, position,

width, height, and position of the parent bone of the model's skeleton within the image.

4. Rendering of the Impostor Model

The main problem with using a pre-generated impostor approach is the consumption of texture memory. In order to render a dynamically lit impostor, an impostor detail image and a normal map image are required for each frame of animation. The RGBA impostor detail image contains four channels ($1024 \times 1024 \times 4$ bytes) and the RGB normal map image contains three channels ($1024 \times 1024 \times 3$ bytes), resulting in 7MB of texture memory being required for a single frame of animation. By using DXT3 texture compression, the memory requirements are reduced by a factor of four for RGBA images and by a factor of six for RGB images, resulting in only 1.5MB ($1024 \times 1024 \times 4 \times 1/4 + 1024 \times 1024 \times 3 \times 1/6$ bytes) of texture memory for each frame. Unfortunately, DXT3 texture compression is not particularly effective at compressing normal maps, as it results in noticeable block artefacts. These artefacts can be avoided by using 3Dc, which is ATI's new compression technology, and provides 4:1 compression of normal maps with image quality that is virtually indistinguishable from the uncompressed version [3Dc]. Another way of reducing these artefacts is to store one of the components in the alpha channel and then use DXT5 compression, which compresses the alpha values independently at a higher accuracy [ATI].

Our impostor representation is capable of using *mipmapping* techniques [Wil83]. Mipmapping avoids visual artefacts that occur when textures are mapped onto smaller dynamic objects, causing them to shimmer. OpenGL allows the generation of a series of pre-filtered texture maps of decreasing resolutions, called *mipmaps*, which are selected based on the size (in pixels) of the object being mapped. Although mipmapping requires some extra computation and texture storage (which is increased by a third), this is necessary to maintain the impostor's realism when displayed at a distance. However, care has to be taken not to generate mipmaps at too low a resolution, as this causes other artefacts due to the averaging of several viewpoint images within the mipmap.

Given the amount of texture memory required by the system, we need a method of improving the variety and visual interest of large crowds of impostors, while keeping memory usage to a minimum and ensuring that rendering speed is uncompromised. Our contribution in this area is that we improve upon existing impostor techniques for adding variety by taking advantage of recent improvements in programmable graphics hardware in order to perform an arbitrary number of colour changes in one pass. Since the colouring regions are encoded in the alpha channel (as described in Section 3), this number is limited only by that channel's precision. Our further contribution is the real-time shading of the impostors implemented in programmable hardware.

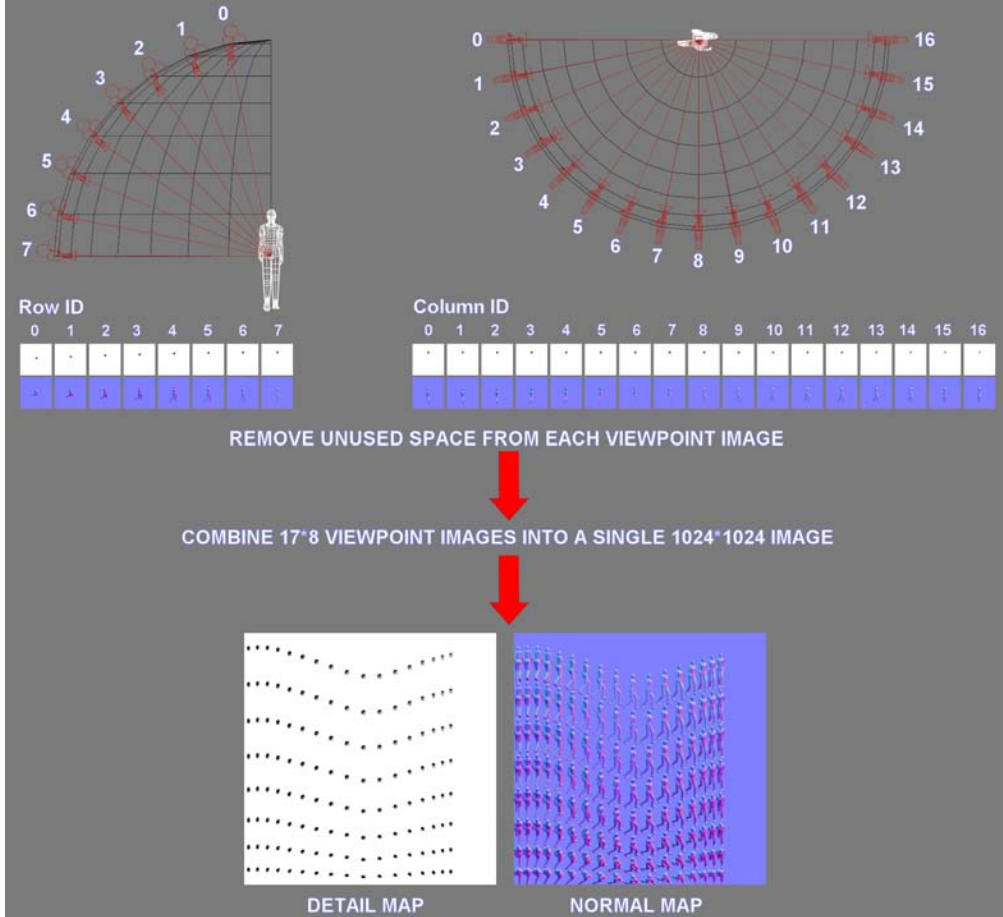


Figure 4: A MaxScript plug-in removes unused space from each viewpoint image and combines 17*8 viewpoint images into a single 1024x1024 image for a particular frame.

To render the impostors, we need to calculate which viewpoint image needs to be displayed and rotate its quadrilateral so that it always faces the viewer. Using the position of the virtual human’s root bone \vec{H} and the camera’s position \vec{C} , the quadrilateral’s normal vector \vec{N} can be calculated using Equation 2.

$$\vec{N} = \frac{\vec{H} - \vec{C}}{|\vec{H} - \vec{C}|} \quad (2)$$

The vector from the camera to the human projected onto the ground plane \vec{CH} can be calculated (Equation 3) using \vec{N} . It should be noted that in Equation 2, it is assumed that the ground is the XZ plane and that the camera’s position cannot be lower than the ground. Therefore, it is not necessary to pre-generate any viewpoint images from these elevations.

$$\vec{CH} = \frac{(N_x, 0, N_z)}{|(N_x, 0, N_z)|} \quad (3)$$

The amount by which to rotate the quadrilateral around the x-axis θ_x and y-axis θ_y is calculated using Equation 4. The viewpoint’s row and column ID (V_{Row} and V_{Column}) can be used to lookup which viewpoint to render using Equation 5, where N_x and N_y are the number of viewpoint images pre-generated around the x- and y-axis.

$$\begin{aligned} \theta_x &= \cos^{-1}(N_y) \\ \theta_y &= \cos^{-1}(CH_z) \end{aligned} \quad (4)$$

S. Dobbyn, R. McDonnell & C. O'Sullivan / Real-Time Crowd Rendering with Pre-Generated Impostors

$$\begin{aligned} V_{Row} &= \theta_x \times \frac{N_x}{90} \\ V_{Column} &= \theta_y \times \frac{N_y}{180} \end{aligned} \quad (5)$$

For improving realism, interactive lighting of impostors is highly desirable. Additionally, since we are presenting a hybrid system that switches between two representations, it is crucial that there is no difference in the shading of each representation for the interchange to be imperceptible to the viewer. By using a per-pixel dot product between the light vector and a normal map image, Tecchia et al. [TLC02] computed the final shaded value of a pixel through multi-pass rendering, which required a minimum of five rendering passes. However, multi-pass rendering can have a detrimental effect on rendering time, which limits the number of impostors that can be shaded in real-time.

We improve upon this technique by taking advantage of programmable graphics hardware and shade the impostors in a single pass. The impostors are rendered with the same lighting and material properties as the mesh representation, and thus the shading of the impostor is based on Equation 6.

$$\begin{aligned} Pixel_{Colour} &= DetailTexture_{RGB} * \\ & (Ambient_{LightModel} * Ambient_{Material} + \\ & (MAX(Vector_{Light} \cdot Normal_{Vertex}), 0) * \\ & (Diffuse_{Light} * Diffuse_{Material})) \end{aligned} \quad (6)$$

Similar to the mesh representation, the lighting of the impostor representation has been implemented in hardware using both texture shaders and register combiners [NVR99], and vertex and fragment programs [Ver02, Fra02]. This involves implementing Equation 7 in hardware, whereby the per-pixel dot products of the light vector and the pre-generated normal map is multiplied with each pixel in the coloured region map (which will be discussed in the next section) to produce a shaded coloured region map. This result is added to an ambient term, and multiplied with the detail map to yield the final lit, coloured pixels. The overall shading and colouring sequence is illustrated in Figure 5.

$$\begin{aligned} Pixel_{Colour} &= DetailMap_{RGB} * \\ & (Ambient_{LightModel} * Ambient_{Material} + \\ & (MAX((Vector_{Light} \cdot NormalMap_{RGB}), 0) * \\ & (ColourMap [DetailMap_{\alpha}] * Diffuse_{Light})) \end{aligned} \quad (7)$$

Similar to the mesh model, we optimise the rendering of the impostors by precalculating and storing each of the key-frame's viewpoint data in a single VBO object. Since dynamically orientating the quad involves the computationally

expensive \cos^{-1} function (see Equation 2), we use a lookuptable (LUT) of \cos^{-1} values instead. A LUT is typically an array used to replace a run-time computation with a simpler lookup operation and can provide a significant speed gain.

5. Variation LOD: Adding Variety to the Impostor Model's Appearance

At the lowest level of variety ($Variation_{LOD}$), individuals in a crowd use the same model and are a carbon copy of each other. While this level (or lack) of variety reduces the load on the limited computational resources per frame, this is only suitable for a specific type of crowd without having a disconcerting effect on the viewer e.g., the army of droids in Star Wars: Attack of the Clones. To increase a model's level of variety regarding its appearance, changing the colours of a virtual human's clothing and skin is a method that is simple and yet has high visual impact when viewed in a crowd.

In order to do this, we use a set of different template human meshes and change their appearance by using different "outfits". Outfits define a set of colours for the virtual human's skin and clothes, where each colour is associated with a specific body part material. The production of these outfits is controlled entirely by artist-drawn textures produced in an 'Outfit Editor' application, allowing a quick and easy method of producing many different colour maps that are realistic and suitable to the model being rendered. The outfit editor is a 3D Studio MAX plug-in that allows the artist to select particular colors for each body material from a colour palette (see Figure 6). The impostor can be rendered in 3D Studio MAX's viewport in real-time using a shader written in HLSL to give the artist immediate feedback.

A multi-pass method, as described in [TLC02], achieves this goal by performing a rendering pass for every different region of colour that needs to be changed. We exploit the programmability of graphics hardware to efficiently increase the variety and interest of each impostor. In order to match the virtual human's geometric representation, the impostors must also be able to change colour, depending on the human model and outfit materials. We achieve this by storing distinct material IDs in the alpha channel of the impostor detail image upon generation, and use these IDs to address a changeable colour map at run-time. We perform a lookup on the detail map, using the alpha-encoded material IDs to address a colour map texture that can be altered to match the outfit of the virtual human currently being rendered (Figure 7). It should be noted that, since the alpha channel of the impostor's detail map contains alpha encoded regions, nearest filtering needs to be used. Otherwise, linear filtering results in the linear interpolation of these values when the impostor representation is at a distance, causing shading artefacts due to the wrong outfit colour being looked up. This problem can be solved by using a high-level shader written in the OpenGL shading language to linear filter the looked up color values [Gui05].

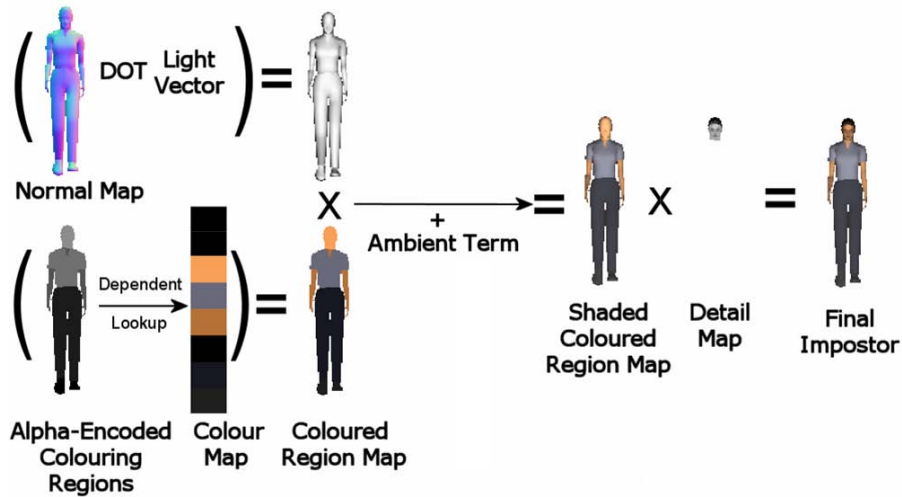


Figure 5: Impostor shading and colouring sequence.

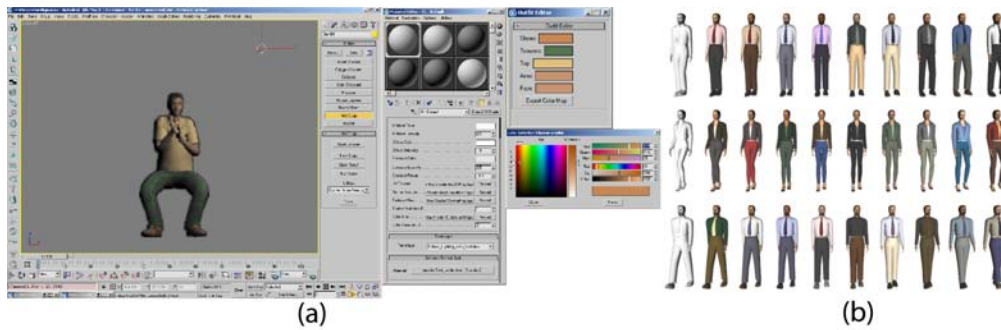


Figure 6: (a) Example of an artist in progress of generating an outfit for a model using the ‘Outfit Editor’ plug-in. (b) Nine outfits for three template meshes.

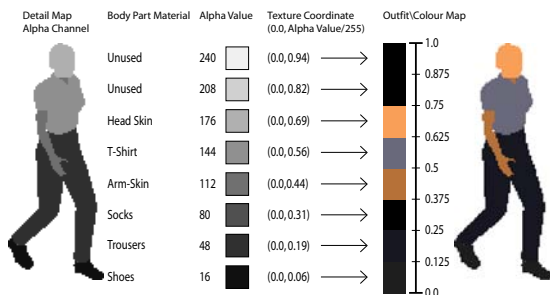


Figure 7: Using programmable texture addressing to add variety to the impostor representation.

6. Real-time Clothed Crowds with Pattern Variation

In the past, skinned meshes for the clothing of individuals in a simulated crowd were used, often resulting in rigid and un-

natural motion. While there have been advances in the area of cloth simulation, both offline and in real-time, interactive cloth simulation for hundreds or thousands of clothed characters would not be possible with current methods. In [DMK*06], we addressed this problem and devised a system for animating large numbers of clothed characters.

The majority of games that implement cloth dynamics on current generation hardware do so in a highly constrained manner, often ignoring the issues of collision detection by allowing the cloth surface to penetrate nearby surfaces. Game developers favour cloth that is highly controllable and tend to use more traditional methods of bone based skinning and pre-simulated vertex mesh animations if the performance of the cloth is critical to the game. In simulated crowd scenes for games, cloth is rarely if ever used due to the large numbers of polygons required to accurately capture the deformation of the cloth. However, deformable clothing adds greatly to the realism of the characters.

We have added realism to our crowd simulations by dressing the individuals in realistically simulated clothing, using an offline commercial cloth simulator, and integrating this into our real-time hybrid geometry/impostor rendering system. Additionally, we developed a technique for generating cyclical motion for pre-simulated cloth, which therefore moves in a fluid, realistic manner. Furthermore, we have added variety to our impostor representation by developing a new hardware rendering technique for adding pattern variety to the same cloth for different humans in a crowd. Our results show a system capable of rendering large realistic clothed crowds at interactive frame rates.

6.1. Brief overview of Cloth Simulation

Implementing realistic cloth dynamics in real-time game applications still represents a significant challenge for game developers. Simulating cloth deformation is a complex process both in terms of dynamics simulation and collision detection for the changing cloth shape. Many game developers have relied on more tractable but approximate solutions including the Verlet method [Jak01], and have restricted the simulation's complexity by only using a subset of the mesh to represent the cloth. Another method to introduce complex clothing is to generate the folds using cloth simulation and then use skinning to attach the clothing to the character. This method works well in some cases, but often results in the unrealistic bending of folds in the cloth, as the folds have to deform according to the skeleton of the character. This can make a long flowing skirt look like trousers with folds. Also, the important secondary motion of the cloth is lost in this case. In the animation research community, Cordier and Magnat-Thalmann [CMT05] use a data-driven approach for real-time processing of clothes. Vassilev et al. [VSC01] developed an efficient technique for dynamic cloth simulation using a mass-spring model.

6.2. Cyclical Cloth

In a real-time crowd system, the characters' animations are often cyclical in nature, so that they can be smoothly linked to allow them to move in a fluid manner. Cyclical animations are commonly obtained by manually altering the underlying skeletal motion so that they loop in a realistic looking manner. However, making looping animations using characters with pre-simulated clothing is a more difficult task, as manual cleanup of the cloth to make it cyclical is very time-consuming, particularly for very deformable items of clothing like skirts, and can result in unrealistic effects.

We wanted a more automatic way of generating cyclical cloth and began by creating a very long animated sequence, repeating the animation of the human many times and simulating the cloth in response to the repeating animation, in the expectation that it would at some point become periodic. On viewing these long sequences, it was found that the cloth did not always settle to a periodic state, particularly for highly

deformable clothing such as long flowing skirts. A more robust method was needed in order to obtain a good cycle in the cloth. In a good cloth cycle, the cloth at the start frame F_s and at the end frame F_e of the animation cycle of length l should be the same, and be travelling at the same velocity. The long cloth sequence needed to be searched using a distance metric that took into account all of the vertices on the cloth mesh between the two frames of animation, in order to find one correctly cyclical loop.



Figure 8: Edge Images taken from 5 different viewpoints.

We used a distance metric similar to Kovar et al. [KGP02] to compute the differences between all frames that were of length l apart in the sequence, and chose a set of candidate cycles whose distances were below a user set threshold. Usually, we chose the 5 cycles with the lowest distance metric as candidate cycles. In the case of stiff clothing, where the motion is very restricted, picking the cycle with the smallest distance metric was often enough to produce a good cyclical motion. For other more deformable, flowing clothing, this metric was often insufficient, as it did not weight the importance of the folds in the cloth, but rather, weighted all points equally.

Bhat et al. [BTH*03] showed that the human perceptual system is sensitive to moving edges, and used this to compare the folds and silhouettes of simulated cloth to that of video cloth sequences to find the difference between them. We based the second pass of our algorithm on this idea of matching folds and silhouettes, and devised a metric for comparing the candidate cloth cycles at F_s and at F_e . For each of the candidate cycles, we generated images of the cloth at F_s and F_e from 5 different viewpoints around the skirt. The images at F_s and at F_e for each of the viewpoints were then converted into edge images (Figure 8), using the standard Canny edge detection algorithm [Can86]. The mean distance between edges in the corresponding images of F_s and F_e were then found using an edge distance estimator, and the resulting differences in the 5 images were summed together, to give a final difference metric for the candidate cycle. The cloth cycle with the smallest edge difference was chosen as the final cycle. In most cases, this final cloth cycle was good enough for use, but in certain cases, an extra linear blend step between F_e and F_s was needed to produce the final cycle. We tried to avoid linear blending where possible, as it often resulted in the cloth intersecting with the human model, and we also felt that the results were more natural when blending was not used.

This method produced cloth that appeared cyclical from all viewpoints for all of the clothing that we tested. Cyclical appearance was judged by whether or not it was possible to notice a discontinuity in the cloth motion at the start and end of

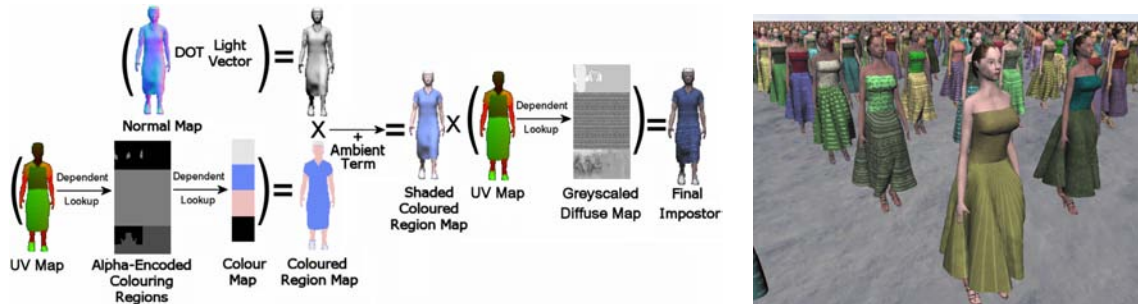


Figure 9: (a) UV mapped impostor rendering sequence and (b) an example of adding variation to one character using 8 different diffuse textures.

the cycle from all viewpoints (i.e., the looped animation did not exhibit a flicker in the cloth). We found that tighter clothing needed few animation cycles to produce cyclical cloth (sometimes as few as 4 cycles), as the cloth settled to a near periodic state quickly. Whereas looser cloth needed up to 40 animation cycles to produce a nice cyclical animation. The cyclical cloth and human animations could then be exported into a real-time system and replayed, and impostors were also generated as described next.

6.3. Cloth Geopostors

Our clothed crowd system builds on our Geopostor system [DHOO05], described earlier. Furthermore, our system includes clothed characters by using commercial software [CloFX] to obtain our cloth simulations, but any high quality offline simulator could be used to produce the cloth animation. We pre-simulate the deformation of both the virtual human's skin mesh using linear blend skinning and its cloth mesh using the physical simulator, based on the motion of its underlying skeleton. However, while the secondary motion of the character's cloth greatly adds to our crowd's visual realism, cyclical cloth motion is necessary to avoid any jerky motion artefacts and we present a technique to solve this in Section 6.2.

Once the character's meshes are pre-simulated, they are then exported and stored in separate keyframed meshes or "poses". By pre-calculating and storing the deformation of the skin and cloth mesh in poses, this avoids the cost of deforming the character's body and simulating its clothes at run-time. Generating the impostor representation of our clothed character involves capturing two types of images from a number of viewpoints around the model: a detail map image to capture the detail of the model's diffuse texture and a normal map image whereby the model's surface normals are encoded as an RGB value. At run-time, the clothed virtual humans switch between the two level of detail (LOD) representations depending on their position with respect to the viewer.

Adding colour variation to an impostor representation has already been achieved through the encoding of colouring regions in the alpha channel of the detail image, as described above. This is used at run-time to address a colour or "outfit"

map through programmable texture addressing. In the case of a mesh, another method to add texture variation is to provide it with a set of different diffuse textures with which it can be texture mapped. However, applying this type of variation to the impostor would require exporting a set of detail maps for each different diffuse texture used, resulting in the rapid consumption of texture memory. To solve this problem, we propose replacing the detail map with a UV map.

6.4. UV Mapping Technique

We improve upon existing impostor techniques for adding variety by replacing the detail map images (described in Section 6.3) with a texture coordinate map or *UV map*. This is similar to a normal map whereby it is generated for each viewpoint and contains the texture coordinates of the model's surface encoded as a RGB value. At run-time, these values are used to lookup the same set of diffuse textures used by the mesh, allowing texture variation for both the human's skin and cloth. To also allow for colour variation, the alpha channel of the mesh's diffuse textures is encoded with alpha encoded regions which are used to lookup the colour map. The overall sequence for shading and adding both colour and texture variation to the impostor representation is shown in Figure 9. Before the impostor's UV mapped images are pre-generated, texture seams should be kept to a minimum when texture mapping the associated mesh. Otherwise, these seams result in incorrect texture coordinates being stored in the UV map, which causes rendering artefacts to arise at run-time due to the wrong pixels in the diffuse texture being addressed. A similar type of artefact also occurs when linear filtering is used, causing the background pixels and the impostor's silhouette to be linearly interpolated and generating incorrect texture coordinates. However, this is only noticeable when the impostor is close to the viewer.

This new type of image allows the texture variety and interest of each clothed impostor to be greatly increased. Replacing the detail map with the UV map image ameliorates the problem of trying to add the same type of variation using detail maps, which results in the consumption of large amounts of texture memory (see Section 6.5). Additionally, these UV maps could be further utilised to enhance the impostor's realism by applying various per-pixel lighting tex-

tures as specular maps, which are commonly used by high resolution game characters.

6.5. Results

Frame rate tests were carried out on the clothed crowd system to investigate how many clothed humans could be displayed using different LOD representations at 30 fps (see Figure 10). All of our tests were performed using a PentiumIV 3.6Ghz processor, with 2.0GB RAM and an ATI Radeon X850 XT Platinum Edition graphics card with 256MB of video memory. In each test, all of the humans were on the screen walking on the spot, and were dynamically lit.

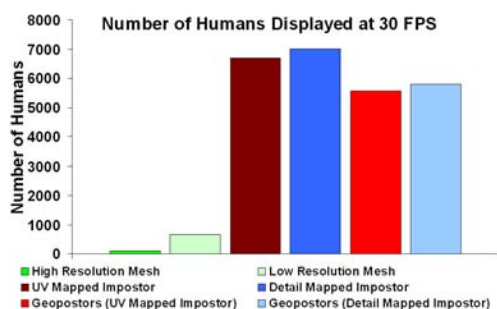


Figure 10: Number of humans displayed at 30 fps using different LOD representations.

It was found that 90 high resolution geometric models of clothed characters (13,056 triangles each) could be displayed onscreen at one time. With loss of visual quality, this number could be increased to 655 when low resolution geometry clothed characters were displayed (1,899 triangles each). Approximately 6,600 UV mapped impostors could be displayed at the same frame rate, but the closer humans appeared very pixellated. As expected, the number of detail mapped impostors displayed is higher due to the new UV map approach requiring extra texture lookups and per-pixel operations. When we used a hybrid geometry/impostor approach as in [DHOO05], up to 6,000 humans could be displayed (15 high resolution, 5,985 UV mapped impostors) which allowed visual quality and real-time performance to be maintained. Visual fidelity was maintained as impostors were displayed at the 1:1 pixel-to-textel ratio, where they are perceptually equivalent to high resolution meshes [Ham05]. Furthermore, in the case of switching between a clothed character's mesh and impostor representation, adding variation to the cloth using the detail mapped approach requires a substantially larger amount of texture memory in comparison to the UV mapped impostor (see Figure 11). These calculations use a single clothed character and assume that the detail mapped and UV mapped impostor consist of 10 frames of animation, each normal map being a 1024x1024 sized RGB image and both the detail map and UV map are 1024x1024 sized RGBA images. Additionally, each diffuse texture used by the clothed character's mesh representation

is a 1024x1024 RGBA sized image. DXT3 texture compression is used in these calculations to reduce the memory requirements by 4 for all RGBA images and by 6 for all RGB images.

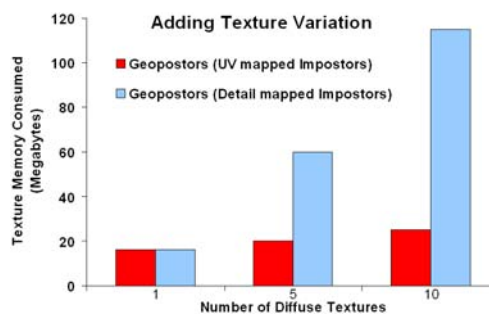


Figure 11: Texture memory consumed by adding pattern variation to the Geopostor system using UV mapped and detail mapped impostors for a single clothed character.

7. Animation LOD: Adding Variety to the Impostor Model's Animation

Similar to the mesh model, we add variety to the animation at a lower level of detail by pre-generating the template model's impostor images using the same default animations, that can reflect the age and gender of the model. To avoid the impostors moving in step, each virtual human's animation is offset by a particular number of frames to achieve a more varied crowd motion. However, since each animation key-frame is stored in a separate texture, this type of variation is limited depending on the number of textures needed in a single frame.

Increasing an impostor representation's sense of individualism is a tricky problem, since it is limited to the animation used in the pre-generation of its images. We solve this problem by layering head and arm gestures on top of the default impostor animation, whereby a particular body-part in the impostor image is replaced with a gesturing mesh representing the body-part. Since each body-part of the impostor is represented by a particular alpha value in the detail image's alpha channel, the impostor can be rendered without these body-parts by changing the alpha function accordingly. Using the corresponding mesh's skeleton, the gesturing bones are updated and the affected part of the mesh is deformed and rendered (Figure 12). The main advantage of this approach is that it avoids the cost of deforming and rendering the entire mesh by replacing it with the impostor representation. Thus, only the triangles affected by the gesturing bones need to be rendered. While minor rendering artefacts can appear caused by the layering of the mesh on top of the impostor, these can be removed through blending.

The problem with this method is that, depending on the viewpoint being displayed, holes appear when a body part is not rendered since the body part may sometimes be occluding other areas of the impostor. When the virtual human



Figure 12: Adding variety to the virtual human model's animation by layering head and arm gestures on top of the default walk animation.

performs a head gesture this artefact is not as much of a problem as when they are performing an arm gesture. Currently, virtual humans that are rendered with an impostor representation switch to a low resolution mesh representation when they request an arm animation. As a possible solution, dynamically generated impostors could be used to render the virtual human's body without its arms and this will be investigated in future work.

8. Virtual Human LOD Shadows

Our run-time system enhances the realism of the virtual humans and the environment they inhabit by creating shadows on the ground wherever the light is blocked. Our shadow technique is based on the planar projected shadow algorithm and is implemented in hardware using per-pixel stencil testing. This section will describe how this technique is used to render the virtual humans' shadows.

The planar projected shadow algorithm is used to cast a geometric model's shadow onto a ground plane based on the light's position. In order to achieve this, a planar projected shadow matrix can be constructed. Given the equation for a ground plane $G: \vec{N} + d = 0$ and the homogenous position of the light \vec{L} , a 4×4 planar projected shadow matrix S can be constructed using Equation 8 (see [Bli88] [HMAM02] for the derivation of the matrix).

$$S = \begin{pmatrix} D - L_x * N_x & -L_x * N_y & L_x * N_z & -L_x * d \\ -L_y * N_x & D - L_y * N_y & -L_y * N_z & -L_y * d \\ -L_z * N_x & -L_z * N_y & D - L_z * N_z & -L_z * d \\ -L_w * N_x & -L_w * N_y & -L_w * N_z & D - L_w * d \end{pmatrix} \quad (8)$$

$$\text{where } D = N_x * L_x + N_y * L_y + N_z * L_z + d * L_w$$

Stenciling works by tagging pixels in one rendering pass to control their update in subsequent rendering passes. It is an

extra per-pixel test that uses the stencil buffer to track the stencil value of each pixel. When the stencil test is enabled, the frame buffer's stencil values are used to accept or reject rasterized fragments. When rendering the scene, the stencil buffer is cleared at the beginning and a unique non-zero stencil value is assigned to pixels belonging to the ground plane. In the first rendering pass, the shadow cast by each virtual human's geometric representation is rendered. Using the matrix S , the geometry is projected onto the ground plane and rendered into the stencil buffer, where each pixel is tagged with the ground plane's unique stencil value. In the subsequent rendering pass, each virtual human's representation is rendered and the appropriate areas of the stencil buffer are simultaneously cleared. This prevents an artefact whereby shadows might overwrite real objects, damaging the realism of the scene. Finally, a single semi-transparent quad is rendered over the whole scene (where the stencil buffer pixels have been set to the unique stencil value) resulting in realistically blended shadows.

Our shadow technique uses a LOD approach, where either the impostor or mesh representation is projected onto the ground plane depending on which LOD representation the virtual human is currently using (see Figure 13 (a) and (b)). To render the virtual human's shadow using the impostor representation, we need to calculate which viewpoint image needs to be displayed with respect to the light's position and rotate its quadrilateral so that it always faces the light. Using the virtual human's position \vec{H} and the light's position \vec{L} , the quadrilateral's normal vector \vec{N} can be calculated using Equation 9. The projection of the impostor onto the ground plane with respect to the light position can be calculated using \vec{N} and Equations 3 and 5 (previously described in Section 4). The impostor's shadow requires no more than a single textured quad, and therefore is extremely fast to render.

$$\vec{N} = \frac{\vec{H} - \vec{L}}{|\vec{H} - \vec{L}|} \quad (9)$$

While this method is similar to that employed by Loscos et al. [LTC01], our use of the stencil buffer instead of darkened textures results in shadows that blend realistically with both the underlying world and each other (see Figure 13 (d)). The main advantage of implementing this shadow algorithm with the stencil buffer is that it can avoid artefacts caused by double blending and can limit the shadow to an arbitrary ground plane surface. Unfortunately, unlike full geometric stencil shadows, our projection shadows are restricted to the ground plane and do not project onto nearby static objects, or other dynamic objects. While shadow mapping could be used to solve this problem, a LOD approach would be needed to deal with the many hundreds or thousands of shadows. It should be noted that shadow volumes were not considered in the system as this technique can decrease the pixel fill rate and

S. Dobbyn, R. McDonnell & C. O'Sullivan / Real-Time Crowd Rendering with Pre-Generated Impostors

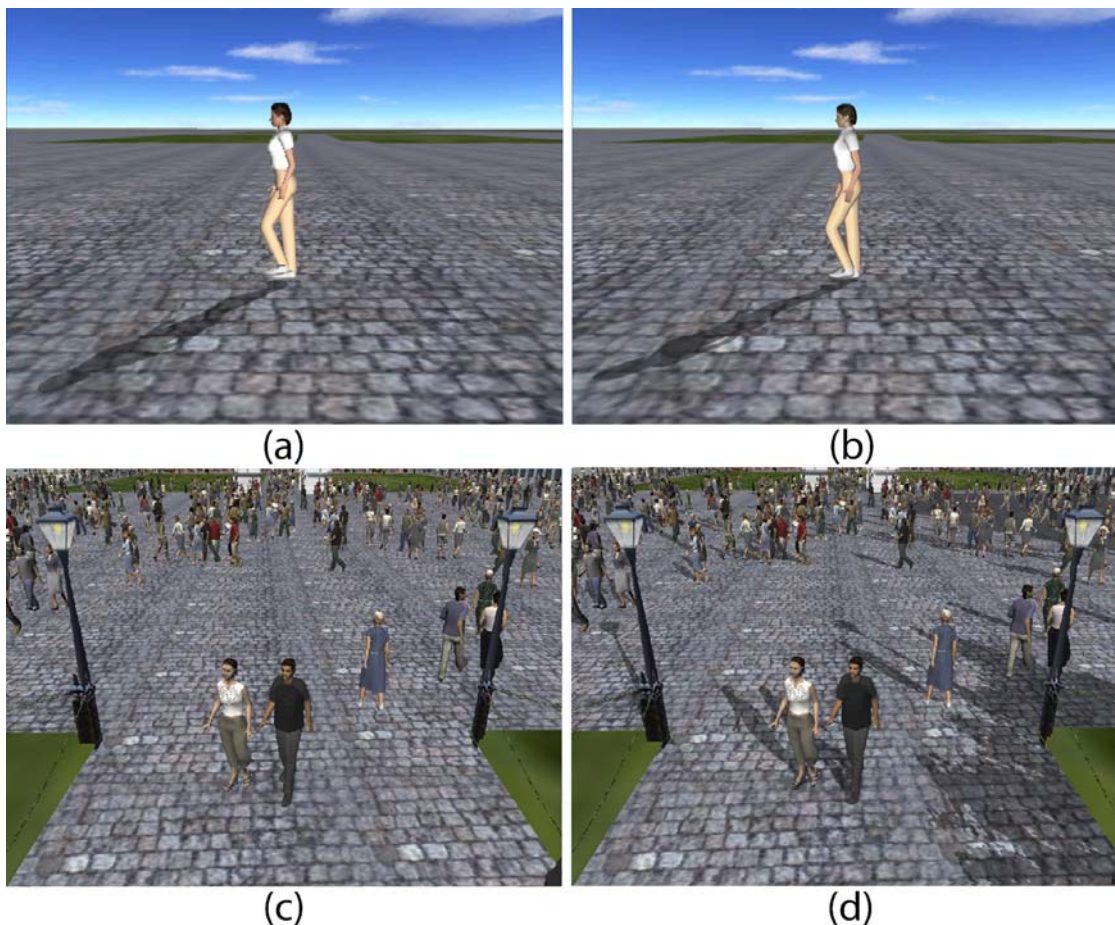


Figure 13: (a) Projected impostor shadow. (b) Projected mesh shadow. (c) Crowd and city without shadows. (d) Crowd and city with projected LOD shadows.

the constructed shadow volume for an impostor is incorrect as a result of being a semi-transparent quadrilateral.

9. Performance Optimisations

9.1. Virtual Human Occlusion Culling

As a first step towards improving performance, view frustum culling can be used to eliminate those humans that are not potentially on screen. However, due to the densely occluded nature of an urban environment, large groups of humans may be in the frustum but occluded by buildings and therefore rendered unnecessarily. By avoiding the rendering of these humans using occlusion culling techniques, this should greatly improve the performance of the system [CT97, BHS98, SVN99, WS99, Zha98].

We make use of hardware accelerated occlusion culling similar to the technique used by Saulters et al. [SF02]

to cull large sections of the crowd. We utilise the *ARB_occlusion_query* extension to determine the visibility of an object. This extension defines a mechanism whereby an application can query the number of pixels drawn by a primitive or group of primitives. Typically, the major occluders are rendered and an occlusion query for the bounding box of an object in the scene is performed. If a pixel is drawn for that object's bounding box, then the object is not occluded and therefore should be displayed. The main performance advantage of this extension is that it allows for parallelism between the CPU and GPU, since many queries can be issued before asking for the result of any one. This means that more useful work, such as the rendering of other objects or other computations on the CPU, can be carried out while waiting for the occlusion query results to be returned.

Since the city is populated by several thousand humans, there could potentially be a large number of humans in the

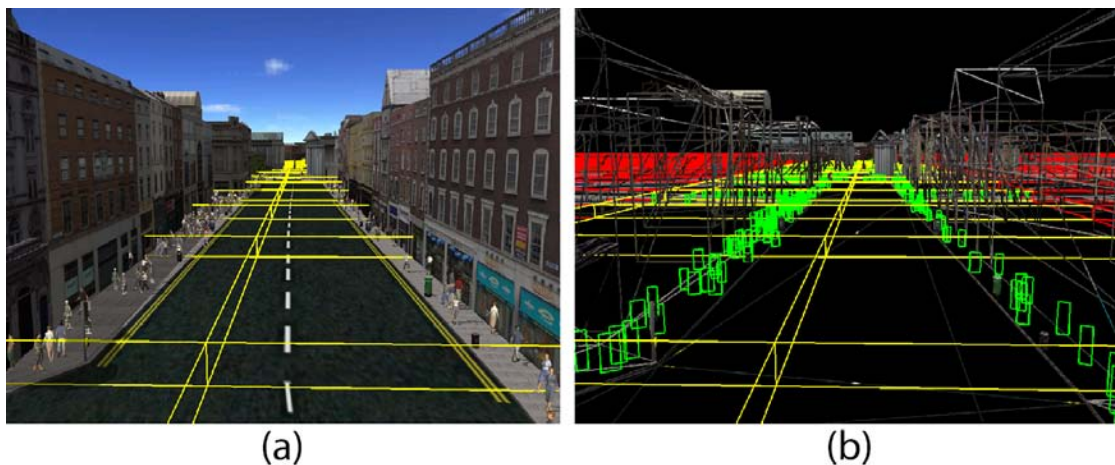


Figure 14: Occlusion Culling: (a) Environment is divided into nodes to facilitate occlusion culling. (b) Characters that are in unoccluded nodes (shown in yellow) are drawn while those that are in occluded nodes (shown in red) are discarded.

view frustum and therefore it would be computationally inefficient to perform a separate occlusion query for each human. To facilitate the occlusion culling of buildings, the virtual city is divided into a grid of regular-sized nodes (see Figure 14(a)). By re-using these nodes so that they record which virtual humans inhabit them, this can help to avoid performing separate occlusion culling queries for each human. Having initially rendered the static environment, we perform occlusion queries on the bounding volume of any nodes in the view-frustum, thus allowing us to rapidly discard those nodes hidden by the environment and the humans within them (as shown in Figure 14(b)). With regards to the unoccluded nodes, we perform view-frustum culling on the virtual humans within these nodes, since parts of these nodes may not be within the view frustum. It should be noted that the height of each node's bounding volume is set to the height of the tallest virtual human used in the system to allow humans to still be displayed when they are behind an occluding object whose height is less (e.g., walls). This occlusion culling method could be extended so that the number of pixels drawn for a node could be used as a metric to decide on what level of detail the humans in the node should use, with regards to representation, behaviour, and animation.

9.2. Virtual Human Simulation LOD

While frustum and occlusion culling decrease the rendering workload, there are still overheads associated with updating the positions of thousands of humans in motion. To lighten the workload we pause humans within nodes that have not been visible for more than a certain number of seconds. This technique takes advantage of the fact that a large number of humans are occluded per frame and therefore their position in the world can remain unchanged without the viewer

noticing. By storing the time each node was last unoccluded, the position of a human is only updated if the node it inhabits has been unoccluded for the last five seconds. This time delay prevents temporal artefacts becoming noticeable amongst the nearby humans when performing rapid camera rotation. In addition to this, checking whether a node is occlusion culled is only performed every 100 milliseconds if the camera has moved or rotated, since the same nodes will be occluded if the camera remains stationary. Since the humans only move every 100 milliseconds, we reduce the number of times we check whether a human is within the view-frustum by performing this test every time the humans move instead of every frame.

However, simulation artefacts can arise when the camera's position remains static for a period of time and the humans move from an unoccluded node to an occluded node. This results in the congregating of humans on the boundary of these occluded nodes since their steering behaviour is not being updated. A potential solution to this problem would involve a LOD simulation approach whereby humans are updated at a frequency dependent on the last time the node was unoccluded.

9.3. Minimising OpenGL State Changes

OpenGL is a simple state machine with two operations: setting a state, and rendering utilizing that state. By minimizing the number of times a state needs to be set, this can maximize performance since it minimizes the amount of work the driver and the graphics card have to do. This technique is generally referred to as *state sorting* and attempts to organize rendering requests based around the types of state that will need to be updated. Generally, the goal is to attempt to

sort the render requests and state settings based upon the cost of setting that particular part of the OpenGL state.

With regards to our crowd, rendering is optimized by sorting the virtual humans in the following order based on the most to least expensive state changes: binding a shader, binding a texture, and setting VBO data pointers. By organising the rendering of our crowd in this manner, our approach sorts each virtual human by LOD representation, then by template model, and finally by the current key-frame of animation. Sorting the virtual humans by LOD representation minimizes the number of times that the following states have to be changed: the setting of lighting parameters, alpha test enabling and disabling, and vertex and fragment programs. Next, sorting the LOD representations based on template model minimizes texture loads and binds. Finally, sorting virtual humans using the same template model by animation key-frame reduces the setting of VBO data pointers, since each VBO stores the data for a particular key-frame. In the case of rendering virtual humans using the same model and animated with the same key-frame, an extra step needs to be implemented to sort them based on the viewpoint required with respect to the camera. This is necessary, since certain viewpoints for the current key-frame are obtained by mirroring the same viewpoint for the symmetrical key-frame. By sorting impostors based on whether the viewpoint is mirrored, this minimizes texture loads and binds.

9.4. Minimising Texture Thrashing

Texture thrashing can become a serious problem when populating a virtual city with crowds using a number of different pre-generated impostor models. In addition to each impostor model requiring 1.5MB of texture memory every frame, the city model will also require a certain amount of texture memory. Therefore, as the number of template models within the virtual city increases, texture thrashing will occur much sooner as a result of the extra texture memory being consumed by the city model. It should be noted that, in the case of real-time applications where the camera is fixed, say at eye-level, only 17 viewpoint images are needed for each frame of animation and therefore the consumption of texture memory is less of a problem. Since we wanted to implement a more generic system, where the camera can view the city from any height, 17 by 8 viewpoints are needed for the impostor representation.

However, as only a subset of the viewpoints in the impostor textures is being used every frame, we propose splitting the impostor detail and the normal map images into eight separate smaller *elevation* images containing the set of viewpoints pre-generated at each camera height. To facilitate the creation of these elevation images, an application was written in C to allow the positioning of viewpoint images within a larger image. The application reads in the 17 viewpoint images for a particular camera height and, based on the sum of these images' area, the minimum dimensions of the elevation image are calculated. Once the viewpoints have been

loaded in, the application allows the user to organise the viewpoints within the new elevation image. Unfortunately, since the area of each viewpoint image varies, it is not guaranteed that they will all fit within the minimum dimensions and therefore have to be increased by a factor of two along a single dimension. Once the user has got all the 17 images to fit, the new elevation image is exported (shown in Figure 15).

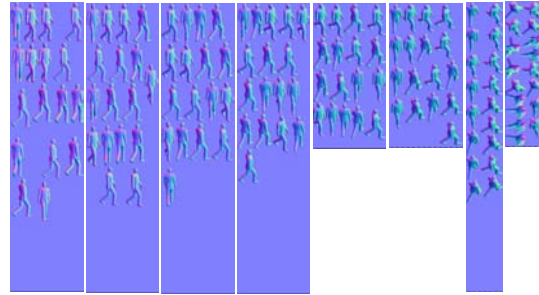


Figure 15: Normal map split into smaller elevation images.

The number of elevation images needed to render impostors using a particular human model type depends on the height of the camera and the distance of the camera from each impostor. Since buildings in a city environment generally occlude humans in the distance, all elevation images should never be needed simultaneously. The angle (θ_E) between the impostor and the camera around the horizontal axis, can be calculated using Equation 10, where h_{cam} is the camera height and d_{xz} is the distance on the x-z plane from the camera to the impostor. Using θ_E , the elevation image needed for that impostor can be calculated. As the camera's height decreases, the number of elevation images needed is reduced dramatically (see Equation 10). Taking advantage of the occluding nature of city environments, this method of separating impostor and normal map images for each elevation permits greater variety, without texture thrashing, as a result of each human model type consuming less texture memory. It should be noted that in order for the transitions between viewpoints to appear smooth, the perceptual metrics detailed in [MDCO06] should be employed. These metrics are dependent on character dimensions, with characters of large width to depth ratios requiring more viewpoint images than those with small width to depth ratios.

$$\theta_E = \tan^{-1}\left(\frac{h_{cam}}{d_{xz}}\right) \quad (10)$$

9.5. Optimisations For Spectator Crowds

In the case of crowds that do not move within the virtual environment, such as those found in sports games, viewpoint selection and the orienting of the billboard for each character can be done on the vertex processor. For each individ-



Figure 16: (a) Rendering thousands of characters in a single draw call. (b) Frame rate results.

ual, the billboard's vertices are set to the individual's position and the corresponding texture coordinates are set to the character's directional vector. These values are subsequently used by the vertex processor to dynamically rotate the vertices towards the camera view and lookup the texture coordinates for the most suitable viewpoint image. This means that the billboards of individuals can be batched together in a single vertex buffer object and rendered in a single draw call. For more details see [MR06]. It should be noted that smaller numbers of animation frames will result in bigger batch sizes, since only individuals using the same impostor texture can be batched together. Additionally, since the camera is typically limited to the playing field in sports games, pre-generating viewpoints from behind the character is not necessary, thus reducing the amount of texture memory consumed by the impostors.

As shown in Figure 16, rendering multiple instances in a single draw call greatly improves performance resulting in tens of thousands of characters drawn in real-time (see Figure 16). Two scenarios were tested to show the effect of batch size. The first test scenario involved one template model performing a single animation. The second scenario involved smaller batches as a result of using 2 template models performing one of 3 different animations. In both cases each animation was one second long and consisted of 30 key-frames. All of our tests were performed using a PentiumIV 3.6Ghz processor, with 2.0GB RAM and an NVidia Quadro FX 4400 graphics card with 512MB of video memory.

10. Short-Comings of the Pre-Generated Impostor Representation

While the impostor used in the Geopostor system is computationally efficient to render, the following short-comings are associated with this representation:

- **Anti-Aliasing:** Since the impostors are not rendered without anti-aliasing, this results in the silhouette being pixelated in appearance and is especially noticeable when the impostor is close to the viewer. Future work will investigate how anti-aliasing techniques would improve the impostor's visual appeal.

- **Models and animations need to be symmetric:** To reduce the number of viewpoint images needed, both the model and animation have to be symmetric in the XZ plane. If this is not possible then the impostor's texture will consume twice as much memory in order to fit the additional viewpoint images that are needed.
- **No viewpoint images generated from directly above or below the ground-plane:** No viewpoint images were generated from directly above the virtual human model or from below the ground-plane, resulting in parallax artefacts when the impostor is viewed from these camera angles. However, these viewpoints were not needed since the camera is not allowed to move below the ground plane in the city simulation system. The number of viewpoint images needed depends on what camera angles the impostors will be viewed from and this should be considered when generating the impostor's textures to minimize memory consumption.
- **Pixellated shadows when the sun is low in the sky:** Since the impostor texture are used in projecting ground-plane shadows (see Section 8), this results in the shadows being pixelated when the sun is low in the sky and is especially noticeable when the shadows are close to the viewer. In this case, the virtual human's mesh representation should be used in the projection of the shadow.

References

- [3Dc] 3dc white paper, ATI Technologies. <http://www.ati.com/products/radeonx800/3DcWhitePaper.pdf>.
- [ATI] Normal map compression, ATI Technologies. <http://ati.de/developer/NormalMapCompression.pdf>.
- [BHS98] BITTNER J., HAVRAN V., SLAVÍK P.: Hierarchical visibility culling with occlusion trees. pp. 207–219.
- [Bli88] BLINN J.: Me and my (fake) shadow. *IEEE Comput. Graph. Appl.* 8, 1 (1988), 82–86.
- [BTH*03] BHAT K. S., TWIGG C. D., HODGINS J. K., KHOSLA P. K., POPOVIC Z., SEITZ S. M.: Estimating cloth simulation parameters from video. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation* (2003), 37–51.

- [Can86] CANNY J.: A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI* 8 (1986), 679–698.
- [CloFX] Cloth simulation software, ClothFX.
- [CMT05] CORDIER F., MAGNENAT-THALMANN N.: A data-driven approach for real-time clothes simulation. *Computer Graphics Forum* 24, 2 (2005), 173–183.
- [CT97] COORG S., TELLER S.: Real-time occlusion culling for models with large occluders. *SI3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics* (1997), 83–90.
- [DHO05] DOBBYN S., HAMILL J., O'CONOR K., O'SULLIVAN C.: Geopostors: A real-time geometry / impostor crowd rendering system. *SI3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games* (April 2005), 95–102.
- [DMK*06] DOBBYN S., MCDONNELL R., KAVAN L., COLLINS S., O'SULLIVAN C.: Clothing the masses: Real-time clothed crowds with variation. In *Eurographics Short Papers (EG'06)* (September 2006), pp. 103 – 106.
- [Fra02] Arb_fragment_programs extension, Silicon Graphics. http://oss.sgi.com/projects/ogl-sample/registry/ARB/fragment_program.txt (2002).
- [Gui05] GUINOT J.: Image filtering with GLSL - convolution kernels. http://www.ozone3d.net/tutorials/image_filtering.php (2005).
- [Ham05] HAMILL J.: *Level of Detail Techniques for Real-Time Urban Simulation*. PhD thesis, University of Dublin, Trinity College, 2005.
- [HMAM02] HAINES E., MÖLLER T., AKENINE-MÖLLER T.: *Real-Time Rendering*. A.K. Peters, 2002.
- [Jak01] JAKOBSEN T.: Advanced character physics. In *Proceedings of the Game Developers Conference* (2001).
- [KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 473–482.
- [Lee02] LEESON W.: *Games Programming Gems III - Subdivision Surfaces for Character Animation*. Charles River Media, 2002, pp. 372–383.
- [LTC01] LOSCOS C., TECCHIA F., CHRYSANTHOU Y.: Real-time shadows for animated crowds in virtual cities. *VRST '01: Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (2001), 85–92.
- [MDC06] MCDONNELL R., DOBBYN S., COLLINS S., O'SULLIVAN C.: Perceptual evaluation of LOD clothing for virtual humans. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2006), pp. 117–126.
- [MR06] MILLÁN E., RUDOMÍN I.: Impostors and pseudo-instancing for gpu crowd rendering. In *Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia* (2006), pp. 49–55.
- [NVR99] NV_register_combiners extension, Silicon Graphics. http://oss.sgi.com/projects/ogl-sample/registry/NV/register_combiners.txt (1999).
- [OCV*02] O'SULLIVAN C., CASSELL J., VILHJÁLMS-SON H., DINGLIANA J., DOBBYN S., MCNAMEE B., PETERS C., GIANG T.: Levels of detail for crowds and groups. *Computer Graphics Forum* 21, 4 (2002), 733–742.
- [SF02] SAULTERS S., FERGUSON R.: Real-time rendering of dynamically variable scenes using hardware occlusion queries. *Proceedings of the Eurographics Ireland Rendering Workshop* (2002), 47–52.
- [SVNB99] SAONA-VÁZQUEZ C., NAVAZO I., BRUNET P.: The visibility octree: a data structure for 3d navigation. *Computers & Graphics* 23, 5 (1999), 635–643.
- [TC00] TECCHIA F., CHRYSANTHOU Y.: Real-time rendering of densely populated urban environments. *Proceedings of the Eurographics Workshop on Rendering Techniques* (2000), 83–88.
- [TLC02] TECCHIA F., LOSCOS C., CHRYSANTHOU Y.: Visualizing crowds in real-time. *Computer Graphics Forum* 21, 4 (2002), 753–765.
- [Ver02] Arb_vertex_programs extension, Silicon Graphics. http://oss.sgi.com/projects/ogl-sample/registry/ARB/vertex_program.txt (2002).
- [VSC01] VASSILEV T., SPANLANG B., CHRYSANTHOU Y.: Fast cloth animation on walking avatars. *Computer Graphics Forum* 20, 3 (2001), 260–267.
- [Wil83] WILLIAMS L.: Pyramidal parametrics. *SIGGRAPH '83: Proceedings of the 10th Annual Conference on Computer Graphics and Interactive Techniques* (1983), 1–11.
- [WS99] WONKA P., SCHMALSTIEG D.: Occluder shadows for fast walkthroughs of urban environments. *Computer Graphics Forum (Eurographics '99)* 18, 3 (1999), 51–60.
- [Zha98] ZHANG H.: *Effective Occlusion Culling for the Interactive Display of Arbitrary Models*. PhD thesis, 1998.

New Trends in 3D Video

Eurographics 2007 Tutorial T5

Organizers

Christian Theobalt (Stanford University, USA)
Stephan Würmlin (Liberovision AG, Switzerland)

Speakers

Christian Theobalt (Stanford University, USA)
Stephan Würmlin (Liberovision AG, Switzerland)
Edilson de Aguiar (MPI Informatik, Germany)
Christoph Niederberger (Liberovision AG, Switzerland)

Table of Contents**Tutorial Introduction**

Course Abstract	III
Syllabus	IV

Annotated Tutorial Slides

Introduction	1
Silhouette-based Methods	13
Stereo-based Methods	42
Model-based 3D Video – part I	94
Model-based 3D Video – part II	124
Free-Viewpoint Video Relighting	161
Applications	200
Outlook and Discussion	228
Presenters' Contact Information	232
Acknowledgements	233

Course Abstract

3D Video is an emerging and challenging research discipline that lives on the boundary between computer vision and computer graphics. The goal of researchers working in the field is to extract spatio-temporal models of dynamic scenes from multi-video footage in order to display them from user-controlled synthetic perspectives. 3D Video technology has the potential to lay the algorithmic foundations for a variety of intriguing new applications. This includes stunning novel visual effects for movies and computer games, as well as, facilitating the entire movie production pipeline by enabling virtual rearranging of cameras and lighting during post-processing. Furthermore, 3D Video processing will revolutionize visual media by enabling 3D TV and movies with interactive viewpoint control, or by enabling virtual fly-arounds during sports-broadcasts.

To achieve this purpose, several challenging problems from vision and graphics have to be solved simultaneously. The speakers in this course will explain the foundations of dynamic scene acquisition, dynamic scene reconstruction and dynamic scene rendering based on their own seminal work, as well as related approaches from the literature. They will explain in more detail three important categories of algorithms for dynamic shape and appearance reconstruction, namely silhouette-based, stereo-based, and model-based approaches. Alternative methods, such as data-driven approaches, will also be reviewed. The tutorial will focus on latest 3D Video techniques that were not yet covered in a tutorial, including algorithms for free-viewpoint video relighting, model-based deformable mesh tracking, as well as high-quality scene reconstruction with camera/projector setups. The course keeps a balance between the explanation of theoretical foundations, engineering problems and emerging applications of 3D Video technology. We therefore believe that the course will be a valuable and entertaining source of information for students, researchers and practitioners alike.

Syllabus

1. **Introduction (15 min)** - Speaker: Christian Theobalt
 - 3D Video - Why bother?
2. **Silhouette-based Methods (25 min)** - Speaker: Stephan Würmlin
 - Silhouette-based Methods - Foundations
 - Point Primitives for 3D Video
 - Real-time Applications in tele-presence systems (the blue-c)
3. **Stereo-based Methods (25 min)** - Speaker: Stephan Würmlin
 - Stereo-based Methods - Foundations
 - Using Camera Systems and Structured Light for High-quality 3D Video
 - Postprocessing Methods
4. **Model-based 3D Video I (25 min)** - Speaker: Christian Theobalt
 - Foundations
 - Marker-less Tracking and Dynamic Scene Reconstruction
 - Model-based 3D Video Rendering
5. **Break**
6. **Model-based 3D Video II (25 min)** - Speaker: Edilson de Aguiar
 - Alternative Model-based Approaches
 - Deformable Mesh Tracking for 3D Video
7. **Free-Viewpoint Video Relighting (25 min)** - Speaker: Christian Theobalt
 - Data-driven Dynamic Scene Relighting
 - Model-based Free-Viewpoint Video Relighting
 -
8. **Applications (30 min)** - Speaker: Christoph Niederberger
 - Authoring and Editing 3D Video
 - Applications of 3D Video in Movie and TV Production
 -
9. **Outlook And Discussion (10 min)** - Speaker: Stephan Würmlin

EG:130

- Conclusions
- Questions



New Trends in 3D Video

Half-day Tutorial

Christian Theobalt
Stanford University

Stephan Würmlin
ETH Zürich/LiberoVision

Edilson de Aguiar
MPI Informatik

Christoph Niederberger
ETH Zürich/LiberoVision

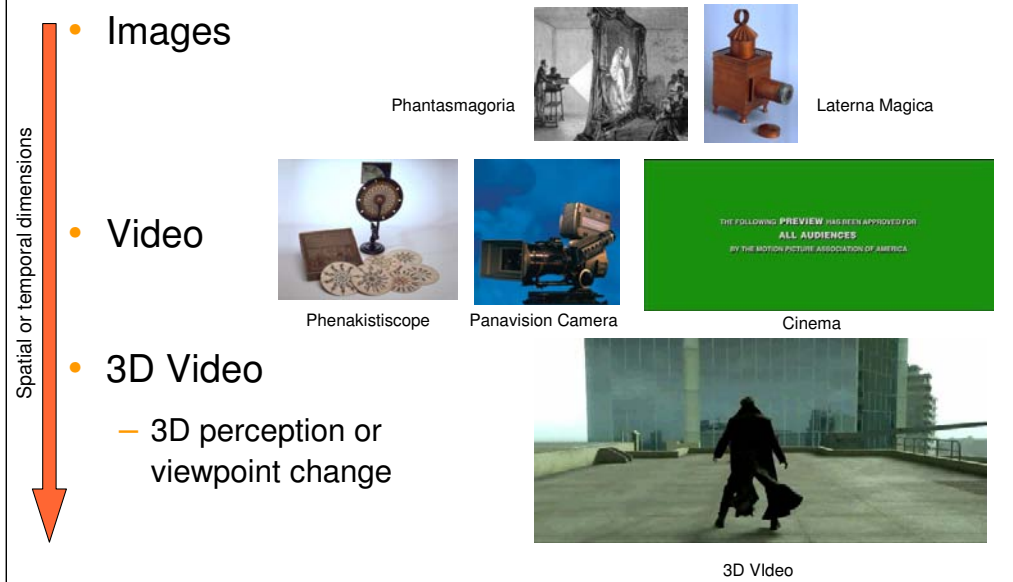


Introduction (15 min)

Christian Theobalt
Stanford University



Development of Visual Media



Vision is one of the most powerful senses that humans possess as it is one of the richest sources of psychological and physical stimuli. Visual media such as video or television capitalize on this fact and allow viewers to immerse with their imagination into scenes and events displayed to them.

In history, the ever ongoing technical improvement has caused several major changes in the way how visual media are produced and perceived. However, the most important change so far was due to the introduction of time as an additional dimension. While humans have been and are still fascinated to look at photographs, the first devices that were able to reproduce and capture moving images caused a major revolution that still dominates the type of visual media that we mostly use today, namely video (in its most general sense).

The availability of ever more powerful acquisition, computation and display hardware, has spawned a new field of research that aims at adding one more dimension to visual media, namely the third spatial dimension. This young and challenging field is still in its early days but, as we will show in this course, bears great potential to revolutionize visual media once more.

3D Video is a multi-faceted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

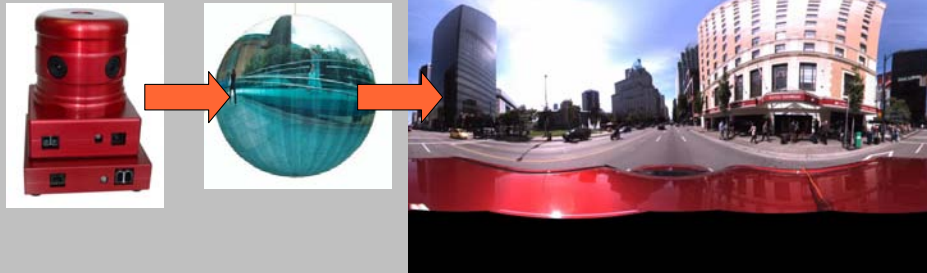
3D Video

The field of 3D video is multi-faceted as there exist several ways how the third dimension can be added. Here “3D” can, for instance, mean that the viewer is given the possibility to interact with a video and change his viewing direction on the fly while playing the content.

3D Video is a multi-faceted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

Omnidirectional Video



Free panning – predetermined viewpoints – no explicit 3D model

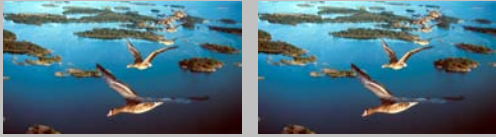
This type of immersive experience is, for example, generated by a technique known as omni-directional or panoramic video. Typically, this type of footage is recorded with an omni-directional camera. Such a camera either comprises of several synchronized cameras that simultaneously record all spherical directions (as the one in the image above), or of a normal camera and an attached panoramic mirror that also enables multidirectional recording. During display the captured footage is typically mapped onto a spherical or cylindrical surface such that the viewer can perform arbitrary rotations while traveling along a fixed path of camera positions.

Please refer to [1] for a detailed study of panoramic imaging techniques.


3D Video is a multi-faceted Field

Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation

3D Cinema




Left eye video Right eye video

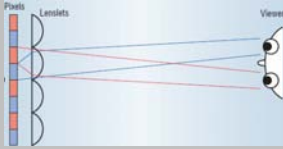


Polarized Glasses

Depth-image-based Video



Color video Depth video



Micro-Lens Display

A different type of 3D video is provided by 3D Cinema or related depth-image based methods. Here, the main goal is to generate a true 3D depth perception while displaying video streams. However, the viewer cannot change a virtual camera viewpoint interactively, but can merely move his head in a very confined space to experience parallax effects.

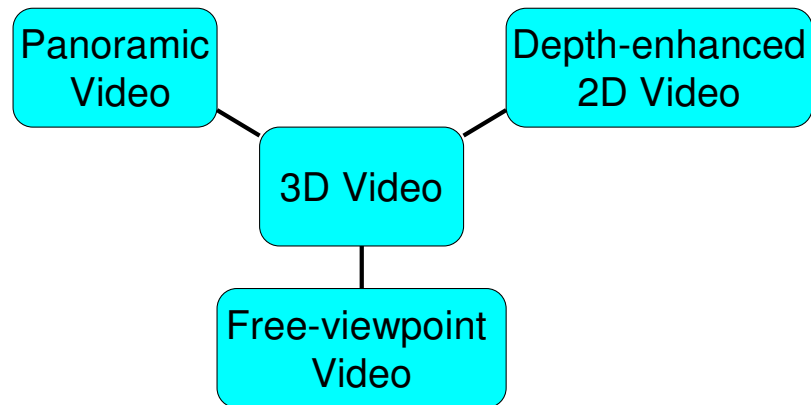
While capturing a movie for 3D cinema, a stereo camera records independent video streams for the left and the right eye. During display, both streams are simultaneously rendered. Typically, some kind of stereo splitter technology is used to separate the left and the right signals from the displayed footage. A common method is to use two projectors with different polarizations and a pair of glasses with appropriate polarization filters for each eye.

Depth-image-based rendering [2] uses hybrid video streams comprising of a color stream and a synchronized depth map stream. During display, virtual images for the left and the right eye can be reconstructed on-the-fly thereby creating a similar depth-enhanced viewing experience as 3D cinema, for instance on an auto-stereoscopic micro-lens display.

3D Video is a multi-faceted Field



Techniques differ in range of possible (virtual) viewpoints, ability to change viewpoint interactively, and complexity/completeness of employed scene representation



The previous two categories were mainly representative examples. A sea of other techniques exists that combines ideas from the two to perform, for instance, panoramic stereo to name just one example.

The techniques we will talk about in this course reconstruct and render the most general type of 3D videos, so-called free-viewpoint videos. This type of dynamic scene representation enables the display of captured real-world footage from arbitrary novel viewing positions and directions. As such, a free-viewpoint video representation is the most general type of 3D video as all other types of 3D video that we talked about before can be derived from it.

Free-viewpoint Video

- Reproduce Arbitrary Virtual Viewpoints
- Explicit Reconstruction Approaches



Dynamic Shape Model



texture

reflectance

Dynamic Appearance Model

- Data-driven Approaches

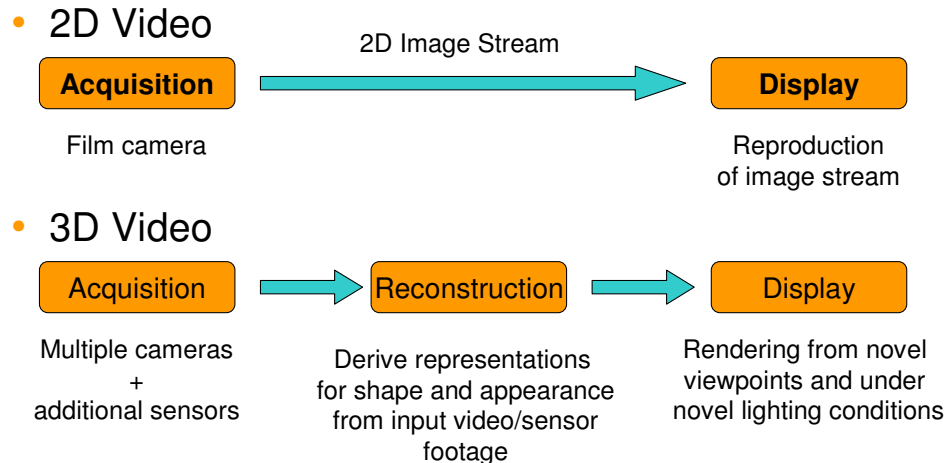


Most 3D video approaches capture a full dynamic 3D representation of real-world scenes that comprises, for instance, of a dynamic shape model as well as a dynamic appearance model. As we will see later in the course, the dynamic shape models are typically dynamic 3D meshes or point primitive presentations. Multi-view appearance is normally represented by a set of multi-view textures. Recently, even dynamic surface reflectance could be reconstructed which we will also show in this course.

Instead of representing scene geometry and appearance explicitly, data-driven approaches sample the space of capturing viewpoints densely and reconstruct novel views by appropriately combining the captured raw image data. In the remainder of this course, we will see examples for either of these category of approaches.

Some images on this slide were kindly provided by Larry Zitnick from Microsoft Research and Paul Debevec from the University of Southern California.

3D Video - An Algorithmic Challenge



The generation of 3D Video requires the solution to hard algorithmic problems that live on the boundary between the fields of Computer Vision and Computer Graphics.

The standard 2D video production pipeline shown above is fairly well understood and comprises of an acquisition and a display step. Acquisition is performed using standard camera systems and display, for the most part, is a replay of the captured streams on a display device.

The production of 3D video requires a fundamental rethinking of this pipeline. While still there is an acquisition and a display step involved, they have to be redesigned from scratch in terms of both the required engineering (sensors etc.) and the employed algorithms. Additionally, there is a reconstruction step involved which infers from the captured footage the underlying dynamic scene descriptions. It is this step which poses the hardest challenges as it requires the proper solution to several computer vision problems known to be notoriously hard.

In the remainder of this course, we will explain in more detail several possible solutions to each of the three steps.

Why bother ? - Applications



3D Video will revolutionize Visual Media

- 3D Digital Cinema
- 3D Enhancement of Live Broadcasts
- Interactive 3D Video
- Visual Effects in Movies and Games
- 3D City Mapping
- ...

Apart from the fact that 3D Video raises challenging algorithmic problems, the authors of this course believe that the technology has the potential to revolutionize the way How visual media are produced and presented.

There is a variety of intriguing applications of 3D video technology in movie, TV and game productions that are currently developed. The list above just names of few of them. In the remainder of the course, we will have a closer look at some of these applications.



Schedule

- Introduction – Theobalt (15 min)
- Silhouette-based Methods – Würmlin (25 min)
- Stereo-based Methods - Würmlin (25 min)
- Model-based 3D Video I – Theobalt (25 min)
- Break
- Model-based 3D Video II – de Aguiar (25 min)
- Free-Viewpoint Video Relighting – Theobalt (25 min)
- Applications – Niederberger (30 min)
- Outlook and Discussion - Würmlin (10 min)

This slide illustrates the further schedule of the course. Please also refer to the beginning of the course notes for a more detailed schedule.

Course Webpage



[http://www.mpi-inf.mpg.de/departments/d4/
3dvideo EG course/](http://www.mpi-inf.mpg.de/departments/d4/3dvideo_EG_course/)

Many links, test sequences, tools, additional
background information on camera
systems...

The webpage accompanying this course lists some interesting links and also some test data set that the people who attended the course may want to use in their own work. The web page also features a detailed list of references to related work from the literature.

References

- [1] O. Faugeras, R. Benosman, S. B. Kang, *Panoramic Vision*, Springer, 2001.
- [2] C. Fehn. 3D-TV Using Depth-Image-Based Rendering (DIBR). In *Proceedings of Picture Coding Symposium*, San Francisco, CA, USA, December 2004.

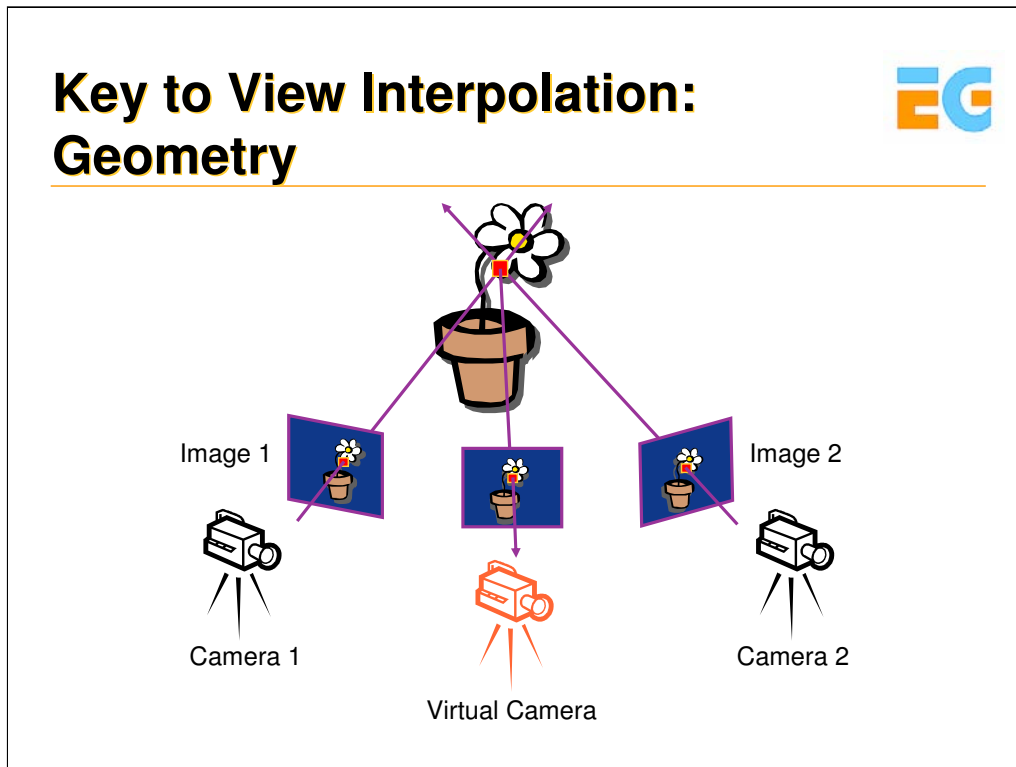


Silhouette-based Methods (25 min)

Stephan Würmlin

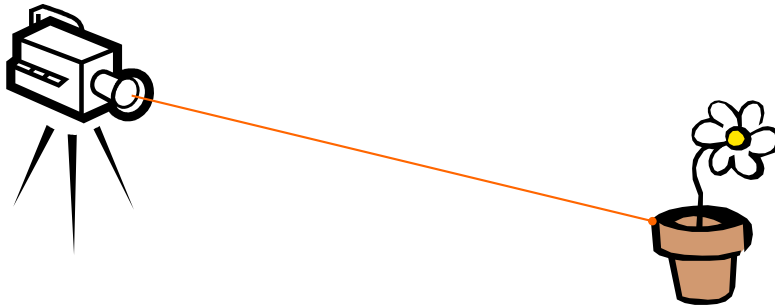
LiberoVision AG and

ETH Zürich



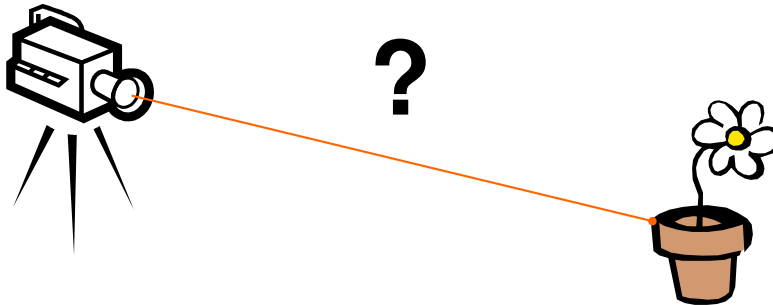
3D video is mainly about how to generate or interpolate arbitrary views from a set of multiple camera images. There are many different methods that can achieve that, however, purely image-based approaches such as the Lightfield or the Lumigraph need a huge amount of different input images to smoothly interpolate novel views. Most researchers intend to design more practical systems, and for them it is key to include some sort of geometry or 3D information in the data.

Image Acquisition



From images acquired by cameras...

Geometry: 3D Reconstruction



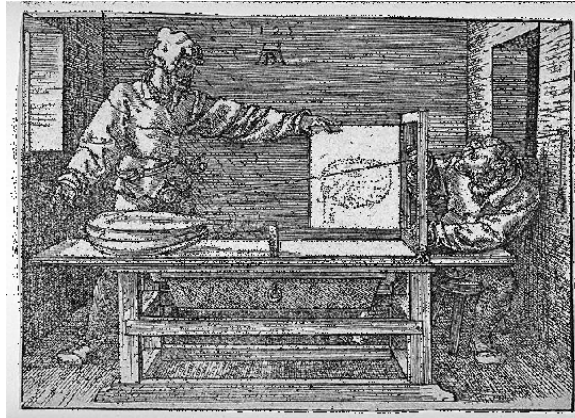
- Different computer vision algorithms out there
- Mostly used:
 - **Depth-from-Stereo** or
 - **Shape-from-Silhouettes**

... we want to know where each 3D scene point is that is image by the camera. In other words we want to compute the distance from the camera (or more precise the image plane of the camera) to the scene point.

There are basically two classes of algorithms that can compute this information from the images alone: (1) depth-from-stereo and (2) shape-from-silhouettes.

We will explain the fundamentals of both classes of algorithms and show some example methods and systems for 3D video.

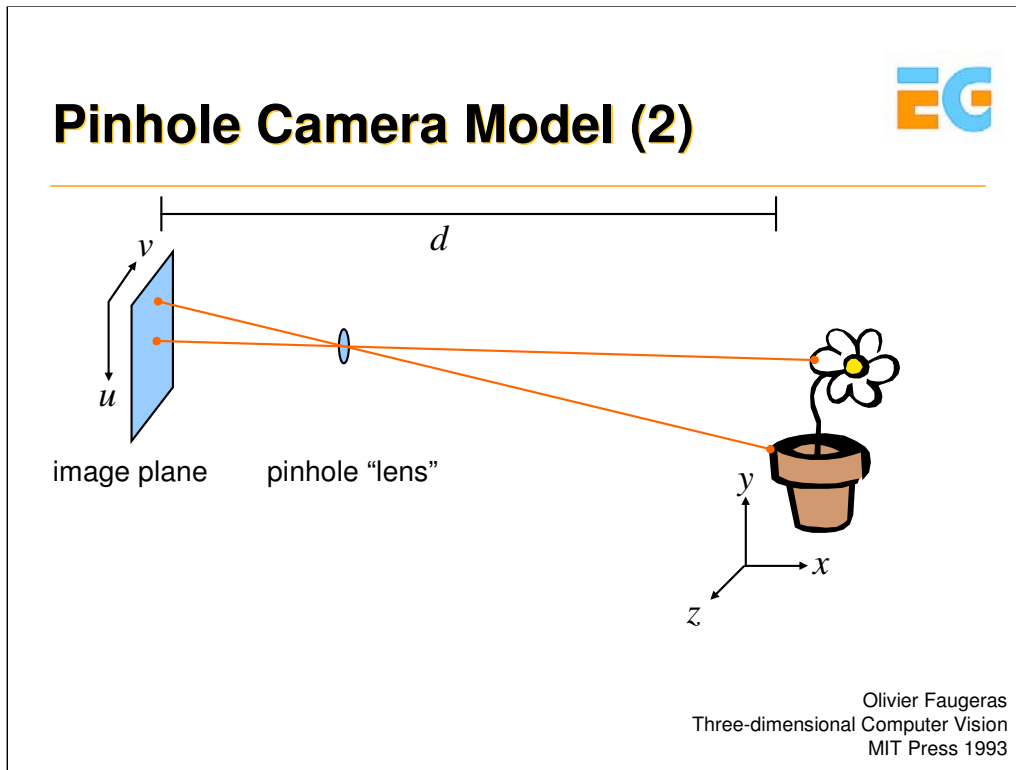
Pinhole Camera Model



Albrecht Dürer, Man Drawing a Lute (The Draughtsman of the Lute), woodcut, 1525

Before we can do that we need to know how we model the camera. The most used model is the ideal pinhole camera model which is a sufficiently close approximation of a real camera. The geometric process for image formation in a pinhole camera has been nicely illustrated by Dürer. The process is completely determined by choosing a perspective projection center and a retinal plane. The projection of a scene point is then obtained as the intersection of a line passing through this point and the center of projection C with the retinal plane P .

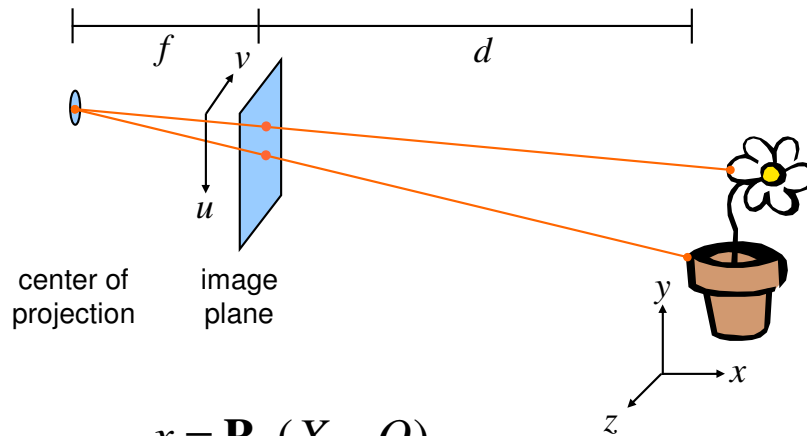
Most cameras are described relatively well by this model. In some cases additional effects (e.g. radial distortion) have to be taken into account



Here is a more schematic overview.

There is a perspective transformation that transforms points in 3-space X, Y, Z to image plane pixels u, v .

Frontal Pinhole Camera Model



$$x = \mathbf{P} \cdot (X - O)$$

Olivier Faugeras
Three-dimensional Computer Vision
MIT Press 1993

The frontal pinhole camera model is more easy to understand if one thinks about this. There all viewing rays converge in the pinhole which is now called the center of projection.

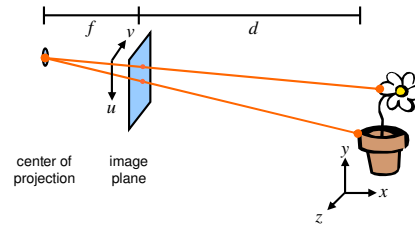
The projection of a camera (transforming 3D points into the camera's image plane) is defined by this equation.

Where:

\mathbf{P} is the matrix projecting viewing rays to image coordinates. The inverse of \mathbf{P} would be the matrix transforming image coordinates to rays in 3D world space.

O represents the center of projection of the pinhole camera.

Frontal Pinhole Camera Model (2)



$$\lambda \cdot x = \mathbf{P} \cdot X = [K | 0] \cdot \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix} \cdot X$$

$$K = \begin{bmatrix} fc_x & 0 & cc_x \\ 0 & fc_y & cc_y \\ 0 & 0 & 1 \end{bmatrix} \quad O = -R^T \cdot t$$

The mapping between a point in 3D space and the corresponding camera pixel can also be rewritten as...

Where:

K is an upper triangular 3x3 matrix containing the camera intrinsic parameters and R and t denote the rotation and translation between a world coordinate system W and the camera coordinate system C .

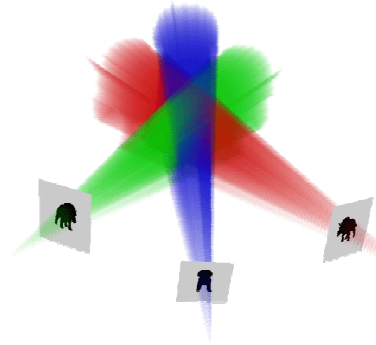
fc_x and fc_y are the focal lengths in effective horizontal and vertical pixel size units, and $[cc_x, cc_y]^T$ represents the image center coordinates, i.e. the principal point.

The center of projection can easily be determined by...



The Visual Hull

- Shape-from-Silhouettes
 - Intersection of silhouette volumes seen from multiple points of view
 - Reconstructs the **Visual Hull**
- Voxel representation
 - Volume carving
- Image-based representation
 - Silhouette image with occupancy intervals at every pixel



Aldo Laurentini
The Visual Hull Concept for Silhouette-Based Image Understanding
IEEE Transactions on Pattern Analysis and Machine Intelligence 1994

First we will tackle shape-from-silhouettes methods. Starting from the silhouettes extracted from the camera pictures, a conservative shell enveloping the true geometry of the object is computed. This generated shell is called the visual hull [Laurentini, 1994]. For 2D scenes, the visual hull is equal to the convex hull of the object, and for 3D scenes the visual hull is contained in the convex hull, where concavities are not removed but hyperbolic regions are. Even convex or hyperbolic points that are below the rim of a concavity (e.g. a marble inside a bowl) cannot be reconstructed. While the visual hull algorithms are efficient, the geometry they reconstruct is not very accurate. When observed by only a few cameras, the scene's visual hull is often much larger than the true scene. When rendering new views, one can partially compensate for such geometric inaccuracies by view-dependent texture-mapping [Debevec et al., 1996, Debevec et al., 1998]

Strictly, the visual hull is the maximal volume constructed from all possible silhouettes. In almost any practical setting, the visual hull of an object is computed with respect to a finite number of silhouettes. We call this type of visual hull the inferred visual hull.

There exist two classes of methods to compute the visual hull, (1) voxel carving methods, which carve away all voxels that are not contained in the silhouettes of the acquisition cameras and (2) image-based methods, that exploit epipolar geometry and store so-called occupancy intervals at every pixel.

What is a Visual Hull?



Here is an animated illustration of how a visual hull is carved...



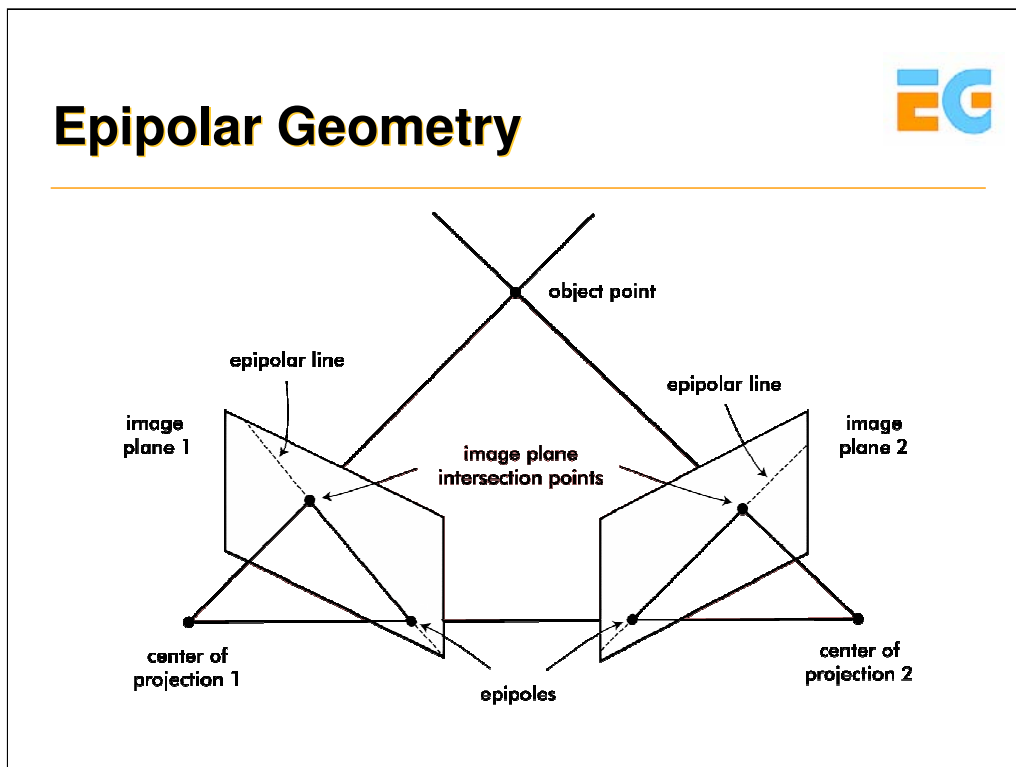
Image-based Visual Hulls

- Given k silhouettes, their associated viewpoints and the desired viewpoint:
 1. Cast a ray into space for each pixel in desired view
 2. Intersect this ray with the k silhouette cones and record intersection intervals
 3. Intersect the k lists of intervals
- Doing this in 3D is too expensive (projection of silhouettes into 3-space)
→ In 2D: **Epipolar Geometry**, projects 3D rays into 2D space of the silhouettes

Matusik et al.
Image-based Visual Hulls
SIGGRAPH 2000

We explain a particularly fast shape-from-silhouettes algorithm – which is able to perform in real-time - the image-based visual hulls method as presented by [Matusik et al., 2000].

The IBVH method takes advantage of epipolar geometry to accelerate calculation of depth values and to achieve real-time performance. As opposed to volumetric reconstruction techniques, e.g. voxel carving, the IBVH algorithm does not suffer from limited resolution, or quantization artifacts due to the underlying explicit voxel representation.

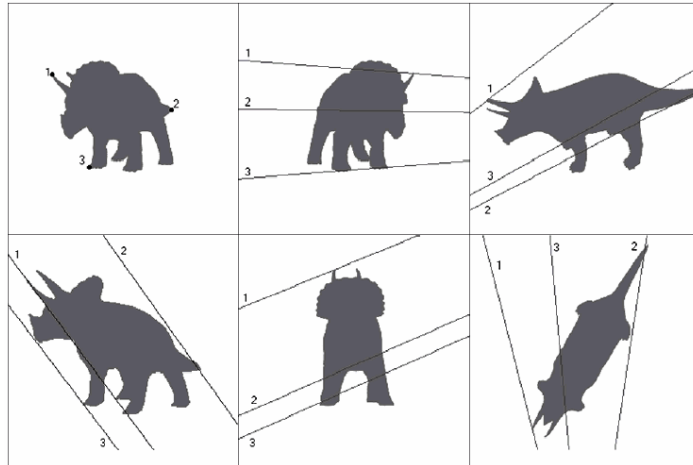


An *epipole* lies at the intersection of the baseline of the two cameras with the image plane of one of the cameras. Therefore, the epipole is the projection of the center of projection of one camera into the image plane of another camera.

An *epipolar plane* is defined by both centers of projection and a 3D point. Each plane containing the baseline is an epipolar plane, and intersects the image planes in corresponding epipolar lines, which also represent the projection of the ray from the center of projection of the other camera to the point. As the position of the 3D point varies, the epipolar planes “rotate” around the baseline. This one-parameter family of planes is known as an *epipolar pencil*. The respective epipolar lines intersect at the epipole.

The benefit of epipolar geometry in terms of a 3D reconstruction algorithm is that the search for a point corresponding to a point in another image plane need not cover the entire image plane, but can be restricted to an epipolar line.

Epipolar Lines in Reference Views



Here's an illustration of the epipolar lines of some points in one reference image, projected into the other images.

IBVH: Exploiting Epipolar Geometry



Creating Image-based Visual Hulls:

1. Projection of the desired 3D viewing ray onto a reference image (epipolar line)
2. Determination of the intervals where the projected ray crosses the silhouette
3. Intersect with intervals from other reference images
4. Reconstruct texture by projecting the IBVH to the k reference images and sample the color values

For estimation depth for a given pixel or fragment a ray has to be cast into space from that pixel. By making use of epipolar geometry this ray is projected to line segments in all other reference images (1).

There, the intersection/intervals is calculated with the binary silhouette (2). The resulting intersection points are lifted back onto the original ray where intersection intervals are built. They are represented as pairs of enter/exit points.

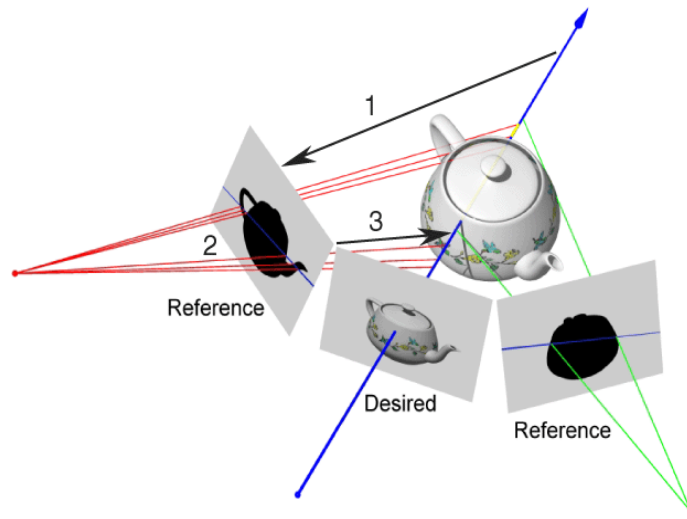
The intervals can be intersected with intervals from all other reference images (3).

Finally, texture is reconstructed in the desired view by projecting the IBVH data to all reference images and blending the color values together.

The result is basically a LDI representation of the geometry as seen from a specific camera. The key aspect

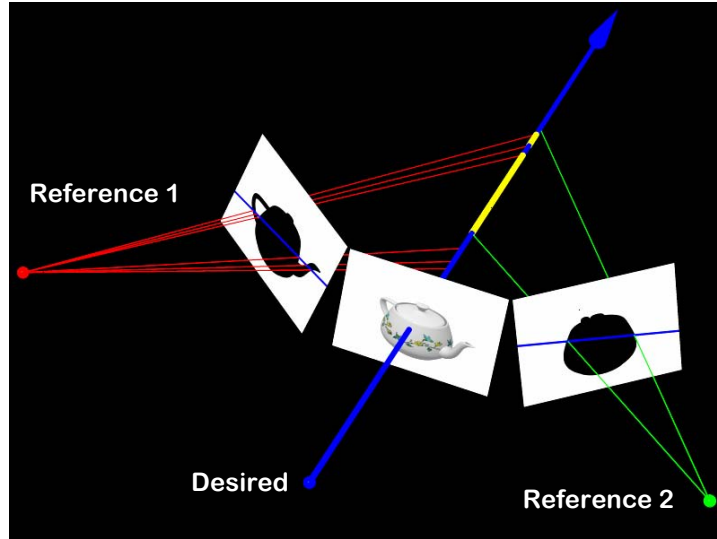
of the IBVH algorithm is that all intersection calculations can be done in two dimensions rather than three.

IBVH: Algorithm Illustration



And here's an illustration of the IBVH process for the notes.

Image-Based Computation

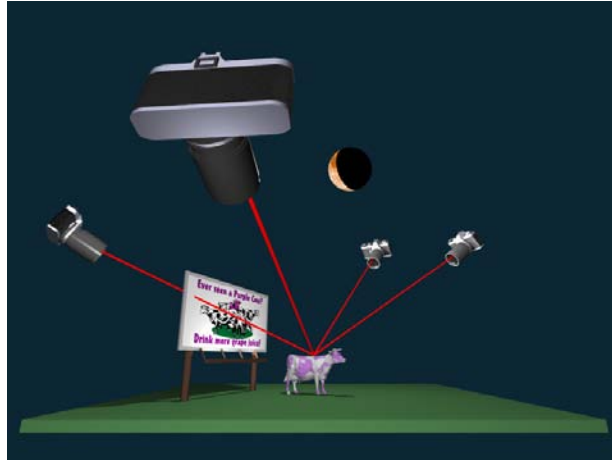


And here's an animated illustration of the IBVH process.



Shading Algorithm

- A view-dependent strategy

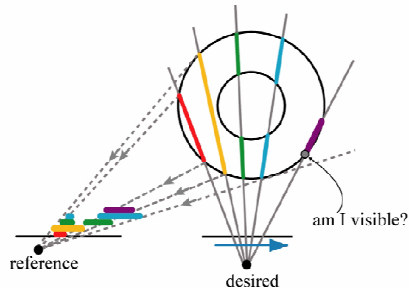


Finally, texture is reconstructed in the desired view by projecting the IBVH data to all reference images and blending the color values together.

Different techniques exist to blend the textures together, mostly applied is the Unstructured Lumigraph Rendering framework.

Care has to be taken for visibility.

IBVH: Visibility



- Visibility determination
 - Project all pixels' depth ranges into reference image
 - Built z-buffer in reference image plane
 - Desired pixel location on top?
- Implicit depth

In order to compute the visibility of an IBVH sample with respect to a given reference image, a series of IBVH intervals are projected back onto the reference image in an occlusion-compatible order. The front-most point of the interval is visible if it lies outside of the unions of all preceding intervals.

Once more we can take advantage of the epipolar geometry in order to incrementally determine the visibility of points on the visual hull.

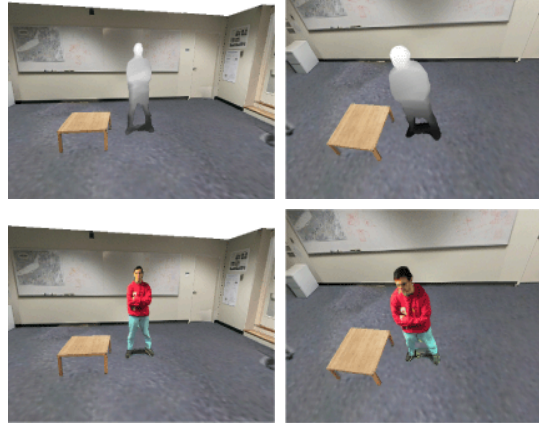


IBVH: Original Results

Input:



4 segmented
reference images



Upper: depth maps of the computed visual hulls
Lower: shaded renderings from the same viewpoint

And here are some results for the notes.

IBVH: Original Results Video



**Image-Based
Visual Hulls**



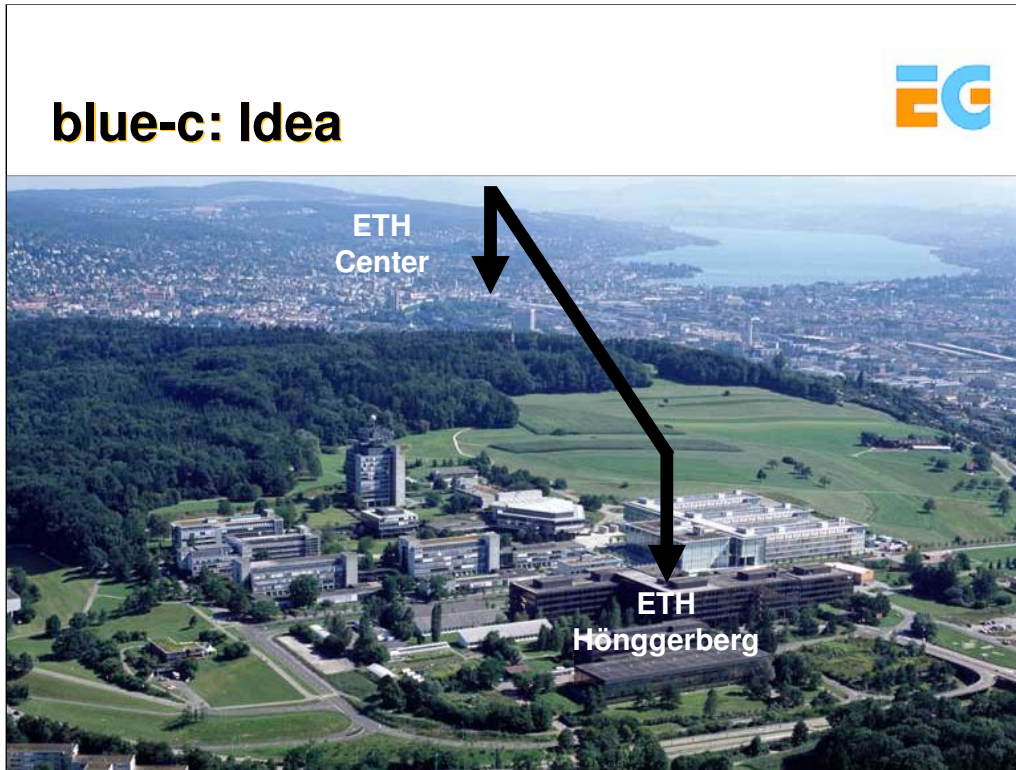
The blue-c



Gross et al.
The blue-c
SIGGRAPH 2003

Such techniques can now be exploited for telepresence applications since they provide 360 degree viewing of persons. ETH Zurich conducted a huge project in 1999-2004 which is called the blue-c. It exploited shape-from-silhouettes algorithms to connect two spatially immersive environments to be able to have telecollaboration sessions – seeing the remote participant in full 3D.

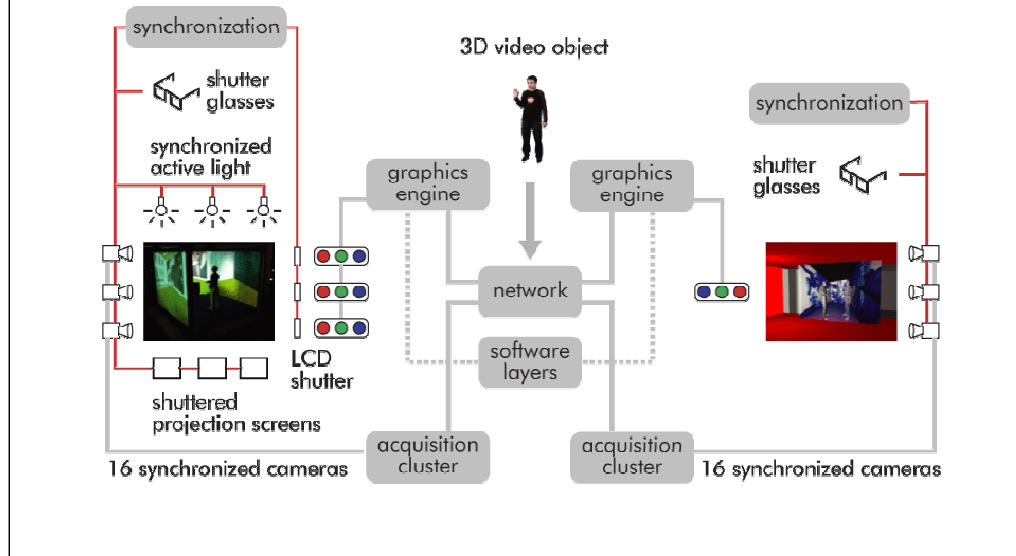
Here is example of how blue-c works. It connects a 3 sided CAVE environment located in the ETH computer center downtown Zurich with a second site on our campus outside Zurich. This second site consists of a single stereo projection panel only. As a central feature of our system, both sites are equipped with 16 video cameras capturing 3D video of the blue-c users. This allows for immersive 3D telepresence applications as the one you can see in the video clips.



The blue-c connects the two physically remote ETH campuses in Zurich, a distance of approx. 10 miles. The basic idea was to connect both campuses with a next-generation telepresence system.



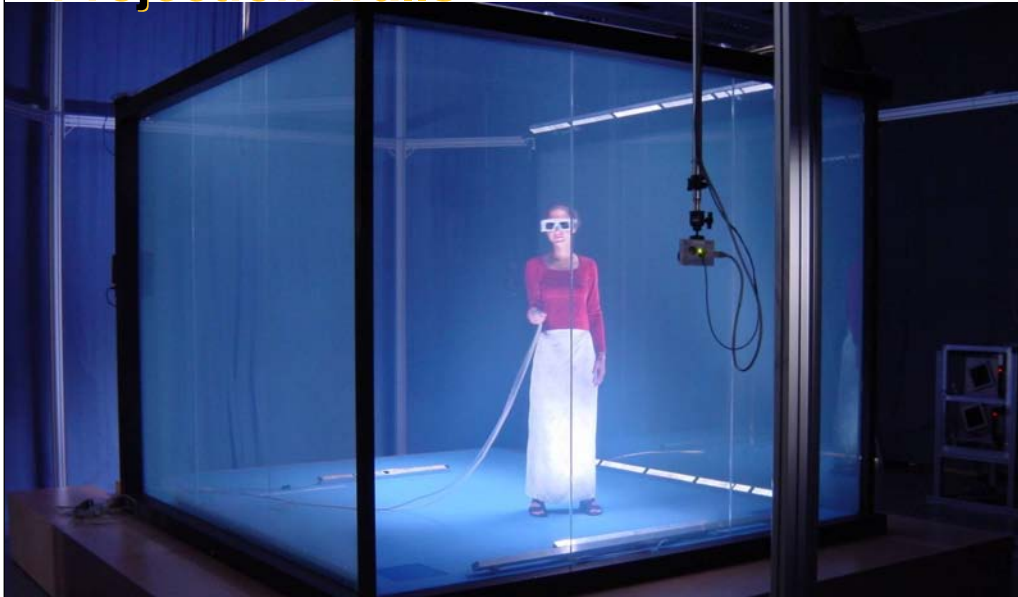
blue-c: System Overview



This picture displays an overview of the system architecture. We can clearly see that our setup is asymmetric. Besides costs, the major reason for this asymmetric design was to demonstrate scalability.

On the left we see the core hardware components being involved to accomplish simultaneous immersive projection and acquisition. This includes multiple cameras, shuttered projection screens, shutter glasses, an active lighting system, and an actively shuttered projection system. All hardware components are synchronized using a specially designed sync hardware. The cameras transfer 2D video frames to a PC cluster which computes a 3D video inlay of the user in realtime. This inlay is streamed over the network to the partner site and is composited into the synthetic scene by the graphics engine. We use both PCs and SGI Onyx 3200.

blue-c: Switchable Projection Walls

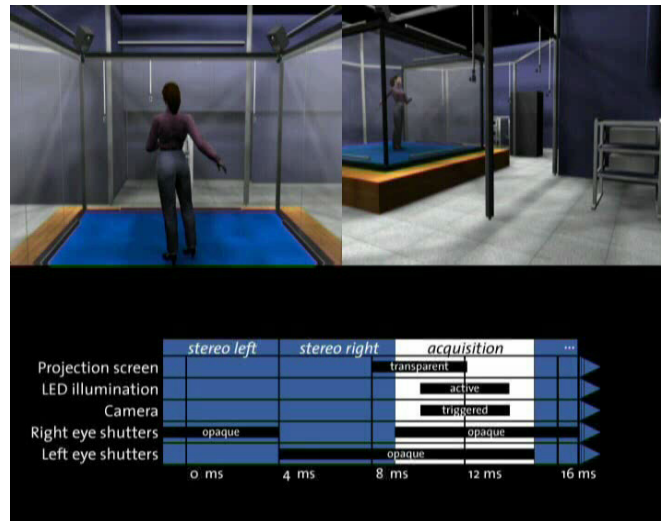


One of the core technical challenges when combining video acquisition and immersive projection is the placement of the cameras. As a central part of our design we place most of the cameras outside the projection space, which are, hence, not visible to the user. 5 remaining cameras are attached to the upper corners and to the ceiling to facilitate color calibration and texture acquisition.

It is easy to see that the projection screens occlude the user from the outside cameras. We solve this problem by using phase dispersed liquid crystal panels. These panels are switched from an opaque state during projection to an transparent state during acquisition. We do this at 62.5 Hz which is well above the fusion frequency of the human visual system.



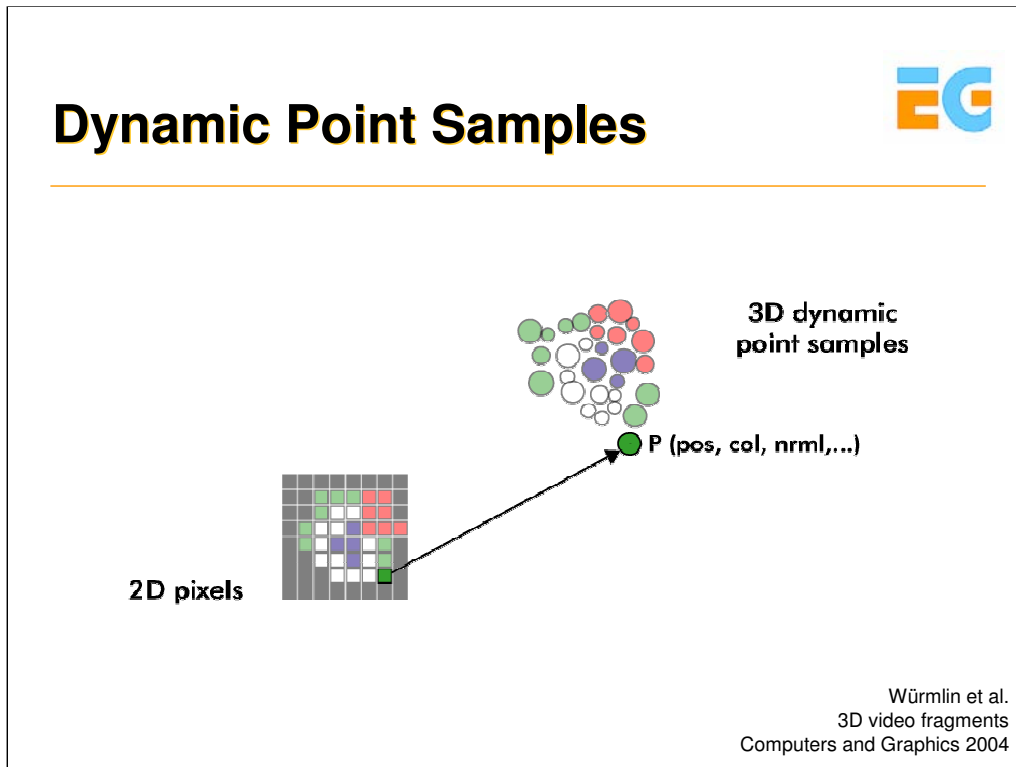
blue-c: Timing



The following video illustrates the timing and synchronization of the involved hardware components.

We first project the image for the left eye then the image for the right eye.

During a small time window of about 4 ms between the projection cycles, we open the walls and acquire the video frame. Due to hardware limitations the system currently grabs frames in every 7th window resulting in 9 Hz update rate. To improve the quality of the texture acquisition, we built an active lighting system which is synchronized with the video acquisition.



The basic primitives of the pipeline are 3D video fragments, which are dynamic point samples with attributes like, e.g., a position, a surface normal vector, and a color. 3D Video Fragments are a generalization of 2D video pixels towards 3D irregular point samples and we can therefore benefit from earlier work on point-based graphics.

Dynamic Point Samples: Advantages

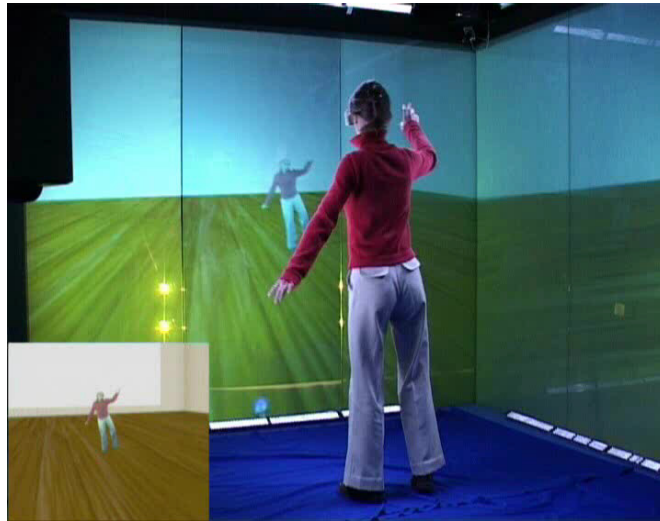


- Unified geometry and appearance
 - Amenable to topological changes of the scene's geometry
- Needs less acquisition cameras for even broader viewing ranges
 - Compared to purely image-based approaches
- Efficient coding
 - e.g. by using conventional video coding algorithms

Dynamic points have some advantages over other primitives.

- (1) It is a unified representation, holding geometry and appearance as one, and are amenable to topological changes of the scene's geometry
- (2) It needs less acquisition cameras for even broader viewing ranges compared to purely image-based approaches because it explicitly encodes the scene's geometry information.
- (3) It has potential for efficient coding schemes due to its simplicity, e.g. by using conventional video coding algorithms when stored in an image-space representation.

blue-c: 3D Mirror



The following example shows a 3D mirror application we built to demonstrate the concept. The user can experience herself in full 3D. She can freely move the camera and fly around herself. The cameras are now looking through the projection screens. This video gives a good feeling of the projection quality. It was recorded in realtime using a conventional unsynchronized camcorder.



Results – blue-c Video



And here's a video with results.

Acquisition was done at ETH Hoengerberg outside of Zurich as illustrated in the video inlay in the bottom-left corner.

The 3D video inlay is then streamed to the blue-c installation at ETH Computing Center in real-time and composited with the virtual scene.



Stereo-based Methods (25 min)

Stephan Würmlin

LiberoVision AG and
ETH Zürich



Overview

- Stereo Fundamentals
- Stereo-based 3D video
 - Dense camera setup
 - Sparse camera setup

Stereo-based 3D video (Dense)



- Video-View Interpolation
- Working volume?
 - Walls of a room:
Virtualized Reality
 - 2D “window”:
Light Field Array
 - 1D “rail”:
Video-View Interpolation

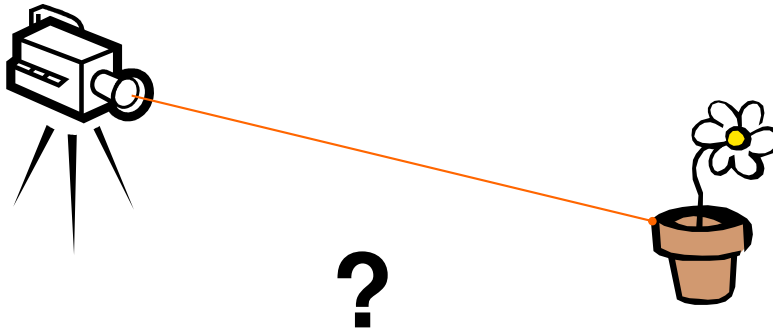


Zitnick et al.
High-quality Video View Interpolation
SIGGRAPH 2004

Stereo-based 3D video is able to not only capture one or two objects – due to the constraint of separable silhouettes for the employed shape-from-silhouettes algorithm – but can handle entire scenes. Techniques vary depending on the amount of freedom in navigation that a system wants to achieve. As an example, the Virtualized Reality project at CMU tried to enable full 360 degree freedom while a light field array only gives the possibility to give the user a 2D window into the world.

An interesting approach where I want to go in a little more detail is the video-view interpolation project at MSR where they tried to come up with a production-quality 3D video system but give the viewer only the ability to navigate on a 1D “rail”. For that they employed depth-from-stereo algorithms and the camera’s were placed rather dense as you can see in the image.

3D Reconstruction

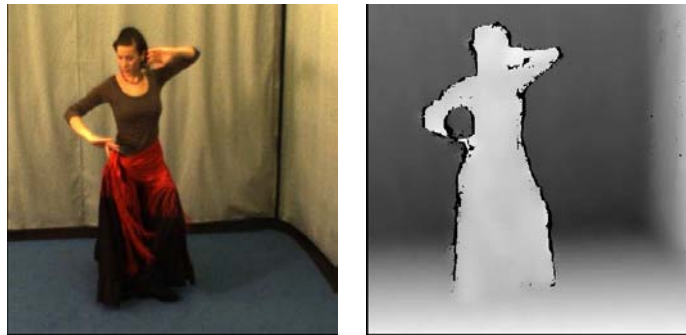


Again we have to know where the 3D points are that we image by the camera.

Depth Map



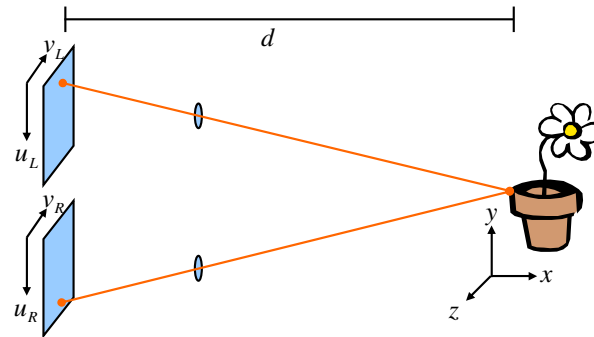
- Gray value encodes distance from camera



This means that we should calculate a depth map, indicating the distance from each pixel to the 3D surface point. On the right you see gray-coded the distance from the camera, with darker regions indicating farther away surface points and brighter regions indicate closer surfaces.



Depth From Stereo



- Basic Principle: Triangulation
- Requires:
 - Calibration
 - Point correspondence

$$d \approx \frac{1}{u_L - u_R}$$

The basic principle of depth-from-stereo is triangulation. When you know where a surface point is projected in two camera images you can – with an appropriate calibration of the cameras triangulate the distance that point has from the image. But to be able to do that you need point correspondences.

Stereo Vision



- Search for corresponding pixels



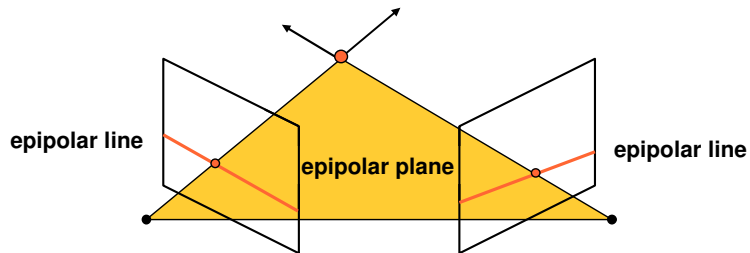
- Use windows to help you
 - But can still fail due to lack of texture!

Here is an example of what an algorithm should do. Instead of only calculating color similarities on single pixels, many methods employ a window-based approach. However, this can still lead to ambiguities and false depths in regions where there is not enough texture detail.



Stereo Correspondence

- Determine Pixel Correspondence
 - Pairs of points that correspond to same scene point



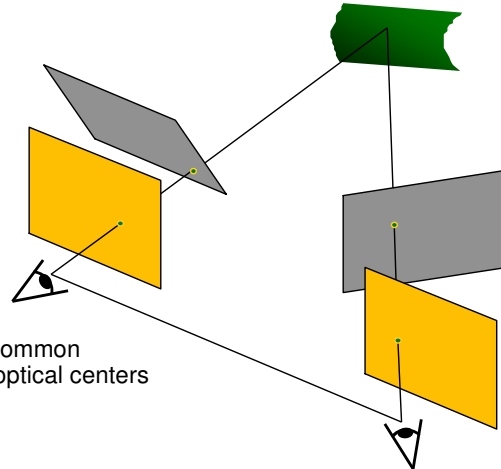
- Epipolar Constraint
 - Reduces correspondence problem to 1D search along conjugate epipolar lines

To determine pixel correspondences you need to search for pairs of points that correspond to the same scene point. This can be arbitrarily difficult to find in general – and hence arbitrarily time consuming because you need to do an exhaustive search. By employing again an epipolar constraint we can reduce the correspondence problem to a 1D search along conjugate epipolar lines as indicated in the picture.

Stereo Image Rectification



- Image Reprojection
 - Reproject image planes onto common plane parallel to line between optical centers
 - A homography (3x3 transform) applied to both input images
 - Pixel motion is horizontal after this transformation

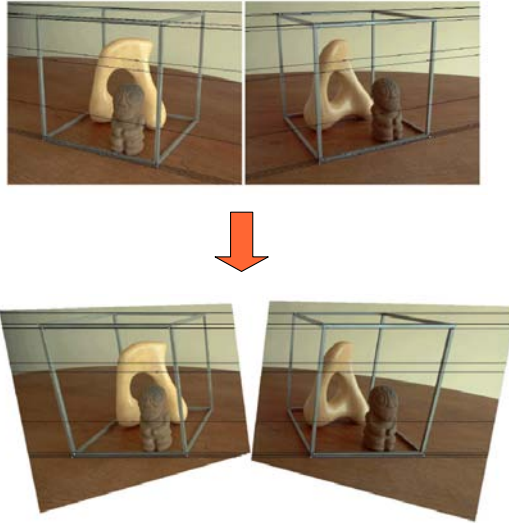


Loop and Zhang
Computing Rectifying Homographies for Stereo Vision
IEEE Conf. Computer Vision and Pattern Recognition 1999

For that we need to rectify the image pair which means that we reproject the image planes onto a common plane parallel to the line between the optical centers. This can be performed by applying a homography – a 3x3 transform – applied to both images. After rectification pixel motion is horizontal and we can search the correspondence on the same horizontal lines in the other image.




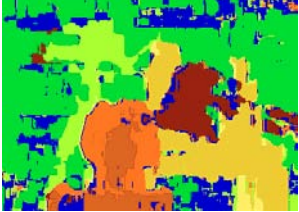
Stereo Rectification




This is an example of a stereo rectification where you clearly see that the features are afterwards located on horizontal lines.

Different Stereo Methods Exist...






Window-based
matching



Ground truth



State of the art

Middlebury Stereo Vision Page:
<http://cat.middlebury.edu/stereo/>

State of the art method:
 Boykov et al.
 Fast Approximate Energy Minimization via Graph Cuts
 International Conference on Computer Vision, 1999.

Based on this basic principle researchers developed a multitude of different stereo methods, some of it using the already mentioned window-based correlation methods for better robustness or by applying graph cut based optimization schemes.

The Middlebury Stereo Vision Page contains material for taxonomy and experimental comparison of stereo correspondence algorithms. It contains stereo data sets with ground truth, the overall comparison of algorithms, instructions on how to evaluate stereo algorithms, and stereo correspondence software.

Segmentation-based Depth-from-Stereo



- Don't match Pixels – Match Segments
- Segments contain more information, so they're easier to match.

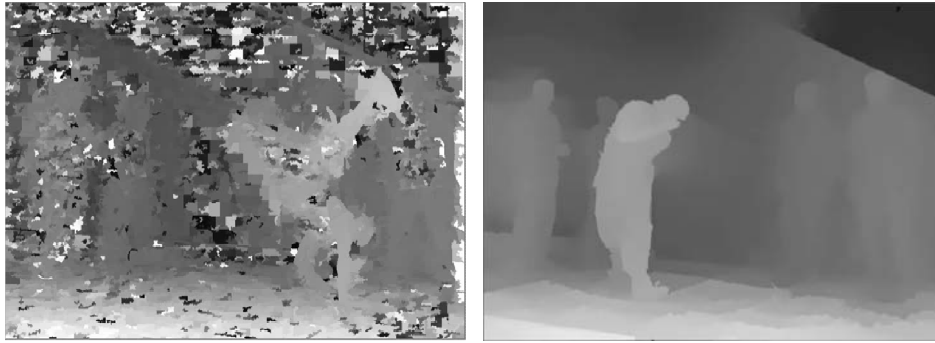


Zitnick et al.
High-quality Video View Interpolation
SIGGRAPH 2004

Now the approach by Zitnick et al. was probably one of the first really high quality 3D video systems out there. It combined a novel segmentation-based stereo algorithm with a multi-layered representation which could then interpolate views along a 1D-rail.

Segmentation-based approaches to stereo try to overcome some of the limitations of the pixel-based algorithms. Pixels are inherently hard to match and by correlating entire segments the algorithm produces much better depth maps. However, it relies on the assumption that all pixels of a segment belong to the same surface – so no discontinuities are allowed in the segments. Hence, a over-segmentation has to be produced during a pre-process step.

Iteratively Update Depths



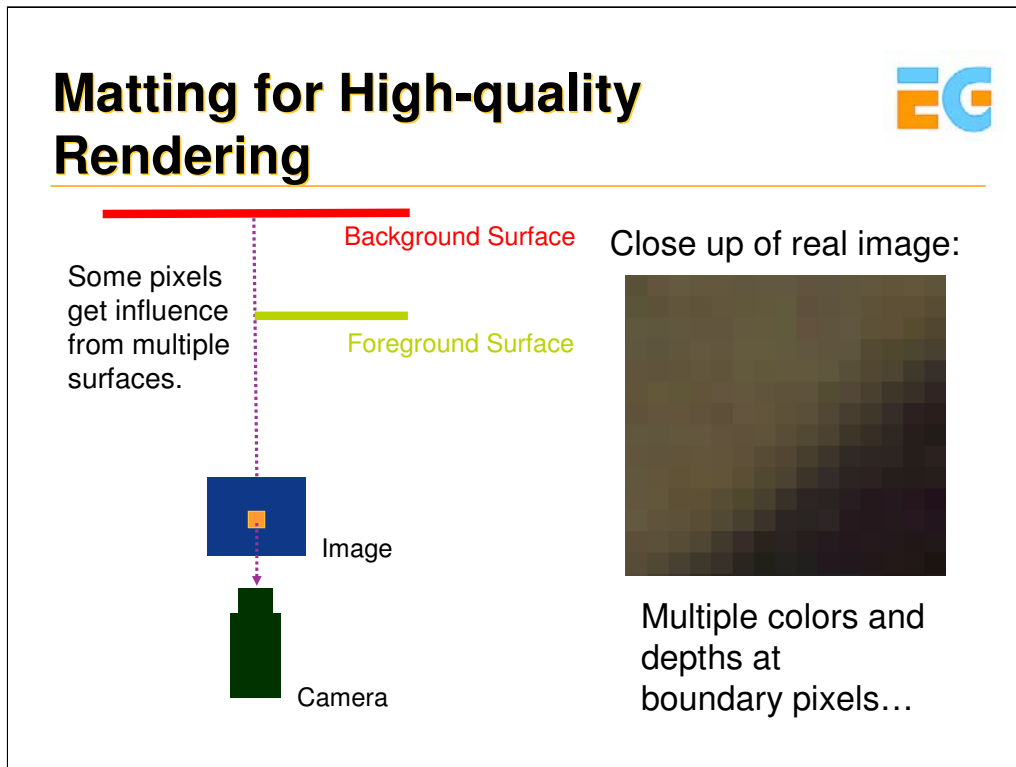
Here is an example of that work and how they can iteratively update the depths by taking into account all camera pairs of their system (for course notes).



Depth Through Time



Here is an example of that work and how they can iteratively update the depths by taking into account all camera pairs of their system.

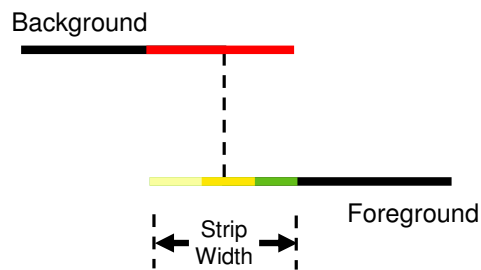
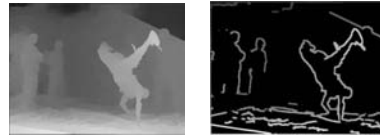


To be able to achieve high-quality re-renderings they apply a novel alpha matting technique. This helps significantly to reduce the artifacts around the depth discontinuities. There, some pixels get influence from multiple surfaces, they are called mixed pixels.



Finding Matting Information

1. Find boundary strips using depth.
2. Within boundary strips compute the colors and depths of the foreground and background object.



The algorithm first extracts a thin boundary strip around these surfaces which can be easily extracted using the depth information.

Then, within the boundary strips they compute colors and depth for both the foreground and the background object. In other words they try to separate this mixed pixel.

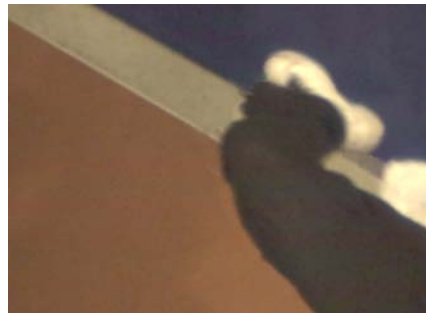
Why Matting is Important



No Matting



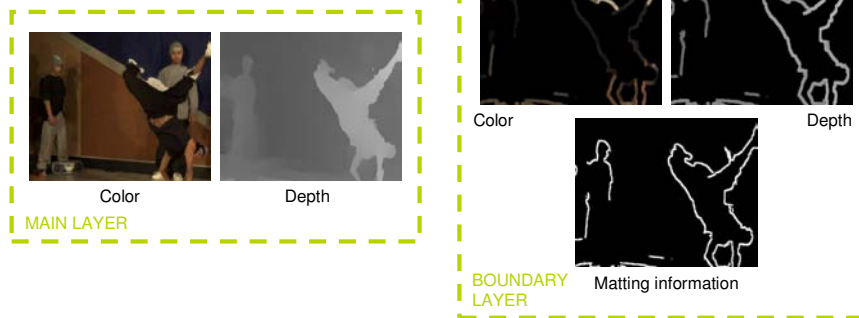
Matting



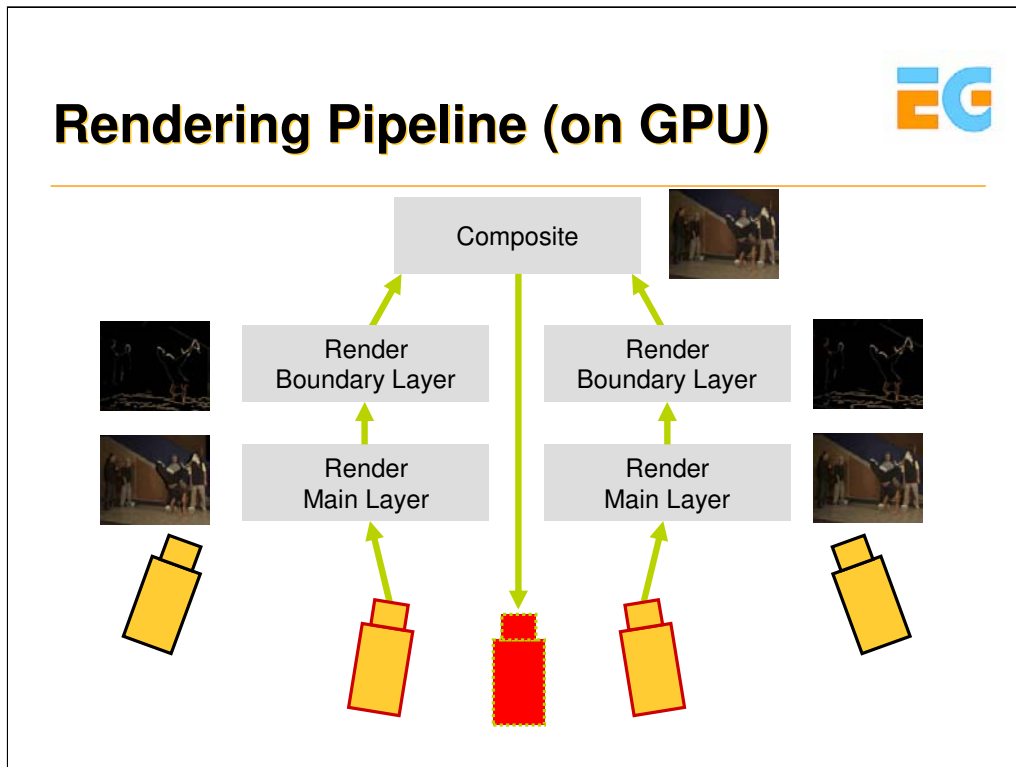
And here you see why matting is so important for high-quality renderings. On the left – without matting – the image has artifacts around the depth discontinuities which is visible as ghosting artifacts.

On the right you see how matting can improve the final image quality.

Layered-depth Representation



All this information – color, depth, matting information – is put into a layered-depth representation, one layer holds the main data, and another layer holds the boundary strip information.



This can then be rendered efficiently on the GPU by a multi-pass approach and final composite on the fragment level.



Interactive Session Video



Here is an interactive session of this system...

Free-Viewpoint Video



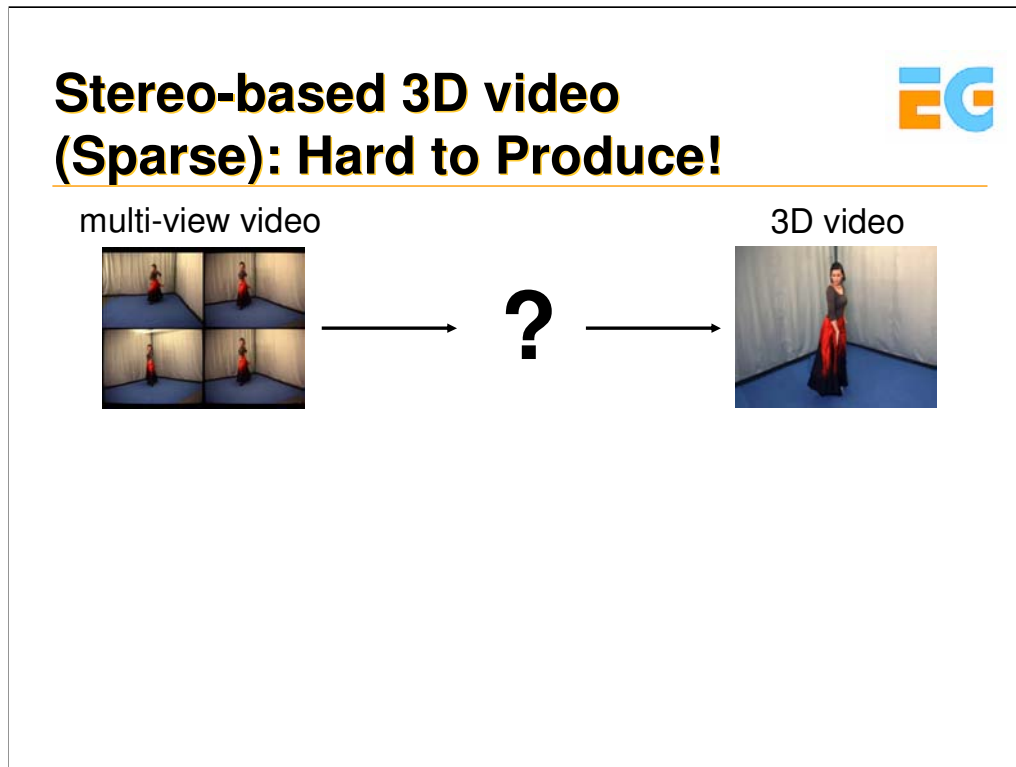
And here is a final result where the system was applied for a commercial music video (for course notes).

1D-Rail “3D” Video

A black rectangular frame containing the text "Massive Arabesque" in a white, serif font, centered horizontally and vertically.

Massive Arabesque

And here is a final result where the system was applied for a commercial music video.

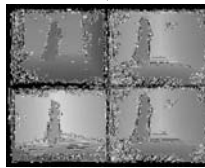
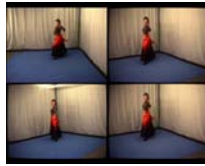


Now if we want to capture scenes not with a dense setup of cameras but instead with a sparse setup, this is even harder to do. Why? The re-rendering has to interpolate even broader views and much more occlusions can occur.



3D Extraction Error Prone

multi-view video



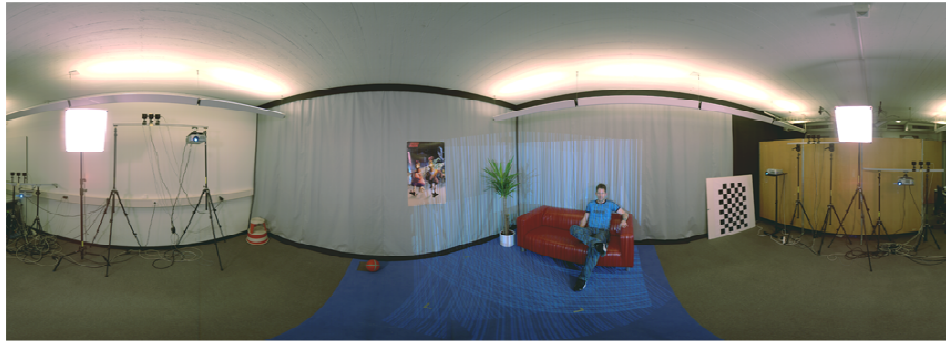
"3D" extraction

3D video



Moreover, the 3D extraction – whether silhouette or depth based – is highly error prone. You see an example in the bottom left and the resulting 3D video in the top right.

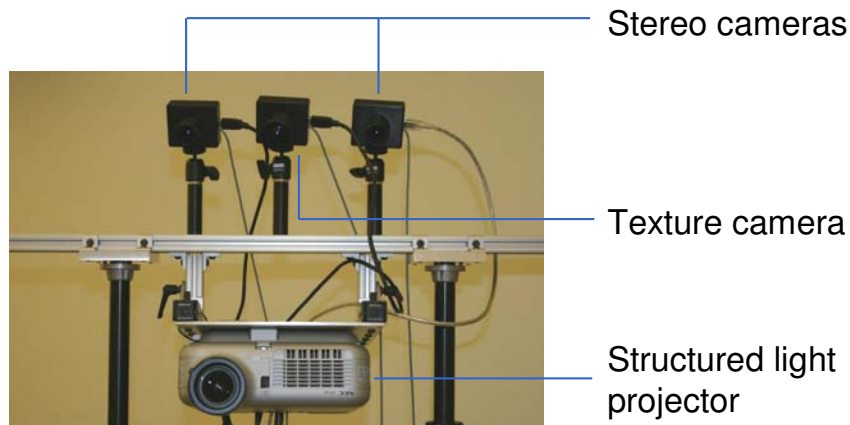
3D Video Studio at ETH Zurich



Here you see a snapshot of the 3D video studio we built at ETH Zurich. The main idea here was to overcome these 3D extraction problems by adding projectors to the studio that help to extract higher quality depth maps. The concept is based on so-called 3D video bricks.



Scalable 3D Video Bricks



Waschbüsch et al.
Scalable 3D video of dynamic scenes
Pacific Graphics 2005

Here you see an image of one such 3D video brick.

Each brick is equipped with one calibrated color camera for acquiring textures.

Two calibrated grayscale cameras together with a projector acquire stereo structured-light images used for depth extraction.

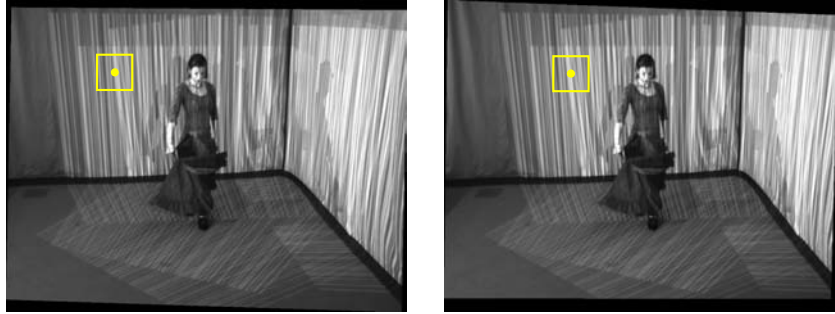
The projector additionally serves as an illumination source for the texture camera.

Furthermore, each brick is equipped with one PC for doing the acquisition and depth extraction.

Stereo Vision



- Structured illumination resolves ambiguities



Discontinuity-preserving stereo on structured light

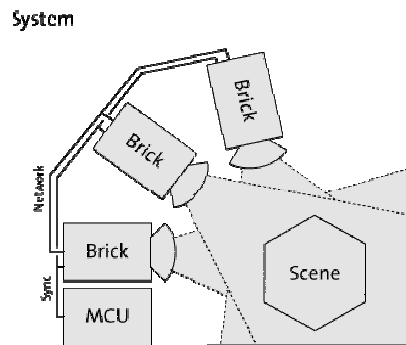
Waschbüsch et al.
Point-Sampled 3D Video of Real-World Scenes
Signal Processing: Image Communication 2007

Why do we want to use projectors in the studio? They can project structured light onto the scene which helps us to resolve ambiguities in regions where there is no texture.



System Configuration

- Multiple bricks
- Overlapping projections
- Common synchronization clock

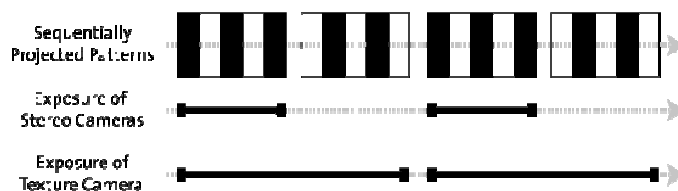


Here you see a system configuration schematic of the 3D video studio with three bricks and overlapping projections. Note that since the structured light is only used to improve the stereo computation, this is no problem.

Simultaneous Texture & Depth Acquisition



- Project random vertical stripe patterns
 - Multiple projectors prevent shadows
 - Stereo insensitive to projection overlaps
- Synchronize cameras shutters with different exposures
 - Invisible for texture cameras
 - Interchangeably project inverse patterns

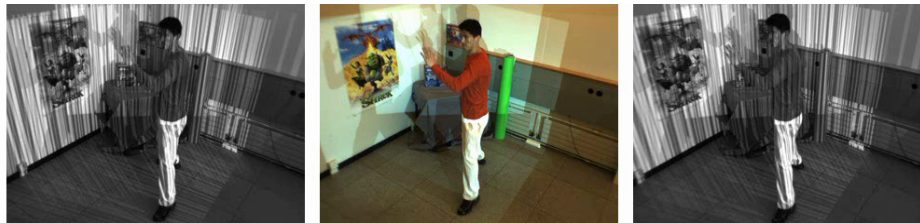


To simultaneously acquire texture and depth maps, the structured light patterns should only be visible to the stereo but not to the texture cameras.

We achieve this by interchangeably projecting a pattern and its inverse while exposing the texture cameras to both projections. Thus they acquire the integral image of both patterns which has a uniform white color.

The stereo cameras, in contrast, are only exposed to the first structured light projection.

Structured Light & Texture Acquisition



Grayscale camera

Color camera

Grayscale camera

↓
Textures

→ Stereo ←

Here you see the result of the acquisition.

Notice that the texture camera does not see the projected pattern.

However, the white projector lights are still clearly visible. This was mainly caused by space restrictions of our laboratory which forced us to put the projectors quite close to the scene. Moving the projectors further away or equipping them with wide-angle lenses would cover the whole scene in uniform white light which would be less disturbing.

Depth Extraction



This video shows the final depth maps of the sequence acquired by all three bricks.

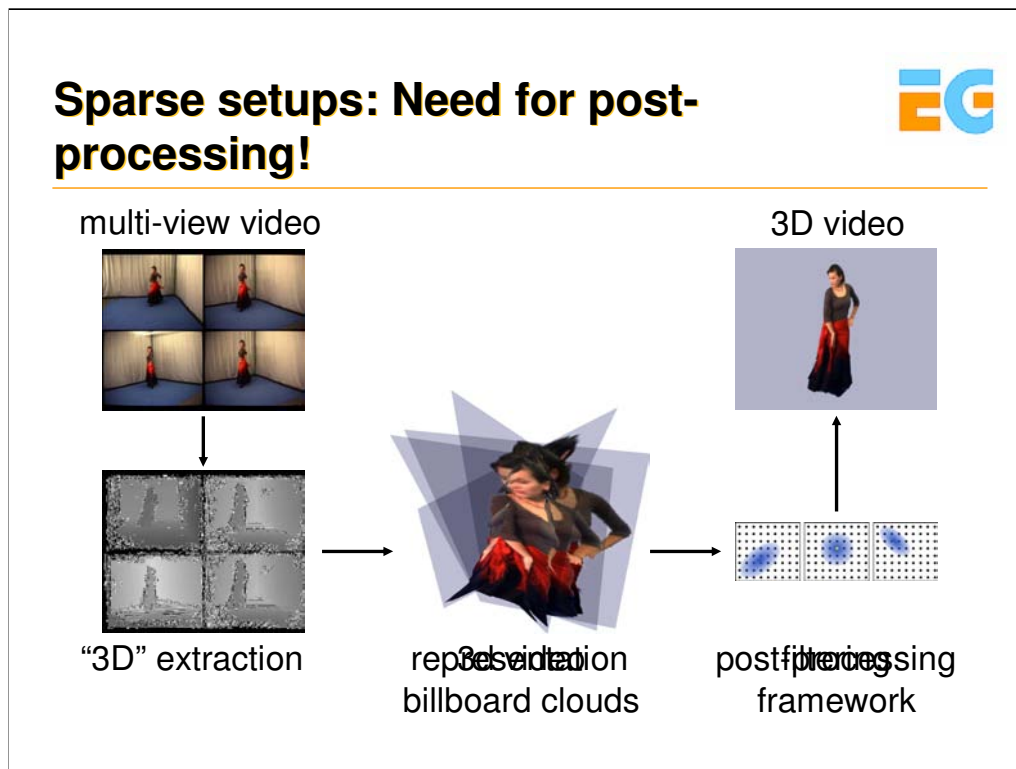
We still have some outliers at discontinuities. They are reduced during reconstruction of the 3d model of the scene.



Depth Extraction – Results



And here are the colors and depths of all three bricks used in our original setup.



We will now explain a novel representation which is also introduced here at Eurographics in a talk by Michael Waschbuesch – the 3D video billboard clouds – and how you can exploit this for post-processing the data to achieve high-quality re-renderings.



3D Video Billboard Cloud

- One billboard from each input viewpoint
- Planar geometric proxy
- Displacement map

Mantler et al., Displacement-mapped Billboard Clouds,
TR Vienna Uni. Of Technology, 2007



Waschbüsch et al.
3D video billboard clouds
Eurographics 2007

A 3D video billboard represents the 3D structure and texture of an object at a specific point in time as observed from a single viewpoint.

It consists of an arbitrarily placed and oriented texture-mapped rectangle or proxy approximating the real geometry of the object. Its associated textures are a displacement map for adding fine scale geometric detail, a color map modeling the surface appearance, and an alpha map holding a smooth alpha matte representing the object's boundary.

The latter is employed for seamless blending with the background of the scene.



Requirements

1. Simple geometric proxy
→ texture parameterization
2. Regular sampling
→ signal processing
3. Uniform error model
→ geometry filtering
4. Minimal displacements
→ compression, level of detail

We impose a set of requirements for an optimal 3D video billboard clouds representation:

1) Simple geometric proxy. The geometric proxy should be as simple as possible, i.e. a rectangle. This permits an easy parameterization for texture mapping.

2) Regular sampling. By ensuring a regular sampling we can exploit standard signal processing methods for easy post-processing of the geometry without the need of resampling.

In particular, we would like to directly exploit the existing regular sampling from the acquisition cameras.

3) Uniform error model. 3D reconstruction introduces noise which is usually not uniform in world coordinates. The uncertainty of depth values reconstructed by triangulation increases with their absolute value. Our billboard representation should be defined in a space where the reconstruction error can be assumed to be uniform, independent from the distance of a surface from the camera. This allows for easy reconstruction of filters using a uniform, linear kernel for smoothing the acquired geometry.

4) Minimal displacements. A minimal displacement of the proxy to the real surface ensures a good approximation of the geometry and can improve future compression and level-of-detail algorithms.



Billboard Space

- Requirement 2: regular sampling
- Requirement 3: uniform error model

→ Define billboards in disparity space!

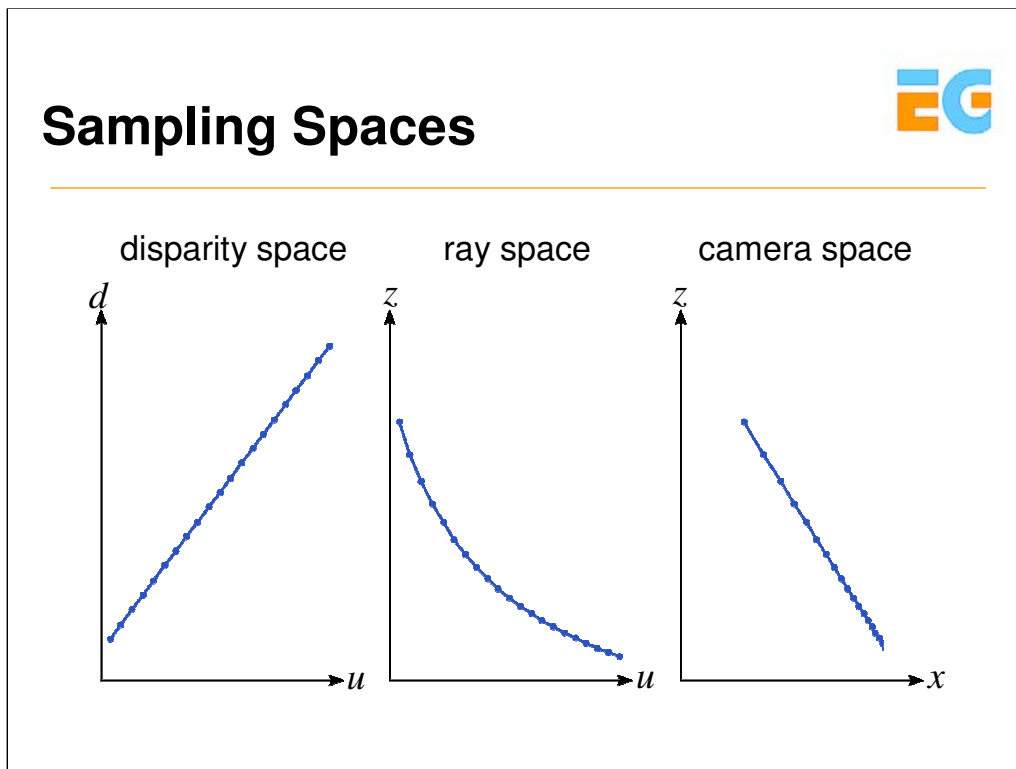
- Proxy plane in disparity space

$$B(u, v) = b_u u + b_v v + b_0$$

- Displacement map \equiv stereo disparities

Requirement (1) can be guaranteed by definition.

To fulfill requirements (2) and (3) the billboards are not defined in conventional 3D space of the scene but in the so-called disparity space of the acquisition camera. There, the displacements are simply the stereo disparities.



The transformation from camera to ray space is nonlinear, i.e. linear functions in camera space are not linear in ray space anymore. Hence, if we would define the billboard plane in ray space and use the depth values as displacements, it would not be planar in world coordinates and thus it would be difficult to use it as an approximation for the real geometry. On the other hand, if we would place it in camera space, the sampling would become irregular.

Instead, we define a disparity space of a camera as coordinates (u_i, v_i, z_i') with $z_i' = 1/z_i$. If we use this representation and store the reciprocal of the z -coordinate from ray space, we can observe that planes in disparity space stay planar in camera space.

Moreover, sampling in disparity space is identical to the regular sampling of the acquisition cameras. Thus, requirement (2) is fulfilled if we define the billboard planes in these coordinates.

In camera space it can be observed that the resulting uncertainty of the geometry is not constant anymore but depending on the absolute value of the disparity.



Billboard Placement

- How to place the billboard plane?
 - Noise in displacement map should result in small errors in camera space
 - Useful for compression, level of detail, ...
- Wrong position in disparity space can lead to large displacements in camera space!
- Minimize sum of displacements in camera space
 - Non-linear least-squares problem
 - Solve with Levenberg-Marquardt

We are still free to choose the position and orientation of the billboard plane. A bad choice of these values can lead to arbitrarily large displacements in world coordinates. This becomes an important issue as soon as the values of the displacement map should be processed, e.g. for compression, level of detail, ...

Hence, noise in displacement map should result in small errors in camera space

We minimize the sum of displacements in camera space – non-linear least-squares problem – and solve it with Levenberg-Marquardt algorithm.

Geometry Filtering



- Input displacements are noisy
- Regular sampling and uniform error model allow for easy signal processing tools
- Spatio-temporal bilateral filter for smoothing the geometry



The displacement values generated by the acquisition system are subject to quantization errors, noise, and calibration inaccuracies, resulting in several kinds of artifacts in the re-rendered image: The object surfaces do not appear smooth and their geometry is very noisy, which is especially visible as flickering over time. Moreover, overlapping parts of surfaces from different scanning directions do not necessarily fit to each other.

To improve this, we apply a four-dimensional smoothing filter yielding better spatial coherence within surfaces and between overlapping surfaces, and better coherence over time.



Filtering over Multiple Views

- Problem
 - Billboards from multiple views are filtered independently
 - Overall geometry may diverge
- Solution
 - Filter over all billboards at the same time

However, when filtering each acquisition view independently, corresponding surfaces reconstructed from different views would not fit to each other. Thus, we extend the filter by an additional domain by accumulating the data from all acquisition views.



Bilateral Disparity Filter

$$\tilde{d}(\mathbf{x}) = \frac{1}{s} \cdot \iiint d(\xi) \cdot c(\xi, \mathbf{x}) \cdot s(d(\xi), d(\mathbf{x})) \cdot \delta\xi$$

All this is now put together into this formula, where you see how a bilateral filter is applied over the displacement map and over time.

Bilateral filter: The domain filter kernel c smoothes the disparities over space and time while the range filter kernel s tries to retain geometric discontinuities



Filter Kernels

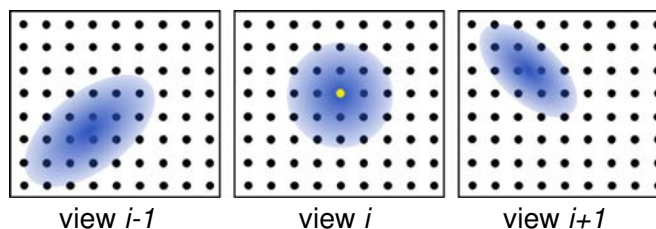
- Domain filter kernel
 - Cubic B-spline weighted by alpha matte
$$c(\xi, \mathbf{x}) = \alpha(\xi) \cdot B(\xi - \mathbf{x})$$
 - Low-pass filter
 - Lower weight at more inaccurate boundaries
- Range filter kernel
 - Step function
 - Preserve depth discontinuities

For c we use a cubic b-spline low-pass filter kernel B weighted by the alpha values of the current billboard.

The range filter kernel s does not only maintain discontinuities in the disparity maps but also ensures a correct handling of occlusions that occur during warping. We use a simple step function.

Filter Implementation

- For filtering view i
 - Warp domain filter kernel to all views j
 - Convolve all views j with warped kernel
 - Accumulate in view i , using range filter kernel



- **Splatting**
Zwicker et al., Surface Splatting, SIGGRAPH 2002

Intuitively, for computing a disparity d_i , this filter does not only compute a convolution in the current view i but it convolves all views with warped versions of the domain filter kernel c and accumulates all values weighted by the range filter kernel s .

[Image: For computing the value of the yellow pixel in view i , the values all pixels in all views weighted by the warped domain filter kernel (blue) are accumulated.]

The implementation of the filter process is done via splatting. For filtering a view i , instead of projecting the filter kernel into all other views, we do the inverse and splat all views into view i . This has the advantage that we can use a uniform splatting kernel. Splatting can be performed efficiently in the GPU.



Filtering Comparison

no disparities,
only planes



raw
disparities



views filtered
independently



views filtered
together



Here is a comparison of different filtering techniques where you clearly see the difference.

View-dependent Rendering



- **Unstructured Lumigraph (UL) framework**
Buehler et al., Unstructured Lumigraph Rendering, SIGGRAPH 2001
- **Render consistent depths**
 - Render views into separate depth buffers
 - Resolve occlusions via fuzzy z-buffer
 - Blend remaining depths using UL
- **Render consistent colors**
 - Projective texturing
 - Blend colors using UL

We render the billboard clouds using the Unstructured Lumigraph Rendering framework.

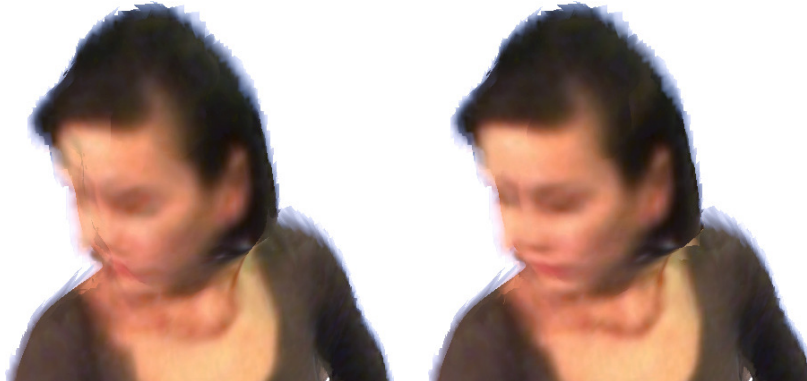
In contrast to the original UL algorithm, our method does not only blend the colors but first reconstructs a consistent, view-dependent geometry of the scene, where each pixel has a uniquely assigned depth value. This results in much crisper renderings. If multiple fragments are rendered at the same pixel, its depth buffer value d is computed in a fragment program by blending all fragment depths d_i .

Rendering Comparison



color blending only

color & depth blending



Here is a comparison of rendering with only color blending and with color & depth blending.

Results – No Disparities





Results – Raw Disparities



Results – Filtered Disparities





Handling Scenes

- Segment scene using semi-automatic video cutout techniques

Li et al. Video Object Cut and Paste SIGGRAPH 2005

Wang et al., Interactive Video Cutout, SIGGRAPH 2005

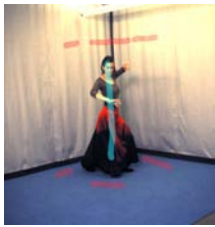


To represent scenes and not only stand-alone objects, each view has to be decomposed into multiple billboards. We use a semi-automatic video cutout technique to segment the input videos into distinct objects. After the user has marked the objects in a single input frame of each view by applying a few brush strokes, a graph-cut optimization automatically computes the segments over time. The segment boundaries are refined by a Bayesian matting algorithm [Chuang et al., A bayesian approach to digital matting. CVPR 2001] yielding alpha mattes for the billboards.

Graph Cut Segmentation



- Graph cut on color and depth maps
 - Colors for accurate boundaries
 - Depths for increased stability



input



colors only



colors & depths

As well we include the depths in this process which yields significantly higher stability.

Results with Complete Scenes



Final result.



Model-based 3D Video I **(25 min)**

Christian Theobalt
Stanford University



Motivation

- Previous methods in course
 - General scenes
- Problem
 - Multi-view correspondence problem
 - Remaining artifacts
- Model-based approaches
 - Exploit prior knowledge about type of scene

The approaches presented so far in the course did either not at all or just to a small extent exploit prior knowledge about the type of scene during reconstruction. This has the great benefit that arbitrary types of scenes can be processed. However, the multi-view reconstruction (and thus the multi-view correspondence problem) in this case is particularly hard and thus there may be noticeable artifacts.

For certain types of scenes it may therefore be beneficial to exploit prior knowledge about the type of subjects in the scene being reconstructed. Although by this means generality may be sacrificed, it may still be beneficial in terms of reconstruction quality.

The approaches in this part of the course utilize prior shape and motion models for 3D video reconstruction. While the approaches in the first part of this section mainly deal with human actors, towards the end some more recent approaches are shown that can handle arbitrarily dressed people and even a broader category of subjects.

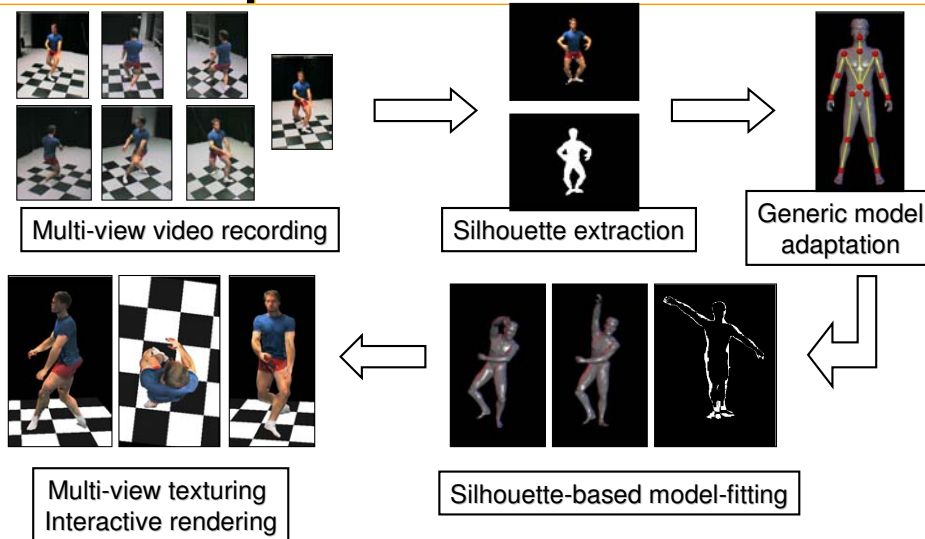


Overview

- 3D Video using a kinematic template model
- Alternative model-based approaches
- Mesh-based dynamic scene capture and 3D Video

The section on model-based 3D Video is subdivided into three main parts summarized above.

Model-based Free-Viewpoint Video



[Carranza et al., Siggraph 2003]

In free-viewpoint video the viewer is given the possibility to freely change his viewpoint onto a 3D rendition of a dynamic real-world scene. In order to generate a free-viewpoint video, the problems of acquisition of input data, reconstruction of a dynamic scene descriptions, and rendering in real-time have to be solved simultaneously.

This slide illustrates the workflow between algorithmic components of a model-based system to reconstruct and render free-viewpoint videos of human actors [3]. Inputs to our system are multiple frame-synchronized video streams showing a moving person that have been captured with calibrated video cameras. Image silhouettes of the person in the foreground are extracted from each video frame. A generic human body model consisting of 16 segments, a triangle mesh surface geometry with roughly 21000 triangles, and a kinematic skeleton made of 17 joints is employed to represent the time-varying appearance of the human. An analysis-by-synthesis approach based on silhouette-overlap is used to adapt the model in shape and proportions to the actor, and to determine pose parameters for each time step of video. Real-time renditions of the captured scene are generated by projectively texturing the moving body model from the input video frames that are appropriately blended.

Acquisition



- 8 synced CCD cameras;
currently:
 - Currently: 1004x1004, 25fps,
12 bit
 - old: 320x240, 15 fps
- Calibrated
- Controlled lighting



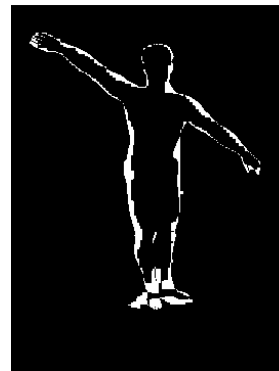
Multi-view video footage is recorded in a 3D video acquisition studio. This studio features 8 synchronized calibrated video cameras. Currently, we employ a system with eight CameraLink 1-Megapixel video cameras. In a previous version of the studio, we used 8 IEEE1394 video cameras with VGA resolution. Some of the older footage presented in this course has been recorded with our old setup.

In the remainder of the course, we will also show important hardware and software extensions to the studio in case they are relevant for a specific topic.



Silhouette Matching

- Model Initialization & Marker-free Motion Capture
 - Non-linear minimization of pose/scaling parameters
 - Criterion: overlap between image silhouettes and body model projection
 - Area of intersection
 - Robust against silhouette inaccuracies
 - Graphics hardware acceleration
 - Pixel-wise XOR (stencil buffer)
 - 8 camera views / frame buffer read-write
 - 105 pose evaluations / sec. (GeForce 3)

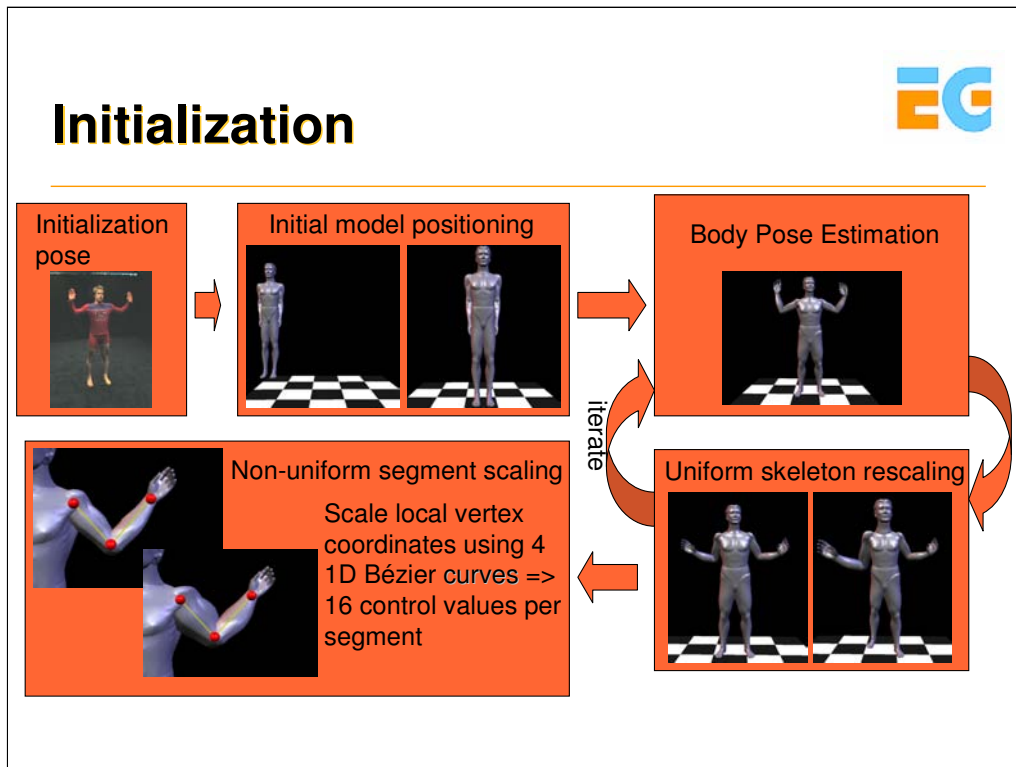


Silhouette XOR

Besides 35 parameters to specify the pose, our body model features for each segment a uniform scaling parameter, as well as 16 Bézier parameters for fine-tuning the appearance of the surface. Our analysis-by-synthesis approach employs the overlap between the projected model silhouettes and the image silhouettes in all camera views for two purposes:

- 1) Initialization: The geometry of the model is adapted to optimally represent the appearance of the real-world equivalent.
- 2) Marker-free Motion Capture: after the model has been customized, its pose is matched to the pose of the actor at each time step of the input video footage.

Both tasks require non-linear optimizations in the model parameters, the first one being performed in the pose and scaling parameters simultaneously, the second one being performed in the pose parameters only. The error function to be minimized computes the non-intersecting areas between the projected model and input silhouettes in all camera views. Conveniently, an estimate for this is obtained via the binary XOR between image and model silhouettes in all views. It can be efficiently evaluated on state-of-the-art graphics hardware using the stencil buffer. On a PC featuring an Intel Xeon 1.8 GHz CPU and an Nvidia GeForce3 GPU, we can perform 105 energy function evaluations using 8 cameras and a frame size of 320x240 pixels.



The shape and proportions of the body model are adapted to the shape of the actor in an iterative optimization procedure. Inputs are silhouette images of the actor striking a dedicated initialization pose. In a first step the position and orientation of the torso segment in 3D space is determined. The space of pose parameters for the torso is sampled regularly to find an optimal starting point for a subsequent downhill optimization that determines the final torso pose.

Thereafter, the method iterates between determining a new set of pose parameters for the whole model (explained on the next slide), and determining a new set of uniform scaling parameters for each segment.

Finally, optimal parameters for the 1D Bézier scaling functions are found that bring the segment outlines into optimal accordance with the multi-view silhouettes. On the hands and feet no Bézier scaling is performed.

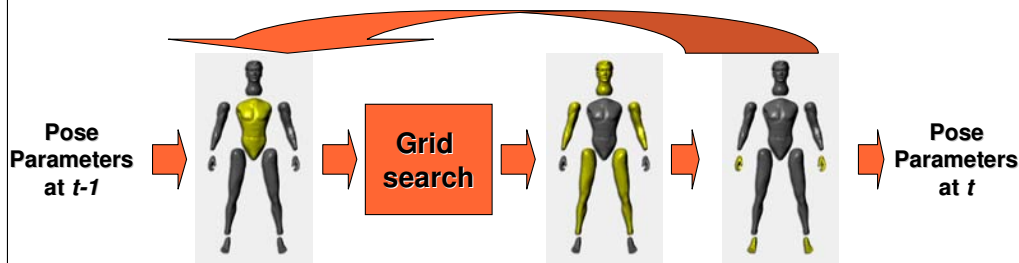
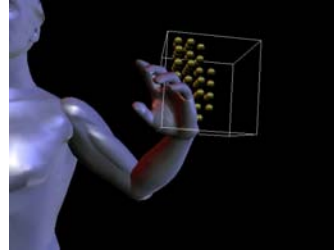
The estimation of the optimal scaling parameters is performed hierarchically. It starts with the root of the skeleton (located in the torso) and proceeds layer-by-layer further down the skeleton hierarchy until the terminating nodes (head, hands and feet are reached).

For numerical minimization we employ a standard downhill direction set method such as Powell's method. The shape parameters of the model remain fixed for the duration of a free-viewpoint video.



Body Pose Estimation

- Challenges
 - Many local minima
 - Fast limb motion
 - Constraint: no inter-penetrations
- Hierarchical decomposition
- Grid search



Only with a completely passive marker-free motion capture approach the same video material can be used for motion and texture estimation. Determining the 35 pose parameters for each time step of video is a challenging problem since the non-convex energy-functional exhibits many local minima. Potentially rapid limb motion and constraints on allowable body poses require the parameter search space itself to be constrained. Only this way robust convergence to the correct solution can be assured.

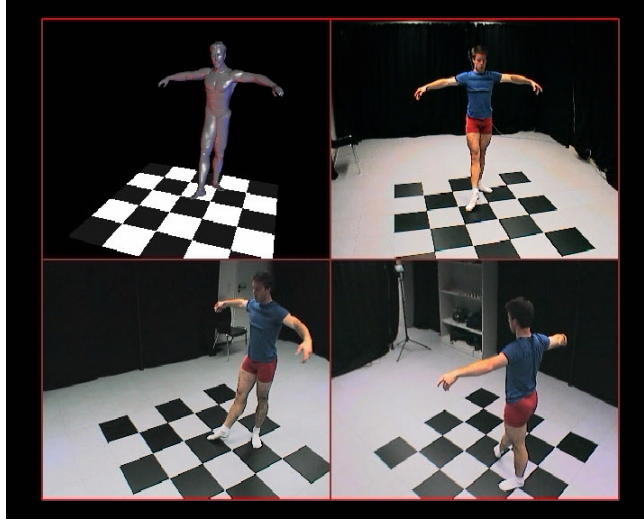
We initialize the optimization search for one time step t with the parameters found at $t-1$. The problem's search space is constrained by performing a hierarchical optimization that exploits structural knowledge about the body. The poses of individual kinematic sub-chains in the skeleton are determined separately. Body segments are considered in descending order with respect to the skeleton hierarchy and their respective influence on the silhouette appearance. In consequence, we first determine the torso pose, thereafter the poses of arms, legs, and head, and finally the parameters for the hands and feet.

For arms and legs we employ a custom 4-degree-of-freedom parameterization.

In order to handle rapid body motion the pose determination for each limb is preceded with a regular grid search in the 4D pose space. The best grid point found is used as starting point for the subsequent downhill optimization. Interpenetrating limb poses are also discarded during grid sampling.

Optionally, the complete pose estimation scheme is iterated.

Silhouette-based Motion Capture



The video shows three of the input camera views and the body model correctly following the motion of the dancer.



Model Fitting Acceleration

- Performance bottlenecks
 - Energy function
 - Transfer bandwidth (frame buffer read/write)
 - Rendering model geometry
 - Optimization
- Improvements
 - Modified XOR evaluation
 - Distributed Model Fitting

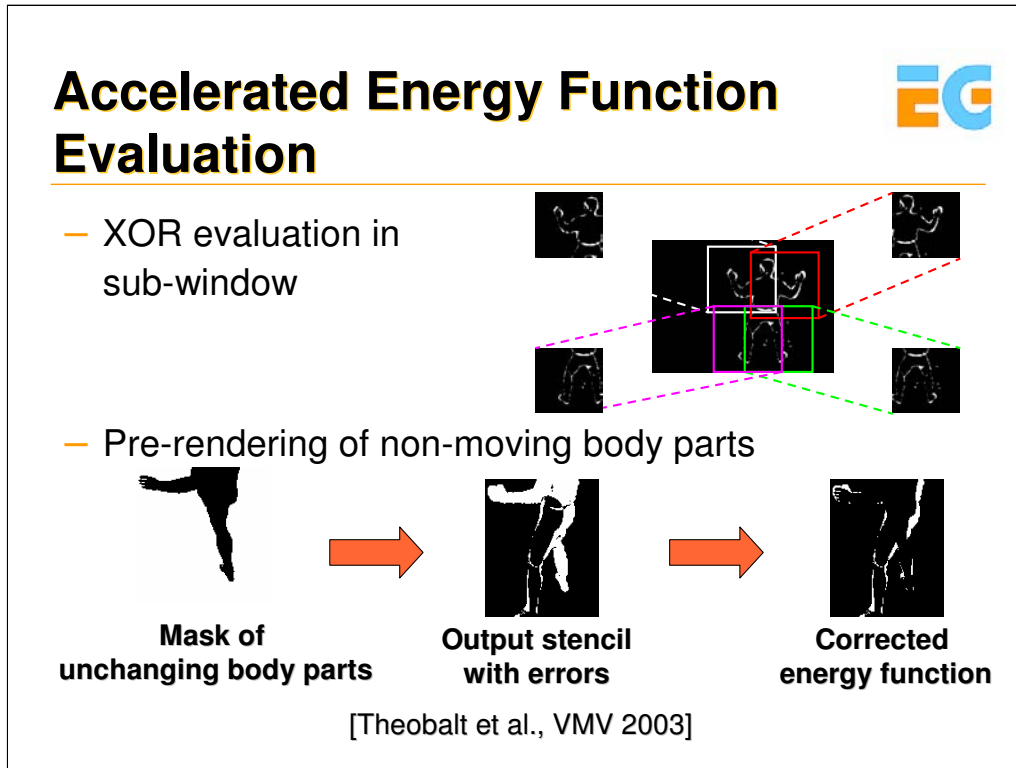
[Theobalt et al., VMV 2003]

The performance of the model fitting method is limited by two factors: the time needed to evaluate the XOR energy function and the runtime of the numerical minimizer itself.

The performance of the energy function evaluation on the GPU is constrained by the overhead inflicted by the necessary frame-buffer read/write operations, as well as the overhead to render the model geometry.

The performance of the pose determination procedure is constrained by the fact that on a single PC one can only optimize the pose for one body segment at a time. Given more CPUs, however, the parameters for independent segments on the same level of the skeleton hierarchy could be efficiently estimated in parallel.

Thus, we have enhanced the GPU-based XOR computation in order to capitalize on the compartmentalized nature of the pose determination problem [1]. The implicit parallel structure of the problem also suggests a distributed implementation of the motion capture sub-system as a whole.



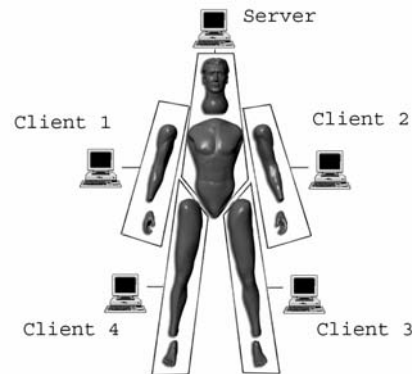
We modify the error function evaluation in two ways in order to exploit the compartmentalized nature of the pose determination problem:

- 1) Instead of rendering the frames in full size, the rendering window for each camera views is confined to a limited area around the image plane location of that kinematic sub-chain which is currently optimized. By this means the amount of data that has to be transferred during frame-buffer read/write is significantly reduced.
- 2) The rendering overhead for the body model can be reduced if only the geometry of those body parts is displayed whose pose is currently optimized. The additional error in the XOR value in each camera view that is inflicted by not showing large parts of the model has to be compensated. We do this by applying an image-mask of unchanging body parts that is generated prior to the optimization. The bottom figure illustrates the process for one camera view.



Distributed Implementation

- Client-server setup
- 5 PCs – 5 CPUs and 5 GPUs
- Fitting procedure
 - Server : torso
 - Clients/server in parallel: arms, legs, head
 - Clients in parallel: feet, hands



[Theobalt et al., VMV 2003]

Our distributed pose determination system is a client-server setup using 5 PCs, i.e. 5 CPUs and 5 GPUs. The hierarchical pose estimation procedure first determines the parameters of the torso on the server. The result is transferred to all clients. Now, the server and the clients determine the parameters of legs, arms and head in parallel. The results are broadcasted via the server before, in a final step, the poses of hands and feet are determined.



Results

- Average pose estimation time (s)

Method	Seq. A	Seq. B
XOR single computer	7.98	14.10
Sub-window, pre-render, single computer	3.30	10.10
Sub-window, pre-render, 5 computers	1.16	1.76

Xeon 1.8 Ghz , 512 MB RAM, GeForce3

The table shows average times needed for determining the pose of the body model at a single time step with the different algorithmic alternatives for silhouette fitting. The results we obtained with two test sequences are shown. Seq.A exhibits mostly slow body motion, whereas Seq.B shows an expressive jazz dance performance. Motion capture with the non-accelerated XOR computation on a single computer takes between 8s and 14s. A significant speed-up is obtained if we apply the enhanced energy function evaluation with sub-window constraint and body-part pre-rendering. By employing our distributed implementation we achieve average fitting times close to 1s.



Refined Model Fitting

- Silhouettes
 - Robustly detectable
 - Sensitive to large-scale motion
 - Object texture
 - Highly detailed
 - Sensitive to small movements
- ⇒ Exploit texture for fine-tuning model parameters

The silhouette-based analysis-by-synthesis approach described on the previous slides enables us to reconstruct a dynamic scene description without having to modify the input video footage in any form, e.g. with optical markings in the scene. This is a necessary precondition if texture information is taken from the video streams as well.

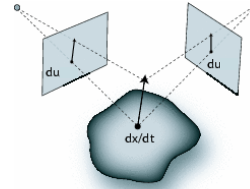
Image silhouettes can be computed robustly and our motion capture approach is fairly insensitive to measurement noise in the image data. However, although a silhouette-based approach robustly captures poses on a large scale, the exact pose of some body parts is hard to infer exactly. Slight pose inaccuracies can often be observed for those segments whose shape exhibits very few features on the silhouette outline that guide the optimization towards the correct parameters, such as the head.

We thus propose to enhance the original motion capture method into a hybrid approach that jointly uses silhouettes and texture information from the video frames for pose determination. The texture information enables the correction of slight pose inaccuracies that exist in the silhouette fit.

Texture-enhanced Motion Capture



- 2D Optical Flow u_i
 - projection of 3D point motion dx/dt into 2D image plane i
- Optical Flow vs. Scene Flow
 - Jacobian matrix
 - Known from camera model
 - Optical Flow, Jacobian & Geometry
 - Solve for 3D motion dx/dt



$$\mathbf{J}_i = \frac{\partial \mathbf{u}_i}{\partial x, y, z}$$

$$\mathbf{u}_i = \mathbf{J}_i \frac{d\mathbf{x}}{dt}$$

[Theobalt et al., PG 2003]

Our texture-enhanced motion capture method employs the *scene flow*, the 3D equivalent of the 2D optical flow in the image plane, to compute corrective pose updates.

The 2D optical flow is the projection of the 3D motion field of a dynamic scene into the image plane of a recording camera.

A sea of algorithms has been proposed in the computer vision literature to compute this 2D flow field between two subsequent depictions of the scene. In our implementation we employ the method by Lukas and Kanade (see [2] for references to the original literature).

The scene flow corresponding to a set of optical flows in multiple camera views is the set of 3D motion vectors, one for each point in the scene, whose projections are the 2D optical flows. The differential relationship between the optical flow and the scene flow is described by a Jacobian matrix whose entries can be determined from the matrix of a calibrated camera.

Given a set of optical flows from multiple calibrated camera views, and full knowledge about the 3D geometry, it is possible to infer the scene flow vector for each point on the geometry by solving a linear equation system.



Motion Field-guided Model Fitting Algorithm

- Estimate model pose for $t+1$
 - Silhouette fitting
- Render image estimates $I'_{j,t+1}$
 - Texture model with images $I_{j,t}$
- For each model vertex
 - Determine projection coordinates and visibility in each image
 - Compute optical flow between $I_{j,t+1}$ and $I'_{j,t+1}$
 - Calculate 3D vertex motion from motion field
- Update model to conform with motion field

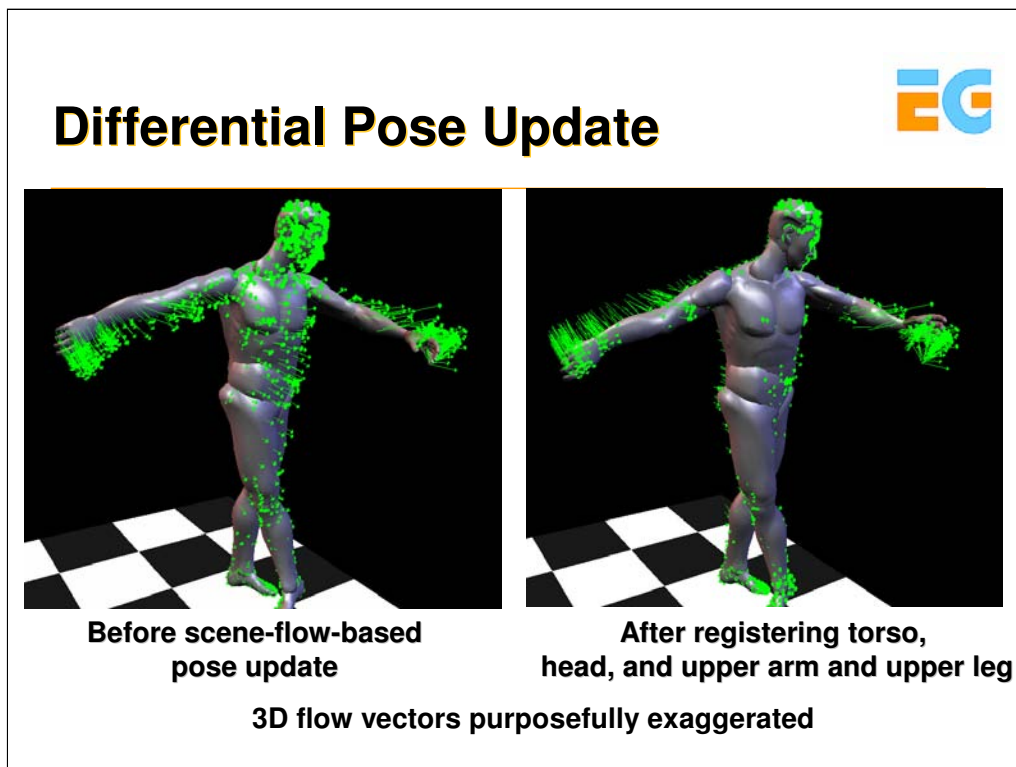
In a predictor-corrector-scheme, we employ the scene flow determination method for computing differential pose updates that correct inaccuracies in the silhouette-fit.

In a first step, an estimate of the pose parameters at time $t+1$ is computed using the original silhouette-based motion capture technique.

Using this first pose estimate, we predict the appearance of the model by rendering and projectively texturing it with the camera images from the previous time step $I_{j,t}$. This way, we generate novel images $I'_{j,t+1}$ showing the predicted model appearance at time $t+1$ in each camera view j .

For each vertex it is determined in which cameras it is visible. For each camera that sees the vertex, the optical flow between its projection into $I'_{j,t+1}$ and $I_{j,t+1}$, is computed. Using all the 2D flow vectors found for this vertex, the corresponding 3D scene flow vector is computed using the method outlined on the previous slide.

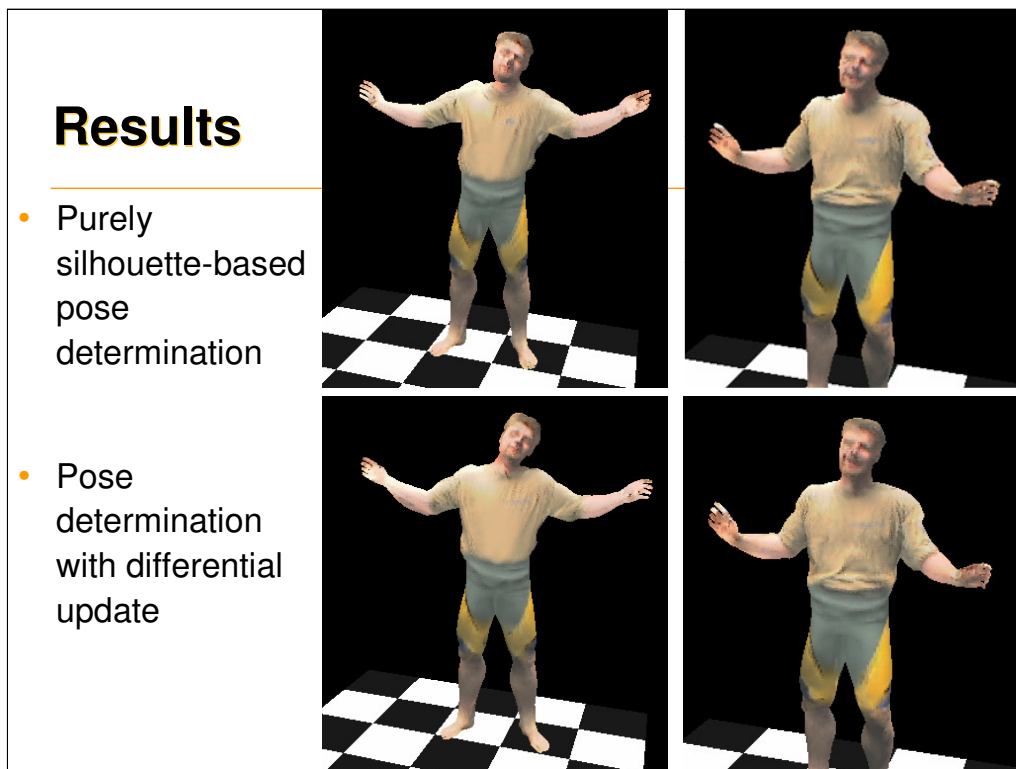
This way, a scene flow field is generated which describes for each vertex a position update that brings the model into a pose that is photo-consistent with all camera images. The corrective flow field is translated into a set of corrective pose parameters for the model by means of a shape registration method originally proposed by Horn (see [2]).



The figure on the left shows the body model in the pose that was found via silhouette-fitting.

The green arrows are corrective scene flow vectors that have been computed for all vertices in the body model using the predictor-corrector scheme described on the previous slide. The length of the flow vectors has been purposefully exaggerated in order to better visualize the flow field.

The figure on the right shows the pose of the model after the corrective pose update parameters for the torso, the head, the upper arms and the upper legs have been applied to the skeleton.



The four figures illustrate the visual improvements achievable with the texture-enhanced motion capture method. The top row shows screen-shots of free-viewpoint videos that have been reconstructed with pure silhouette-fitting.

The bottom row shows renditions from the same virtual camera views but with the scene-flow-based pose correction applied. Improvements are mainly visible in the face and on the torso. Block artifacts in the images are due to the limited camera resolution of 320x240 pixels.



Rendering

- Render model geometry in captured pose
- Dynamic projective texturing with camera images
 - Per-vertex blending and interpolation in fragments
 - Texture information (camera image) $tex_j(i)$
 - Visibility $Vis_j(i)$
 - View-independent spatial blending weights $\omega_j(i)$
 - Optional: view-dependent weights $\rho_j(i)$

$$\text{Final color } c_i = \sum_j^N Vis_j(i) \cdot \rho_j(i) \cdot \omega_j(i) \cdot tex_j(i)$$

The reconstructed 3D videos are rendered by displaying the body model in the sequence of captured poses, and projectively texturing it with the video frames at each time step of video.

For vertex i in the model we compute the final color c_i by appropriately weighting and summing the color contributions from each input camera view $tex_j(i)$. $Vis_j(i)$ is the visibility of vertex i in camera view j , $\omega_j(i)$ is a spatial blending weight depending on the relative orientation of the vertex with respect to the input camera. We can assume that the reflectance of most garments is close to lambertian. It is thus valid to compute the spatial texture blending weights only based on the orientation of a vertex with respect to the input cameras. The camera which sees a surface element most head-on is assigned the highest blending weight.

However, in order to model view-dependent reflectance effects appearing in some garments, an optional view-dependent rescaling factor $\rho_j(i)$ can be included into the color computation. It weights the camera contributions depending on the relation between outgoing viewing direction and camera viewing direction. The per-vertex blending weights are interpolated in the fragment stage of the GPU.



Video Texturing

- Time-varying texture

- Cloth folds
- Local shadows
- Facial expressions



- Detail preservation

- Small inaccuracies between geometry and silhouettes
- Soft shadow visibility
- Camera image dilation
- Controllable spatial blending weights



By generating a novel texture for each time step of video, subtle dynamic details in surface appearance, such as wrinkles in clothing, local shadows and facial expressions are faithfully reproduced (see three Figs. In top row, block artifacts are due to the limited camera resolution of 320x240).

Although our automatic model initialization approach generates a body geometry that matches the appearance of the actor very well, small geometry misalignments between the virtual and the real human may still exist. In These mismatches may lead to erroneous projections of parts of the texture belonging to occluding geometry onto actually more distant parts of the body (bottom left Fig.). Furthermore, seams originating from projected silhouette boundaries may appear on the body.

We employ three techniques to counter these effects. Firstly, we compute the visibility of each vertex from a set of slightly displaced camera views (soft shadow visibility). This way, projection artifacts at occlusion boundaries are prevented (bottom right Fig.). Secondly, we dilate the segmented video frames at the silhouette boundaries to prevent the seams. Thirdly, we apply a special method to compute controllable spatial texture blending weights which is illustrated on the next slide.

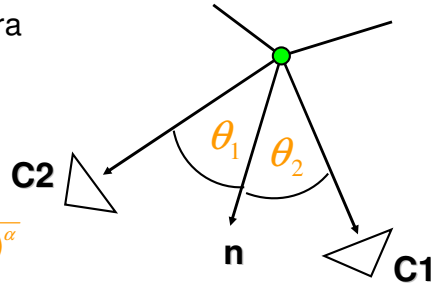
Controllable View-independent Spatial Texture Blending



- dependent on surface-to-camera orientation

$$\omega_j(i) = \frac{1}{\Theta_j(i)}$$

$$\omega'_j(i) = \frac{1}{(\max_k(\omega_k(i)) + 1 - \omega_j(i))^\alpha}$$



The easiest way to compute a view-independent spatial blending weight for vertex i and camera j , $\omega_j(i)$, is to apply the reciprocal of the angle $\theta_j(i)$ between the vertex normal and the viewing vector towards the camera.

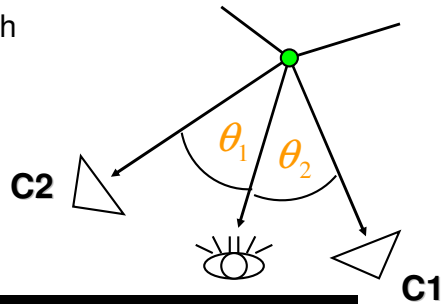
We propose alternative spatial blending weights, $\omega'_j(i)$, that give a better control over the influence of each camera on the appearance of the final texture. The alternative weight computation assigns a proportionally high weight to the camera which sees the vertex most head-on. The sharpness value α controls the amount by which the influence of the best camera is exaggerated. In the limit, $\alpha \rightarrow \infty$, only the best camera is contributing to the color of the vertex. The three figures on the bottom of the slide illustrate the influence of the sharpness value on the final renditions.

View-dependent Rescaling Weights



- Dependent on orientation with respect to output viewpoint

$$\rho_j(i) = \frac{1}{\Theta_j(i)}$$



View-independent blending only



View-dependent rescaling applied

Optionally, a weight $\rho_j(i)$ can be computed that view-dependently rescales the view-independent blending weights.

The weight $\rho_j(i)$ for vertex i and camera j is the reciprocal of the angle $\Theta_j(i)$ between the viewing direction towards the camera and the current outgoing viewing direction.

Free-Viewpoint Video Viewer



- Rendering: 30 fps (GeForce3)



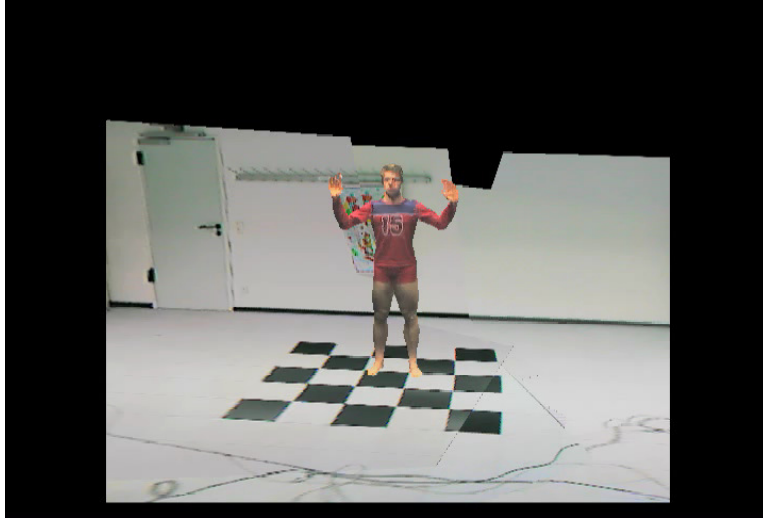
The top right image shows a screen-shot of our renderer which displays free-viewpoint videos at 30 fps on an Nvidia GeForce3 GPU.

The bottom row shows screen-shots of free-viewpoint videos that have been rendered from virtual camera views different from any input camera view.

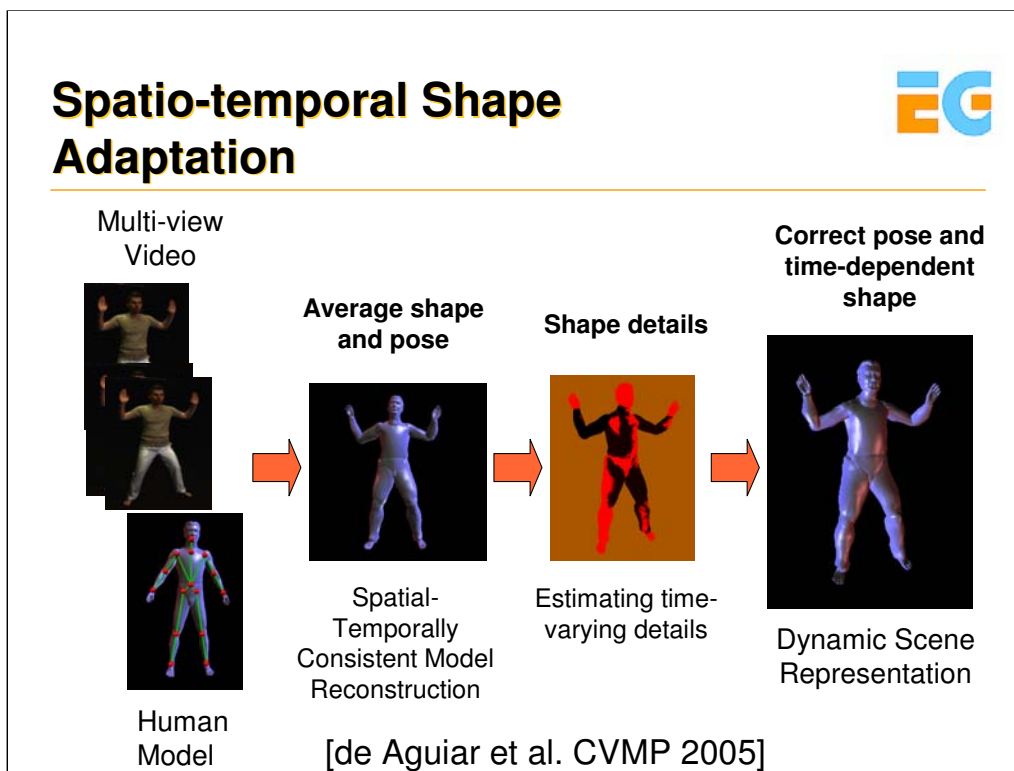
The input video footage for these sequences (as well as all the other sequences shown on previous slides) has been captured with 8 video cameras that provide an image resolution of 320x240 pixels each.



Result



A free-viewpoint video of ballet dancer that we have reconstructed with our method.



In our original pipeline, we adapted the kinematic template model to the shape of the actor in one reference pose only. A natural extension to this approach is to try to recover spatio-temporally varying scene geometry in order to handle at least the coarsest per-time-step deformations.

This has been explored in [4]

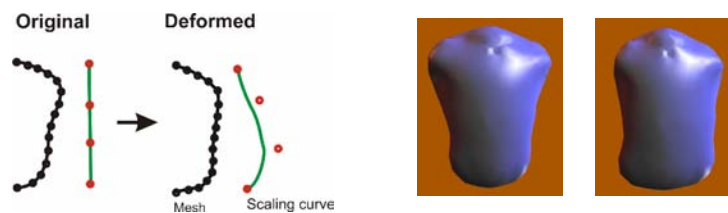
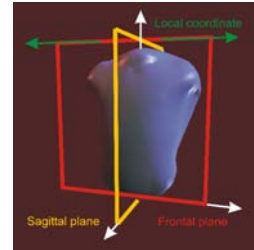
Our algorithm consists of two main steps. In the first step, a spatio-temporally consistent shape model is reconstructed by adapting the scaling parameters of the template to all video frames instead of just the reference one.

In a second step, per-time step multi-view photo-inconsistent regions are identified and their shape adjusted to recover per-time step deformations on isolated surface parts. These two main steps of the deformation method are detailed on the following slides.



STEP1: STC Model Reconstruction

- Find spline/scaling parameters that match the shape of the person „on average“ over several frames
- Iterate fitting and shape refinement over whole sequence several times
- Spatio-temporally consistent model



As briefly outlined earlier, the idea is to first reconstruct a spatio-temporally consistent shape model. This model is still a model with static shape parameters. The main difference to the original pipeline is that these parameters are now reconstructed by fitting the model to several time steps instead of just one time step. In consequence, pose estimation and shape refinement are iterated several times to yield the shape-adapted geometry.

STEP 2: Per-time Step Shape Refinement



- Diffuse surface → Identify photo-inconsistent regions [Fua and Leclerc, 1995]



- Deform vertices to
 - Increase photo-consistency
 - Preserve silhouette-consistency
 - Preserve model quality
- Find vertex displacements r_i for each vertex v_i minimizing

$$E(v_j, \vec{r}_j) = w_I E_I(v_j + \vec{r}_j) + w_S E_S(v_j + \vec{r}_j) + w_D E_D(v_j + \vec{r}_j) + w_P E_P(v_j, v_j + \vec{r}_j)$$

In step2, the spatio-temporally consistent model is deformed to recover per-time-step shape variations. To do so, for a set of seed vertices, optimal displacements are computed such that a multi-view consistency criterion is optimized. After multi-view consistency is optimized, the non-multi-view-consistent regions are smoothly deformed. For details on this deformation, please refer to [4]. The multi-view consistency criterion is represented by an energy functional comprising of 4 different error terms illustrated on the following slide.

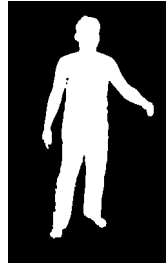
STEP 2: Per-time Step Shape Refinement



- Energy terms

$w_I E_I(v_j + \vec{r}_j)$ Photo-consistency

$w_S E_S(v_j + \vec{r}_j)$ Silhouette distance



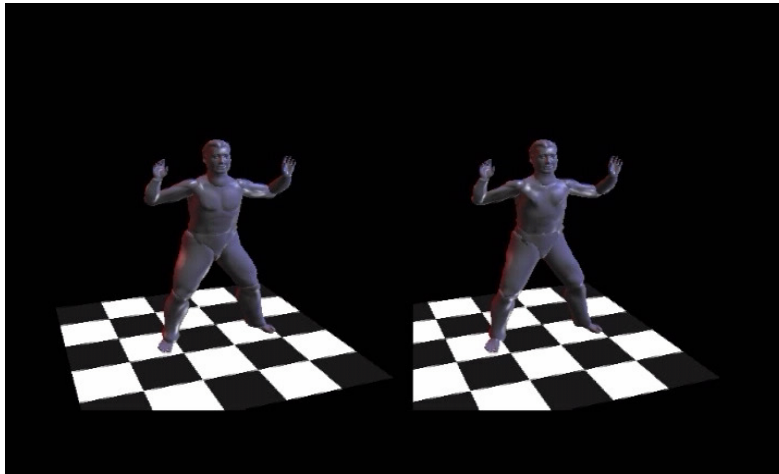
$w_D E_D(v_j + \vec{r}_j)$ Triangle distortion

$w_P E_P(v_j, v_j + \vec{r}_j)$ Change of visible camera set

The four error terms being minimized while solving for the optimal displacements for the seed vertices are shown here.

The first one measures the multi-view color consistency across all cameras in which a vertex is visible. Silhouette consistency is measured in terms of the distance (inner and outer distance field) to the silhouette boundary. Displacements that move a vertex closer to the silhouette boundary are favored. Triangle distortion is taken into account as well to preserve shape integrity. Finally, strong displacements leading to changes in the set of cameras that see a vertex are also penalized.

Results – Comparison



The video on this slide shows a side-by-side comparison between the captured human shape using the original pipeline, i.e. without spatio-temporal shape adaptation and the new spatio-temporally adapted human model.

The per-time step shape adaptation helps prevent certain texturing artifacts that are due to starkly incorrect shape. These inaccuracies may not be fully corrected by texture-based means only.

However, the underlying template geometry still imposes limits on how well per-time-step deformations can be captured. In the following, we will discuss novel directions in 3D video which allow us to capture more detailed dynamic model representations from input footage.



Discussion

- A priori model simplifies geometry + texture reconstruction
- Convincing free-viewpoint renderings
- Simple capture of time-varying geometry feasible
- Model general but limited accuracy
→ alternative approaches (after break)

References

- [1] C. Theobalt, J. Carranza, M. Magnor, H.P. Seidel, *A Parallel Framework for Silhouette-based Human Motion Capture*, in Proc. of Vision, Modeling and Visualization, p.207-214, Munich, Germany, 2003.
- [2] C. Theobalt, J. Carranza, M. Magnor, H.P. Seidel, *Enhancing Silhouette-based Human Motion Capture with 3D Motion Fields*, Proc. of Pacific Graphics 2003, p.185-193, Canmore, Canada.
- [3] J. Carranza, C. Theobalt, M. Magnor, H.P. Seidel, *Free-Viewpoint Video of Human Actors*. in Proc. of ACM SIGGRAPH 2003, p.569-577, San Diego, CA.
- [4] E. de Aguiar, C. Theobalt, M. Magnor, H.-P. Seidel, *Reconstructing Human Shape and Motion from Multi-view video*. 2nd European Conference on Visual Media Production (CVMP), p. 42-49. London, UK. 2005.



Model-based 3D Video II
(25 min)

Edilson de Aguiar

MPI Informatik



Alternative Model-based Approaches

Motivation

- Human Models for 3D Video
 - Coarse approximation (3K-10KΔ)
 - Lack of important details
 - Clothing
 - Surface deformations
- Improving realism
 - Better models
 - Ideally the true geometry
 - Need to capture
 - Time-varying surface detail, cloth motion
 - Skinning deformation



The previously presented template model for 3D video, even after reconstruction of some per-time step deformation, was only a coarse approximation of the true geometry of the subject performing. Important details, like the appearance of the apparel that the actor is wearing or the way the skin deforms over time are not easily reproduced. To improve realism, a better representation for the subject is needed. Ideally, one would want to use the true shape of the subject as a model, since it contains all important details. In the following we will show some algorithmic ways to come closer to such a better model.



Improving 3D Video Applications

- Problem:
 - Models are coarse (3K-10KΔ)
- Goal:
 - Use better models (true shape)
- Solution
 - Laser-scanned models
 - Technology is more available
 - True geometry – more details
- How to integrate a human scan?

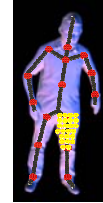


Since current models used in 3D Video applications are relatively coarse, in order to improve realism a better representation for the subject is needed. Recently, laser scanning technology is becoming cheaper and easier available. The main advantage is that a laser scanner can easily capture the true shape of the subject, including most of the important surface detail at fine resolution. By incorporating such better input models in current 3D Video systems one would expect a considerable amount of improvement in terms of quality and realism.

How to integrate human scans?

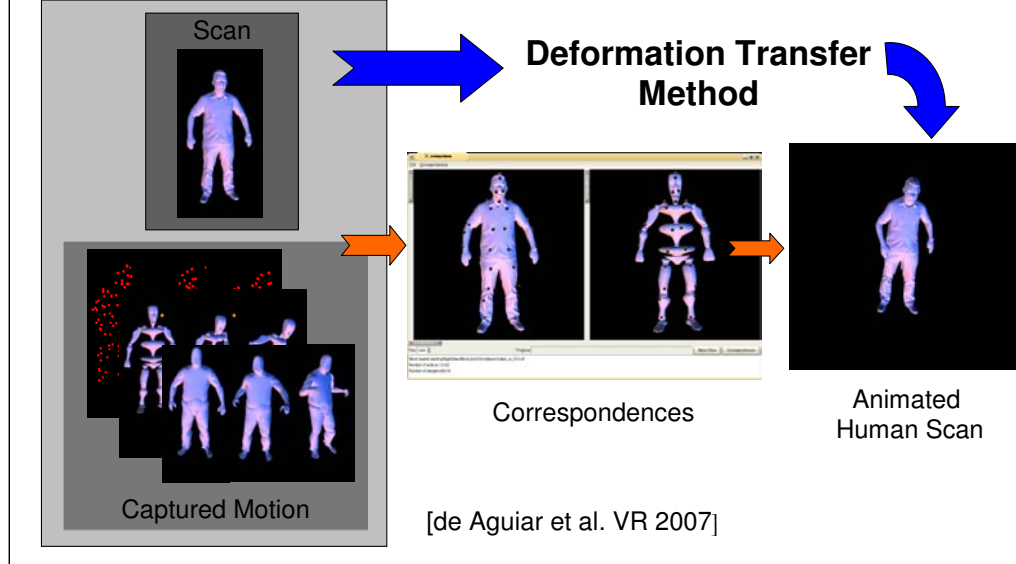


- Using traditional methods
 - Kinematic structure
 - Skinning weights
- Time-consuming and expensive process
- Need to do it for each model
- Alternative
 - Guided mesh-based deformation interpolation method
 - Fast approach
 - No need for underlying skeleton



Traditional 3D Video systems are mostly built around models with an underlying kinematic structure. If we want to replace the model with the coarse surface geometry by a laser-scanned model, we need to fit a kinematic structure to it and thereafter determine the skinning weights for the vertices in order to correctly deform it during tracking or animation. This process is time-consuming and expensive being at best a semi-automatic procedure. Another limitation is that the same procedure should be applied to each new scanned model. An alternative to this expensive process is a mesh-based deformation scheme, which is efficient and does not require the specification of any underlying skeleton structure for each particular subject.

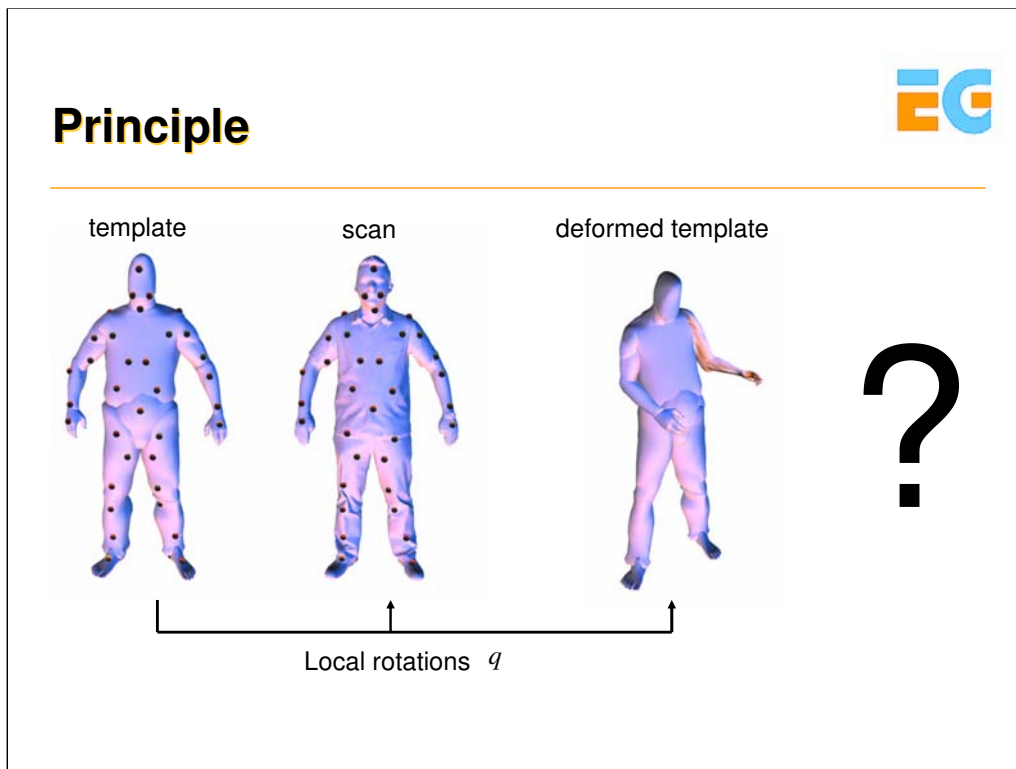
Overview



A simple and easy way to integrate a laser-scanned human in the traditional pipeline of a model-based 3D Video system can be achieved through a mesh-based Laplacian deformation scheme [1]. Having as inputs a scanned model of the subject and a description of her motion (for instance captured with the original marker-free approach described in part I of the model-based section), the main goal is to transfer the input motion to the scanned model. By formulating the motion transfer problem as a deformation transfer problem and by setting a small amount of corresponding marked vertices between both models (template and scan), an automatic guided deformation interpolation technique can be used to animate the scanned human model efficiently.

Reference:

[1] E. de Aguiar, C. Theobalt, C. Stoll and H.-P. Seidel, *Rapid Animation of Laser-scanned Humans*. In Proc. of IEEE Virtual Reality 2007, pp. 223-226, Charlotte, USA.

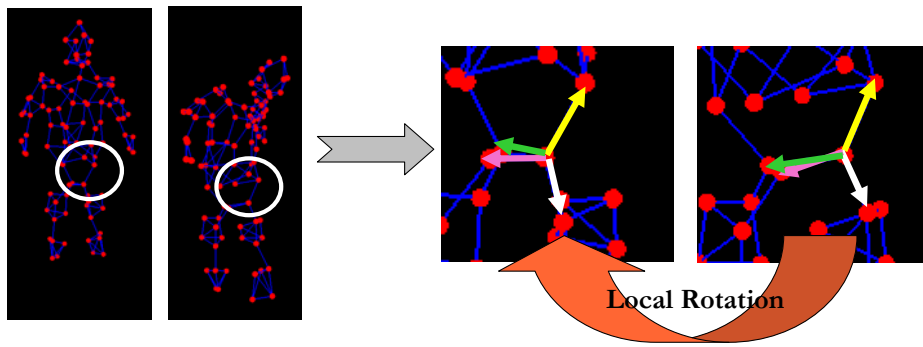


After first automatically aligning both models (template and scan) in a given reference pose, the user marks a set of vertices on the template and assigns to each of them a corresponding vertex in the scanned model. Through placement of markers the characteristics of the motion and the surface skinning are defined, but also retargeting constraints can be set. For each time instant of the input motion sequence, represented by a deformed template, it is our goal to transfer this relative deformation to the human scan, bringing it to the same pose as the deformed template. A good solution is to use a Laplacian mesh deformation scheme that jointly employs rotational and positional constraints on the markers to compute the new sequence of poses for the human scan. This method is divided in three steps and it is performed for each time step of the input motion sequence. In the first step, local rotations for all markers belonging to the scan are estimated from the rotations of the corresponding markers on the template between its reference pose and its pose at the current time step.

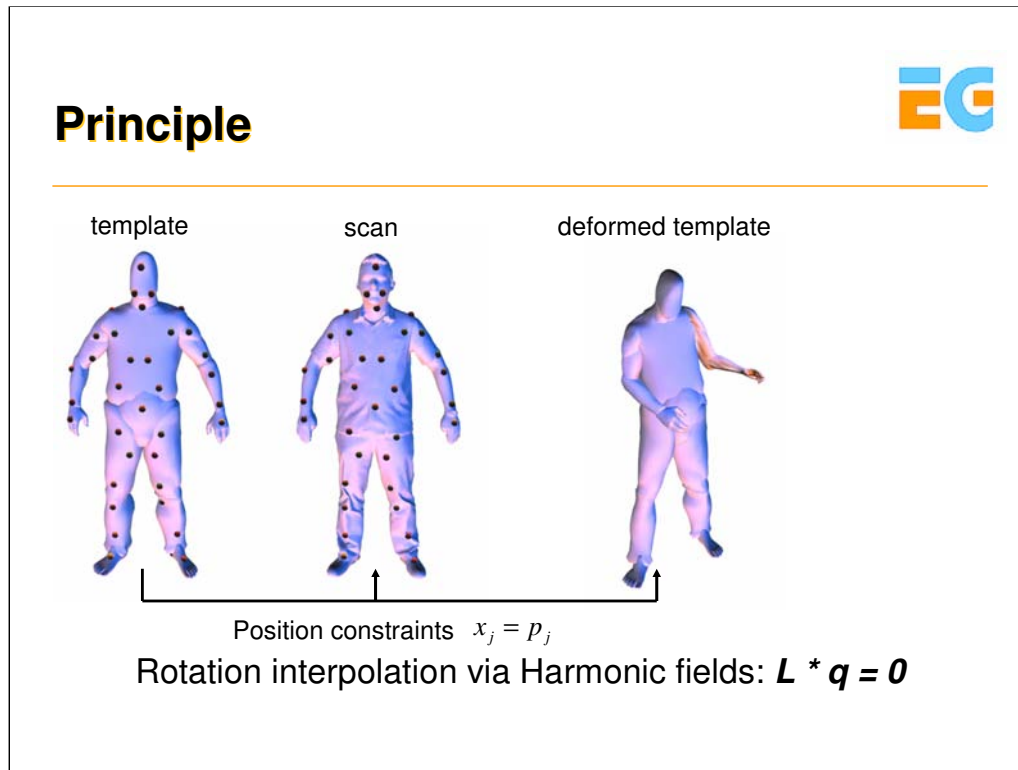


Principle

- Estimating rotations
 - Graph-based method
 - Local frames – Minimum Spanning Tree
- Minimal Rotation -- Jacobian



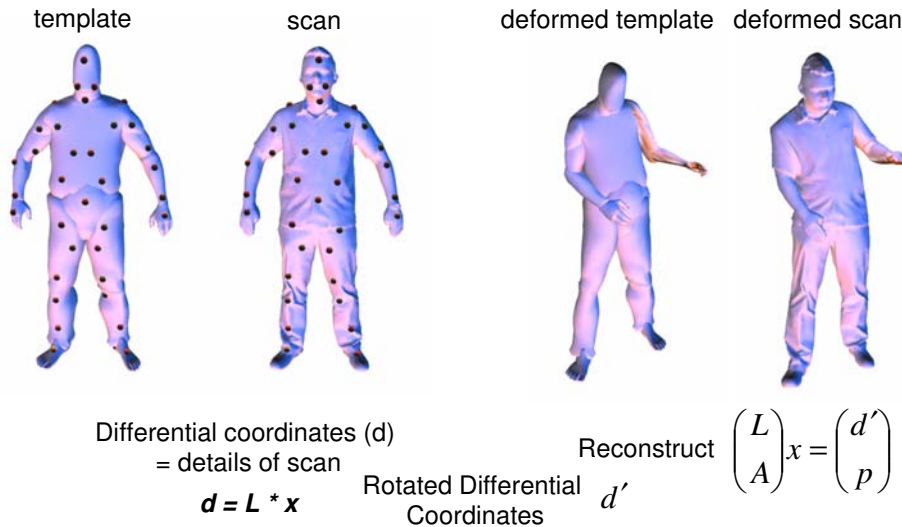
A local rotation for each marker belonging to the scan can be calculated from the rotation of the corresponding markers on the template between its reference pose and its pose at a different time by means of a graph-based method. Template markers can be considered as nodes in a graph and edges between them can be determined by constructing the minimal spanning tree. For each marker, the minimal rotation that makes its outgoing edges at the reference time matches its outgoing edges at time t can be found. Thereafter, local rotations are converted to quaternions and assigned to the corresponding partners in the human scan.



In the second step, rotations are interpolated over the scanned mesh. Regarding each component of a quaternion as a scalar field defined over the entire mesh, a smooth interpolation is guaranteed by regarding these scalar fields as harmonic fields. The interpolation is performed efficiently by solving the Laplace equation ($Lq = 0$) over the whole mesh with constraints at the marked vertices. In the equation, L is the discrete Laplace operator based on the cotangent-weights. Position constraints for the Laplacian deformation scheme are calculated from the displacements of the corresponding markers on the template between its reference pose and its pose at the current time step.



Principle



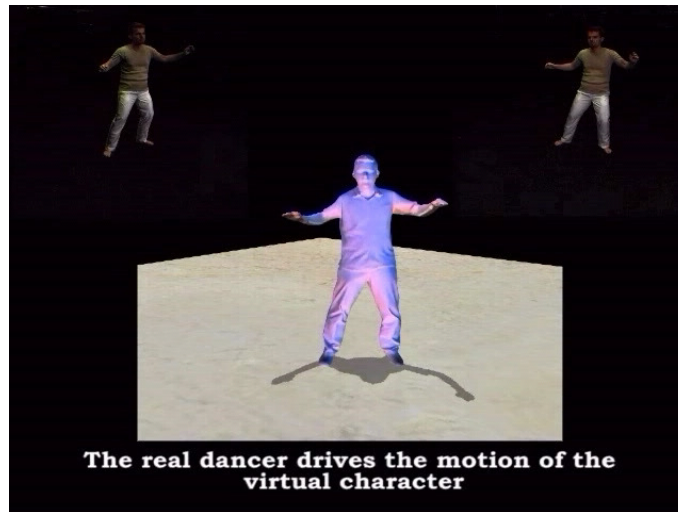
The differential coordinates d are computed once at the beginning of the sequence by solving the given linear system. In the last step, the vertex positions of the scanned model are reconstructed such that it best approximates the rotated differential coordinates, as well as the positional constraints. This can be formulated as a least-squares problem and transformed into a linear system. During the whole process the Laplacian matrix L does not change, which allows a sparse matrix decomposition and execution of only back substitution for each frame.

Marker-less Motion Capture Results



- Skinning + implicit deformation

5 FPS

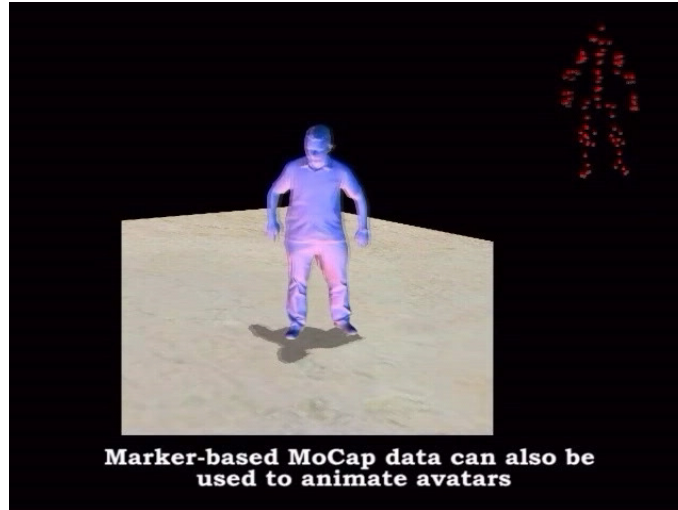


Using the silhouette-based marker-free motion capture system described in part I and [2] that employs a coarse template body model, only eight video cameras are needed to capture the subject's motion. Since the template is already used for tracking, the proposed algorithm is straightforwardly applied to map the captured motion to scanned models. The video shows that this method can accurately transfer poses captured on video to scanned models of even different subjects.

Reference:

[2] J. Carranza, C. Theobalt, M. Magnor, H.P. Seidel, *Free-Viewpoint Video of Human Actors*. In Proc. of ACM SIGGRAPH 2003, p.569-577, San Diego, CA

Also useful for Character Animation



Marker-based MoCap data can also be used to animate avatars

[MoCap data from the Eyes Japan Co. Ltd. database]

There are also many applications for the proposed method in fields related to 3D video, such as computer animation. Nowadays, optical motion capture data is presumably amongst the most widely-used motion descriptions in animation production. By using the presented method, MoCap data can be easily used to animate high-quality surface models while bypassing the drawbacks of the traditional animation pipeline. After transforming the MoCap data into a moving template model (straightforwardly done by transforming the actual bone skeleton into a triangle mesh using any standard animation software), the guided deformation interpolation method can be applied to produce the animation. Note that even raw marker-trajectories can be used as input to this method. The video shows that the scanned model realistically performs the motion while exhibiting lifelike non-rigid surface deformations. Motion retargeting is feasible by appropriately placing constraints.

Guided Mesh-based Deformation Interpolation method



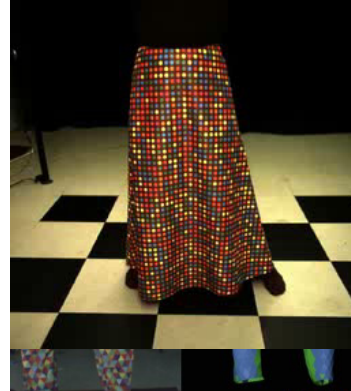
- Simple and efficient
 - Fits into original model-based 3D Video system
 - Can be used with many acquisition systems
 - Easy and intuitive to control
 - Flexible method
 - Different subjects and even animals
- Main drawbacks
 - Only transfer input motion to scanned models
 - Improve realism in 3D Video even further
 - Capture cloth motion
 - Capture skinning deformation

The guided mesh-based deformation interpolation approach is a flexible, simple and fast scheme that allows the integration of high-quality scanned models into the pipeline of conventional 3D Video systems and consequently a considerable amount of improvement in terms of quality and realism. The method requires a minimum of manual interaction, it is flexible, easy and intuitive to use, and simultaneously solves the animation, the surface deformation, and the motion retargeting problems. Although this method allows the integration of the true shape into the pipeline, it is not able to correctly reproduce the temporal behavior of the garments and the deformation of the skin while the subject is performing, in case the input motion description does not have this information. In order to achieve this, new methods will be presented.



Marker-based Cloth Capture

- Simulation methods
 - Parameter tweaking
 - Non-homogeneous textiles
- Marker-based Capturing
 - Motion from real subject
 - Cloth = initial triangle mesh
 - Color-coded markers
- Recent improvements
 - Better results by filling occluded regions



Capturing and Animating Occluded Cloth R. White, K. Crane, D. Forsyth, *ACM Transactions on Graphics (SIGGRAPH)*, 2007
 Garment Motion Capture using Color-Coded Patterns
 V. Scholz, T. Stich, M. Keckeisen, M. Wacker and M. Magnor.
 In Eurographics 2005

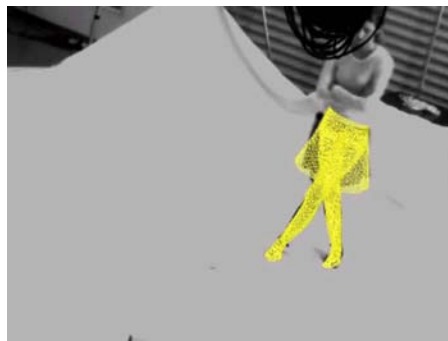
As pointed out earlier, capturing the motion of garment is a big challenge in 3D video reconstruction. In computer graphics, the behavior of the garments while the subject is moving is usually simulated. However, tweaking the parameters to achieve a particular look is fairly difficult and numerical instabilities may frequently happen. In order to reproduce garment motion more faithfully, marker-based capturing techniques [3] were developed that use color-coded markers and a video camera acquisition setup to capture true cloth deformation, thereby allowing for a higher animation quality. While this allows for very detailed reconstruction, traditional multi-view acquisition problems, like occlusion, make the reconstruction still a hard problem. Recently, a new approach explicitly addressed the latter problem and fills in occluded regions of the tracked garment [4].

References:

- [3] V. Scholz, T. Stich, M. Keckeisen, M. Wacker and M. Magnor, Garment Motion Capture Using Color-Coded Patterns, *Computer Graphics Forum (special issue Eurographics 2005)*, vol. 24, no. 3, pp. 439-448, August 2005.
- [4] R. White, K. Crane, D. Forsyth, Capturing and Animating Occluded Cloth, *ACM Transaction on Graphics (SIGGRAPH)*, 2007.

Marker-less Cloth Capture

- Marker-based methods
 - Inappropriate for 3D Video
- Marker-less Capturing
 - No markers
 - Two models
 - Human model
 - Cloth model
 - Jointly optimizing for the human pose and cloth dynamics.
- Main disadvantage of cloth motion capture
 - One model for each component of the scene → more complex



A system for articulated tracking incorporating a clothing model. B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette and H.-P. Seidel. In *Machine Vision and Applications (MVA)*

While marker-based cloth motion capture methods are able to generate good results for animation purposes, the use of the color-coded markers make them inappropriate for 3D Video applications. Instead, marker-less methods have been developed [5] where no intrusion in the scene is necessary. The behavior of the cloth while the subject is moving can be captured by using a template human model (with underlying kinematics structure) and a template cloth model. By jointly optimizing both models at each frame of the video footage, these methods are able to capture the deformation of the cloth, e.g. wrinkles. After that, simulation methods [6] or mesh deformation methods [1] can also be used for transferring the motion to different clothing styles.

Despite some very good capturing results on specific types of scenes, cloth motion capture approaches bear the disadvantage that a separate type of model and often a separate type of tracking method has to be used in combination with the already difficult body motion tracking itself. This greatly reduces the flexibility of the method.

References:

[5] B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette and H.-P. Seidel. A system for articulated tracking incorporating a clothing model *Machine Vision and Applications (MVA)* Vol. 18, No. 1, pp. 25-40

[6] N. Hasler, B. Rosenhahn, H.-P. Seidel: Reverse Engineering Garments, *Mirage 2007*, pages 200-211, Rocquencourt, France, 2007.



Capturing Skinning deformation

- Skinning deformation
 - Skeleton- or muscle-induced non-rigid surface deformation
- Motion capture approaches don't capture these
- Possible solutions
 - Hundreds of optical markers [Park and Hodgins 2006]
 - Combine a MoCap system with a shape-from-silhouette approach [Sand et al. 2003]
 - Combine a MoCap system with a laser-scanner [Allen et al. 2002] [Anguelov et al. 2005]
- Markers → Inappropriate 3D Video applications

Another category of related approaches tries to infer geometry deformation at a slightly reduced level of complexity, however for the whole body. Skinning is a technique used in computer animation to correctly reproduce non-rigid surface deformation around joints or in the vicinity of bulging muscles.

Traditional motion capture systems are not able to measure such time-varying body shape deformations and therefore their acquisition principle has to be augmented. Some methods use hundreds of optical markings for deformation capture [7], or combine a motion capture system with a range scanner [8,9] or a shape-from-silhouette approach [10] to measure a model of skinning deformation in dependence on skeletal pose parameters. Although achieving good results, most of these marker-based methods require active interference with the scene which makes them inappropriate for 3D Video applications. Also, skinning is a very reduced form to describe surface deformations and it is again inappropriate to model the surface deformation of people wearing wide apparel.

References:

- [7] S. I. Park and J. K. Hodgins: Capturing and Animating Skin Deformation in Human Motion, *ACM Transactions on Graphics*, 25(3): 881-889 (2006)
- [8] B. Allen, B. Curless, and Z. Popovic. Articulated body deformation from range scan data. *ACM Transactions on Graphics (ACM SIGGRAPH 2002)*, 21, 3, 612-619.
- [9] D. Anguelov, P.Srinivasan, D.Koller, S.Thrun, J. Rodgers, J.Davis. SCAPE: Shape Completion and Animation of People. Proceedings of the *SIGGRAPH Conference*, 2005.
- [10] P. Sand, L. McMillan, and J. Popovic. Continuous Capture of Skin Deformation. *ACM Transactions on Graphics (TOG)* 22, 3, 578-586.

Jointly Capturing Motion, Cloth Deformation and Surface Deformation



- Specific method for each subpart of scene
 - Capture cloth motion
 - Capture surface deformations
 - Difficult
- Recently, new approaches
 - Jointly capture motion and time-varying surface deformation (cloth or skin)
 - Data-driven approach [Starck and Hilton 2007]
 - Marker-less deformable mesh tracking [de Aguiar et al. 2007]

Capturing the time-varying shape of a scene featuring all components for which we previously showed individual capturing methods is a difficult task. It would be one option to jointly use all of the given methods in a single scene. However, this may be difficult in practice and not very easy to implement.

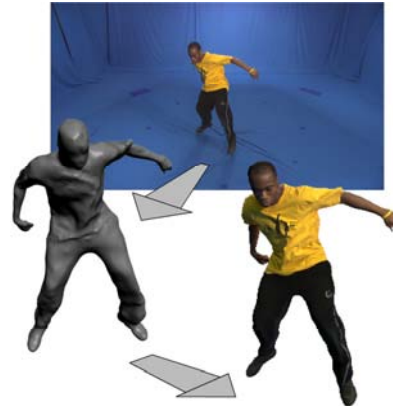
Recently new approaches have been developed that aim at jointly capturing motion and time-varying surface deformations even for subjects wearing wide apparel with complex shape. Most of these methods use a fairly general scene representation, for instance only a 3D triangle mesh instead of dedicated models for each component. By this means, the reconstruction principle remains the same, regardless of the complexity and assembly of the scene.

In the following, we will briefly look at two of these approaches.



Surface Motion Capture

- Data-driven approach
 - Jointly capture motion, varying deformation and appearance
 - No a priori model: VH mesh + stereo
- Pros
 - Marker-less system
 - Cloth motion
- Cons
 - Relatively coarse models
 - Topology changes



Surface Capture for Performance-based Animation
J. Starck and A. Hilton. To appear *IEEE CG&A*, 2007.

The Surface Motion Capture system [11] is a data-driven approach aiming at jointly capturing motion, time-varying deformation and appearance of a moving subject. Using this system no intrusion into the scene is necessary, which makes it appropriate for 3D Video applications. After reconstructing a triangle mesh from the visual hull (VH) and stereo information for each time step separately, additional time-consuming procedures (spherical parameterization and remapping) are used to construct a spatio-temporally coherent model. Although the system is able to nicely capture the motion and temporal cloth deformations for a relatively coarse reconstructed model, its main limitation comes from the topology changes in the model over time. This is an important issue to be considered when developing more advance applications, like 3D Video relighting.

Reference:

[11] Surface Capture for Performance-Based Animation, J. Starck and A. Hilton. to appear *IEEE Computer Graphics and Applications (CG&A)*, 2007.

Marker-less Deformable Mesh Tracking



- Model-based approach
 - Jointly captures motion and surface deformations
 - Mesh only - No kinematic skeleton
- Pros
 - High-quality a priori model: human scan
 - Spatio-temporal coherence implicit
- Cons
 - Range of motions
 - Low-frequency surface details



Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture E. de Aguiar, C. Theobalt, C. Stoll and H.-P. Seidel, In Proc. of IEEE CVPR 2007

The marker-less deformable mesh tracking system [12] is a novel algorithm to jointly capture the motion and the dynamic shape of humans from multiple video streams without using optical markers. Instead of relying on kinematic skeletons, as traditional motion capture methods, it uses a deformable high-quality mesh of a human as scene representation. As opposed to many related methods, it can track people wearing wide apparel, it can straightforwardly be applied to any type of subject, e.g. animals, and it straightforwardly preserves the connectivity of the mesh over time. The main limitations are still the range of motions that can be correctly captured, since only eight cameras are used for recording the multi-view video sequences, and that the system cannot capture the true shape variation of low-frequency surface details, such as wrinkles in clothing. While the cloth and skin globally deform with the model, they seem to be ‘baked in’ to the surface. However, a combination of this method with a per-time-step stereo algorithm would overcome this limitation.

Reference:

[12] E. de Aguiar, C. Theobalt, C. Stoll and H.-P. Seidel, *Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture*. In Proc. of IEEE CVPR 2007, Minneapolis, USA.

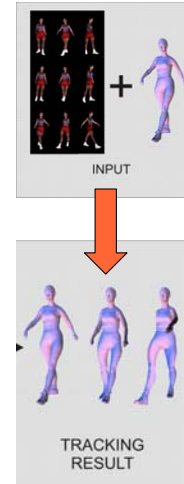


Deformable Mesh Tracking for 3D Video

Since it is more suitable for model-based 3D Video applications, the second algorithm will be described in more details.

Introduction

- Input
 - Static scanned model
 - Multi-view video
- Output
 - Animated scan (motion + non-rigid deformation)
- Problems to be solved:
 - Align scan and recorded images
 - Track vertex positions over time

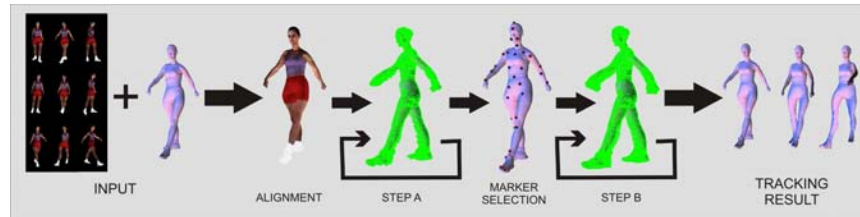


[de Aguiar et al. CVPR 2007]

The input to the system comprises of a static laser-scanned triangle mesh of the moving subject and a multi-view video sequence that shows her moving arbitrarily. As a result it outputs an animated mesh sequence where the model correctly follow the motion of the actor in all video frames. In order to accomplish that, first the laser scan should be aligned to the pose of the subject in the first time step of video and, thereafter, the vertices should be robustly tracked over time.



Algorithm pipeline



- Major steps:
 - Align scan and recorded images
 - Track vertex positions over time
 - Select best set of tracked vertices
 - Use reliable vertices to drive the tracking procedure

The method is composed of four steps. First, the scanned mesh is registered to the pose of the person in the first time step of video. Then, an iterative 3D flow-based deformation scheme is used to extract the motion information of each vertex over time from the images. Thereafter, N marker vertices that are tracked reliably over time are automatically identified. At the end, a more robust method that implicitly enforces structural integrity of the underlying mesh is used to track all vertices correctly.

Data acquisition

- Human scan:
 - Vitus Smart™ full body laser scanner
 - Fast reconstruction (less than 10s)
 - Resolution of 1-2 mm
- Multi-View Video sequences
 - *Post-processing: Silhouette images*



For each test subject, the scanned model and several multi-view video sequences are acquired. The triangle mesh is captured with a Vitus Smart full body laser scanner. After scanning, the subject immediately moves to the nearby area where she is recorded with eight synchronized video cameras that run at 25 fps and provide 1004x1004 pixels frame resolution. The calibrated cameras are placed in an approximately circular arrangement around the center of the scene. After acquiring the multi-view sequences, silhouette images are calculated via color-based background subtraction.

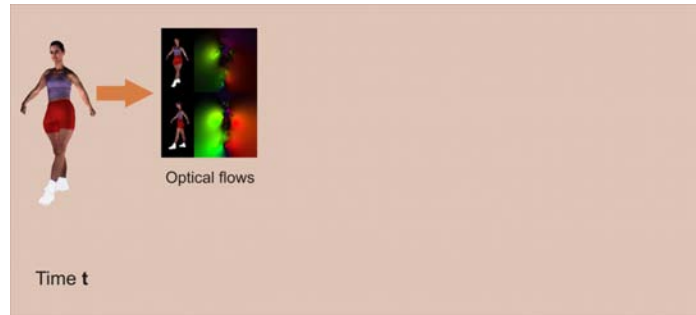


Alignment

- Goal:
 - Align scan with actor's pose in the first frame
- Coarse alignment:
 - ICP-like registration
 - SFS reconstruction of the subject
- Fine alignment
 - Flow-based Laplacian deformation scheme
 - Subtle pose differences are corrected

In an initial alignment we register the scanned mesh with the pose of the person in the first time step of video. To this end, she initially strikes the same pose that she was scanned in. By means of an ICP-like registration the mesh is first coarsely aligned to a shape-from-silhouette reconstruction of the person. Thereafter, the flow-based Laplacian deformation scheme is applied to correct for subtle pose differences.

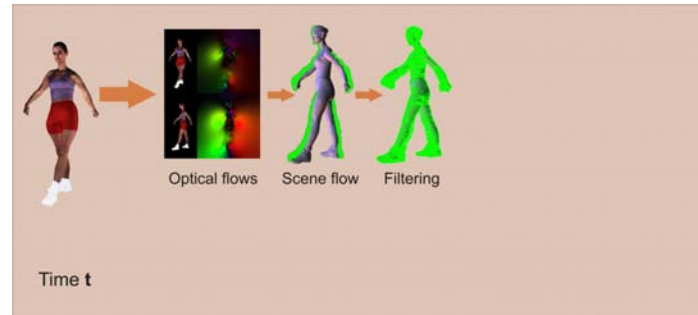
Tracking vertex positions (I)



- Texture model
- Temporary images: project model back into camera views
- Optical flow between original and temporary images

After initial alignment, each individual vertex of the mesh is iteratively deformed based on the 3D optical flow fields that have been reconstructed from the multi-view images. Using subsequent time steps t and $t + 1$, the purely flow-driven mesh tracking approach consists of the following steps: first, the model is projectively textured using the images recorded with the K cameras at time step t . Then, K temporary images are generated by projecting the textured model back into all K camera views. After that, K 2D optical flow fields are calculated between temporary and original images.

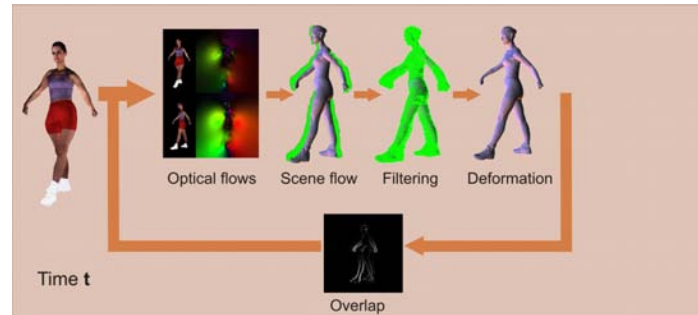
Tracking vertex positions (I)



- Scene flow
- Filter scene flow:
 - Silhouettes
 - Gaussian low-pass filter

Given the model, calibrated cameras and the optical flow fields for all camera views, the scene flow can be computed by solving a linear system for each vertex that is visible from at least two camera views. The generated 3D flow field is parameterized over the mesh's surface and it describes the displacement by which the vertex should move from its current position. Thereafter, the 3D motion field is filtered in order to remove noise and outliers according to a silhouette-consistency criterion, and a Gaussian low-pass kernel is applied over the entire flow field.

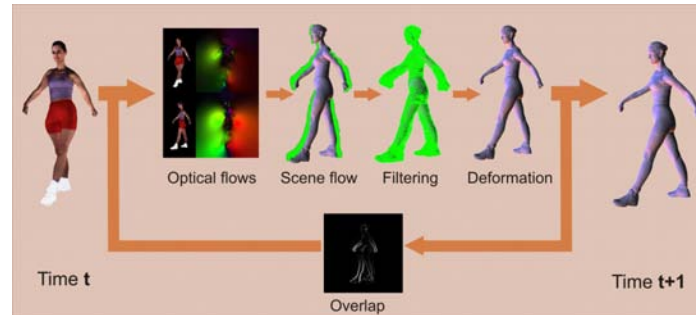
Tracking vertex positions (I)



- Deform scan using scene flow
- Iterate steps:
 - Overlap between silhouettes and rendered model silhouettes

Using the filtered scene flow the model is updated by moving its vertices according to the computed displacements and the iterative process starts again until the overlap error between the rendered model silhouettes and the video-image silhouettes at time $t + 1$ in all camera views is below a threshold.

Tracking vertex positions (I)



- Scanned model is tracked over time!
- However:
 - Quality of animated scan deteriorates over time

At the end, after applying the previously described steps to all pairs of subsequent time steps the model is tracked over the whole sequence. However, since this scheme calculates 3D displacements without taking into account a priori information about the shape of the scanned model, deformation errors accumulate over time.



Tracking vertex positions (I)

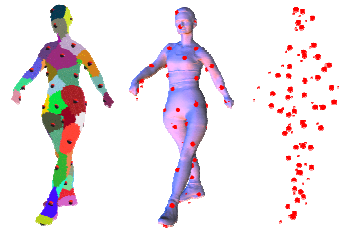
- Quality deteriorates over time
 - Accumulation of correspondence estimation errors
- No a priori information about the scan
- Two-step procedure:
 - Initial 3D motion field
 - Do not deform the model
 - Identify reliable vertices
 - Improved 3D motion field
 - Selected tracked vertices
 - Laplacian deformation scheme

The results of this simple tracking scheme quickly deteriorate due to accumulation of correspondence estimation errors. Since 3D displacements are calculated without taking into account a priori information about the shape of the model, the overall mesh quality is limited. Nonetheless, using this scheme it is possible to automatically identify N marker vertices that can be tracked reliably. Thereafter, an improved tracking scheme, more robust against flow errors, can be used which implicitly enforces structural integrity of the underlying model. This improved method uses the moving marked vertices as deformation constraints to drive a Laplacian deformation framework that makes all vertices correctly follow the motion of the actor.



Selecting best tracked vertices

- Objective:
 - Identify N reliably tracked vertices
- Procedure:
 - Initial 3D motion field
 - 1) Seed points
 - 2) Test seed
 - Silhouette-consistent
 - Motion-consistent
 - 3) Accept if seed passes both tests

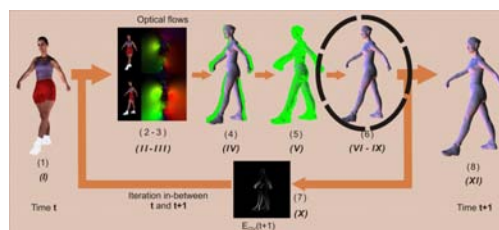


Based on the initial deformation results, this scheme selects N marked vertices of the model that were accurately tracked over time. To this end, first L candidate vertices are chosen that are regularly distributed over the model's surface. A candidate vertex is considered a marked vertex if it has a low error according to two spatio-temporal selection criteria based on silhouette-consistency and motion-consistency measures.

Tracking vertex positions (II)



- Laplacian scheme
 - Differential coordinates ($d = Lv$)
 - Structural details of scan
- Laplacian deformation scheme
 - Constraints from marked vertices
 - Interpolate remaining vertices
- Pipeline:
 - Similar to previous one
 - Deformation step



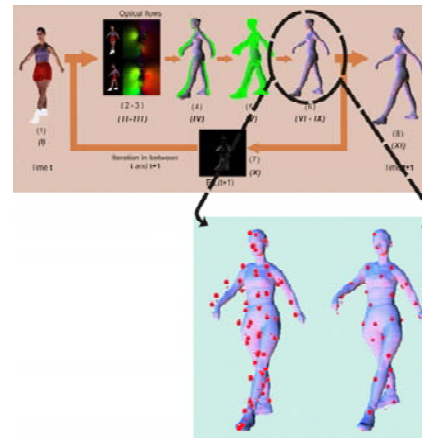
In the final step, the algorithm uses a Laplacian scheme to encode the knowledge about the structural details of the scanned model in terms of the mesh's differential coordinates d . They are computed by solving a linear system of the form $d = Lv$, where L is the discrete Laplace operator based on the cotangent-weights and v is the vector of vertex coordinates.

The main idea is to extract rotation and translation constraints from the motion of the N marked vertices to drive the Laplacian mesh deformation approach. By this means we can extract novel motion fields for each vertex that make the model correctly move and deform like the recorded subject. The individual steps of the Laplacian tracking scheme are very similar to the steps of the previous approach, differing however, in how the deformations are applied.



Tracking vertex positions (II)

- Difference
 - Do not deform all vertices
 - Deform only marked vertices
 - Interpolate other vertices
- Interpolating vertices
 - Rotation and translation constraints for the markers
 - Interpolate rotations
 - Reconstruct pose by solving the Laplace equation



The main difference between the first and second tracking schemes is the way the calculated deformations are applied. In the first scheme, deformations are directly applied to all vertices. In the second scheme, the deformations for the marked vertices are used to guide our Laplacian deformation interpolation method: from the motion of each marked vertex a set of rotation and translation constraints is computed. Then, rotations for all markers are interpolated over the model by regarding the quaternion components as harmonic fields. At the end the model in its new target pose is reconstructed by solving the Laplace equation, subject to the constraints derived from the motion of the markers.

Tracking vertex positions (II)



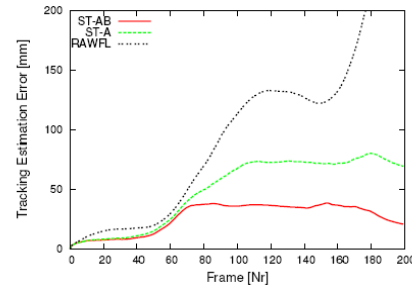
- Tracking result
 - Mesh reconstructed in all poses
 - Differential surface properties preserved
 - Features and details
 - Mesh connectivity preserved
 - Tracking robust against flow estimate errors
- Output
 - Human scan deforms according with its real-world counterpart

By applying these steps to all subsequent time steps the mesh is tracked over the whole video sequence. The Laplacian scheme reconstructs the mesh in its new pose in a way that preserves the differential surface properties of the original scan. Due to this implicit shape regularization, this improved tracking approach is robust against inaccurate flow estimates and deforms the mesh in accordance to its real-world counterpart in the video streams.



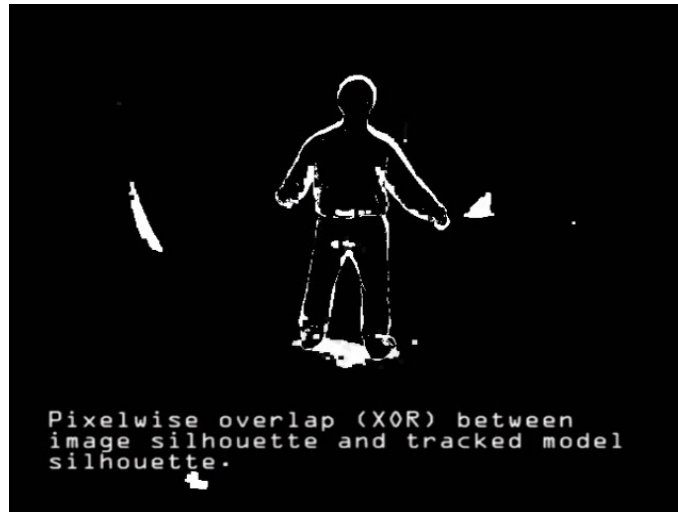
Results

- Synthetic sequences
 - Evaluate the performance
 - Accuracy
 - Efficiency
 - Reliable deformable mesh tracking
- Captured real-world sequences



The algorithm was tested on several synthetic and captured real-world data sets. The synthetic sequences allow us to compare the results against the ground truth and evaluate the performance of the algorithm in terms of efficiency and accuracy. Different deformation alternatives were also compared, namely deformation along the unfiltered flow (*RAWFL*), deformation according to the first tracking method (*ST-A*), and deformation with our complete pipeline (*ST-AB*). Using *RAWFL*, the measurement error grows almost exponentially. Tracking with the first scheme leads to significantly better results, but the absolute inaccuracy is still comparably high. In contrast, the complete pipeline leads to satisfactory results. These experiments confirm that the complete tracking pipeline in combination with a high-quality dense flow method can reliably track human motion from raw unmodified video streams, as seen for the captured real-world sequence results.

Results



For captured real-world results, the captured video sequences are between 300 and 600 frames long and show a variety of different clothing styles, including normal everyday apparel and a traditional Japanese kimono. Many different motions have been captured ranging from simple walking to gymnastic moves. The algorithm reliably recovers the pose and surface deformation for the subjects wearing comparably wide apparel, e.g. it can capture the motion and the cloth deformation for a woman wearing a kimono. The results show that this purely passive mesh-based tracking approach can automatically capture both pose and surface deformation of human actors. The combination of an a priori model, a fast Laplacian mesh deformation scheme, and a 3D flow-based correspondence estimation method enables us to capture complex shape deformations from only a few cameras.

Deformable Mesh Tracking for 3D Video



- Automatic marker-less system
 - Scan deforms as the subject
 - Robust method
 - 3D scene flow method
 - Laplacian scheme
 - Large range of motions and clothing styles
 - Preserve mesh connectivity
 - Flexible: human, animals
- Limitations
 - Reconstructing fast motion
 - Capturing low-frequency surface details (e.g. wrinkles)

The deformable mesh tracking algorithm is a new solution for automatic marker-less tracking of deformable human models from a handful of video streams. The combination of a 3D scene flow-based correspondence estimation approach with a Laplacian mesh deformation scheme enables this method to make a laser scan of a subject move and deform in the same way as its real-world counterpart in video. The algorithm is easy to implement, preserve the mesh's connectivity and can handle a large range of human motions and clothing styles.

Nonetheless, this algorithm is subject to a few limitations. Problems may arise if the subject in the scene moves very quickly. In these situations, optical flow tracking may fail. However, this can be solved by using a high-speed camera that is available today for capturing fast scenes. Also, the algorithm cannot capture the true shape variation of low-frequency surface details, such as wrinkles in clothing. While they globally deform with the model, they seem to be 'baked in' to the surface. Although in typical 3D video applications, this inaccuracy does not play a major role, the authors are planning to extend the method in the future to capture these small details by means of a multi-view stereo algorithm. Despite these limitations this method is a flexible, easy to implement and reliable purely passive method to capture the time-varying shape of subjects from video.



Discussion

- Improving 3D Video realism
 - Laser-scanned models
 - Time-varying deformations (skin/cloth)
- Presented methods
 - Hybrid approach: coarse template + scan
 - Capture time-varying deformations (skin/cloth)
 - Jointly capture motion and time-varying deformations
- Open Challenges
 - Multiple objects in the scene
 - Combination of model-based and non-model-based approaches

To conclude, we presented some algorithmic alternatives that can improve the quality and realism of dynamic geometry in model-based 3D Video applications. As seen in this part of the course, by using a more detailed shape representation of the subject and by properly capturing time-varying deformations (cloth or skin), the realism of model-based 3D video can be increased. We first presented a hybrid method that can be used to incorporate a laser-scanned shape prior into the original 3D Video system. Thereafter, we showed model-based methods that are able to capture different types of non-trivially deforming complex surface geometry, such as the motion of cloth or non-rigid skinning. Finally, we presented a model-based marker-less mesh tracking approach that enables faithful reconstruction of the motion and time-varying geometry of even people wearing complex apparel.

However, many challenges remain open, two of them are named on the slide above.



Free-viewpoint Video Relighting (25 min)

Christian Theobalt
Stanford University

Relightable Free-Viewpoint Video



- So far ...
 - Arbitrary viewpoint
 - **Fixed lighting conditions**



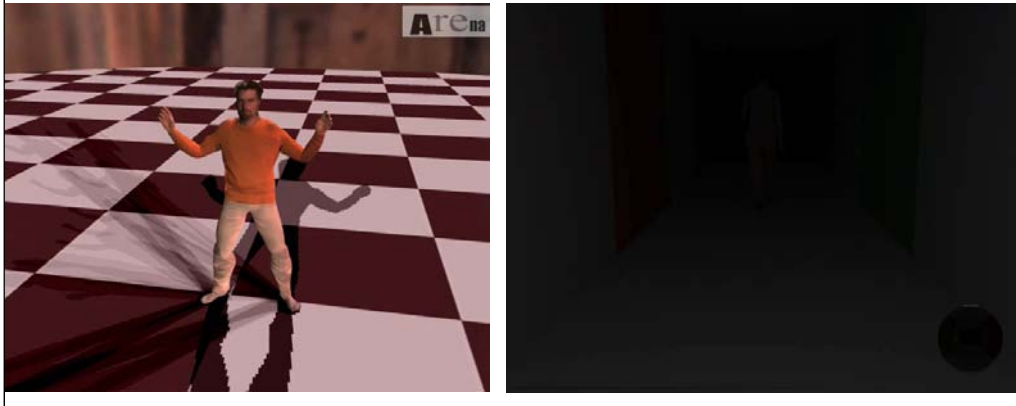
All approaches so far are able to reconstruct the dynamic and sometimes even view-dependent appearance of real-world scenes under the lighting conditions that prevailed at the time of recording.

Although this is one big step forward already, in many applications (e.g. 3D video compositing and postproduction, as well as many applications in games) one would like to be able to implant the captured 3D video footage into novel virtual scenes in which completely different lighting conditions exist.

Relightable Free-Viewpoint Video



- New Lighting Conditions
 - Dynamic Reflectance Capture



To achieve this goal, one has to reconstruct more than only the dynamic surface appearance from the captured footage, namely information on the actual dynamic surface reflectance. This means one needs to know how the scene's appearance varies under changing incident lighting and outgoing viewing directions.

Only recently has acquisition and computation hardware become powerful enough to enable researchers to attack this even more challenging reconstruction problem. Just few approaches attacking this problem have therefore been published so far, and we will review the most important ones in this part of the course. The videos on this slide show exemplary results obtained with methods whose working principles we will detail in the following.



Overview

- **Model-based Methods**
 - Surfel-based Dynamic Scene Capture
 - Model-based 3D Video Relighting
- **Data-driven Methods**
 - Image-based Dynamic Scene Relighting

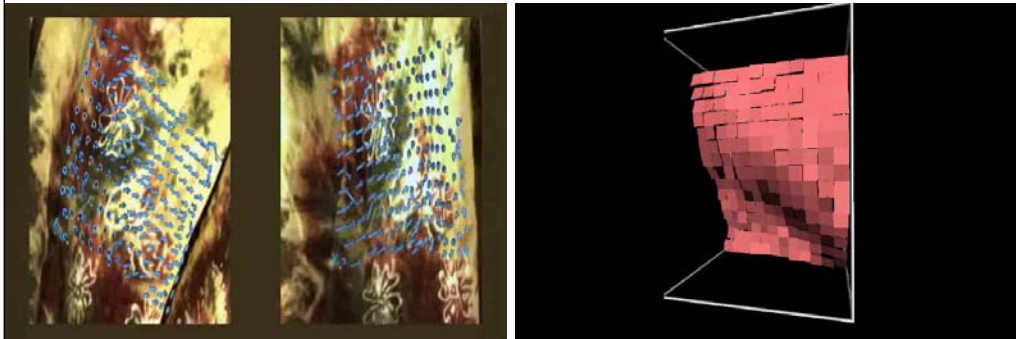
Two major categories of approaches have been proposed to attack the challenging problem of reconstructing dynamic relightable scene representations. The first category of approaches is an extension of the original model-based 3D video pipeline. This will be the subject of the first part of this section.

An alternative way of attacking the problem is to take a data-driven or image-based approach. Instead of explicitly reconstructing shape and appearance models, these algorithms densely sample the space of camera viewpoints and lighting conditions and reconstruct novel views by appropriately combining the recorded input image streams. Data-driven methods will be the subject of the second part of this section.

Surfel-based Non-rigid Dynamic Scene Capture



- Input: Multi-view video, calibrated camera + lighting
- Output: Moving surfel set approximating surface



Video footage with overlaid surfel motion

Reconstructed moving surfels

Courtesy of K. Kutulakos – taken from [Carceroni et al. ICCV 2001]

One of the first approaches that lays the path for model-based dynamic scene relighting is presented in [1]

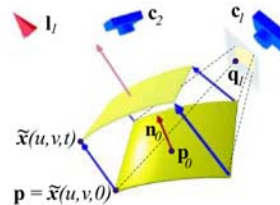
The goal of their algorithm is the reconstruction of non-rigidly deforming dynamic scene geometry from multi-view video footage that was recorded under calibrated lighting. Although the focus of this paper is not relighting itself, the proposed reconstruction algorithm also infers reflectance properties for discrete surface elements in order to recover the dynamic scene geometry as accurately as possible. Therefore, the ideas proposed in this paper can be regarded as a motivation for many of the concepts employed in the 3D video relighting approach discussed in the last part of this section.

This slide and the two following slides contain material that was kindly provided to the authors by Kyriakos Kutulakos from the University of Toronto. The videos shown on this slide can also be found here: <http://homepages.dcc.ufmg.br/~carceron/surfels/>

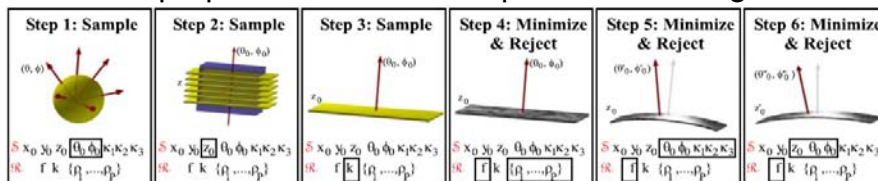
Surfel-based Non-rigid Motion Capture



- *Dynamic Surfel* has associated shape, motion and reflectance (Phong)



- Multi-Step optimization – Find optimal surfel alignment



Courtesy of K. Kutulakos – taken from [Carceroni et al. 2001]

The method tries to find the motion and orientation of so-called dynamic surfels relative to a discretized voxel grid in space. The mathematical description of each surfel features: A 3D shape component, more specifically position, normal orientation and curvature information, a reflectance component modeled by the Phong reflectance model [2], a description of the surfel’s motion in terms of an instantaneous 3D velocity vector.

The approach now explores the space of possible surfel orientations and motions to reconstruct for each time step a complete hole-free approximation to the moving surface. This exploration, i.e. the search for optimal surfel parameters, is performed in the specific manner illustrated on the bottom of the slide.

The details of the method are described in [1]. Although scene relighting is not the primary goal in their work, many of the proposed ideas motivated the development of the model-based dynamic scene relighting approach shown in the following.



Model-based 3D Video Relighting

- Extension of the model-based Free-viewpoint Video approach described earlier
- Dynamic Reflectance instead of Dynamic Surface Textures

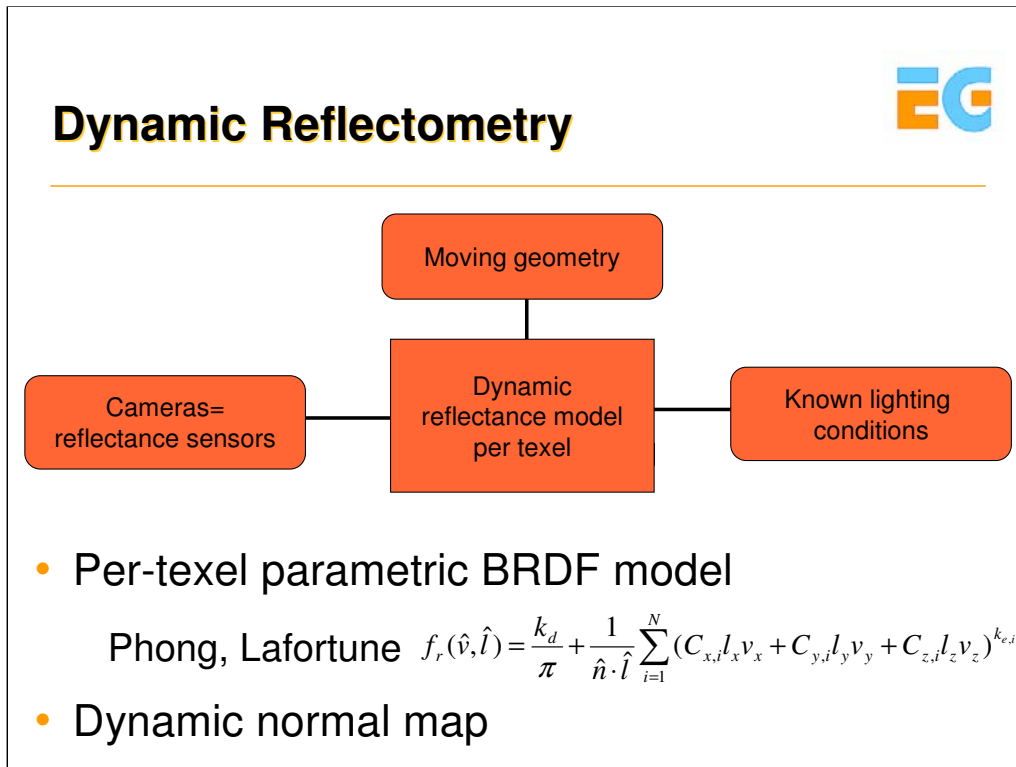


New: Single-skin model

[Theobalt et al. TVCG 2007]

Inspired by ideas from the paper by Carceroni et al. and recent progress in model-based reflectance reconstruction for static scenes we extended our original model-based 3D video pipeline in order to capture relightable scene representations [3].

Dynamic scene geometry is still reconstructed by capturing the motion of a kinematic template model. However, instead of using a segmented surface representation, we now use a single-skin model to represent the surface. This model is created from the shape-adapted segmented template model in a semi-automatic procedure.



From the marker-based motion capture step we know how the human shape model moves with respect to the recording cameras. If we now record the input sequences under calibrated lighting, the cameras are not only texture sensors but turn into reflectance sensors. As the person moves with respect to the acquisition setup, each point on the surface is seen under different incoming lighting and outgoing viewing directions. Therefore, each pixel value in combination with the information on light and viewing directions represents a reflectance sample. Many of such samples are acquired in subsequent video frames as the person moves.

Using a process called dynamic reflectometry we fit to each texel on the model's surface a time-varying reflectance model which comprises of two main components.

The first component is a static parametric representation of the bidirectional reflectance distribution function (BRDF). The BRDF at a surface point is defined as the quotient of the outgoing radiance in a particular direction to the irradiance incoming from a specific direction. It is usually represented as a 6D function of incoming and outgoing directions as well as location on the surface. By computing an integral over the hemisphere of incoming light directions, the BRDF can be used to compute the outgoing radiance for any novel viewing direction. Due to their compactness and modeling power, we employ in our work parametric BRDF models that enable us to represent surface reflectance in terms of a few tunable parameters only [2,4].

The second component of our model is a time-varying normal direction for each texel. By this means, dynamic changes in surface geometry, such as folds in the apparel, can also be represented and realistically relit.



Acquisition

- 8 video cameras + calibrated lighting



Input footage was recorded in the 3D Video recording studio at MPI. Eight 1-Megapixel video cameras running at 25 fps are arranged in a approximately circular setup around the center of the scene. All scenes are recorded under a calibrated lighting setup. We employ two light sources that are placed at opposite corners of the setup to illuminate the scene. The positions of cameras and light sources are calibrated. In addition to geometric calibration, we also perform photometric and color calibration prior to recording.

Acquisition

- Reflectance Estimation Sequence (RES)
 - one for each type of apparel
 - BRDF model



We record two types of multi-view video sequence to reconstruct relightable 3D videos. The first type of sequence, called reflectance estimation sequence (RES), is later used to estimate the per-textel BRDF parameters. We record one RES for each person and each type of apparel. In the RES, the person slowly rotates in front of the acquisition setup while trying to maintain a static upper body pose. By this means, we can capture many reflectance samples and prevent unwanted changes in local surface detail during acquisition. The pose of the body model in each frame is captured using our silhouette-based approach.



Acquisition

- Dynamic Scene Sequence (DSS)
 - Arbitrary motion
 - Motion sequence to be relit
 - Dynamic normal map



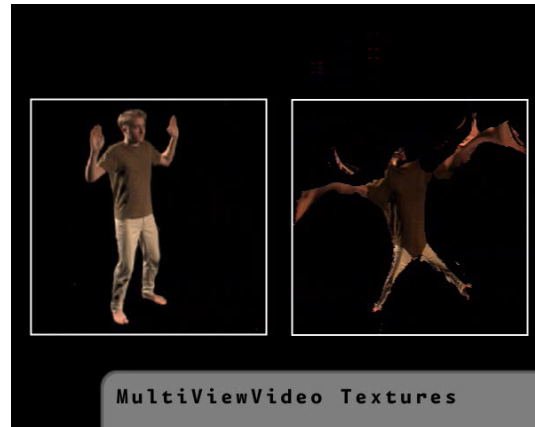
The second type of sequence is the so-called dynamic scene sequence (DSS). One DSS is captured for each performance of the actor that shall be reconstructed. Besides the actual motion, we also reconstruct the dynamic normal map from the DSS.

Multi-view Video Texture Generation



- Transform each input video frame into texture domain to obtain a

Multi-view video (MVV) texture



To facilitate reconstruction and rendering, we are transforming each captured video frame into the texture domain and are thereby creating so-called multi-view video textures. One multi-view video texture is created for each frame and each camera.



Image-based Warp-Correction

- Problem: approximate geometry causes texture registration errors
- One solution: deform geometry
- Our solution: warp correction of input video frames



[Ahmed et al. ICIP 2007]

Before we reconstruct reflectance information from the input data, we have to solve a couple of registration problems. The first problem that we are facing is the fact that, due to approximate body geometry, there might be artifacts when projecting the textures back onto the model.

One solution would be to deform the geometry until the body model's shape best matches the input footage. Instead of doing the adjustment on the geometry level, we do the adjustment in image space. The idea is to warp the input images such that they optimally correspond to the multi-view image material captured. The warping of input images is performed as part of the multi-view texture generation process. It is described on the following slides and in [5].

Image-based Warp Correction



**MVV texture generation
for camera 0**

**Trivial case:
Camera 0 sees
surface point best**



Warp correction happens during multi-view video texture assembly. In the following, the warp correction steps are illustrated using a single surface point of the model as an example.

Let's assume we would like to assemble a multi-view video texture for camera 0. In case it is camera 0 that sees a surface point best, which in this case means most head-on, we look up the appropriate color in the image captured by camera 0. This is the trivial case and no warp correction step is required.



Image-based Warp Correction

**MVV texture generation
for camera 0**

**Warping case:
Camera 1 sees
surface point best**

- Texture model from camera 0
- Reproject into camera 1
- Warp reprojected textured model to match camera 1
- Lookup in warped image



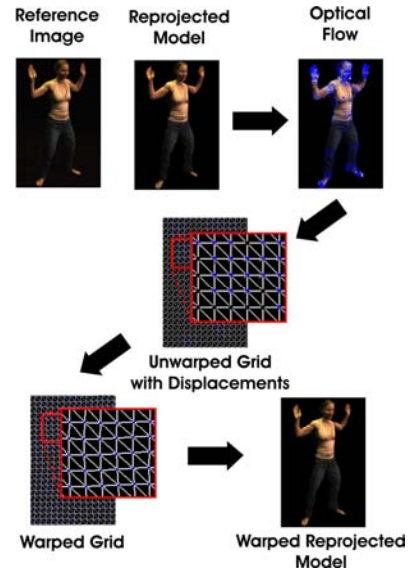
The warping case occurs if it is not camera 0 that sees a point best, but (without loss of generality) for instance camera 1 (red circle in image above).

In this case, the warping procedure is applied. To this end, the model is textured with the image from camera 0 and the so-textured model is projected back into the image of camera 1. The textured back-projected image is warped such that it optimally overlaps with the image actually captured from camera 1. The texel color is now looked up from the warped re-projected image. By this means, we make sure that the texture information always comes from camera 0 although we warp it into multiple camera views.

Warp-Computation



- Warp of a camera image into the reference view
- Regular mesh over image
- Dense optical flow field
- Warping – Mesh deformation
- GPU:
 - Textured rendition → Warped image



The image warping procedure itself, i.e. the procedure used to align the re-projected and captured images from camera 1, is based on optical flow computation and a subsequent image warping step. The working principle of the warping algorithm is illustrated in the flow diagram on this slide. To produce the final warped images, we make use of the texturing and filtering capabilities of the GPU and render image-aligned textured triangle meshes. Image-warping itself is implemented as a 2D smooth deformation of textured meshes.




Image-based Warp-Correction

- Improvements

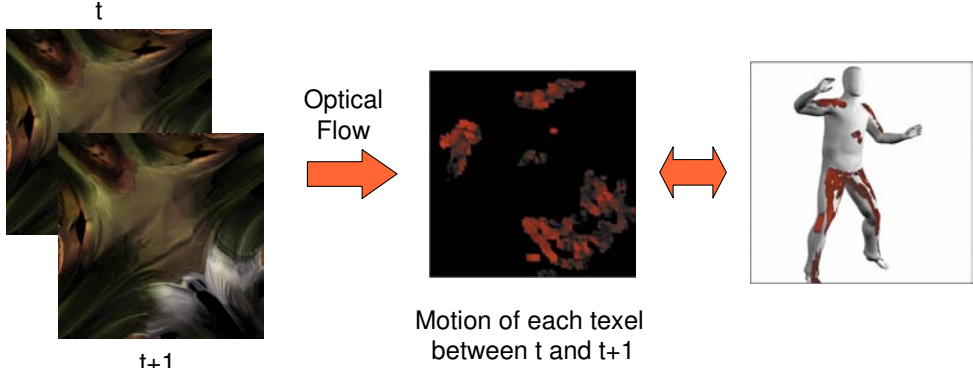


As one can see in the image above the “ghosting artifact” seen on the pants of the actor, which are due to shape misalignments, are corrected if our warping method is applied.



Cloth Shift Detection

- Detect cloth motion in texture domain
- Store time-varying texture coordinates



Motion of each texel between t and t+1

[Ahmed et al. ICIP 2007]

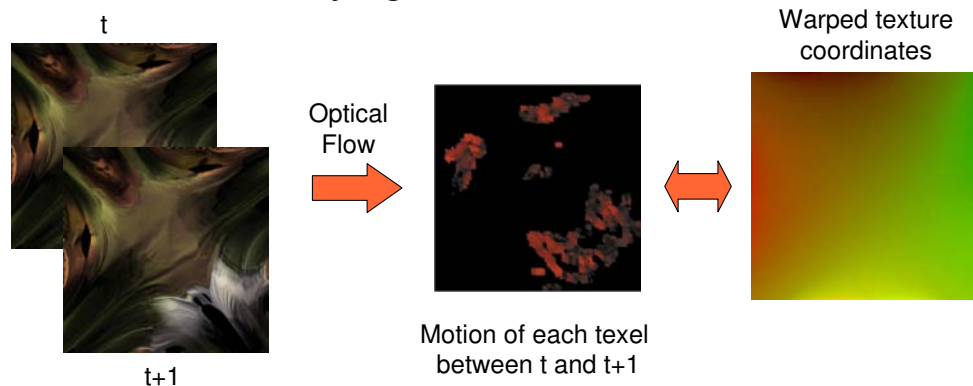
The second spatio-temporal registration problem we are confronted with is due to the fact that when the person is moving his apparel shifts across the body surface. This contradicts our assumption that we can assign a constant set of BRDF parameters to each location on the model's surface. In order to extract this cloth shifting information and in order to make it accessible to both our reflectance estimation framework and our renderer, we employ the cloth shift detection procedure illustrated in the slide above.

We identify the motion of apparel by means of a dense optical flow field between the complete surface texture images of the person in subsequent time steps. The complete surface texture is computed by weightedly blending all input images. An example of such a flow field is visualized in the image above. The 2D sections marked in red are areas in which motion has been detected. These areas correspond to specific surface areas on the model which are illustrated in the image to the right.



Cloth Shift Detection


- Detect cloth motion in texture domain
- Store time-varying texture coordinates




[Ahmed et al. ICIP 2007]

Flow fields are computed between all successive frame pairs. In the end, we make the information on cloth motion available to our renderer by warping the texture coordinates according to the recovered flow fields. The stream of warped texture coordinates is our final representation for cloth motion that enables us to do proper sample lookups during reconstruction and to render moving apparel despite a static assignment of material properties to the model surface. In both cases, i.e. reconstruction and rendering, texture information is looked up using the appropriately warped texture coordinates.


Cloth Shift Detection




- Improvement



Input



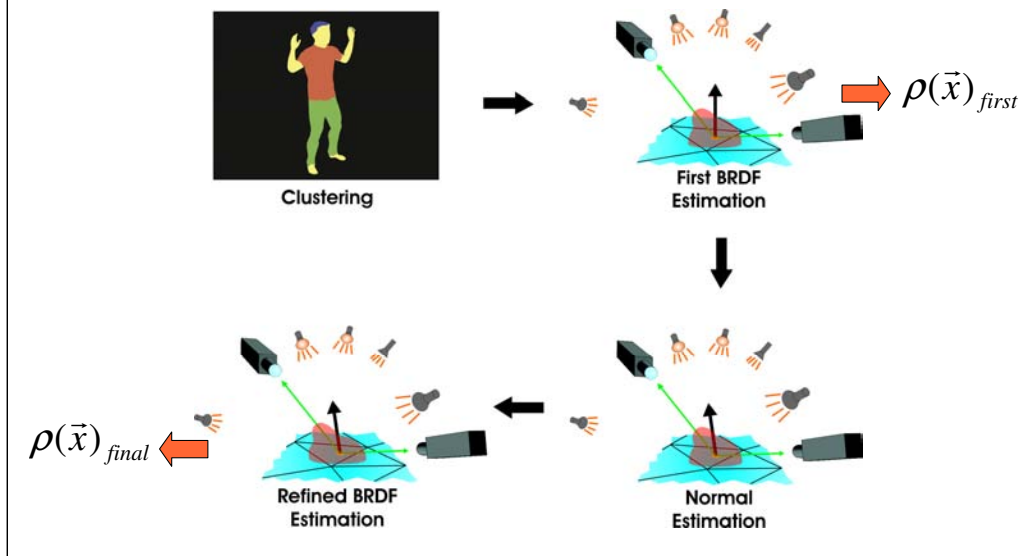
Rendering without
shift detection



Rendering with
shift detection

This set of images shows a close-up view of the waist area of a reconstructed relightable 3D video. The images illustrate the usefulness of our cloth motion detection method. The leftmost image is a ground truth image in which the boundary of the t-shirt shifted upwards in comparison to the first frame of the sequence. The image in the middle shows a reconstruction without cloth shift correction where the boundary of the shirt is improperly reproduced. The image to the right shows the correct assignment of the t-shirt color to the boundary location when cloth shift detection is applied during reconstruction and rendering.

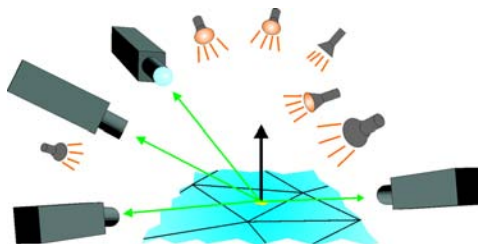
Dynamic Reflectometry



After all pre-processing steps are completed and the spatio-temporal reconstruction problems are solved, the actual reflectance estimation commences. In a process called dynamic reflectometry a separate set of BRDF parameters is estimated for each texel on the model's surface. The input to the dynamic reflectometry procedure is the reflectance estimation sequence.

The first step in the dynamic reflectometry process is the clustering of texels into groups of similar surface material according to the average diffuse color. After clustering, a first set of BRDF parameters is estimated via an energy minimization procedure. Given the first BRDF estimates, the default normal field of the template model is refined to better reproduce the true surface geometry in the RES. In a final estimation step, a new set of BRDF parameters is estimated using the now refined geometry description.

Dynamic Reflectometry - BRDF Estimation

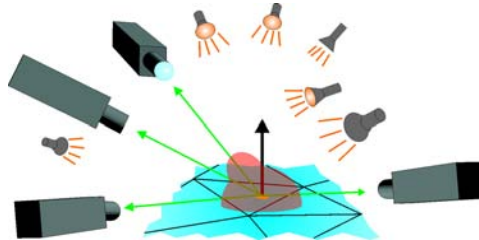


$$\text{Minimize } E_{BRDF}(\bar{x}, \rho(\bar{x})) = \sum_t^N \sum_c^8 (S_c(t) - P_c(t, \rho(\bar{x})))^2$$

BRDF
params
Reflectance
sample
Predicted
appearance

BRDF estimation is formulated as an energy minimization problem as it was proposed in a similar way in [6]. For each texel, the functional shown above is minimized in the BRDF parameters. The functional measures the quadratic error between the measured samples for that texel and the prediction according to the current BRDF parameters across all time steps of video and all camera views. Visibility from light sources and cameras is taken into account.

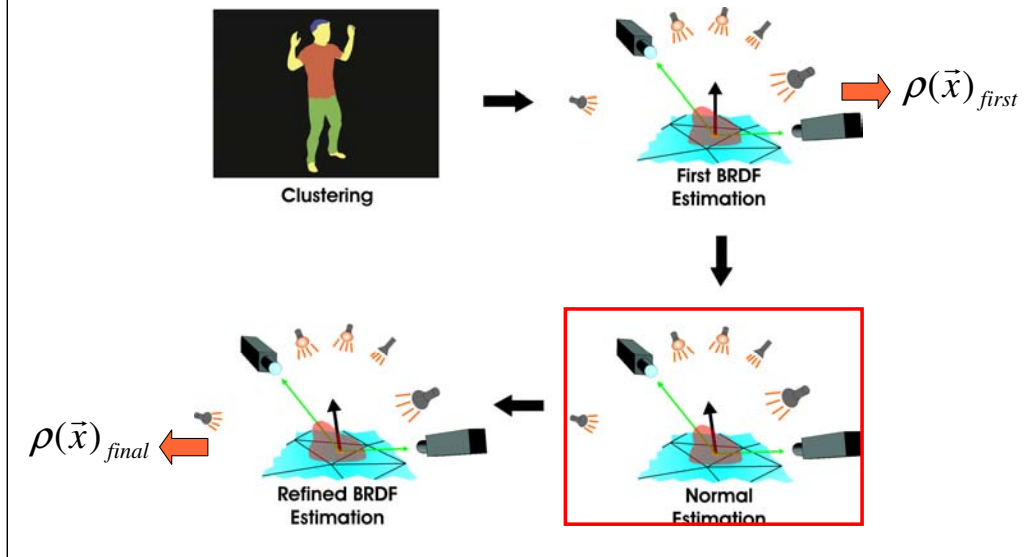
Dynamic Reflectometry - BRDF Estimation



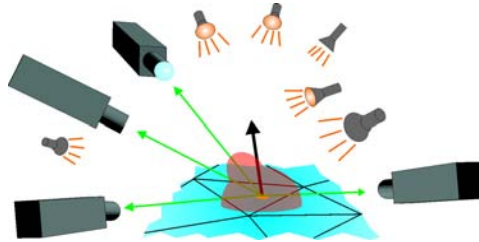
- In practice: Average BRDF per material
- Subtract specular component from samples
- Re-estimate diffuse \rightarrow per-textel diffuse

Our BRDF models [2,4] treat specular and diffuse reflectance as separate terms. Specular reflectance is a high frequency signal and one has to make sure that a sufficient number of samples is available in order to properly reconstruct it. To obtain a sufficient sampling density of the space of reflectance samples we therefore estimate an average specular BRDF for each material cluster, but an individual diffuse component for each texel. In practice, BRDF estimation itself therefore comprises of two sub-steps. The first sub-step is the estimation of an average BRDF for each material. Using this average BRDF, purely diffuse reflectance samples are created by subtracting the estimated average specular components. Finally, a separate diffuse component for each texel is estimated using the purely diffuse samples.

Dynamic Reflectometry



Dynamic Reflectometry – Normal Estimation



Minimize $E_{normal}(\vec{x}, \hat{n}) = \alpha E_{BRDF}(\vec{x}, \rho(\vec{x})_{first}) + \beta \Delta(\hat{n})^\gamma$

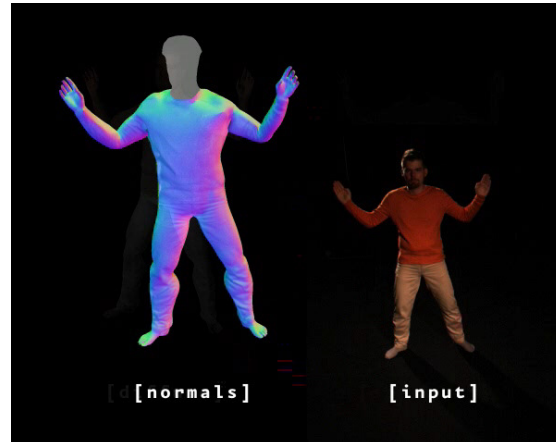
Normal direction
Penalize deviation from default normal

The normal estimation step capitalizes on the first set of estimated reflectance parameters to create a refined estimate of 3D shape. To this end, a new energy functional is employed which is shown in the slide above. The new functional is a weighted sum of original BRDF error functional and a second term which is used for regularization. The regularization term penalizes strong deviations of the estimated normal direction (the delta term) from the template normal. As before, the energy functional is minimized separately for each texel.

Dynamic Normal Maps



- Input: DSS
- Per-textel:
Minimize energy



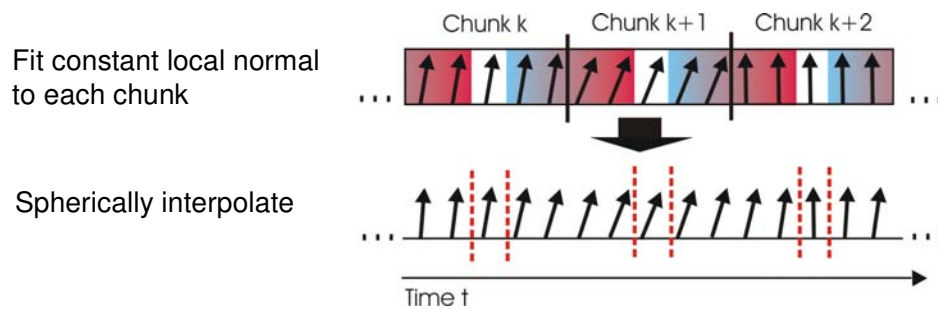
Once the BRDF parameters for each surface point are estimated and a refined estimate of surface geometry is available, we can reconstruct the time-varying normal field from the dynamic scene sequence in order to complete our reflectance description.

The procedure applied to recover the per-time step normal direction is again the energy minimization described on the previous slides. In contrast to the normal estimation which was part of the dynamic reflectometry process, however, we are now facing the problem of reconstructing a dynamic normal field.



Photometric Stereo

- Assumption: Local normal direction constant in short time interval



In order to reconstruct a temporally smooth normal field, we employ the following procedure. First, we assume that the local normal direction of a texel does not change within a short window in time. By this means, we can combine reflectance samples from several subsequent time steps to estimate a single normal direction. Once the normal directions have been estimated on this coarser scale, we employ spherical linear interpolation to obtain normals that smoothly change their orientation. Please refer to [3] for details on this reconstruction procedure.

Results



- Lighting
 - Spot lights
 - Environment map
- On Pentium 4 3 GHz, GeForce 6800
 - 16 fps (4 lights)
 - 6 fps (16 lights)



[Environment maps courtesy of Paul Debevec]

This and the following slides show several examples of relightable 3D videos rendered in real-time and lit from simulated and captured real-world illumination.



Results

- Estimation times
 - BRDF estimation
~2h
 - Input Warping
~10s/image pair
 - Cloth shift
~35s/time step





Discussion

- Model + calibrated lighting
→ Relighting with Sparse Set of Cameras
- High-frequency relighting
- Compact representation
(330 frames = 528 MB)
- Local illumination effects only
- Difficult for general scenes

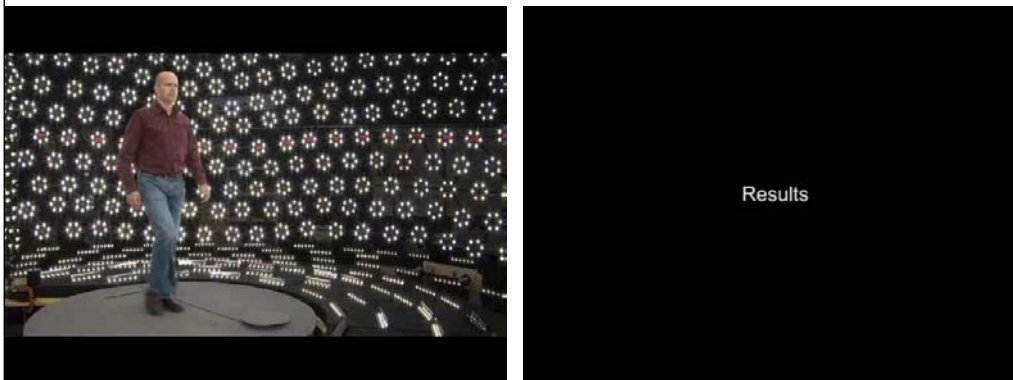
The model-based approach to dynamic scene relighting bears a couple of important advantages. First, 3D videos can be reconstructed using a moderately complex acquisition setup comprising of only a handful of cameras. Furthermore, the captured reflectance descriptions enable us to reproduce all-frequency lighting effects. An additional benefit is the fairly compact scene representation in terms of a dynamic geometry model, a static set of BRDF textures, and dynamic normal field and texture coordinate textures. The dancing sequence shown before that comprises of 330 frames of video has a total size of 528 MB. Our specific scene representation is very well suited for rendering on state-of-the-art graphics hardware.

Nonetheless, the model-based approaches also has several disadvantages. First, we can currently only reproduce local illumination effects. Secondly, for each type of subject in the scene a dedicated a priori model has to be available. Some of these limitations can be overcome by a data-driven approach to dynamic scene relighting.

Data-driven Dynamic Scene Relighting



- Capture 7D reflectance field (motion, image location, viewpoints, lighting conditions)
- Relighting + Interpolation → new viewing and lighting conditions



Courtesy of Paul Debevec (www.debevec.org) [Einarsson et al. EGSR 2006]

A data-driven approach to do dynamic scene relighting was developed in the lab of Paul Debevec at USC Los Angeles and is described in [7].

They built a device called Lightstage 6 that enables them to capture a 7-dimensional full dynamic reflectance field. This means they capture a complete set of images for dynamic scenes that spans 2 dimensions for the images themselves, 2 dimensions (hemispherical directions) for the incident lighting, 2 dimensions for the viewpoints (hemispherical directions) and 1 dimension for time.

When rendering a novel viewpoint of a particular captured moment under novel lighting conditions, the novel lighting conditions are projected into the employed lighting basis and the images in the 7D data set are appropriately combined to generate the output view.

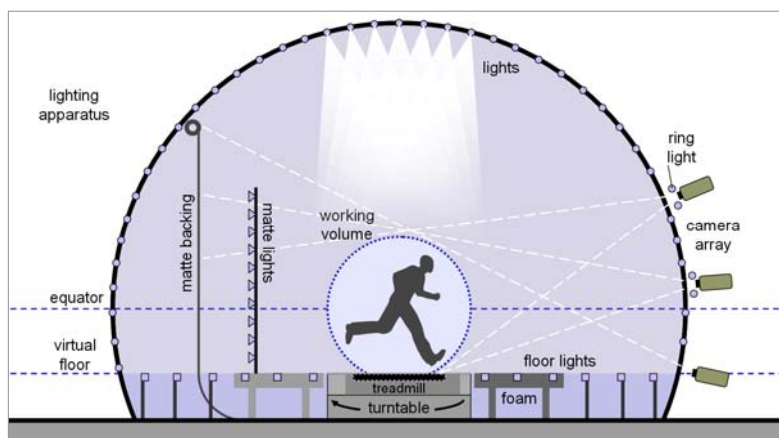
This and the following slides on this project were kindly provided by Paul Debevec.

Acquisition - Light Stage 6



2/3 of a 8m geodesic dome
931 dome lights
140 floor lights

2 meter wide working volume
3 high-speed cameras



The acquisition setup, called Light stage 6, features 931 LED dome lights that can be switched on and off very rapidly. The lights are all mounted in a 2/3 full geodesic dome, 140 floor light simulate light bouncing from Lambertian ground plane.

For video recording, three high-speed cameras are used that are running at roughly 1000 frames per second. The light sources are triggered in synchronization with the cameras. Due to the high frame rate of the cameras, each pose of the actor can be illuminated and recorded under a full set of 26 different basis lighting conditions. The frame rate of the cameras is enough to capture 30 complete lighting cycles per second.

As high-speed cameras are still very expensive, the authors restrict their setup and only use 3 cameras instead of a full dome of imaging sensors. They also use a treadmill which can be rotated in the setup and restrict captured scenes to cyclic human motion like walking. This way, the authors can simulate $N \times 3$ virtual recording cameras although only 3 cameras are physically available. However, this simplification also comes at the cost of restrictions in motion generality.



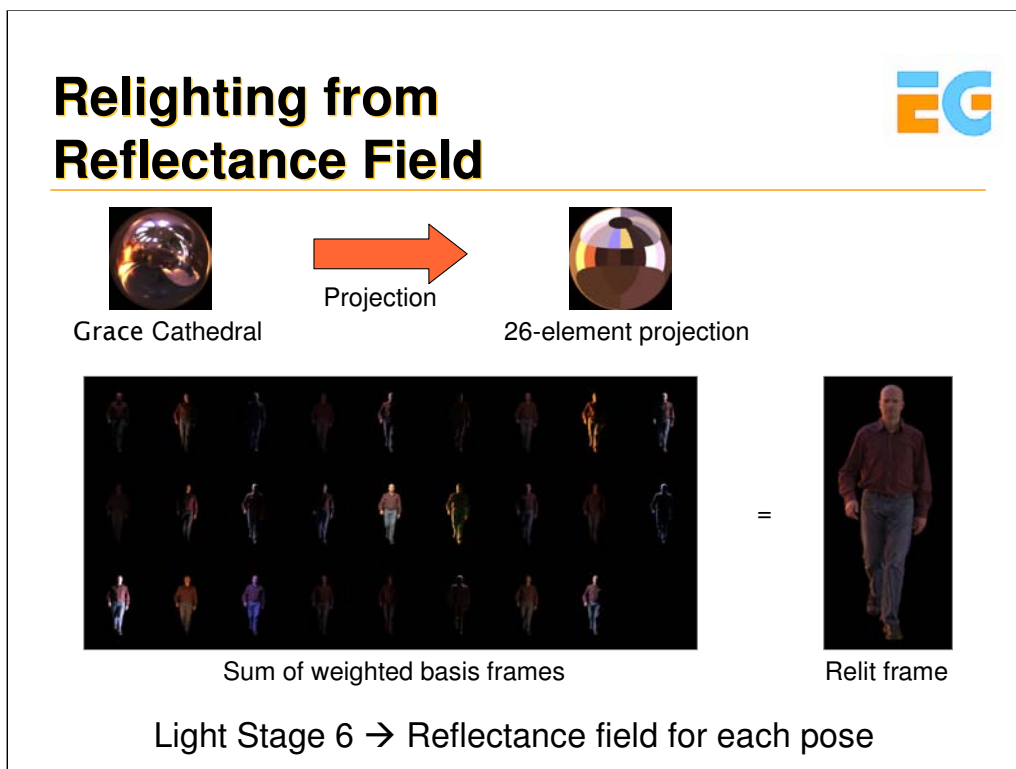
Acquisition

- 4D reflectance field for each pose and each “virtual” camera view
- 3x36 reflectance fields for each moment in a walk-cycle



For each pose of the walking cycle, i.e. each 1/30 of a second, a full 4D reflectance field is captured. A reflectance field is a function that transforms an incident light field hitting a surface to an outgoing light field exiting the surface [8,9]. Therefore, in its most general form, it is an 8-dimensional function (position + direction incoming; position + direction outgoing). If the light can be assumed to arrive from infinity and if the output viewpoint is fixed, as in this case, the dimensionality of the reflectance field reduces to 4. In this particular case, the reflectance field is simply parameterized as a set of images, each one being illuminated by one of the basis conditions.

In other words, the light stage enables capturing a 3x36 array of 4D light fields for each moment of a walking cycle.

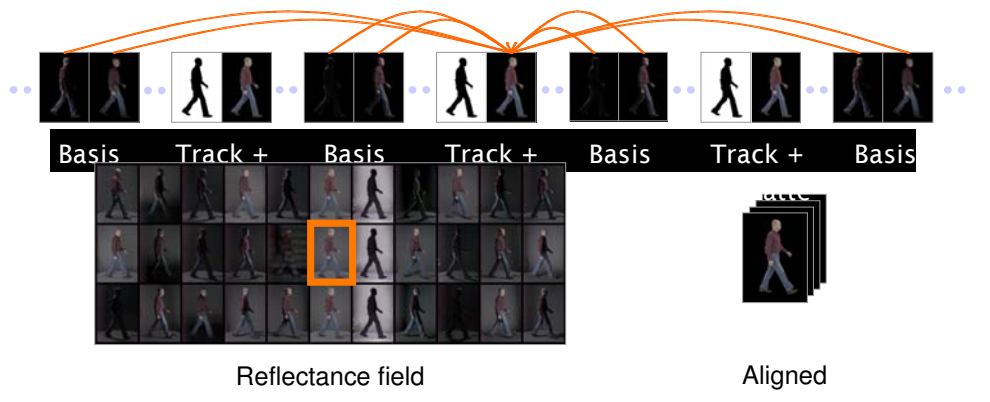


As mentioned before, a reflectance field transforms incident light fields to exitant light fields (see [9] for a definition of light field). Before the person is rendered under a novel illumination condition, the new illumination is projected into the lighting basis and the basis images in all 4D reflectance fields are accordingly scaled. A novel view of the actor is obtained by weightedly summing up the scaled basis images. By this means the flowed reflectance fields originally captured turn into a flowed light field, i.e. a set of relit images showing the person from each possible camera view and at each possible moment of the walking cycle. Please refer to [8] for details.



Registration

- Person moves and turntable rotates
- Spatio-temporal registration of images in 4D reflectance fields via optical flow warping

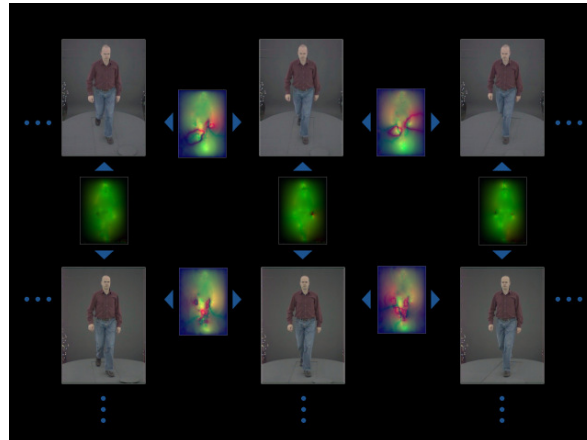


Since the turntable is rotating all the time, and the person is permanently moving a couple of registration problems have to be solved. For each complete lighting cycle (1/30s), a complete 4D reflectance field is acquired. However, as the person is permanently moving and the turntable is permanently rotating, the acquired images in the reflectance field are not properly aligned. Therefore, in a post-processing step all images captured during one lighting cycle (i.e. for each pose and each rotational increment) are aligned with the center frame by using an optical flow based warp correction. By this means, a so-called flowed reflectance field is obtained.

Flowed Reflectance Fields



- Bidirectional flow fields between 36x3 recorded viewpoints



During recording, new viewpoints are captured at 10 degree rotational increments of the turntable. Therefore, 36x3 virtual viewpoints of the scene are obtained. In order to be able to later generate arbitrary viewpoints in-between recording cameras, optical flow fields between adjacent 4D reflectance fields in the set of 108 viewpoints are generated. By this means a so-called flowed reflectance field is obtained.

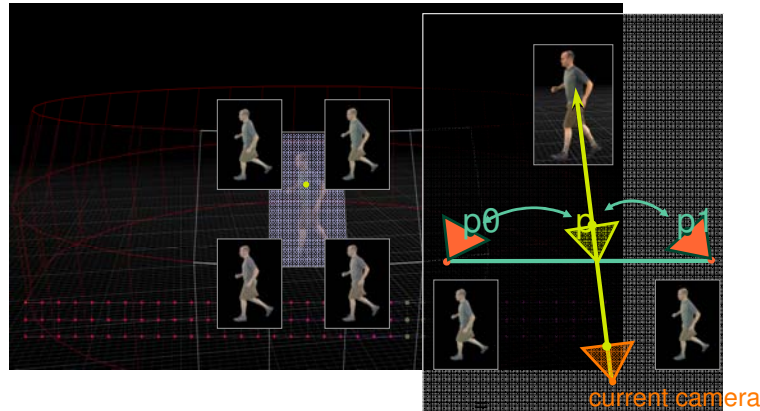
Rendering Flowed Light Field Interpolation



- Image-based relighting



- Flowed light field interpolation



Finally, a novel viewpoint of a person in-between a recording camera viewpoint and under novel lighting conditions is obtained by using a process termed flowed light field interpolation:

First, the set of captured data is relit according to the new synthetic lighting conditions using the image-based relighting approach presented before [8] which yields a flowed light field.

Second, to generate any arbitrary viewpoint in-between true recording cameras, a warping-based approach is used to combine pixel information from the closest nearby relit images in the flowed light field. The pixel information from the closest nearby true cameras is combined using a light field rendering process similar to [9, 10]. To this end, geometric relations between the virtual camera and the four surrounding recording cameras are used to determine how much influence the pixels from each camera get in the final output view. Proper pixel locations for blending are looked up in the true camera views by following the previously computed bidirectional flow fields in a backward direction.



Discussion

- Pros
 - High visual quality
 - Arbitrary global illumination effects
- Cons
 - Low-frequency relighting
 - Periodic motions
 - Huge amount of data (24 GB for a walk cycle, 320x448 pixels)

The data-driven relighting approach has a couple of advantages. First, the quality of the rendered and relit views is fairly high if the array of recording cameras is sufficiently dense. In addition to that, being based on the captured images themselves, the data-driven representation can inherently reproduce both local and global lighting effects under any novel incident illumination.

On the other hand, the proposed method also has a couple of disadvantages. First, only low frequency lighting effects can be reproduced as the employed lighting basis is very coarse. Furthermore, the huge amount of data even at low image resolutions, the high engineering overhead and the current limitation to very short cyclic motion may make the approach infeasible for many applications.



Discussion

- Pros
 - High visual quality
 - Arbitrary global illumination effects
- Cons
 - Low-frequency relighting
 - Periodic motions
 - Huge amount of data (24 GB for a walk cycle, 320x448 pixels)

References

- [1] R. L. Carceroni and K. Kutulakos. *Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape, and reflectance*. In Proc ICCV 2001.
- [2] B.-T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, pages 311-317, 1975.
- [3] C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, H.-P. Seidel, *Seeing People in Different Light: Joint Shape, Motion, and Reflectance Capture*, to appear in IEEE Transactions on Visualization and Computer Graphics, 13(4), p. 663-674, 2007.
- [4] Lafortune, E. P., Foo, S., Torrance, K. E., and Greenberg, D. P. 1997. *Non-linear approximation of reflectance functions*. In Proc. SIGGRAPH. ACM Press, 1997, p. 117-126.
- [5] N Ahmed, C. Theobalt, M. Magnor, H.-P. Seidel, *Spatio-Temporal Registration Techniques for Relightable 3D Video*, Proc. of IEEE ICIP, 2007, to appear.
- [6] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich and Hans-Peter Seidel. *Image-based Reconstruction of Spatial Appearance and Geometric Detail*. In ACM Transactions on Graphics, 22(2), 2003, pages 234-257.
- [7] Einarsson P., Chabert C., Jones A., Lamond B., Ma A., Hawkins T., Sylwan S., Debevec P. *Relighting human locomotion with flowed reflectance fields*. Proc. of EGSR 2006
- [] Wenger, A., Gardner, A., Tchou, C., Unger, J., Hawkins, T., and Debevec, P. 2005. Performance relighting and reflectance transformation with time-multiplexed illumination. In *ACM SIGGRAPH 2005 Papers*, p. 756-764 .
- [8] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, Mark Sagar. *Acquiring the Reflectance Field of a Human Face*, Proceedings of SIGGRAPH 2000, Computer Graphics Proceedings, Annual Conference Series, pp. 145-156 (July 2000). ACM Press
- [9] Levoy, M. and Hanrahan, P. Light field rendering. In Proc. of SIGGRAPH '96.
- [10] Buehler, C., Bosse, M., McMillan, L., Gortler, S., and Cohen, M. Unstructured lumigraph rendering. In Proc. SIGGRAPH 2001.



Applications (30 min)

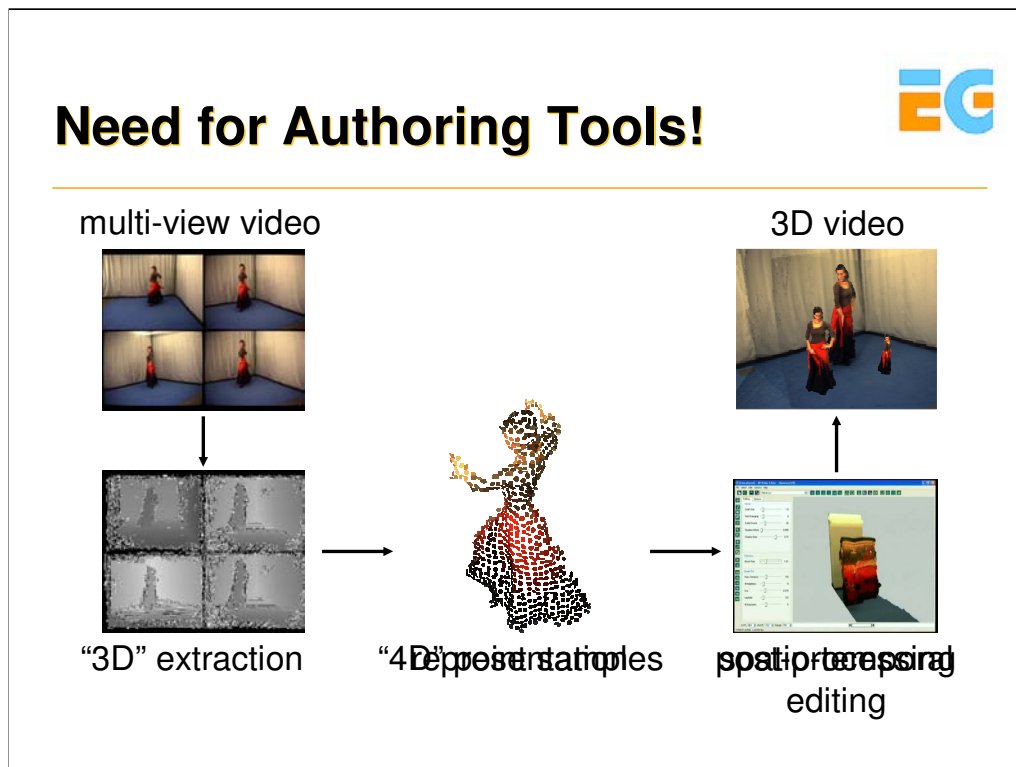
Christoph Niederberger

LiberoVision AG



Overview

- Authoring & editing 3D video
- Commercial applications
 - Early: The matrix
 - Digital Air's camera systems
 - LiberoVision: 3D video in sports broadcasts



If we have captured a scene from multiple viewpoints, authoring tools are necessary to edit this content. In this slide, an overview of the processing pipeline is shown.

Based on a multi-view video acquisition of a scene, a 4D representation is generated based on the extraction of the three-dimensional data. Usually, this relies on the extracted depth information.

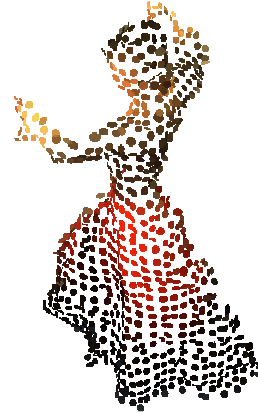
To edit and author such 3D video data, we need a new type of authoring tools. Such a tool must be able to extract objects or parts of a scene over a certain time and place it into another 3D video environment. Thus, such a tool can be envisioned as a cross-over between a video editor and a 3D modeling tool.

Finally, the edited scene can be previewed such that the final trajectory of the camera can be set up.



Point Cloud Representation

- Irregularly point-sampled object surfaces
 - Generalization of pixels
 - Unified storage of diverse attributes (normals, colors, ...)
 - Arbitrary complex topology
 - Adaptive resolution
 - Easy streaming
 - Multi-resolution & compression



To represent the scene geometry, we are not using triangle meshes but we suggest to use points where each point is a sample of a surface in the scene.

Such a point cloud can be generated quite easily from depth maps by back-projecting all depth pixels into a common 3D world coordinate system. As such, points can be considered as a generalization of 2D image pixels.

If the acquisition system does not produce depth maps but uses another approach to for example directly construct a mesh, this can be also converted quite easily into a point cloud by re-sampling.

Unlike meshes, points provide a unified storage container for all the data you need. Apart from geometry, they can carry information like surface normals, colors, or other material properties. You do not need an additional data structure like a texture to represent such attributes.

Points cannot represent the topology of a scene. This is a clear advantage in our case because this allows us for handling arbitrary complex topologies. More specifically, we can even represent dynamically changing topologies in the 3D video without much effort.

Unlike for example voxel grids, points only represent object surfaces and not their interior or the free air. As a consequence, we are not limited in scene resolution. But we can locally adapt the resolution to the scene complexity by irregular sampling.



4D Point Samples

- 4D Surfel [Pfister 00]: Gaussian ellipsoid

- Position $\mathbf{p} = (x, y, z, t)^T$

- Spanning vectors:

- $\mathbf{v}_1, \mathbf{v}_2$ (surface tangents)

- $\mathbf{v}_3 = \mathbf{0}$

- $\mathbf{v}_4 = (0, 0, 0, \Delta t)^T$

→ Covariance matrix

$$\mathbf{V} = \Sigma \cdot \Sigma^T, \quad \Sigma = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \mathbf{v}_3 \quad \mathbf{v}_4]$$

We extend those Surfels to our 4D setting by adding a temporal coordinate t .

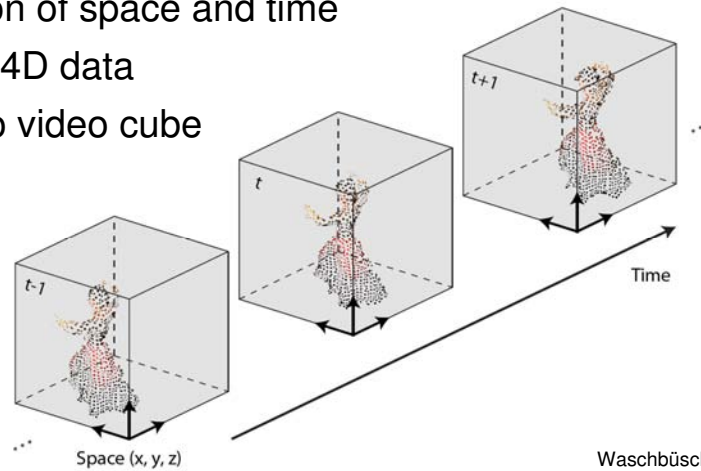
For the computation of the covariance matrix we introduce a fourth vector parallel to the time axis. Its length corresponds to the distance of two successive frames. This gives us smooth renderings if we want to visualize the temporal domain of the video hypervolume.

Alternatively, the covariance matrix can provide a probabilistic model of the geometric uncertainty, as has been introduced in our previous paper. This allows to model inaccuracies in the scanning due to noise or calibration errors.



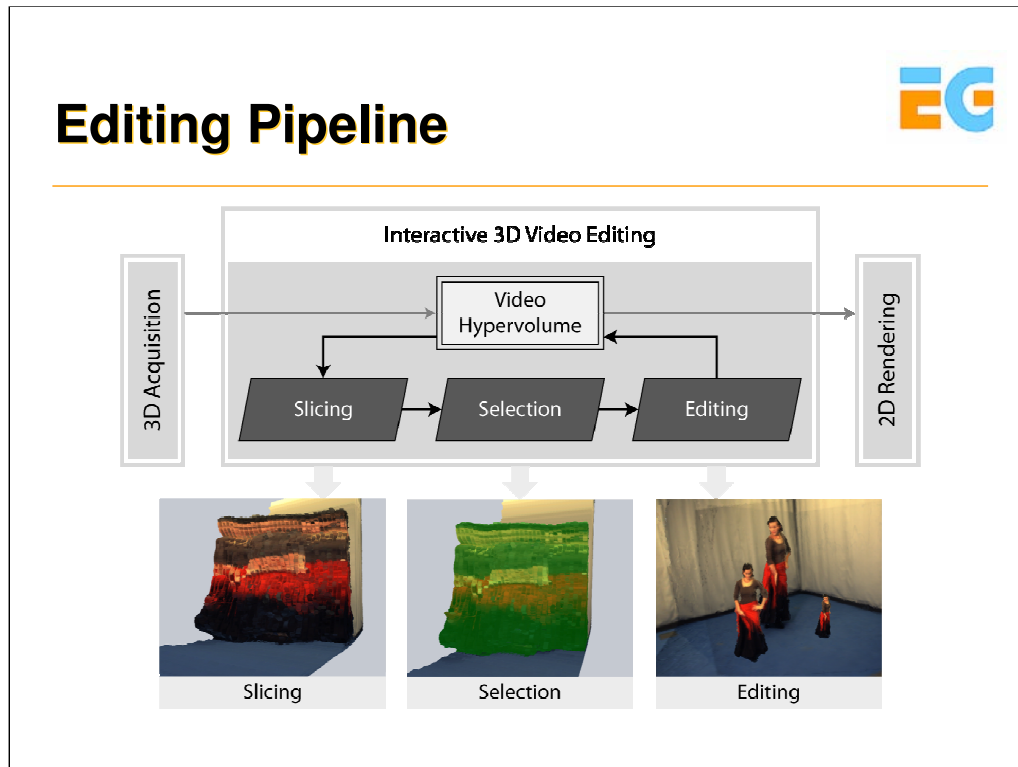
The Video Hypervolume

- Unification of space and time
- Irregular 4D data
- Similar to video cube



Waschbüsch et al.
Interactive 3D video editing
Pacific Graphics 2006

The 3D video hypervolume is a representation unifying point clouds over time. For each moment in time $t-1$, t , $t+1$, ..., we have a complete 3D space irregularly filled with point samples representing the objects in the scene. Thus, the hypervolume becomes a four-dimensional representation of the scene.



When interactively editing a video hypervolume, three different operators help the user to edit a scene with many degrees of freedom.

First, slicing the hypervolume results in a 3D volume not necessarily representing a 3D space but rather a 2D plane over time.

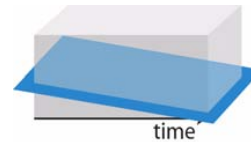
Then, the user can select a subvolume of the sliced part of the hypervolume. This intuitive approach is known from many editing programs for 2D images or 3D models.

Finally, the editing operators allow the user to cut and paste, resize or reposition a previously selected part of the scene.



Slicing

- Visualize the hypervolume
- Navigate in space-time
- Intersect hypervolume with hyperplane



Via slicing, the user can define a part of the four-dimensional hypervolume which should be visualized on the screen.

The slicing process basically intersects the hypervolume with a hyperplane, reducing the dimensionality by one from 4D to 3D.

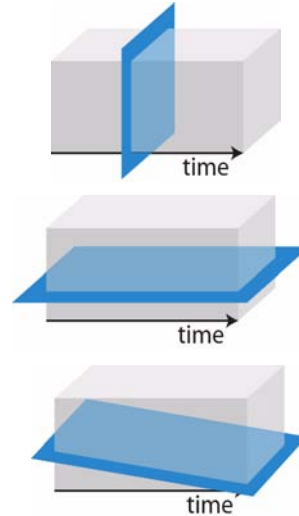
The resulting 3D point cloud is displayed on the screen and can be viewed from all sides using a trackball navigation interface.

By freely choosing the slice orientation, the user can visualize both spatial and temporal aspects of the 3D video.

Different Types of Slices



- Display conventional frame
- Visualize time domain
- General slice



The simplest slice is oriented orthogonal to the time axis. This corresponds to viewing a conventional frame of the 3D video.

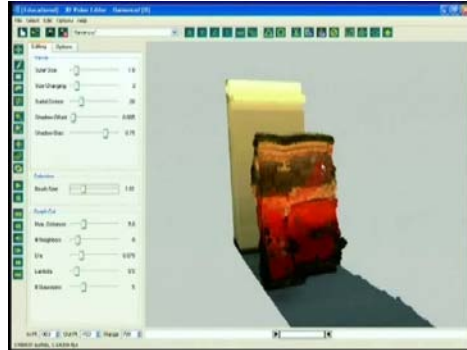
By orienting the slice parallel to the time axis, the user can define views which visualize the temporal domain of the video. This can be used for example to select a moving object over multiple frames.

In general, the user can define arbitrary slice orientations which can be used for example to follow object trajectories.



Selection

- Select points in current slice
- Selection tools
 - Bounding box
 - 3D paintbrush
 - Graph cut segmentation



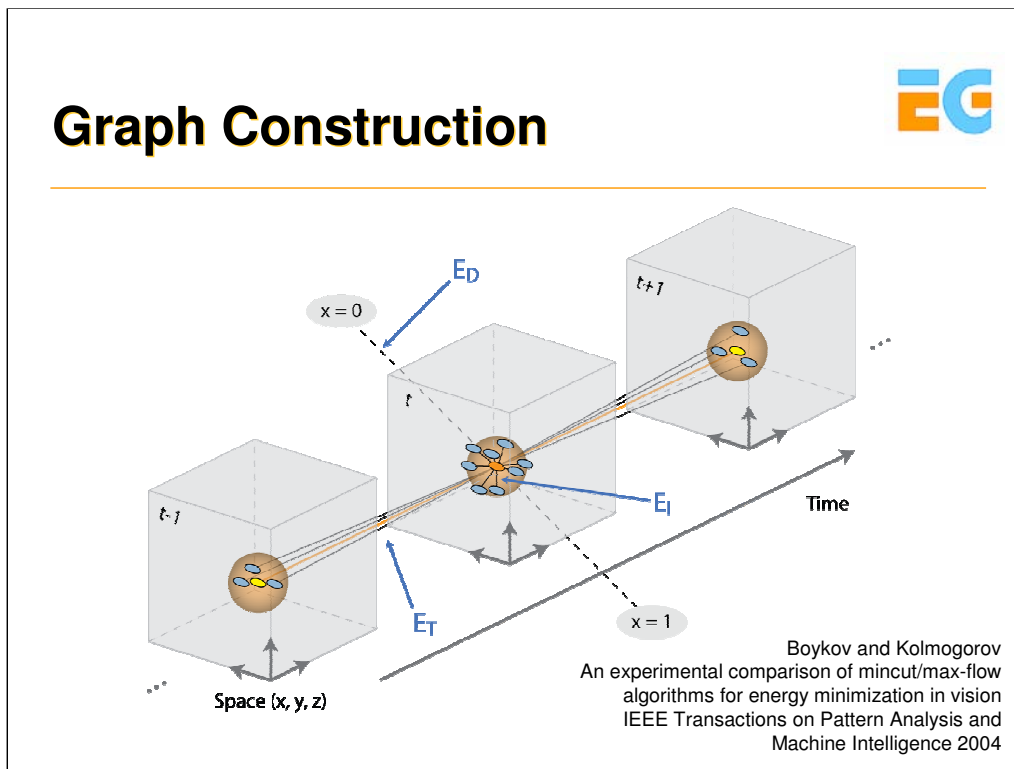
graph cut selection

In order to select parts of the current slice, the user can use different simple approaches. Each of these tools results in a partition of the points in the current slice.

For example, a simple bounding box selects all points inside the volume.

With the 3D paintbrush, the user can select points by simply drawing onto the current 2D projection of the slice. Then, the paintbrush follows the surface of the scene.

Finally, a graph cut segmentation can be used to simply create a partition of the scene.



The underlying graph not only connects the points in each frame but also over time. This yields to smooth segmentation/selection results both spatially as well as temporally.

$E(d)$ stores the so-called data energy which gives an indicator of how similar a point's information is to either the foreground or background ($x=0$ and $x=1$).

$E(i)$ and $E(t)$ are the so-called link energy (as in the cited literature).

$E(i)$ are the intra-frame connections and calculate the likelihood that two spatially close points belong to the same cluster (foreground or background). A k -nearest neighbor search yields the closest points.

$E(t)$ are the inter-frame connections and calculate the likelihood that two spatially close points in different frames belong to the same cluster (foreground or background). A k -nearest neighbor search in the frames $i-1$ and $i+1$ starting from the reference point of frame i yields the closest points.

Besides color we can also exploit geometry information such as the normal similarity of two 3D points for 3D video editing.



Editing Operators

- Cut, copy & paste
 - Object removal
 - Insertion into other scenes
 - Actor cloning

- Translate, rotate, scale
 - Translation in temporal domain → time shift



For editing, our system supports basic operations like copy & paste, translation, or scaling and rotation.

Although those operations are quite simple, they become very powerful in 3D video.

For example, with cut & paste together with our selection tools, you can generate complex effects like object removal or insertion of people into other scenes. After object removal, there are no remaining holes in the background, because we have the complete information of the scene available. Note how difficult such tasks would be with conventional 2D video.

The translation operator benefits from the generality of our hypervolume representation: a time shift operator is identical to the translation operator in the temporal domain.

More Editing Tools



- Shadow mapping
- Insertion of 3D meshes
- Insertion of 2D images and videos



shadow mapping

We implemented some further tools to insert 2D images or videos, which are just converted into point samples, and to generate artificial shadows.

The latter is important if novel objects get inserted into the scene because shadows provide important hints to the scene geometry.



3D Video Editing is Fun!



As a final result, let me show you a demo video which we generated with our editor...

You can clearly observe that we still have artifacts at object boundaries. This is a common weakness of the depth from stereo reconstruction algorithm which has difficulties to cope with depth discontinuities. We are continuously working on improving this issue. Let me mention that those artifacts are not an issue of an inaccurate graph cut segmentation, because you can observe them at the whole silhouette both in the source and the edited material. Graph cut has been only applied where the actor was connected to the floor.

“Bullet-time Photography”



- Application of “Campanile” research project
Debevec et al., Modeling and Rendering Architecture from Photographs, SIGGRAPH 1996
 - photo-grammetric modeling
 - projective texture-mapping
- Array of still-image cameras
- Keyframe interpolation of still images to create continuous motion

A 3D video application called “Bullet-time Photography” is well known under a different term: The “Matrix effect”.

The underlying application is a result of Paul Debevec’s 1996 SIGGRAPH Paper.

An array of still image cameras simultaneously takes pictures of a scene from different viewpoints. The location and orientation of each of the cameras has to be determined before the actual shot since the images can’t be changed afterwards.

Using a keyframe interpolation of all these images, the editor can finally create a continuous motion around an object while it is frozen.

Bullet-time Photography in “The Matrix”



The Matrix
© Warner Bros. Pictures 1999

Here, we see how the matrix special effects were created.

The top row shows how many cameras were necessary and how precisely aligned the array of cameras must have been.

The bottom row shows two intermediate results

Scene form “The Matrix”



The Matrix
© Warner Bros. Pictures 1999

And this is the final scene. The captured person is placed into a synthetic environment.

“Bullet-time Photography” Revisited



Movia® camera systems, www.movia.com
Digital Air (Dayton Taylor), www.digitalair.com
Geneva, Switzerland

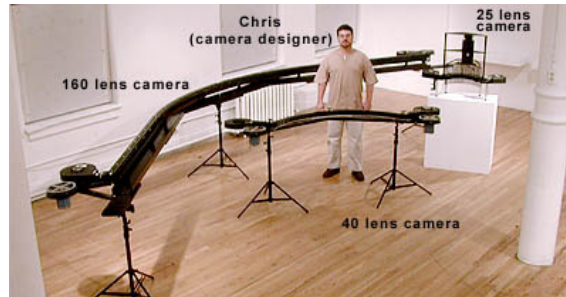
The bullet-time approach of the matrix has been adapted to video cameras by Digital Air in Switzerland. Using an array of many well-aligned video cameras, it is not only possible to fly around a frozen object but also to change the speed and direction of time.

These special effects are used in cinematography, commercial advertisements or trailers as eye-catchers.

Digital Air Camera Systems



- Movia®: digital camera array imaging
 - Digital cameras, original “Bullet-time” equipment
- Timetrack®: virtual camera movements
 - Film camera systems, 25-160 lens cameras



Digital Air offers two different products.

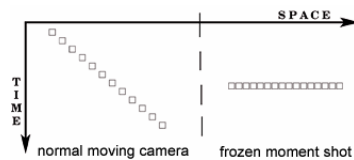
Movia relies on normal still image cameras aligned in an array as we have seen before.

Timetrack replaces the still image cameras with film camera systems where each camera records the scene through 25-160 lenses aligned in an array.

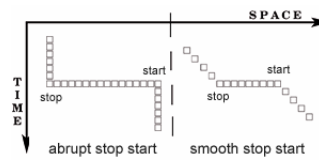


Effects (www.timetrack.com)

- Frozen Moment



- Stop-Start



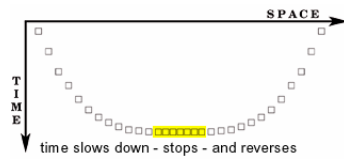
As mentioned before, using the timetrack system allows to change the timely behavior of the result, too. The diagrams below the movies depict the editing of the input imagery.

On the left, we see the classic approach where the camera moves in space and over time and the frozen moment where the camera moves in space in a frozen frame.

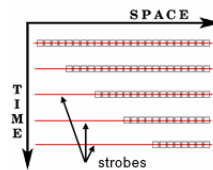
On the right, two combinations of these approaches are shown. The camera stands still and shows a moving scene, then freezes abruptly or smoothly and rotates around the scene before finally standing still again and showing the remaining sequence from a different location.

Effects (2)

- Time Ramp



- Multiple Exposure



Another possible special effect is the „time ramp“ where a continuous motion of the camera shows a scene decelerating, stopping, and reverse accelerating again.

The „multiple exposure“ effect overlays multiple shots of the same scene and creates a stronger feeling of the motion.



A scene of a soccer game with an really tight offside situation. None of the existing cameras can resolve the situation

3D Sports Visualization

Orad *VirtualLive*PVI *EyeVision*Red Bee Media *Piero*LifelnMedia *LifeInSports*Swiss Timing *tvVAT[3D]*BBC R&D *iview*

Different approaches have been developed to visualize sports scenes from novel views not covered by a camera.

VirtualLive and LifeInSports are based on a 3D model which is synthetically rendered, thus, allowing a total freedom of possible viewpoints but a non-realistic looking result.

EyeVision captures a scene with over 30 cameras positioned around the pitch. These cameras are controlled by an operator and are always focused on one spot on the pitch. The final result is a blending through the camera views resulting in a fly-around of the scene.

Piero and iview generate a model of the players based on the original camera feeds. These representations are then placed into a synthetic environment offering more freedom in terms of possible viewpoints.



LiberoVision is one step beyond the previously seen examples by offering a full freedom of viewpoints resulting in a realistic virtual view of the scene.

The video shows the same scene as before. In the moment of the offside decision, the playback stops and the virtual camera moves onto the offside line and shows the view of the referee on the side-line.

LiberoVision Inc.



- Spinoff of ETH Zurich, technology based on the 3D video technology developed at ETH Zurich
- Provides virtual replay functionality to sports broadcasts
- Requires isolated camera feeds
 - 8 or more cameras in the stadium
 - Half of them are on a „high“ position providing an overview
 - Synchronized
 - Arbitrary position, orientation and zoom

The technology of LiberoVision is based on the developments at ETH Zurich, Switzerland.

It requires the isolated feeds of the cameras positioned in the stadium and does not require any additional infrastructure inside the stadium. Usually, more than 8 cameras are used to produce a sports event such as a soccer game, approximately half of them are on a high position and provide an overview of the game. Synchronization of the cameras is a requirement of the broadcast production and makes it easier to recreate a scene. However, the cameras are not fixed but can change their position, orientation and zoom.

Based on existing camera feeds only



Penalty box camera
(left)

LiberoVision
(virtual camera)

Lead camera
(middle line)

On the left, we see a shot of the penalty box camera and on the right, a shot from the lead camera in the middle of the pitch. Based only on both these images, we can create the virtual image inbetween showing the scene from a viewpoint not covered by the physical cameras.



Another example of the LiberoVision technology. A goal situation is shown and the virtual camera provides a birds-eye view to tactically analyze the scene.



Conclusion

- Applications:
 - Three-dimensional special effects for movie industry and commercial advertisement industry
 - Analysis tools for sports broadcast industry
- High-quality vs. processing time

We have shown two different applications of 3D video.

First, the three-dimensional special effects for the movie industry such as the “Matrix effect” and derivatives of the same technology which is used in the commercial advertisement industry as eye-catchers.

A second application are tools for the analysis of sports scenes where virtual views are generated providing perspectives not captured by the available cameras.

Both these industries have different requirements regarding the results. While the movie industry can afford a rather long processing time, the result must be of a really high quality. In sports broadcast on the other hand, the result must be available within minutes or even seconds.



Outlook and Discussion (10 min)

Stephan Würmlin

ETH Zürich and
Liberovision AG



Outlook

- Major challenges
 - Everyday acquisition systems
 - Fully automatic
 - Mobile
 - Real-time
 - Production-quality 3D video
 - Applications in versatile environments (e.g. movies)
 - High-quality geometry extraction

Acquisition and processing of 3D video is still very time consuming, cumbersome and needs years of training and experience. However, for reaching the masses, acquisition needs to be as simple as possible, meaning fully automatic, mobile installations (such as on a mobile equipped with a camera and a GPS) and should at least acquire and process the data in real-time.

Despite the just mentioned commercial applications of 3D video, production-quality 3D video can only be achieved by either focusing on one application (such as soccer or sports in general) or by using special equipment, such as highly sophisticated camera systems and the like. The toughest problem is the extraction of high-quality geometry information out of the images alone. This is probably the biggest challenge of it all!



Outlook (2)

- Major challenges
 - Authoring & Delivery
 - Editing tools as simple as iMovie
 - Coding and Compression (MPEG?)
 - Displays & interaction
 - Novel 3D displays
 - Novel interaction metaphors

Major challenges also include authoring and delivery of 3D video data.

1) On the application side, 3D video gives us novel possibilities for creating special effects but also requires novel editing operations. It is desirable to have an editing software being as user-friendly as today's 2d video editing programs, such as iMovie.

2) To bring 3d video to the masses there is also the interesting question of spatio-temporal coding of the data stream. Those issues are currently investigated by the MPEG committee.

Current display technology does not fully exploit the interactive and three-dimensional nature of 3D video.

1) Hence, novel displays need to be developed such as autostereoscopic TVs or mobile projectors

2) Furthermore, interaction with the mouse is cumbersome in most daily environments, novel user interfaces such overcome this limitation. Microsoft Surface is one possible way of solving these problems.

Questions?





Presenters

- Christian Theobalt
 - theobalt@cs.stanford.edu
- Stephan Würmlin
 - wuermlin@liberovision.com
- Edilson de Aguiar
 - edeaguia@mpi-inf.mpg.de
- Christoph Niederberger
 - niederberger@liberovision.com



Acknowledgements

- EU 3DTV Network of Excellence
- Max-Planck-Center for Visual Computing
- Kyros Kutulakos, Univ. of Toronto
- Paul Debevec, University of Southern California
- Naveed Ahmed, Gernot Ziegler, Art Tevs, Carsten Stoll, Hendrik Lensch, Marcus Magnor, Hans-Peter Seidel, Joel Carranza



Acknowledgements

- E. Lamboray, M. Waschbüsch, D. Cotting, F. Sadlo, M. Gross
- M. Zwicker, C. Lee, W. Matusik, H. Pfister
- T. Weyrich, N. Kern, P. Kaufmann, S. Böhler, R. Küng, A. Smolic, D. Y. Kwon, S. Rondinelli
- C. Niederberger, R. Keiser, M. Germann, M. Ferencik
- ETH research grant No. 0-21020-04 (blue-c-II poly-project)

Capturing Reflectance - From Theory to Practice

Eurographics 2007 Tutorial T6

Organizers & Speakers

Hendrik P. A. Lensch (MPI Informatik)
Michael Goesele (University of Washington)
Gero Mller (University of Bonn)

EG:366

Abstract

One important problem in photorealistic or predictive rendering nowadays is to realistically model the light interaction with objects. Measurements can capture the reflection properties of real world surface, i.e., they are one way of obtaining realistic reflection properties.

For arbitrary (non-fluorescent, non-phosphorescent) materials, the reflection properties can be described by the 8D reflectance field of the surface, also called BSSRDF. Since densely sampling an 8D function is currently not practical various acquisition methods have been proposed which reduce the number of dimensions by restricting the viewing or relighting capabilities of the captured data sets. In this tutorial we will mainly focus on three different approaches, the first allowing to reconstruct opaque surfaces from a very small set of input images, the second allows for arbitrary surfaces but under the assumption of distant light sources and the last which allows for relighting an arbitrary scene with arbitrary spatially varying light patterns.

After a short introduction explaining some fundamental concepts regarding measuring and representing reflection properties, the basics of data acquisition with photographs will be addressed. The tutorial present the set of current state-of-the art algorithms for acquiring and modeling 3D objects. The tutorial investigates the strengths and limitations of each technique and sorts them by their complexity with regard to acquisition costs. Besides describing the theoretical contributions we will furthermore point out the practical issues when acquiring reflectance fields in order to help interested users to build and implement their own acquisition setup.

Syllabus

- 8:30 **Introduction** (Lensch)
material properties
classification of techniques
- 8:45 **Acquisition Basics** (Goesele)
light sources
cameras
HDR
- 9:15 **Reflectance Sharing** (Goesele)
image-based BRDF measurement
spatially varying BRDFs
- 9:45 **BREAK**
- 10:00 **Reflectance Fields for Distant Lights** (Müller)
BTFs
light stage
acquisition, compression, synthesis and rendering
- 10:40 **Near-field Reflectance Fields** (Lensch)
relighting with 4D reflectance fields
dual photography
- 11:15 **Conclusion, Q/A** (all)

Resume of the Presenters

Michael Goesele is a postdoctoral research associate in the computer graphics and vision group at the University of Washington. In 1999, he joined the computer graphics group at the MPI Informatik and received his PhD from Saarland University in 2004. His research is focused on a broad range of acquisition techniques for computer graphics. Among others, he recently published two papers at ACM SIGGRAPH about the acquisition of light sources (Accurate Light Source Acquisition and Presentation) and translucent objects (DISCO – Acquisition of Translucent Objects). He has given several lectures and tutorials (e.g. at Eurographics 2002 and SIGGRAPH 2005) about the topics covered in the tutorial.

Gero Mueller currently works as a research assistant and Ph.D. student in the computer graphics group of Prof. Reinhard Klein at the University of Bonn, Germany. He received his diploma in computer science from the University of Bonn in 2002. His main research interests are realistic material representations, in particular BTFs. He has authored and co-authored several papers about this topic. At Eurographics 2004 he presented a state-of-the-art report covering the acquisition, compression, synthesis and rendering of BTFs and gave tutorials about the topic at various events (e.g. at Siggraph 2005).

Hendrik P. A. Lensch is the head of an independent research group "General Appearance Acquisition and Computational Photography" at the MPI Informatik in Saarbrücken, Germany. The group is part of the Max Planck Center for Visual Computing and Communication. He received his diploma in computers science from the University of Erlangen in 1999 and after joining the computer graphics group at MPI received his PhD from Saarland University in 2003. Dr. Lensch spent two years (2005-2006) as a visiting assistant professor at Stanford University, USA. His research interests include 3D appearance acquisition, image-based rendering and computational photography. For his work on reflectance measurement he received the Eurographics Young Researcher Award 2005. He was awarded an Emmy Noether Fellowship by the German Research Foundation in 2007. He has given several lectures and tutorials at various conferences including SIGGRAPH courses on realistic materials in 2002 and 2005.

Annotated Bibliography

Introduction

The goal of this annotated bibliography is to provide an overview over the most important publications in the areas covered by the course. Our goal was especially to help newcomers to the field to quickly become familiar with the main papers and serve as a starting point for further literature study. This is naturally always a subjective choice and we claim therefore by no means that the list of selected papers is complete and apologize for any important papers we missed.

General References

- [1] Richard S. Hunter and Richard W. Harold. *The Measurement of Appearance*. Wiley, 2. ed., 5. print. edition, 1987.

In this book, the various effects of reflections off surfaces are carefully described and analyzed. The authors provide valuable and intuitive insights on how to distinguish the appearance of two different materials. The book furthermore illustrates how the appearance of real world surfaces can be measured giving examples of techniques commonly applied in print industry. The main focus is on measuring the appearance of planar surfaces.

- [2] Fred E. Nicodemus, Joseph C. Richmond, Jack J. Hsia, I. W. Ginsberg, and T. Limperis. *Geometrical Considerations and Nomenclature for Reflectance*. National Bureau of Standards, 1977.

This report introduces the basic concepts of BSSRDFs, BRDFs, and related functions to describe reflectance. It also defines the nomenclature for all of them and describes their relationships such as the derivation of the BRDF from the BSSRDF.

BRDFs

- [1] James F. Blinn. Models of Light Reflection for Computer Synthesized Pictures. In *SIGGRAPH '77: Proceedings of the 4th annual conference on Computer graphics and interactive techniques*, pages 192–198. ACM Press, 1977.

This paper introduces the empirical Blinn-Phong model (based on the earlier Phong model [18]). It can model more realistic reflections using three parameters (diffuse and specular coefficient, specular exponent). The specular lobe is computed based on the halfway vector.

- [2] Samuel Boivin and André Gagalowicz. Image-based rendering of diffuse, specular and glossy surfaces from a single image. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series*, pages 107–116. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

This paper tries to solve the difficult problem of measuring BRDF in indoor scenes from a single observation. The hope is that the global illumination and grouping of measurements of multiple surface points provide sufficient constraints to estimate a per-patch BRDF. At first a simple diffuse BRDF model is assumed. If the observed error is still insufficient a specular lobe is added. In case of failure, further tests involve anisotropic or mirroring BRDFs.

- [3] R. Cook and K. Torrance. A reflection model for computer graphics. *ACM Transactions On Graphics*, 1(1):7–24, 1982.

The Cook-Torrance model is a modification of earlier reflectance models. The main assumption is that the surface is composed of tiny, perfectly reflective, smooth microfacets oriented at different directions. The facets are assumed to be V-shaped and their distribution is isotropic. The model takes into account the fact that the light might be blocked by other microfacets (shadowing). Similarly, it also considers the fact that the viewer does not see some of the microfacets since they are blocked by the other microfacets (masking effect). The model takes into account an average Fresnel term (polarization is not considered) when modelling the reflectance of individual microfacets. However, it does not allow for multiple light bounces between the microfacets. The orientation of the facets is assumed to have some distribution - Cook and Torrance use the Beckman distribution function.

- [4] P. Debevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the Reflectance Field of a Human Face. In *Proc. SIGGRAPH*, pages 145–156, July 2000. ISBN 1-58113-208-5.

While this paper actually introduced the concept of reflectance fields it also contains a section where a BRDF model is fit to the measured data of each texel. The spatially varying BRDF yields some compression compared to the full reflectance field data set.

- [5] Paul Debevec, Chris Tchou, Andrew Gardner, Tim Hawkins, Charis Poullis, Jessi Stumpfel, Andrew Jones, Nathaniel Yun, Per Einarsson, Therese Lundgren, Marcos Fajardo, and Philippe Martinez. Estimating Surface Reflectance Properties of a Complex Scene under Captured Natural Illumination. Technical Report ICT-TR-06.2004, USC ICT, December 2004.

This reports combines the idea of clustered BRDFs with global inverse illumination. For a number of representative spots/materials the BRDF is captured using standard image-based BRDF measurement techniques under controlled illumination conditions. In order to capture the spatially varying BRDF of a building the incident light onto this building is captured by an environment map which serves as a illumination source in a global illumination framework. Based on the differences between the synthesized images and the captured HDR images the weight for combining the cluster BRDFs are updated for each texel individually.

- [6] A. Gardner, C. Tchou, T. Hawkins, and P. Debevec. Linear light source reflectometry. *ACM Trans. Graphics.*, 22(3):749–758, 2003.

In this paper the fully spatially varying BRDF and a transmission term is estimated for rather flat documents. The illumination is provided by a linear light source which has to be considered during the BRDF estimation. The same data is also used to scan the 3D geometry of the surface.

- [7] Athinodoros S. Georghiades. Recovering 3-d shape and reflectance from a small number of photographs. In *Eurographics Symposium on Rendering: 14th Eurographics Workshop on Rendering*, pages 230–240, June 2003.

Georghiades addresses the problem of estimating shape and reflection properties at the same time. Given a set of images of the scene illuminated by a point light source of unknown position the approach sets up an optimization problem that solves for the diffuse component of the BRDF and the actual surface normal per pixel as well as a global specular component and the light source positions in the individual images. As in other shape-from-shading approaches assuming a continuous surface introduces a regularization term that allows for solving the large optimization problem.

- [8] X. He, K. Torrance, F. Sillion, and D. Greenberg. A comprehensive physical model for light reflection. *Computer Graphics*, 25(Annual Conference Series):175–186, 1991.

This paper presents a reflectance model that accounts for the phenomena that can be explained using both geometrical optics and wave optics (diffraction, interference). The

model supports arbitrary polarization of incident light, but the simplifications for unpolarized light are also presented. In general, the reflectance is modelled as a sum of three components: specular, directional diffuse, and uniform diffuse. The specular component accounts for mirror-like reflection. It depends on the Fresnel reflectivity, roughness, and shadowing factors. The directional diffuse contribution of the reflectance function is the most complex term. It accounts for diffraction and interference effects. It depends on surface statistics (the effective roughness and the autocorrelation length). The uniform-diffuse contribution is a result of multiple microfacet reflections and subsurface reflections. It is expressed as a simple function of wavelength. The resulting isotropic reflectance model for unpolarized light is a function of four parameters. Each of the parameters has some physical meaning and (at least theoretically) can be measured separately.

- [9] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear Approximation of Reflectance Functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126. ACM Press/Addison-Wesley Publishing Co., 1997.

The Lafortune model presented in this paper is an extension of the Phong model [18] with a diffuse term and multiple lobes. Each lobe consists of a weighted dot product between viewing and lighting direction raised to some power. This empirical model can handle off-specular peaks, backscattering and anisotropy and is frequently used to model the reflection properties of real, measured materials.

- [10] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Rendering Techniques 2001: 12th Eurographics Workshop on Rendering*, pages 103–114. Eurographics, June 2001. ISBN 3-211-83709-4.

This paper introduces the concept of capturing cluster BRDFs and expressing the spatial variation by per-texel weighted sums of cluster BRDFs. Making use of the idea of image-based BRDF measurements samples from multiple surface points are combined when determining the cluster BRDFs. This results in more reliable, that is, more plausible BRDF parameters and at the same time reduces the number of required input images. Drastically different materials distributed in the same patch can be reproduced faithfully.

- [11] Hendrik P. A. Lensch, Jan Kautz, Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Image-Based Reconstruction of Spatial Appearance and Geometric Detail. *ACM Transactions on Graphics*, 22(2):234–257, April 2003.

This paper extends the previous work towards estimating per-texel normals. Starting from a scanned and smoothed 3D geometry model the per-texel BRDF is estimated. In a photometric stereo approach the current estimate of the BRDF is used to update the surface normal.

- [12] S. Marschner, S. Westin, E. Lafortune, and K. Torrance. Image-based measurement of the Bidirectional Reflection Distribution Function. *Applied Optics*, 39(16):2592–2600, 2000.

Marschner et al. describe an image-based BRDF measurement system. They use a material sample with different surface normals. Each point with a different surface normal gives a different BRDF measurement. Their system uses a spherical sample of homogeneous material. A fixed camera takes images of the sample under illumination from an orbiting light source. The system, although limited to only isotropic BRDF measurements, is both fast and robust. Furthermore, they extend their method to surface geometry acquired with a laser range scanner to acquire reflectance of a human face.

- [13] S. Marschner, S. Westin, E. Lafortune, K. Torrance, and D. Greenberg. Image-based BRDF Measurement Including Human Skin. In *10th Eurographics Workshop on Rendering*, pages 131–144, June 1999.

This paper applied the idea of image-based BRDF measurement to objects of arbitrary geometry. A 3D scan of the object provides the geometric information. Multiple images illuminated by a flash light are combined in order to estimate a single BRDF for the object.

- [14] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A data-driven reflectance model. *ACM Trans. Graph.*, 22(3):759–769, 2003.

The authors built an automatic measurement setup to densely capture isotropic BRDFs using spherical material samples. They analyze the data and construct a low-dimensional data-driven BRDF model using non-linear dimensionality reduction techniques.

- [15] D. McAllister, A. Lastra, and W. Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. *Graphics Hardware 2002*, 2002.

The authors present the first real-time rendering framework for BTFs. They used the Lafortune model to approximate the spatially varying BRDFs which leads to an extreme compact representation amendable to hardware implementation. Since the Lafortune model does not approximate meso-scale shadowing and masking effects well, it is only suitable for materials with minor depth variation (SVBRDFs).

- [16] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental analysis of brdf models. In *Proceedings of the Eurographics Symposium on Rendering*, pages 117–226. Eurographics Association, 2005.

This paper extends the measurement setup of [14] to anisotropic BRDFs. It furthermore fits the parameters of several BRDF models to the measured materials and analyzes the fitting quality.

- [17] K. Nishino, Z. Zhang, and K. Ikeuchi. ”determining reflectance parameters and illumination distribution from a sparse set of images for view-dependent image synthesis”. In *in Proc. of Eighth IEEE International Conference on Computer Vision ICCV ’01*, pages 599–606, July 2001.

Nishino et al. address the complicated problem of reconstructing BRDF and incident illumination at the same time. Specular highlights observed in the individual images are projected into a global environment map to estimate incident illumination. In the next step the BRDF is estimated. Spatial variation is restricted to the diffuse component.

- [18] Bui Tuong Phong. Illumination for Computer Generated Pictures. *Commun. ACM*, 18(6):311–317, 1975.

This paper introduces the Phong model – one of the earliest empirical lighting models for computer graphics. The model consists of a diffuse term and one specular lobe. It is neither energy conserving nor reciprocal and is only well-suited to approximate plastic-like materials. Improvements and extensions of the model include [1, 9].

- [19] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In Eugene Fiume, editor, *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 117–128. ACM Press / ACM SIGGRAPH, August 2001. ISBN 1-58113-292-1.

This paper as well presents a solution to the problem of estimating BRDF and illumination from the same set of images for which an iterative algorithm has been developed. The paper is mostly well-known for the use of spherical harmonics to represent environment maps as well as BRDFs. This representation allows for computing the convolution of the BRDF with the environment map by a simple dot product.

- [20] Y. Sato, M. Wheeler, and K. Ikeuchi. Object Shape and Reflectance Modeling from Observation. In *Proc. SIGGRAPH*, pages 379–388, August 1997.

In this paper shape and reflectance properties are captured using the same sensor but different illumination. There is no explicit registration step necessary to match 3D geometry and 2D images. The diffuse component of the BRDF is estimated per pixel while the specular component is constant per patch.

- [21] G. Ward. Measuring and modeling anisotropic reflection. *Computer Graphics*, 26(Annual Conference Series):265–273, 1992.

This paper presents one of the first methods to speed up the BRDF measurement process. Ward's measurement device (imaging gonio-reflectometer) consists of a hemispherical mirror and a CCD camera with a fisheye lens. The main advantage of his system is that the CCD camera can take multiple, simultaneous BRDF measurements. Each photosite of the imaging sensor contains a separate BRDF value. Moving the light source and material over all incident angles enables the measurement of arbitrary BRDFs. Ward also presents a BRDF model that is based on the elliptical Gaussian distribution. The model is carefully designed to be physically plausible - it supports energy conservation and reciprocity. It is also relatively simple and can be evaluated efficiently. The parameters of the model have physical meaning and theoretically can be measured independently.

- [22] Y. Yu, P. Debevec, J. Malik, and T. Hawkins. Inverse Global Illumination: Recovering Reflectance Models of Real Scenes From Photographs. In *Proc. SIGGRAPH*, pages 215–224, August 1999.

This paper considers indoor scenes. A few input images are aligned with a geometry model of the room and the furniture. Given the positions of the light sources, the light transport in the room can be simulated incorporating global illumination effects. The estimated BRDF minimizes the error between the measured values and a global illumination forward solution.

- [23] Y. Yu and J. Malik. Recovering Photometric Properties of Architectural Scenes from Photographs. In *Proc. SIGGRAPH*, pages 207–218, July 1998.

The BRDFs of buildings in outdoor scenes are estimated considering the incident illumination from the sun and the sky. Only the diffuse component is allowed to vary freely across the surface.

BTFs

- [1] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 151–157, 1997.

This paper introduced Bidirectional Texture Functions to the computer graphics community. The authors present the CURET reflectance and texture database which made BTF and BRDF measurements publicly available for the first time. The sampling density of the BTFs (205 images per material) was not yet sufficient for high-quality rendering.

- [2] Jefferson Y. Han and Ken Perlin. Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Trans. Graph.*, 22(3):741–748, 2003.

A promising approach for capturing several BTF samples at once using a kaleidoscope is presented. A problem with the approach is that it is quite sensitive to imperfections in the mirrors and their configuration because the light is reflected several times within the kaleidoscope.

- [3] Xinguo Liu, Yaohua Hu, Jingdan Zhang, Xin Tong, Baining Guo, and Heung-Yeung Shum. Synthesis and Rendering of Bidirectional Texture Functions on Arbitrary Surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 10(3):278–289, 2004.

This paper can be regarded as a follow up paper to the BTF synthesis paper of Tong et al. from Siggraph 2002. It uses SVD to compress the BTF data and shows how the BTF can be synthesized and rendered from this compressed representation while achieving a significant speed up compared to the original method. It is also shown how the BTF can be rendered with graphics hardware.

- [4] D. McAllister, A. Lastra, and W. Heidrich. Efficient rendering of spatial bi-directional reflectance distribution functions. *Graphics Hardware 2002*, 2002.

The authors present the first real-time rendering framework for BTFs. They used the Lafortune model to approximate the spatially varying BRDFs which leads to an extreme compact representation amendable to hardware implementation. Since the Lafortune model does not approximate meso-scale shadowing and masking effects well, it is only suitable for materials with minor depth variation (SVBRDFs).

- [5] Jan Meseth, Gero Müller, and Reinhard Klein. Reflectance field based real-time, high-quality rendering of bidirectional texture functions. *Computers and Graphics*, 28(1):103–112, February 2004.

This paper addresses the problem of using parametric functions for representing BTFs with significant meso-structure. They propose to fit parametric functions not to the whole per-textel apparent BRDF but to the per-view per-textel reflectance functions which use to be relatively smooth functions.

- [6] G. Müller, G. H. Bendels, and R. Klein. Rapid synchronous acquisition of geometry and btf for cultural heritage artefacts. In *The 6th International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST)*, pages 13–20. Eurographics Association, Eurographics Association, November 2005.

Based on a camera array of 151 of-the-shelf digital cameras a method for rapidly acquiring the geometry and reflectance of objects with significant meso-scale geometry is presented. It combines image-based 3D reconstruction and BTF compression and rendering techniques.

- [7] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein. Acquisition, synthesis and rendering of bidirectional texture functions. *Computer Graphics Forum*, 24(1):83–109, March 2005.

This comprehensive overview discusses from acquisition, over synthesis to rendering of BTFs most of the topics covered in the BTF-part of this tutorial. The relevant publications in the field of BTFs up to the year 2005 are introduced.

- [8] G. Müller, R. Sarlette, and R. Klein. Data-driven local coordinate systems for image-based rendering. *Computer Graphics Forum*, 25(3), September 2006.

In this paper a data-driven technique for computing local coordinate systems from image-based reflectance measurements is presented. These coordinate systems allow to align the per-textel reflectance measurements which results in increased compression performance with negligible run-time overhead.

- [9] M. Sattler, R. Sarlette, and R. Klein. Efficient and realistic visualization of cloth. *Proceedings of the Eurographics Symposium on Rendering 2003*, 2003.

In this paper the first BTF real-time rendering framework based on statistical data analysis is presented. It describes the whole pipeline from measurement using a fully automatic setup over compression to rendering including image-based illumination and large scale shadows. It also introduces the BTF Database Bonn which still offers the most detailed publicly available BTF data.

- [10] Peter-Pike Sloan, Xinguo Liu, Heung-Yeung Shum, and John Snyder. Bi-Scale Radiance Transfer. *ACM Transactions on Graphics*, 22(3):370–375, 2003.

The authors combine Precomputed Radiance Transfer with BTFs to achieve striking real-time renderings of BTF-covered objects realistically lit by environment maps. They represent the BTF by projecting the data per sampled view direction into the Spherical Harmonics basis.

- [11] Frank Suykens, Karl vom Berge, Ares Lagae, and Philip Dutré. Interactive Rendering of Bidirectional Texture Functions. In *Eurographics 2003*, pages 463–472, September 2003.

This BTF compression and rendering method approximates the BTF data per texel using a sophisticated factorization scheme called Chained Matrix Factorization. The idea is to factorize the data with different parameterizations which are suitable for the different significant features of the per-texel apparent BRDFs. Thereby the data can be reliably represented with a much smaller number of factors compared to standard matrix factorization based on SVD.

- [12] M. A. O. Vasilescu and Demetri Terzopoulos. Tensor textures: Multilinear image-based rendering. In *Proceedings of SIGGRAPH*, August 2004.

This work introduces tensor representations for image-based datasets. In contrast to the classic matrix representation multi-linear tensors allow a so-called strategic dimensionality reduction. This means that e.g. more components can be spent for encoding the view variation which results in perceptually more satisfying reconstructions.

- [13] Hongcheng Wang, Qing Wu, Lin Shi, Yizhou Yu, and Narendra Ahuja. Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Trans. Graph.*, 24(3):527–535, 2005.

This paper improves the 3D tensor representation of Vasilescu et al. by arranging the data in higher-dimensional tensors (e.g. 5D). This allows to exploit the coherence along other dimensions like between the rows and columns of the measured images. The method achieves very high compression rates, generalizes to higher-dimensional datasets like time-varying BTFs and can be implemented as an out-of-core technique. The reconstruction costs are a disadvantage of the method.

Near-field Reflectance Fields

- [1] Billy Chen and Hendrik P. A. Lensch. Light source interpolation for sparsely sampled reflectance fields. In Günther Greiner, Joachim Hornegger, Heinrich Niemann, and Marc Stamminger, editors, *Vision, Modeling, and Visualization 2005 (VMV'05)*, pages 461–469, Erlangen, Germany, November 2005. Aka.

Captured reflectance fields are typically sparsely sampled in the light direction domain. In this paper, a method is presented that allows for smoothly moving light sources in near-field reflectance fields. The system treats high frequency illumination effects such as highlights and shadows separately from slowly moving effects such as the cosine fall-off and interreflections, for which linear blending is sufficient to reproduce the appearance of intermediate light source positions. Highlights and shadows are detected using intrinsic images and then moved according to the detected optical flow. The technique further exploits the properties of near-field reflectance fields to perform virtual 3D scanning.

- [2] Yanyun Chen, Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Shell texture functions. *ACM Transactions on Graphics*, 23(3):343–353, August 2004.

This paper presents an appearance representation approach that is particularly suited for heterogeneous translucent objects. The translucent object is divided into a homogeneous diffusely scattering core surrounded by volume of heterogeneous translucent material. The shell texture function (STF) provides an intermediate data structure representing the light transport and the mesostructure of the outer shell. For each voxel in the shell volume the irradiance due to light impinging from arbitrary directions is precomputed and stored in a 5D data structure.

- [3] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A frequency analysis of light transport. *ACM Transactions on Graphics*, 24(3):1115–1126, August 2005.

This paper analyzes the different effects of occluders, reflectors, or the propagation of light in free space on the spatial and angular frequency content of the transformed light field. The authors propose a signal-processing framework and show a large set of instructive examples. They further show how the analysis of the frequency content of the light field can be used to control sampling rates or the choice of reconstruction kernels in rendering, pre-computed radiance transfer, and inverse rendering.

- [4] Gaurav Garg, Eino-Ville Talvala, Marc Levoy, and Hendrik P. A. Lensch. Symmetric photography: Exploiting data-sparseness in reflectance fields. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 251–262, June 2006.

Capturing dense light transport matrices so far required a sequential sensing of individual incident light rays. In this paper, two techniques are combined in order to allow for fast acquisition of arbitrarily complex reflectance fields. The first is the use of H-matrices which subdivide a matrix hierarchically until each sub-block can be represented sufficiently well using a low-rank approximation of the block. The second ingredient is an symmetric acquisition system where cameras and projectors are coupled by a beam splitter allowing for emitting light and sensing light along exactly the same rays. This turns the captured reflectance field into a symmetric tensor whose sub-blocks can be determined in parallel given that they are of low rank. The paper features one of the first full 8D reflectance fields, at a rather low resolution, though.

- [5] Michael Goesele, Hendrik P. A. Lensch, Jochen Lang, Christian Fuchs, and Hans-Peter Seidel. DISCO – Acquisition of Translucent Objects. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, 23(3), 2004.

This is the first paper that captured the diffuse reflectance R_d of a real translucent object with inhomogeneous material properties. The object is pointwise illuminated and its impulse response function is captured with a HDR camera. A hierarchical model of transfer functions is computed from a large number of input images. Rendering can be performed in real time using an earlier approach.

- [6] Akira Ishimaru. *Wave Propagation and Scattering in Random Media*. Academic Press, 1978.

This book describes the physical principles of single and multiple scattering in various types of media and derives the mathematical formulations.

- [7] Henrik Wann Jensen and Juan Buhler. A Rapid Hierarchical Rendering Technique for Translucent Materials. In *SIGGRAPH 2002*, pages 576–581, 2002.

The authors propose a hierarchical evaluation technique to speed up the rendering of translucent objects using the dipole model [8]. This is the first of a whole series of papers proposing various rendering techniques to speed up evaluation of the dipole model – see e.g. [5] for a list of such publications.

- [8] Henrik Wann Jensen, Stephen R. Marschner, Marc Levoy, and Pat Hanrahan. A Practical Model for Subsurface Light Transport. In *SIGGRAPH 2001*, pages 511–518, 2001.

This paper introduces the dipole model as an approximation for translucent objects in computer graphics. It describes the derivation of the model, its use for rendering, and compares the results to Monte-Carlo simulations. The authors describe also a measurement setup to determine the required parameters for real materials and provide a table of measured values. The dipole model is used in many publications including [7] as a fast method to evaluate the effects of subsurface scattering.

- [9] Shree K. Nayar, Gurunandan Krishnan, Michael D. Grossberg, and Ramesh Raskar. Fast separation of direct and global components of a scene using high frequency illumination. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, pages 935–944, New York, NY, USA, 2006. ACM Press.

In this paper a very efficient method is presented for separating the direct and the global component of the light reflected by a scene due to illumination by a projector. The key observation is that global light transport is due to multiple scattering and therefore dampens high frequency in spatially varying illumination patterns. The technique makes use of multiple shifted high frequency patterns and provides a very simple formula to perform the separation from the minimum and maximum intensity observed for each pixel in the sequence of shifted patterns. The separation results to some extent depend on the frequency of the illumination pattern.

- [10] Pieter Peers, Karl vom Berge, Wojciech Matusik, Ravi Ramamoorthi, Jason Lawrence, Szymon Rusinkiewicz, and Philip Dutré. A compact factored representation of heterogeneous subsurface scattering. *ACM Transactions on Graphics*, 25(3):746–753, July 2006.

This paper presents a method for transferring the reflection properties of one heterogeneous translucent object onto novel geometry. In an initial acquisition the diffuse subsurface reflectance is measured on a planar slab of material by illuminating individual points. The effect of subsurface scattering is assumed to be localized having a well controlled support. In order to compress the captured reflectance function the illumination peaks are aligned to one column and a set of homogeneous BSSRDFs is determined to describe the general shape. Dividing the measured samples by the homogeneous approximation results in a representation of the heterogeneous effects which can be factored in a compact way. When transferring the reflectance function to novel geometry only the light transport in a local neighborhood is considered.

- [11] Steven M. Seitz, Yasuyuki Matsushita, and Kiriakos N. Kutulakos. A theory of inverse light transport. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision*, pages 1440–1447, Washington, DC, USA, 2005. IEEE Computer Society.

Given a captured near-field reflectance field between a projector and a camera, this paper analyzes how the reflectance field can be inverted in order to render the scene after the first, the second, or after multiple light indirections. The results indicate that it is sometimes possible to remove multiple scattering effects from captured reflectance fields. Note that the inversion of the reflectance field is possible only for a couple of special cases.

- [12] Pradeep Sen, Billy Chen, Gaurav Garg, Stephen R. Marschner, Mark Horowitz, Marc Levoy, and Hendrik P. A. Lensch. Dual photography. *ACM Transactions on Graphics*, 24(3):745–755, August 2005.

This paper presents an acquisition system for capturing near-field reflectance fields, i.e. measuring the light transport on a ray-to-ray basis. Using an adaptive algorithm, the reflectance field between a camera and a projector is measured such that the influence of

every projector pixel to every camera pixel is determined, yielding a 4D light transport matrix. Exploiting Helmholtz reciprocity, the light transport direction can be inverted. Instead of sending out light from the projector it is turned virtually into a sensing camera capturing the scene as if illuminated by a virtual projector at the location of the original camera. The adaptive and parallel capturing scheme accelerates the acquisition time for sparse light transport matrices by three orders of magnitude.

- [13] Xin Tong, Jiaping Wang, Stephen Lin, Baining Guo, and Heung-Yeung Shum. Modeling and rendering of quasi-homogeneous materials. *ACM Transactions on Graphics*, 24(3):1054–1061, August 2005.

This paper features an acquisition system and a model for capturing and rendering quasi-homogeneous materials. The model consists of a homogeneous subsurface reflectance function augmented by two functions modeling the mesostructure effects locally, i.e. independently for the incident and the exitant point of the light transport. The subsurface scattering effect is captured by sweeping a line stripe laser over the surface from various directions. In addition, a full BTF is acquired.

Presenters' Slides

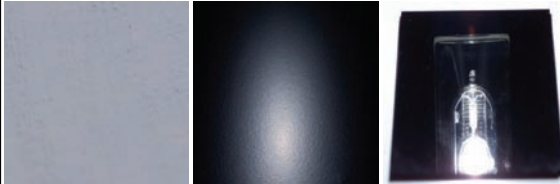
Presenters' Slides

Capturing Reflectance
From Theory to Practice

Introduction

Hendrik P.A. Lensch
MPI Informatik

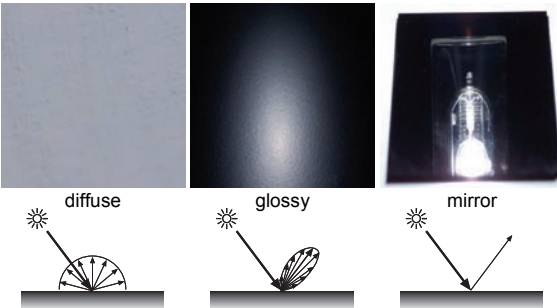
Material Samples



diffuseglossymirror

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

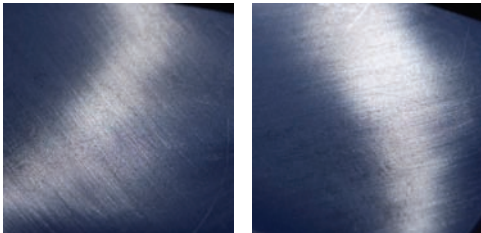
Material Samples



diffuseglossymirror

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Material Samples



anisotropic

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

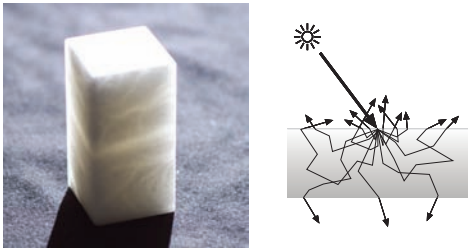
Material Samples



translucent

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

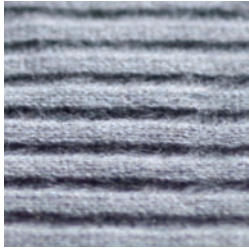
Material Samples



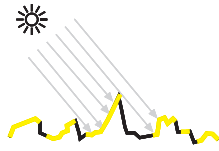
translucent

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Material Samples



complex surface structure



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Material Samples



fibers

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

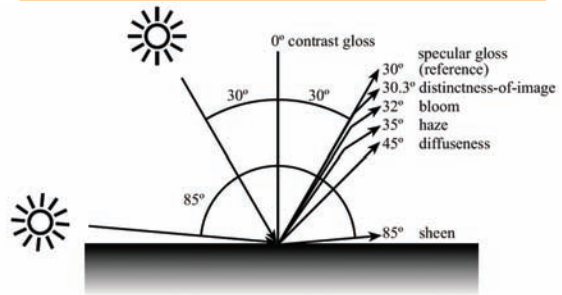
How to describe materials?

- mechanical, chemical, electrical properties
- reflection properties
- surface roughness
- geometry/meso-structure
- **relightable** representation of appearance

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

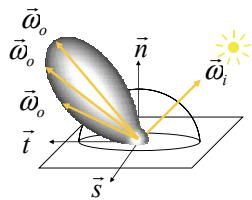
Gloss Model



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

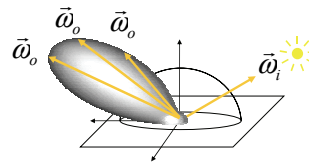
Reflection of an Opaque Surface



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Reflection of an Opaque Surface



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

BRDF – 4D

(bidirectional reflectance distribution function)

$$f_r(\vec{\omega}_i \rightarrow \vec{\omega}_o)$$

The diagram shows a 3D coordinate system with a horizontal plane representing a surface. A normal vector \vec{n} points upwards. An incident direction vector $\vec{\omega}_i$ is shown as a yellow arrow pointing towards the surface, accompanied by a sun icon. An outgoing direction vector $\vec{\omega}_o$ is shown as a grey arrow pointing away from the surface. A shaded lobe represents the reflected radiance distribution.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

BRDF – 4D

(bi-directional reflectance distribution function)

$$f_r(\vec{\omega}_i \rightarrow \vec{\omega}_o) = \frac{dL(\vec{\omega}_o)}{dE(\vec{\omega}_i)}$$

The diagram is identical to the one in the first slide, showing incident direction $\vec{\omega}_i$ and outgoing direction $\vec{\omega}_o$ on a surface.

ratio of reflected radiance to incident irradiance

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Spatially Varying BRDF – 6D

- heterogeneous materials

$$f_r(\vec{\omega}_i \rightarrow \vec{\omega}_o)$$

The diagram is identical to the 4D BRDF diagram, but includes a position vector \vec{x} on the surface plane.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Spatially Varying BRDF – 6D

- heterogeneous materials

$$f_r(\vec{x}(\vec{\omega}_i \rightarrow \vec{\omega}_o))$$

The diagram is identical to the 4D BRDF diagram, but includes a position vector \vec{x}' on the surface plane.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Isotropic BRDF – 3D

- invariant with respect to rotation about the normal

The diagram is identical to the 4D BRDF diagram, but includes a position vector \vec{x} on the surface plane.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Isotropic BRDF – 3D

- invariant with respect to rotation about the normal

$$f_r(\vec{\omega}_i \rightarrow \vec{\omega}_o)$$

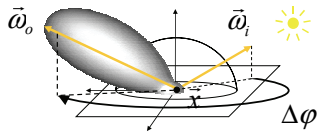
The diagram is identical to the 4D BRDF diagram, but includes a position vector \vec{x} on the surface plane.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Isotropic BRDF – 3D

- invariant with respect to rotation about the normal

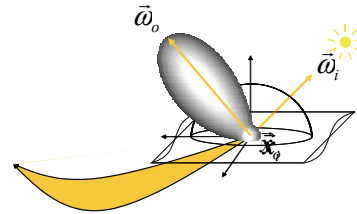
$$f_r((\theta, \phi) \rightarrow (\theta, \phi))$$



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Subsurface Scattering



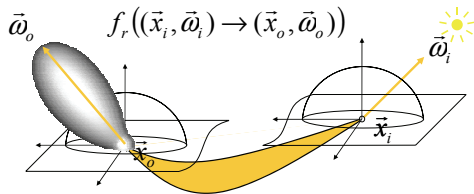
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

BSSRDF – 8D

(bidirectional scattering surface reflectance distribution function)

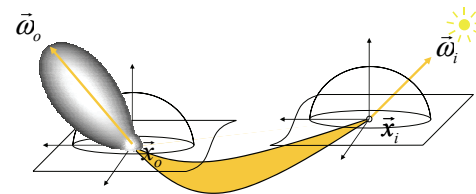
$$f_r(\vec{x}_i, \vec{\omega}_i \rightarrow \vec{x}_o, \vec{\omega}_o)$$



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Subsurface Scattering Homogeneous Material

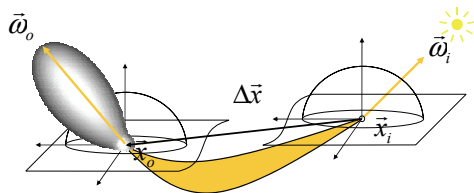


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Homogeneous Material BSSRDF – 6D

$$f_r(\lambda; (\vec{x}_i, \vec{\omega}_i) \rightarrow (\vec{x}_o, \vec{\omega}_o))$$

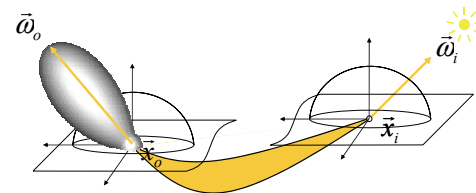


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Generalization – 12D

$$f_r(\lambda; (\vec{x}_i, \vec{\omega}_i) \rightarrow (\vec{x}_o, \vec{\omega}_o))$$



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Generalization – 12D

$$f_r(\lambda; (\vec{x}_i, \vec{\omega}_i) \rightarrow (\vec{x}_o, \vec{\omega}_o))$$

The diagram shows a 3D coordinate system with a surface. An incident light ray is shown as a yellow arrow pointing towards a point \vec{x}_i on the surface, with direction vector $\vec{\omega}_i$. An outgoing light ray is shown as a yellow arrow pointing away from a point \vec{x}_o on the surface, with direction vector $\vec{\omega}_o$. The surface is shaded to show its orientation.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Generalization – 12D

$$f_r((\vec{x}_i, \vec{\omega}_i, \lambda_i) \rightarrow (\vec{x}_o, \vec{\omega}_o, \lambda_o))$$

fluorescence

The diagram is similar to the previous one, but includes a sun icon representing the light source. The incident light is labeled with wavelength λ_i and the outgoing light with wavelength λ_o . The text "fluorescence" is placed above the diagram.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Generalization – 12D

$$f_r(t; (\vec{x}_i, \vec{\omega}_i, \lambda_i) \rightarrow (\vec{x}_o, \vec{\omega}_o, \lambda_o))$$

time-varying scenes

The diagram is similar to the previous one, but includes a sun icon and the text "time-varying scenes" above the diagram.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

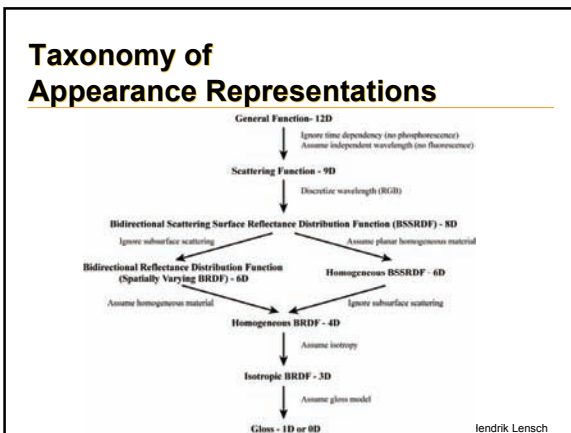
Generalization – 12D

$$f_r((\vec{x}_i, \vec{\omega}_i, t_i, \lambda_i) \rightarrow (\vec{x}_o, \vec{\omega}_o, t_o, \lambda_o))$$

different path length
phosphorescence

The diagram is similar to the previous one, but includes a sun icon and the text "different path length phosphorescence" above the diagram. A jagged line on the surface indicates a complex path.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

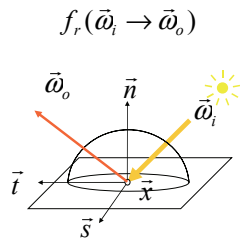


Properties of Reflectance Functions

- Helmholtz reciprocity
- energy conservation
- Fresnel effect

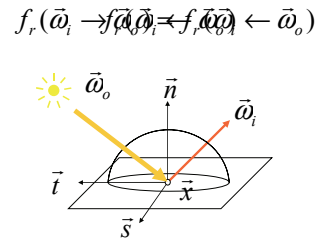
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Helmholtz Reciprocity



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Helmholtz Reciprocity



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Energy Conservation

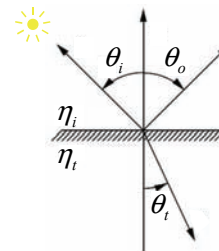
- The sum of energy reflected into all directions has to be smaller or equal than the incident energy.

$$\int_{\Omega_o} f_r(\vec{\omega}_i \rightarrow \vec{\omega}_o) \cos(\theta_i) d\omega_o \leq 1$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

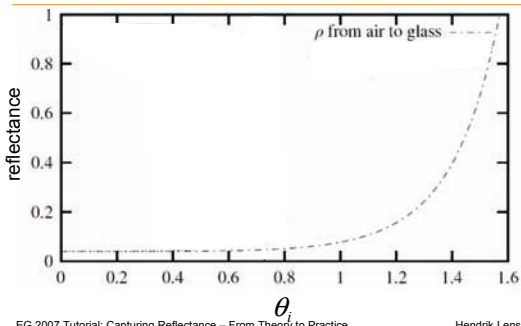
Snell's Law

$$\eta_i(\lambda) \sin \theta_i = \eta_t(\lambda) \sin \theta_t$$



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

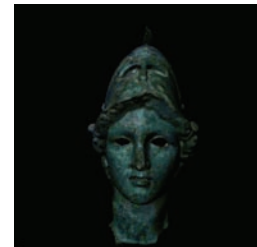
Fresnel Formula



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Material Acquisition

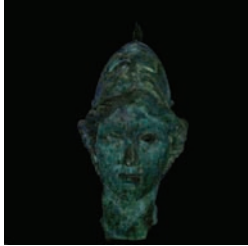
- single picture
 - no interaction



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Material Acquisition

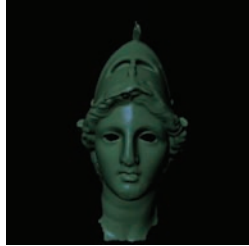
- diffuse color + geometry model
 - no relighting



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Material Acquisition

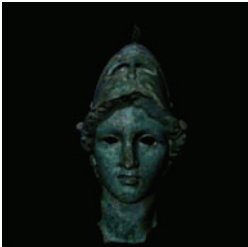
- BRDF + geometry model
 - moving highlights



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Material Acquisition

- spatially-varying BRDF + geometry model
 - moving highlights

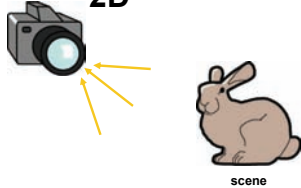


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Digitizing real-world Objects

a single photograph

2D



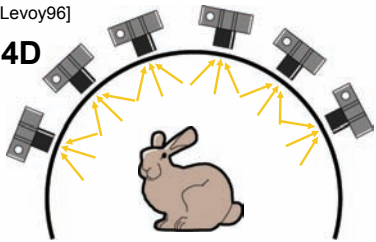
scene

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Light Fields

[Gortler96], [Levoy96]

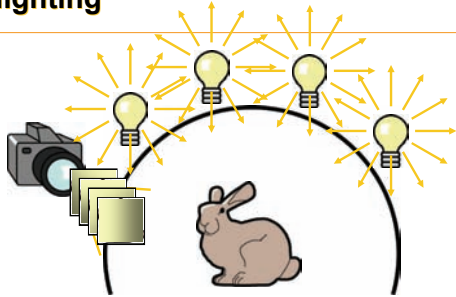
4D



distribution of all reflected light rays

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Relighting



one picture for each light direction

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Relighting

[Debevec2000]

The diagram shows a camera on the left capturing a scene with a rabbit in the center. Above the rabbit, several light bulbs are arranged in a semi-circle, representing different light sources. A plus sign is placed between the camera and the light sources, indicating the superposition principle. Below the rabbit, the text 'superposition principle' is written.

superposition principle

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

4D Reflectance Fields

[Debevec2000]

The diagram shows a camera on the left capturing a scene with a rabbit in the center. Above the rabbit, several light bulbs of different colors (yellow, red, blue) are arranged in a semi-circle, representing different light sources. A plus sign is placed between the camera and the light sources, indicating the superposition principle. The text '2D' is written next to the camera, and '4D' is written next to the light sources.

2D

4D

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Far- vs. Near Field Illumination

The photograph shows a lamp with a pleated shade in a room. In the background, there are window blinds. The lamp is in the foreground, and the blinds are in the background, illustrating the difference between far-field and near-field illumination.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

6D Reflectance Fields

Near Field illumination

[Masselus2003]

The diagram shows a camera on the left capturing a scene with a rabbit in the center. Above the rabbit, several light sources (represented by small squares) are arranged in a semi-circle, representing different light sources. A plus sign is placed between the camera and the light sources, indicating the superposition principle. The text '2D' is written next to the camera, and '4D' is written next to the light sources. Below the rabbit, the text 'relighting with 4D incident light fields' is written.

2D

4D

relighting with 4D incident light fields

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

8D Reflectance Fields

The diagram shows a camera on the left capturing a scene with a rabbit in the center. Above the rabbit, several light sources (represented by small squares) are arranged in a semi-circle, representing different light sources. A plus sign is placed between the camera and the light sources, indicating the superposition principle. The text '4D' is written next to the camera, and '4D' is written next to the light sources. Below the rabbit, the text 'arbitrary perspective + arbitrary illumination' is written.

4D

4D

arbitrary perspective + arbitrary illumination

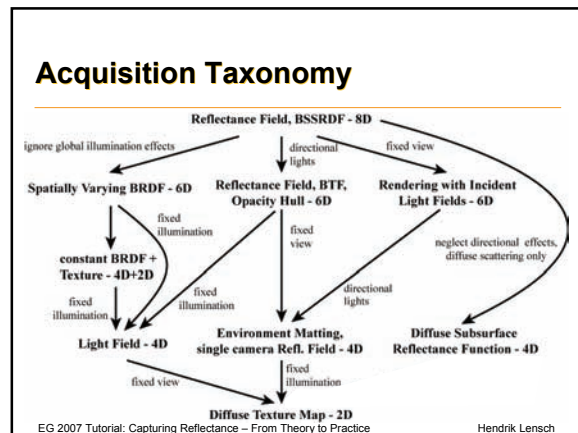
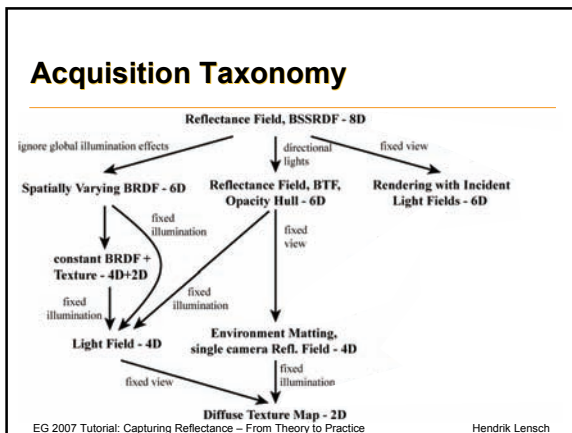
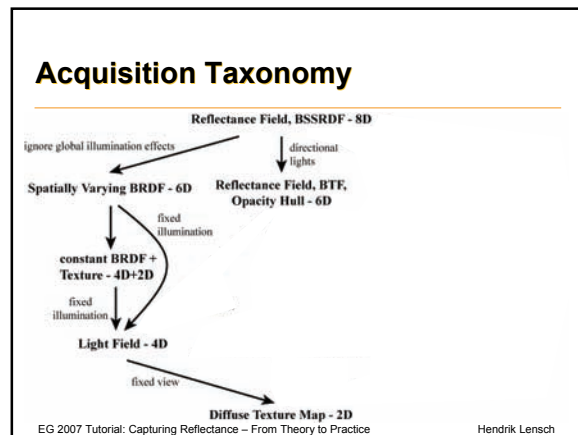
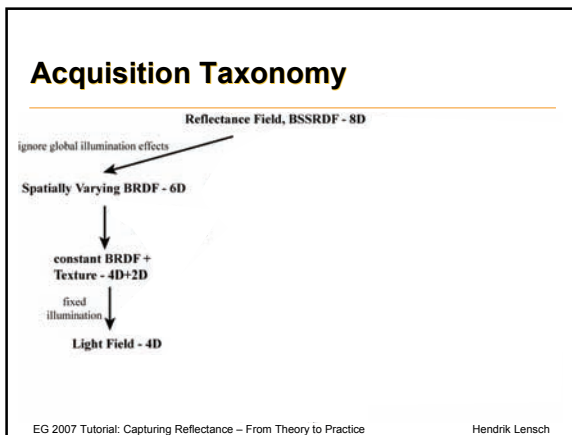
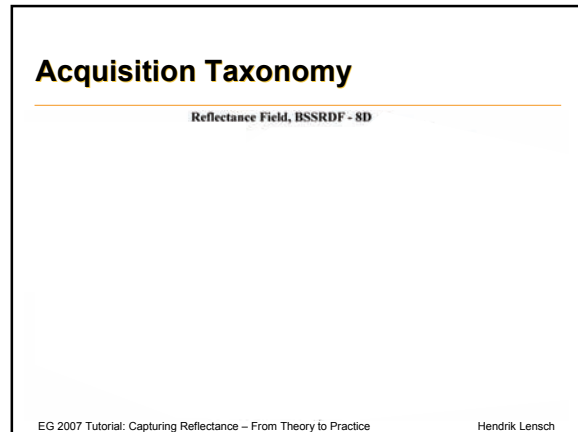
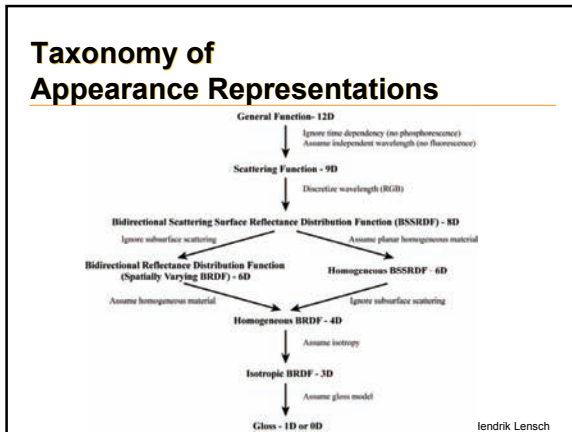
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

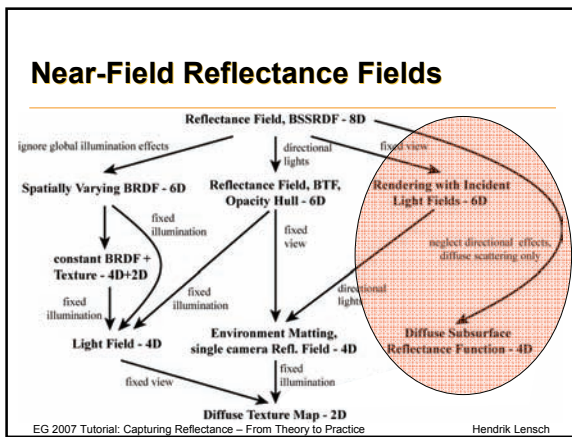
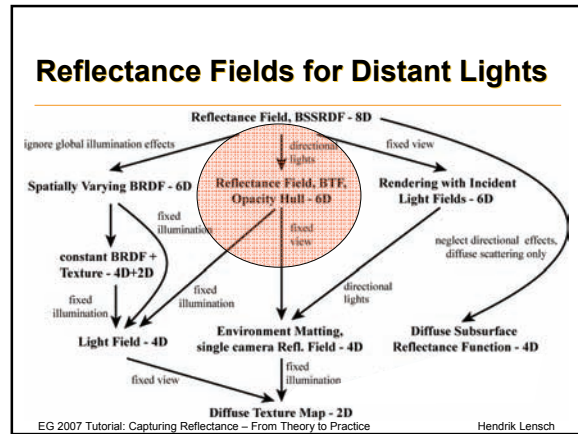
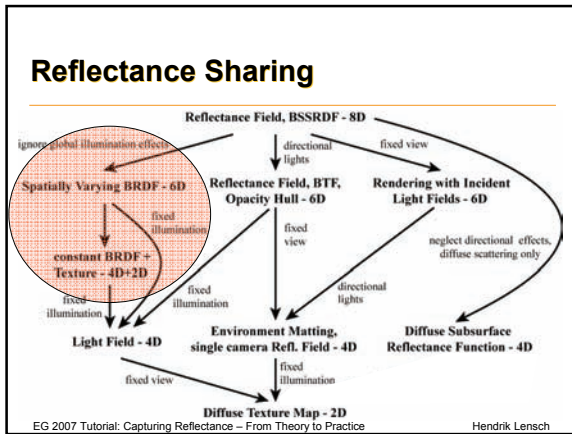
Acquisition Approaches

- hard to sample an 8D function
- dimensionality reduction
- sampling density

- restricted viewing and relighting capabilities
- restriction to a specific class of materials/objects

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch





Summary

- densely sampling 8D functions almost impossible
- less dimensions might be sufficient for specific tasks / materials

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice
Hendrik Lensch

**Capturing Reflectance
From Theory to Practice**

Acquisition Basics

Michael Goesele
University of Washington

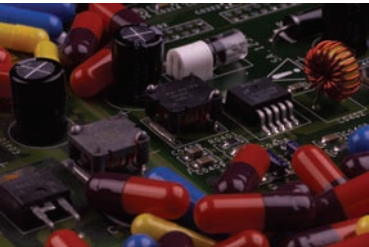
Goal of this Section

- practical, hands-on description of acquisition basics
- general overview, caveats, misconceptions, solutions, hints, ...
- biased to the techniques used in our lab

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

How can we measure material properties?

- color
- texture
- reflection properties
- normals
- ...



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele


Special Purpose Tools

- gloss meter, haze meter, ...
 - various appearance characteristics
- spectrophotometer
 - spectral reflectance of a surface
- often used in industry where single parameters of one material are important

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

General Purpose Tools


- setup with digital camera(s), controlled lighting, ...
- foundation of image-based techniques



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

General Purpose Tools

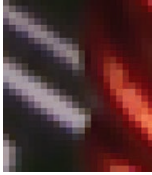
- digital camera as
 - massively parallel sensor
 - mostly tristimulus color
 - often high quality optical system
 - tuned to make good and/or correct pictures



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Overview Acquisition Basics

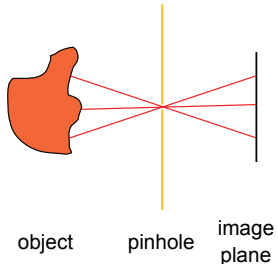
- digital cameras
 - geometric and photometric calibration
 - high dynamic range imaging
- light sources
- lab setup



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Pinhole Camera Model

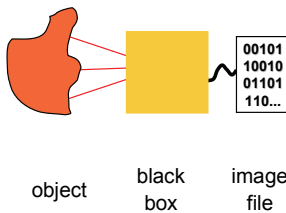
- “each pixel corresponds to one ray through the pinhole onto the object”
- not valid for most digital cameras!!!



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

(Pessimistic) Digital Camera Model


- digital camera as a black box
- take only for granted what you measured (or what is given in the manual)



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

(Pessimistic) Digital Camera Model

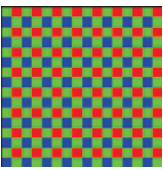
- optical lens system instead of pinhole aperture (aberration, vignetting)
- CCD/CMOS chip and A/D conversion
- normally only one color per pixel (e.g. Bayer pattern) requires demosaicing
- camera image processing
- ...



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Bayer Pattern

- sensor records only one color per pixel
 - higher sampling rate in green channel (luminance channel)
- remaining two color values per pixel must be reconstructed
 - artifacts possible

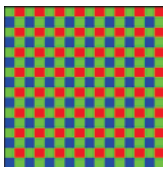


Bayer pattern

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Demosaicing

- common approach
 - combining an interpolation and a pattern matching scheme
 - groups pixels into regions and makes some continuity assumption within the regions
 - “nice pictures”, but no guarantee that two of the R,G,B values per pixel are correct




Bayer pattern

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

(Pessimistic) Digital Camera Model

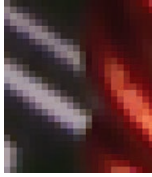
- often globally correct image
- no guarantee that each pixel contains reliable color values
- some issues can be solved using camera calibration
- careful choice of camera for measurements



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Overview Acquisition Basics

- digital cameras
 - geometric and photometric calibration
 - high dynamic range imaging
- light sources
- lab setup



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Geometric Camera Calibration

- get transformation between points in space and image coordinates
- intrinsic camera parameters
 - focal length, distortion coefficients, ...
- extrinsic parameters
 - position, orientation

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Geometric Camera Calibration

- several methods commonly used, e.g., [Tsai '87, Heikkila '97, Zhang '99]
- Matlab calibration toolbox by Jean-Yves Bouguet
 - http://www.vision.caltech.edu/bouguetj/calib_doc/
 - also included in the OpenCV Open Source Computer Vision library distributed by Intel

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Camera Model (simplified from Bouguet)

- point in camera reference frame is mapped to normalized pinhole coordinates

$$x_n = \begin{bmatrix} x_n(1) \\ x_n(2) \end{bmatrix} = \begin{bmatrix} X_c / Z_c \\ Y_c / Z_c \end{bmatrix}$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Camera Model (simplified from Bouguet)

- normalized point coordinates are computed using distortion model
 - only parameterized by distance from center

$$x_d = \begin{bmatrix} x_d(1) \\ x_d(2) \end{bmatrix} = (1 + kc(1)r^2 + kc(2)r^4 + kc(3)r^6)x_n$$

$$r^2 = x_n(1)^2 + x_n(2)^2$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Camera Model (simplified from Bouguet)

- final pixel coordinates are computed using focal length and principal point

$$\begin{bmatrix} x_p(1) \\ x_p(2) \end{bmatrix} = \begin{bmatrix} fc(1) \cdot x_d(1) \\ fc(2) \cdot x_d(2) \end{bmatrix} + \begin{bmatrix} cc(1) \\ cc(2) \end{bmatrix}$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

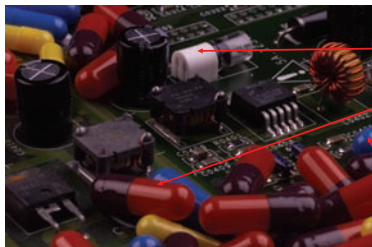
Calibration Approach

- capture images of test target with known geometry
 - cover space and angles with planar target
- solve for intrinsic and extrinsic parameters
- quality can be checked by reprojection



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Photometric Calibration



(225,203,216)

(141,25,4)

(40,70,143)

What do these RGB values (digital counts) mean?

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Camera Response Curve (OECF)

- relationship between digital counts and luminance is unknown (and often non-linear)
 - gamma correction
 - image optimizations
 - ...
- can be described by response curve or OECF (Opto-Electronic Conversion Function)

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Camera Response Curve (OECF)

- direct measurement via test chart
 - patches with known gray levels
 - uniform illumination
- patches arranged in a circle to suppress lens effects (e.g. vignetting)
- inversion using OECF leads to pixel values linearly related to luminance values

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Definition of Dynamic Range

- dynamic range is the ratio of brightest to darkest (non-zero) intensity values in an image
 - assuming linear intensity
- often given as
 - ratio: 1:100.000
 - orders of magnitude: 5 orders of magnitude
 - in decibel: 100 dB

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Sources of Dynamic Range

- diffuse materials reflect 0.5% – >90% of incoming light
 - specular highlights much brighter
 - lit regions vs. in shadow regions
 - moonless night vs. sunny day
- high dynamic range mainly caused by illumination effects

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Sources of Dynamic Range



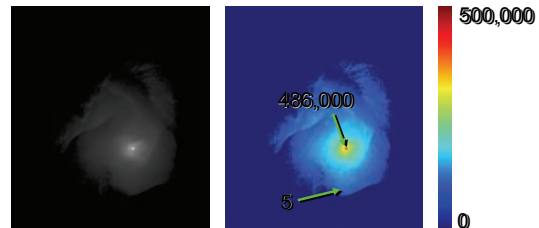
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Dynamic Range of Cameras

- example: photographic camera with standard CCD sensor
 - dynamic range of sensor **1:1000**
 - exposure variation $1/60^{\text{th}}$ s – $1/6000^{\text{th}}$ s (handheld camera/non-static scene) 1:100
 - varying aperture f/2.0 – f/22.0 ~1:100
 - exposure bias/varying “sensitivity” 1:10
 - total (sequential) 1:100,000,000
 - simultaneous dynamic range still only **1:1000**

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

High Dynamic Range (HDR) Imaging



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

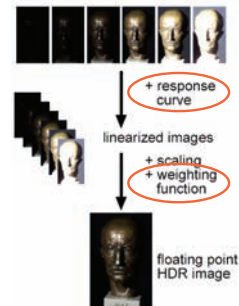
High Dynamic Range (HDR) Imaging

- analog false-color film with several emulsions of different sensitivity levels by Wyckoff in the 1960s
 - dynamic range of about 10^8
- modern CMOS sensors can achieve a dynamic range of 10^6 – 10^8
 - logarithm in analog domain
 - multiple exposure techniques

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

High Dynamic Range Imaging

- extending dynamic range of ordinary camera
- combining multiple images with different exposure



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Determining the Response Curve

- [Madden 1993] assumes linear response
 - correct for raw CCD data
- [Debevec and Malik 1997]
 - selects a small number of pixels from the images
 - performs an optimization of the response curve with a smoothness constraint
- [Robertson et al. 1999, 2003]
 - optimization over all pixels in all images

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Algorithm of Robertson et al.

- Principle of this approach:
 - calculate a HDR image using the response curve
 - find a better response curve using the HDR image
- (to be iterated until convergence)
- assume initially linear response

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Algorithm of Robertson et al.

- input:
 - series of i images with exposure times t_i
 - pixel value at image position j is $y_{ij} = f(t_i x_j)$
- find irradiance x_j and response curve $I(y_{ij})$
 - $t_i x_j$ is proportional to collected charge/radiant energy
 - f maps collected charge to intensity values

$$f^{-1}(y_{ij}) = t_i x_j = I(y_{ij})$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Algorithm of Robertson et al.

- additional input:
 - a weighting function $w(y_{ij})$ (bell shaped curve)
 - an initial camera response curve $I(y_{ij})$ – usually linear
- calculate HDR values x_j from images using

$$x_j = \frac{\sum_i w(y_{ij}) t_i^2 \cdot \frac{I(y_{ij})}{t_i}}{\sum_i w(y_{ij}) t_i^2} \quad y_{ij}$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Algorithm of Robertson et al.

- optimizing the response curve I :
 - start again with definition $f^{-1}(y_{ij}) = t_i x_j = I(y_{ij})$
- minimization of objective function O

$$O = \sum_{i,j} w(y_{ij}) (I(y_{ij}) - t_i x_j)^2$$
- using Gauss-Seidel relaxation yields

$$E_m = \{(i, j) : y_{ij} = m\}$$

$$I(m) = \frac{1}{\text{Card}(E_m)} \sum_{i,j \in E_m} t_i x_j$$
- $\text{Card}(E_m)$ = number of elements in E_m

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Algorithm of Robertson et al.

- both steps are iterated
 - calculation of a HDR image using I
 - optimization of I using the HDR image
 - ➔ I needs to be normalized, e.g., $I(128)=1.0$
- stop iteration after convergence
 - criterion: decrease of O below some threshold
 - usually only a couple of iterations

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

HDR Imaging: Algorithm of Robertson et al.

$\log(I(y_{ij}))$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

HDR Example: Capturing Environment Maps

series of input images

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

HDR Example: Capturing Environment Maps

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Algorithm of Robertson et al.

- choice of weighting function $w(y_{ij})$ for response recovery

$$w_y = \exp\left(-4 \frac{(y_{ij} - 127.5)^2}{127.5^2}\right)$$
 - for 8 bit images
 - possible correction at both ends (over/underexposure)
 - motivated by general noise model

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Algorithm of Robertson et al.

- choice of weighting function $w(y_{ij})$ for HDR reconstruction
 - introduce certainty function c as derivative of the response curve with logarithmic exposure axis
 - approximation of response function by cubic spline to compute derivative
$$w_{ij} = w(y_{ij}) = c(I_{y_{ij}})$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Input Images for Response Recovery

- my favorite:
 - grey card, out of focus, smooth illumination gradient
- advantages
 - uniform histogram of values
 - no color processing or sharpening interfering with the result

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

White Balance

capture the spectral characteristics of the light source to assure correct color reproduction

tungsten daylight
flourescent flash

images taken with different camera settings

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

White Balance

- capture white surface under target illumination
- scale color channels to achieve uniform intensity values
- often built-in function

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Color Calibration

- BRDF model of real object
- long processing pipeline
- which image is (more) correct?

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Color Calibration

- ICC color management system
- capture the properties of all devices
 - camera and lighting
 - monitor settings
 - output properties
- common interchange space

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Color Calibration

- profile connection spaces
 - CIELAB (perceptual linear)
 - linear CIEXYZ color space
- can be used to create an high dynamic range image in the profile connection space
- allows for a color calibrated workflow

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele


Color Calibration

[Goesele et al. 2004]


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Limits of White Balance and Color Calibration

- fluorescence effects
 - signal colors
 - optical brighteners
 - test targets
- color calibration impossible
- cannot be solved using white balance



daylight (HMI)

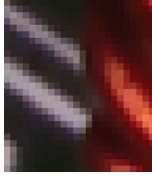


green LED

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Overview Acquisition Basics

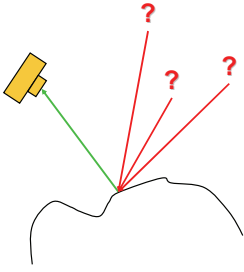
- digital cameras
 - geometric and photometric calibration
 - high dynamic range imaging
- **light sources**
- lab setup
- geometry acquisition



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

General Measurement Approach

- find relation between incoming and outgoing light at a surface point
- derive information from this data
- knowledge and control over light sources needed



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Lighting Requirements

- photometric properties
 - uniform spatial distribution
 - color constant over time
 - even spectral distribution
 - very bright and efficient

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele


Lighting Requirements

- emission pattern
- requirements depend on application, e.g.,
 - well defined light source
 - incident angle as small as possible
 - *parallel light source (e.g. laser beam)*
 - *point light source*
 - lens or reflector based systems are not ideal

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Point Light Source Example

- 800 W HMI light source
- very efficient (equals 2500 W tungsten light)
- (almost) daylight spectrum
- constant colors
- point light source



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Point Light Source Example

- more information about lighting in the individual sections of the course ...

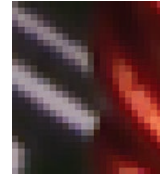


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Overview Acquisition Basics

- digital cameras
 - geometric and photometric calibration
 - high dynamic range imaging
- light sources
- **lab setup**



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

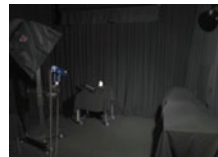
Lab Setup

- part of the lighting considerations
- often low and diffuse reflection required to minimize the influence of the environment
- MPI photo studio
 - walls and ceiling covered with black felt
 - black needle fleece carpet

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Lab Setup



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

Lab Setup

- tuned for efficiency and flexibility
 - enough space
 - enough stands, supporting materials, ...
- have some lighting available in dark areas
 - e.g., radio controlled light switch
- safety concerns

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Michael Goesele

**Capturing Reflectance
From Theory to Practice**

Reflectance Sharing

Michael Goesele
 University of Washington

BRDF

(bi-directional reflectance distribution function)

$$f_r(\vec{\omega}_i \rightarrow \vec{\omega}_o) = \frac{dL(\vec{\omega}_o)}{dE(\vec{\omega}_i)}$$

ratio of reflected radiance to incident irradiance

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Measurement

- Gonioreflectometer

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Image-Based BRDF Measurement

- [Marschner 1999, Lu & Koenderink 1998, ...]
- capture lots of BRDF samples at one shot by a sensor array / camera

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Image-Based BRDF Measurement

- [Marschner 1999, Lu et al. 1998, ...]
- capture lots of BRDF samples at one shot by a sensor array / camera
- homogeneous, isotropic materials only

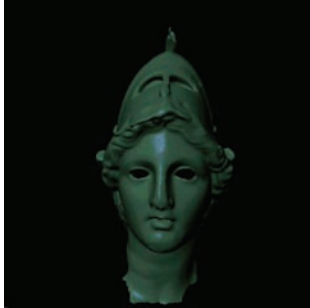
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Image-Based BRDF Measurement

- [Matusik et al. 2003, Ngan et al. 2005]
- systematic capture effort for large number of materials
- includes anisotropic materials
- BRDF database available online
- analysis of captured data using dimensionality reduction techniques

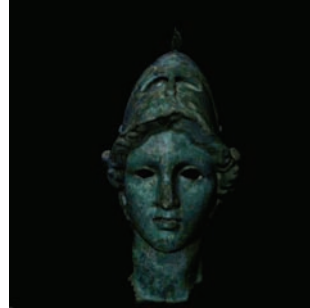
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Homogeneous BRDF



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

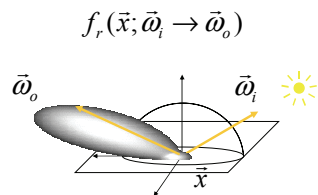
Spatially Varying BRDF



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Spatially Varying BRDF

- heterogeneous materials



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Spatially Varying BRDF

- measurement approach by [Lensch et al. 2003]



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Acquisition Setup

- Camera and light source are moved manually around the object.
- Positions are calibrated with respect to the object.
- The dark room reduces reflections from the environment.



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Fitting and Clustering



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Acquisition

- Capture HDR-images from various viewpoints with different light source positions.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Acquisition

- Capture HDR-images from various viewpoints with different light source positions.

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Fitting and Clustering

```

    graph LR
      A[View Acquisition] --> B[Registration]
      B --> C[Resampling]
      C --> D[BRDF Fitting  
Clustering]
      B --- B1[Visibility/Shadows]
  
```

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

3D-2D Registration

- calibrated gantry
- corresponding points
- silhouette-based method

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Light Source Position

- detect highlights of ring flash reflections
- determine the position of the spheres

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Light Source Position

- detect highlights of light source reflections
- reconstruct light source position


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Light Source Position



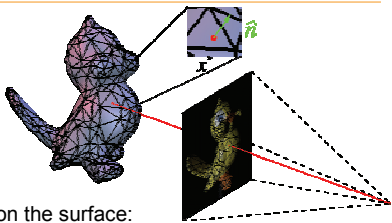
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Fitting and Clustering



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele


Resampling



– for each point on the surface:
 find all images where the point is visible and lit
 take sample at corresponding pixel position
 $(r, \bar{x}, \bar{\omega}_i, \bar{\omega}_o)$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

BRDF Fitting and Clustering



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Key Idea

- Very few radiance samples per texel
 ⇒ no dense sampling of the BRDF
- Most real-world objects consist of a small set of distinct materials.
 ⇒ fit a BRDF model for each basis material
 ⇒ start with the avg. BRDF of the entire surface

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

The Lafortune Model

$$f_r(\hat{\omega}_i, \hat{\omega}_o) = \rho_d + \sum_j C_{x,j} (\omega_{ix} \omega_{ox} + \omega_{iy} \omega_{oy}) + C_{z,j} \omega_{iz} \omega_{oz} |V_j|$$

- physically plausible
- diffuse component plus a number of lobes
- $3 \cdot (1 + i \cdot 3)$ parameters (12 for a single lobe model)
- fit parameters to samples using Levenberg-Marquardt

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Fitting BRDFs to Lumitexels

- define error measure between a BRDF and a lumitexel:

$$E_{f_r}(L) = \frac{1}{|L|} \sum_{R_j \in L} (f_r(\vec{\omega}_{i_j}, \vec{\omega}_{o_j}) \vec{\omega}_{i_z} - r_j)^2$$

= average error over all radiance samples

- perform non-linear least square optimization for a **set** of lumitexels using Levenberg-Marquardt
- yields a single BRDF (i.e. its parameters) per **set** of lumitexels

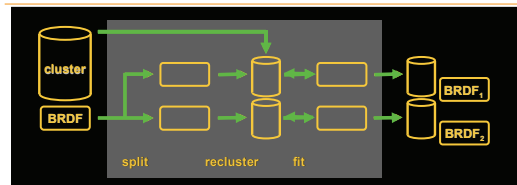
Fitting Result



Clustering

- Goal: separate the different materials
 - similar to Lloyd iteration
 - start with a single cluster containing all lumitexels
 - split cluster along direction of largest variance
 - stop after n clusters have been constructed

Split-Recluster-Fit Cycle



- split into two BRDFs along direction of largest variance of parameters (covariance matrix)
- distribute initial lumitexels forming two new clusters
- refit new BRDFs
- repeat recluster and fitting until clusters are stable

Clustering Results



Spatially Varying Materials



Projection

- Goal: assign a separate BRDF to each lumitexel
 - too few radiance samples for a reliable fit
 - represent the BRDF f_π of every lumitexel by a linear combination of already determined BRDFs of the clusters f_1, f_2, \dots, f_m

$$f_\pi = t_1 f_1 + t_2 f_2 + \dots + t_m f_m$$

- determine linear weights t_1, t_2, \dots, t_m

Projection

- compute the pseudo-inverse using non-negative SVD to get a least squares solution for

$$\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_{|I|} \end{pmatrix} = \begin{pmatrix} f_1(\hat{\omega}_{11}, \hat{\omega}_{o1})\hat{\omega}_{1,z} & f_2(\hat{\omega}_{11}, \hat{\omega}_{o1})\hat{\omega}_{1,z} & \dots & f_m(\hat{\omega}_{11}, \hat{\omega}_{o1})\hat{\omega}_{1,z} \\ f_1(\hat{\omega}_{12}, \hat{\omega}_{o2})\hat{\omega}_{1,z} & f_2(\hat{\omega}_{12}, \hat{\omega}_{o2})\hat{\omega}_{1,z} & \dots & f_m(\hat{\omega}_{12}, \hat{\omega}_{o2})\hat{\omega}_{1,z} \\ \vdots & \vdots & \ddots & \vdots \\ f_1(\hat{\omega}_{|I|1}, \hat{\omega}_{o|I|})\hat{\omega}_{|I|,z} & f_2(\hat{\omega}_{|I|1}, \hat{\omega}_{o|I|})\hat{\omega}_{|I|,z} & \dots & f_m(\hat{\omega}_{|I|1}, \hat{\omega}_{o|I|})\hat{\omega}_{|I|,z} \end{pmatrix} \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{pmatrix}$$

- it is a linear problem!

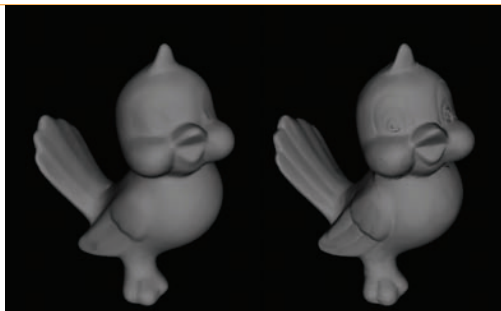
Results



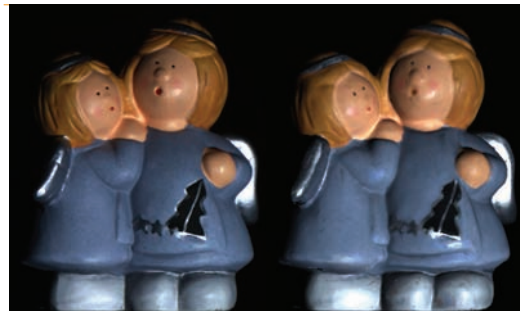
Why to do the complicated clustering?



Normal Fitting



Without Normal Fitting



With Normal Fitting



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele



Conclusion

- determine BRDF of a few basis materials
- spatial variation as a blend of basis BRDFs
- highly efficient acquisition

- model based
- requires geometry model

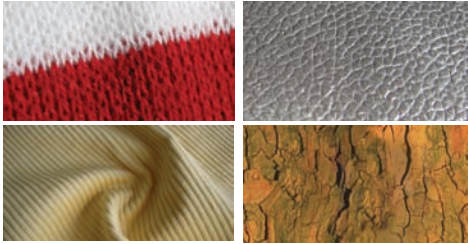
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Michael Goesele

Capturing Reflectance From Theory to Practice

Bidirectional Texture Functions

Gero Müller
University of Bonn

Introduction




- Opaque materials with complex mesogeometry (rough textures)

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Introduction

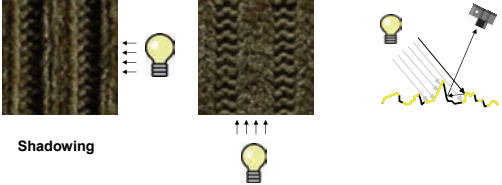
- Goal
 - Capture „look-and-feel“ of those materials independent of a specific physical object
- Capture appearance from material samples
- Standard: single RGB-image
 - Appearance captured only for one view and one lighting situation
 - Valid only for flat and diffuse materials (paper, cardboard,...)



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Introduction

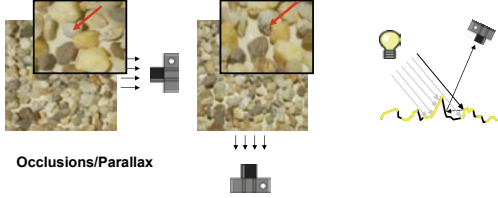
- Images of rough textures with meso-structure contain view- and light dependent shadows, occlusions and local/global illumination effects



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Introduction

- Images of rough textures with meso-structure contain view- and light dependent shadows, occlusions and local/global illumination effects



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Introduction

- Lighting distance large compared to extent of material sample
- Materials are applied to opaque physical objects (furniture, walls, car interior, cloth, ...)
- ➔ Neglect near-field illumination and explicit light-transport between surface points
- ➔ Measure only far-field reflectance field of sample
- ➔ **Bidirectional Texture Function** [Dana et al. 1997]

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Introduction

- BTF \leftrightarrow 6D far-field reflectance field of texture

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Taxonomy

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Overview

- Acquisition
- Compression
- Rendering
- Non-planar objects

- Synthesis not part of this talk

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Acquisition

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

BTF Acquisition

- Sampling a 6D-function $BTF_{rgb}(\mathbf{x}, \mathbf{v}, \mathbf{l})$
 - Take pictures... (spatial dimension)
 - ...under various view and light directions (angular dimensions)

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller


Measurement setups

- Gonioreflectometer-like
 - Advantages
 - fully automatic
 - flexible sampling rate
 - Problems
 - measurement time: ~14h (81x81=6561 images)
 - moving parts: camera, light, sample carrier

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Measurement setups

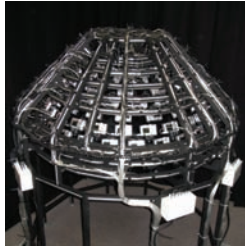
- Using Mirrors [Han et al. 2003]
 - Advantages
 - parallel
 - fast
 - no moving parts
 - Problems
 - small resolution
 - non-perfect mirrors



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Measurement setups

- Using a camera array [Uni Bonn 2005]
 - Advantages
 - fast, parallel ~1 hour (151x151=22501 images)
 - no moving parts
 - high resolution
 - Problems
 - fixed angular sampling
 - complex control apparatus



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

BTF Camera Array

- Custom built hemi-spherical aluminium gantry (80 cm radius) mounted on aluminium base rack
- 151 Canon Powershot A75 digicams (3.2 mpixel)
 - cheapest consumer camera with powerful SDK
 - built-in light source (supports different intensities)
- USB-controllable 160-port relay box for on/off toggling
- Custom built power supply

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

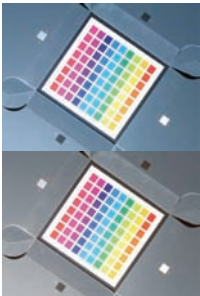
Tasks

- Synchronized control
 - One camera flashes while all cameras take picture
 - High dynamic range
 - 4 passes with different flash intensities and exposures
 - 8 PCs (~19 cameras/PC)
 - 1 Master PC for synchronization

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Tasks

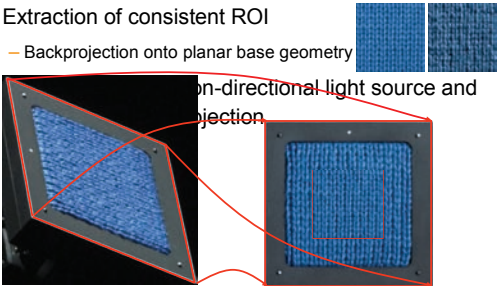
- Calibration
 - Response curve
 - Color calibration
 - Varying flash intensity
 - Camera mapping
 - Lens distortion
 - Intrinsic + extrinsic camera parameters



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

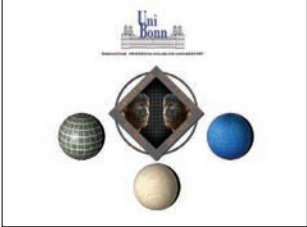
Post-processing

- Extraction of consistent ROI
 - Backprojection onto planar base geometry
- Non-directional light source and projection



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

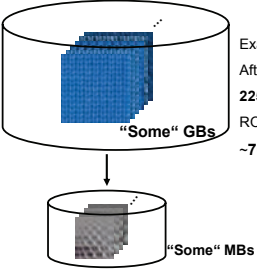
BTF Database Bonn



www.cg.cs.uni-bonn.de/btf

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Compression



Example (Camera array):
After postprocessing:
22501 hdr-images (OpenEXR)
ROI-size 1024x1024
~70-90 GB

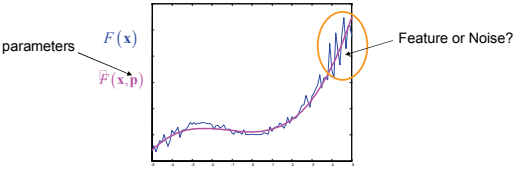
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Compression

- Preferable properties:
 - fast (real-time), random access decompression
 - preservation of visual important features
 - maximum of a few MBs
- Two main approaches
 - Fitting analytical functions
 - Statistical data analysis

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting Analytical Functions



- |p| small, evaluation of $\hat{F}(x,p)$ cheap
- Good compression and fast evaluation

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting Analytical Functions

- Spatial variation (texture domain) too complex
- Fixing spatial position

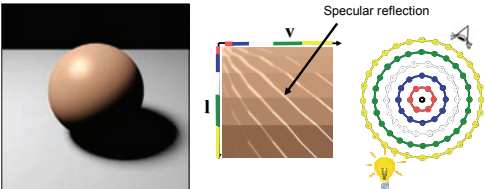
$$B_x(\mathbf{v}, \mathbf{l}) := BTF(\mathbf{x}, \mathbf{v}, \mathbf{l})$$

Apply techniques from BRDF modeling

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting Analytical Functions

- How do these functions look like?
- How do typical BRDFs look like?



shiny plastic

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting Analytical Functions

- How do these functions look like?

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting Analytical Functions

- Are these functions typical BRDFs?

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting Analytical Functions

- ABRDF = Apparent BRDF [Wong et al. 97]
 - Contains also influence from neighborhood:
 - Self-Shadowing
 - Self-Occlusion
 - Sub-Surface Scattering
 - Resampling artefacts
 - ...

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting BRDF-Models

- Generalized Cosine-Lobe Model [Lafortune et al. 97]:

$$\Rightarrow BTF(\mathbf{x}, \mathbf{v}, \mathbf{l}) \approx f_x(\mathbf{v}, \mathbf{l}, \mathbf{p}) = \rho_{d,x} + \sum_{i=1}^k \rho_{s,i,x} \cdot (\mathbf{v}^T \mathbf{D}_{x,i} \mathbf{l})^{n_{x,i}}$$

Labels: diffuse color, specular color, specular lobe

- Non-linear least-squares fitting (Levenberg-Marquardt)
- typically around 2 lobes
- Improvement [Daubert et al. 2001]: view-dependent scale factor per texel to account for shadowing effects

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting BRDF-Models

- Advantages
 - High compression
 - Efficient evaluation
- Problems
 - loss of depth impression
 - Non-linear fitting
 - expensive
 - results depend on initialization

McAllister 2002

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Fitting (Hemi-) Spherical Functions

- Approximate hemispherical slices (fixing e.g. view direction) of per-texel ABRDF separately and blend

$$\Rightarrow BTF(\mathbf{x}, \mathbf{v}, \mathbf{l}) \approx \sum_{\mathbf{v} \in \mathcal{N}(\mathbf{v})} w_{\mathbf{v},\mathbf{v}} HSF_{\mathbf{x},\mathbf{v}}(\mathbf{l})$$


Labels: Hemispherical Function (2D), Interpolation weights

- [Meseth et al. 2004] used polynomials and cosine lobes
- [Sloan et al. 2003] used spherical harmonics (consider also [Masselus et al. EGSR04])

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Summary: Fitting Hemispherical Functions

- Advantages
 - Better preservation of ABRDF features
- Problems
 - Chosen approximation for HSF may introduce artifacts
 - Memory consuming (Apply clustering => quantization artifacts)
 - More expensive evaluation (view-interpolation required)



Meseth et al. 2004

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Statistical Data Analysis

- Motivation
 - Assuming general basis functions (polynomials, lobes, etc...) suboptimal for a given measured BTF-dataset
- Idea
 - Find customized basis functions adapted for the actual data set
 - Exploit the inherent redundancy more effectively

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

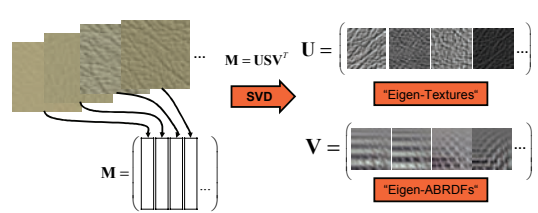
Statistical Data Analysis

- Linear approaches
 - Full BTF-matrix factorization [Koudelka et al. 2003] [Liu et al. 2004]
 - Per-texel ABRDF factorization [Suykens et al. 2003]
 - Per-view factorization [Sattler et al. 2003]
 - Per-cluster factorization [Mueller et al. 2003]
- Tensor approaches
 - TensorTextures [Vasilescu et al. 2004]
 - Out-of-Core Tensor Approximation [Wang et al. 2005]

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Full BTF-Factorization

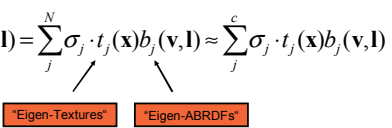
- Stack images as column vectors into large matrix



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Full BTF-Factorization

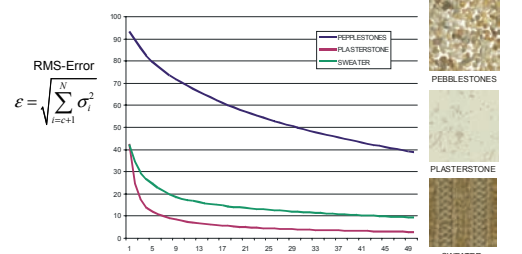
- Write the BTF as sum of products of two functions

$$\Rightarrow BTF(\mathbf{x}, \mathbf{v}, \mathbf{l}) = \sum_j^N \sigma_j \cdot t_j(\mathbf{x}) b_j(\mathbf{v}, \mathbf{l}) \approx \sum_j^c \sigma_j \cdot t_j(\mathbf{x}) b_j(\mathbf{v}, \mathbf{l})$$


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Full BTF-Factorization

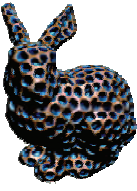
- Number of terms



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Full BTF-Factorization

- Advantages
 - simple and straight-forward
- Problems
 - complex materials require many terms
 - not suitable for real-time reconstruction



Liu et al. TVCG2004

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Per-Texel ABRDF Factorization

- Chained-Matrix Factorization [Suykens et al. 2003]
 - Generalization of common BRDF-Factorization techniques [Kautz&McCool 1999],[McCool et al. 2001]
 - Idea:
 - Apply chain of factorizations (SVD) to reparameterized data
 - Each parameterization accounts for certain ABRDF-features

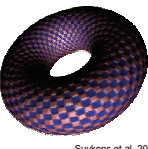
$$BTF(\mathbf{x}, \mathbf{v}, \mathbf{l}) \approx \prod_j \sum_k^{c_j} P_{x,j,k}(\pi_{j,1}(\mathbf{v}, \mathbf{l})) \cdot Q_{x,j,k}(\pi_{j,2}(\mathbf{v}, \mathbf{l}))$$

Different parameterizations for each factor j

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Per-Texel ABRDF Factorization

- Advantages
 - Suitable for real-time rendering: Combination of few factors on GPU
- Problems
 - Resampling artifacts
 - Memory consumption (authors propose clustering of factors ⇒ quantization artifacts)



Suykens et al. 2003

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Per-View Factorization

- Apply SVD to BTF-slices with fixed view direction (Spatially varying Hemispherical Functions)
- Idea
 - Increase quality of low-term factored approximations by factorizing fixed subsets of the data


$$\Rightarrow BTF(\mathbf{x}, \mathbf{v}, \mathbf{l}) \approx \sum_{\mathbf{v} \in N(\mathbf{v})} w_{\mathbf{v}} \sum_j^c r_{\mathbf{v},j}(\mathbf{l}) t_{\mathbf{v},j}(\mathbf{x})$$

"Eigen-Hemispherical-Functions"
"Eigen-Textures" (per view)

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Per-View Factorization

- Advantages
 - Low-term factorization enables high-quality interactive rendering on graphics hardware
- Problems
 - Memory consumption
 - Coherence between different views not exploited

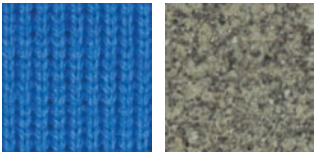


Sattler et al. 2003

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Statistical Data Analysis

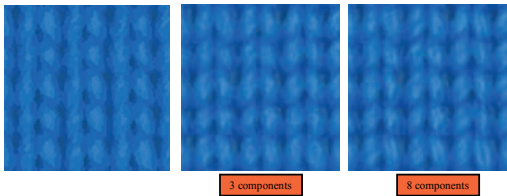
- Per-texel or per-view factorization factorize fixed subsets of the BTF data
- Use clustering across spatial dimension to find better subsets



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Statistical Data Analysis

- Clustering alone leads to quantization artifacts



- Solution: linear approximation of data in each cluster (Local-PCA)

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Per-Cluster Factorization

- Clustering BTF-texels (ABRDFs) leads to

$$\Rightarrow BTF(\mathbf{x}, \mathbf{v}, \mathbf{l}) \approx \sum_j^c t_j(\mathbf{x}) b_{k(\mathbf{x}),j}(\mathbf{v}, \mathbf{l})$$


"Eigen-Textures" (segmented) "Eigen-BRDFs" (per cluster)

- Clustering with generalized Lloyd-algorithm and reconstruction error as distance metric

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Per-Cluster Factorization

- Advantages
 - Low-term factored representation suitable for GPU implementation
 - Good compression
 - Reconstruction per cluster reduces quantization artifacts
- Problems
 - Expensive fitting



Mueller et al. 2003

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Storage Requirements

Model	Storage (L , V , T)	V = L =81, T =256² 8-Bit per channel
Raw BTF	L * V * T	1.2 GB
Analytical BRDF-Model	f(k)* T	(k=2) 2.4 MB
Hemispherical Function	f(k)* V * T	(k=2) 95 MB
BTF Factorization	c*(V * L + T)	(c=40) 8.6 MB
ABRDF Factorization	d*(V + L)* T	(d=2) 63 MB
Per-View Factorization	V *c*(L + T)	(c=4) 64 MB
Per-Cluster Factorization	c*(k*(V * L) + T)	(k=32, c=8) 6.6 MB

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

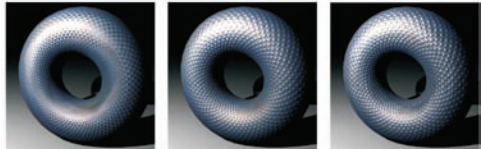
Practical Issues

- Factorization approaches require computing SVD of large matrices (up to several GBs)
- Use incremental/online SVD methods
 - Arnoldi iteration
 - EM-PCA [Roweis 1998]
 - Online SVD [Brand 2003]
 - ...

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Using geometry information


- Fitting local coordinate systems
 - In-between image- and geometry-based BTF representation
- Can be done efficiently using FFT over the group of rotations SO(3) [Müller et al. EG2006]



2 SVD components, 200 KB aligned, 2 SVD components, 200 KB uncompressed 900 MB

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Rendering



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Rendering

- Determine color / visible radiance for every point

„Exitant Radiance = Emitted Rad. + Reflected Rad.“

$$L_r(\mathbf{x}, \mathbf{v}) = L_e(\mathbf{x}, \mathbf{v}) + L_{ref}(\mathbf{x}, \mathbf{v})$$

„Reflected Rad. = Incoming Rad. combined with reflection properties“

$$L_{ref}(\mathbf{x}, \mathbf{v}) = \int_{\Omega_i} \rho_x^*(\mathbf{v}, \mathbf{l}) \cdot L_i(\mathbf{x}, \mathbf{l}) d\mathbf{l}$$

spatially varying reflectance includes foreshortening term

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Point- and Directional Light Sources

- Finite number of light directions

$$L_{ref}(\mathbf{x}, \mathbf{v}) = \int_{\Omega_i} \rho_x^*(\mathbf{v}, \mathbf{l}) \cdot L_i(\mathbf{x}, \mathbf{l}) d\mathbf{l}$$

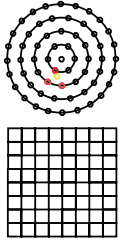
↓

$$L_{ref}(\mathbf{x}, \mathbf{v}) = \sum_{i=1}^n \rho_x^*(\mathbf{v}, \mathbf{l}_i) \cdot \hat{L}_i(\mathbf{x}, \mathbf{l}_i)$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Rendering with GPUs

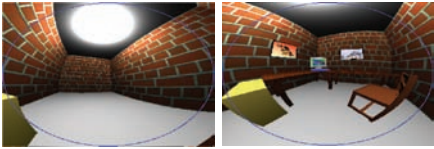
- Measured BTFs
 - Evaluation for directions not in the measured set
 - Interpolation in angular domain
 - Interpolation rather expensive
 - graphics hardware
 - Interpolation from regular samples



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Hardware Supported Angular Interpolation

- Reparameterization
 - Approximately uniform sampling of hemisphere
 - Suitable for hardware filtering
 - Parabolic Maps

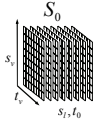


Heidrich + Seidel 1998

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Hardware Supported Angular Interpolation

- 2D Data
 - Bilinear filtering on graphics hardware
- 4D Eigen-ABRDFs
 - Quadrilinear filtering
 - Hardware: trilinear filtering
 - Trilinear filtering of s_v, t_v, s_l
 - 3D textures S_i for fixed t_i
 - Interpolate t_i in fragment shader



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Gero Müller

Anti-Aliasing

- Mip-Mapping compressed BTFs
 - No problem for Eigen-Texture based compression (full-matrix factorization, per-view factorization)
 - Other techniques depend non-linear on compression parameters
 - GPU supported Mip-Mapping not possible
 - ➔ Standard Mip-Mapping on uncompressed data
 - ➔ Compression of each individual Mip-Map level

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Decompression on GPU

- Full-BTF Factorization/Per-Cluster Factorization
 - Store 4D ABRDFs in 3D texture
 - Use 4D interpolation and combine in pixel shader
 - Cluster look-up

$$\text{recon. ABRDF} = \text{mean}^{h_0} + g_1^{h_1} + g_2^{h_2} + g_3^{h_3} + \dots$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Results



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Image-Based Lighting of BTFs



HDR environments wood, beach, kitchen, building and uffizi from www.debevec.org

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Software Rendering

- Global Illumination
 - Decompression on CPU



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Non-Planar Objects






- BTF techniques can be applied to non-planar objects
 - [Furukawa et al. EGRW 2002]
 - [Matusik et al. SIG 2002]
 - [Mueller et al. VAST 2005]
- Use 3D reconstructed base-geometry instead of planar base geometry

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Non-Planar Objects

- Processing steps

- Image acquisition 
- Image-based 3D-reconstruction 
- Mesh parameterization 
- BTF generation 
- BTF compression 

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Non-Planar Objects



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Conclusions

- BTFs capture 6D-slice of the reflectance field of a complex material
- Represents the “look-and feel” of a material
- Several high-quality acquisition setups
- Effective and appearance preserving compression algorithms available
- Real-time rendering possible with point light sources and image-based lighting

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Challenges

- Editing and modeling
 - [Kautz et al. SIG 2007]
 - [Müller et al. EGSR 2007]
- Material Perception
- Time variation (recent work only SVBRDFs)
- Spectral measurements
- Highly reflective materials

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Acknowledgements

- Reinhard Klein, Ralf Sarlette, Dirk Koch, Jan Meseth, Mirko Sattler
- EPOCH NoE

- RealReflect



- University of Bonn



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Gero Müller

Capturing Reflectance
From Theory to Practice

Near-field Reflectance Fields

Hendrik P.A. Lensch
MPI Informatik

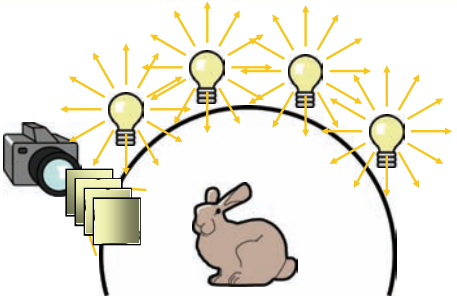
Digitizing Real World Objects



relighting with arbitrary illumination patterns

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

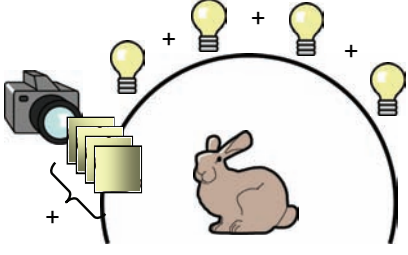
Relighting



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Relighting

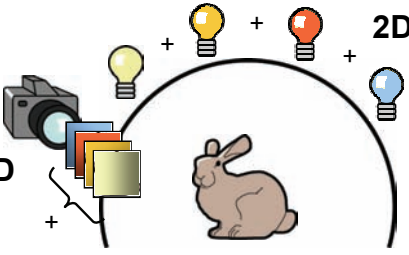
[Debevec2000]



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch


Far-Field Reflectance Fields

[Debevec2000]



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Far and Near Field Illumination



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

6D Reflectance Fields

[Masselus2003]

2D 4D

relighting with 4D incident light fields

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

8D Reflectance Fields

4D 4D

arbitrary view point + arbitrary illumination

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Definition – Reflectance Field

8D function

$$f_r((\vec{x}_i, \vec{\omega}_i) \rightarrow (\vec{x}_o, \vec{\omega}_o))$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Definition – Reflectance Field

ratio of reflected radiance to incident flux

$$f_r((\vec{x}_i, \vec{\omega}_i) \rightarrow (\vec{x}_o, \vec{\omega}_o)) = \frac{dL_o(\vec{x}_o, \vec{\omega}_o)}{d\phi_i(\vec{x}_i, \vec{\omega}_i)}$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Main Problem

- sampling an **8D function**
 - spending 100 samples/dimension → 10^{16} samples
 - hi-res 3D geometry: 10^8 vertices
- coherence in reflectance fields → reduced data complexity
- no complete solution yet

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch


Approaches

- limited reflectance model
- limited reproduction
 - viewer position
 - incident illumination
- adaptive parallel acquisition
- advanced interpolation

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Relighting with 4D Incident Light Fields

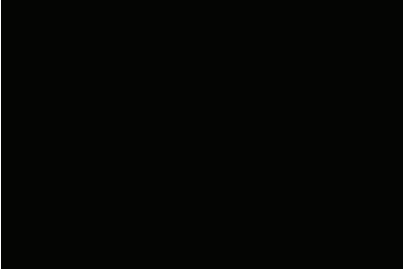
- goal: relighting with spatially varying illumination, e.g. spot lights [Maselus2003]




EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Acquisition with Large Blocks

- fixed camera perspective
- rotating illumination

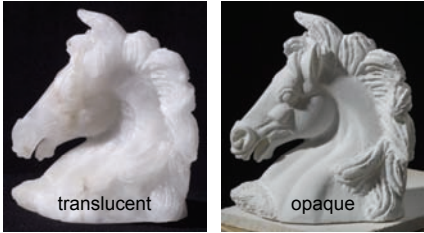


Relighting Results



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Translucent Objects

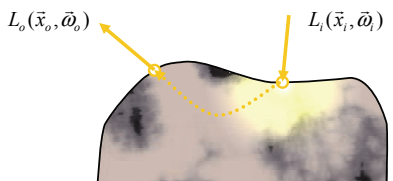


- light transport through the object
- scattering dampens high frequencies

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

BSSRDF – 8D

bidirectional scattering-surface reflectance distribution function [Nicodemus77]

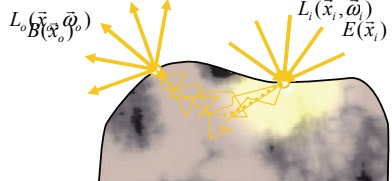
$$f_r((\vec{x}_i, \vec{\omega}_i) \rightarrow (\vec{x}_o, \vec{\omega}_o))$$


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Diffuse Approximation

neglect directional dependency [Jensen 2001]

- multiple scattering leads to diffuse light transport



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

4D - Diffuse Approximation

⇒ diffuse reflectance function $R_d(\vec{x}_i, \vec{x}_o)$

- four dimensions only
- dense sampling is possible

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice
Hendrik Lensch

Diffuse Reflectance Function R_d

- discretize the surface
 - enumerate all surface points
 - vectors for irradiance E and radiosity B
- matrix R_d
 - linear point-to-point transport

$$\begin{bmatrix} B_i \end{bmatrix} = \begin{bmatrix} R_d \end{bmatrix} \begin{bmatrix} E_j \end{bmatrix}$$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice
Hendrik Lensch

Basic Idea

- direct measurement of R_d
 - illuminate individual surface points
 - capture impulse response function

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice
Hendrik Lensch

Basic Idea

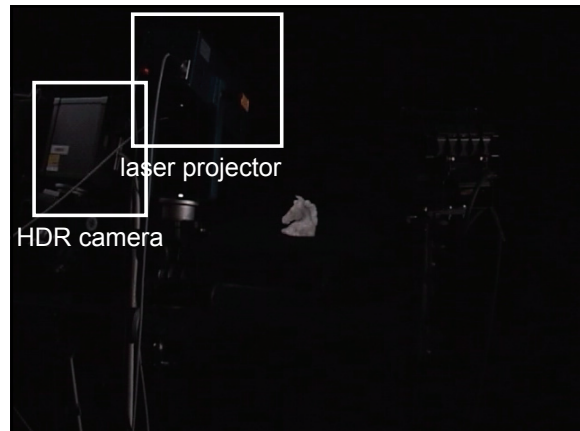
- direct measurement of R_d
 - illuminate individual surface points
 - capture impulse response function

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice
Hendrik Lensch

Basic Idea

- direct measurement of R_d
 - illuminate individual surface points
 - capture impulse response function

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice
Hendrik Lensch





Matrix Representation

- 500.000 – 1.000.000 input images
⇒ $\sim 100.000^2$ entries
- fill up holes (inpainting)
- hierarchical representation
- hardware assisted rendering
 - analysis
 - real-time rendering

[Lensch, Goesele, Bekaert, Magnor, Lang, Seidel – PG2003]

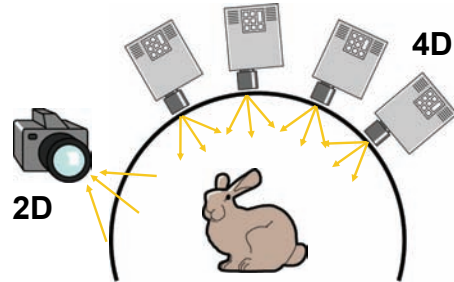
Video

1.000.000 images, 22 hours → model - 800MB



[Goesele, Lensch, Lang, Fuchs, Seidel - SIGGRAPH 2004]

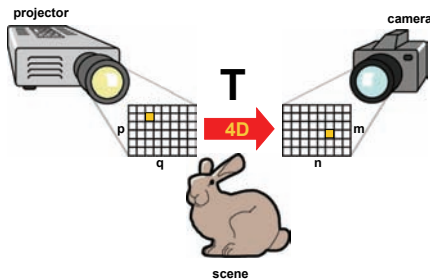
Fixed Perspective + Arbitrary Illumination



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Pixel-to-Pixel Transport

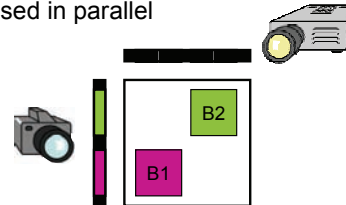


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

Adaptive Parallel Acquisition

- assumption: sparse matrix
- radiometrically independent blocks can be sensed in parallel

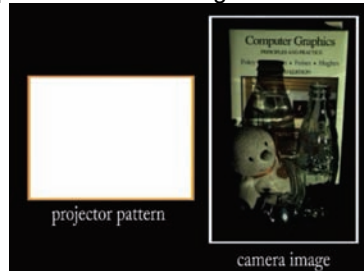


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice

Hendrik Lensch

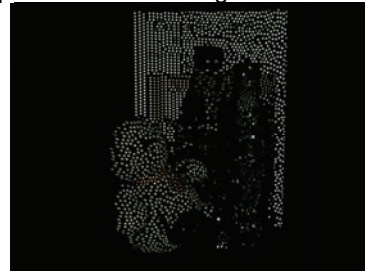
Adaptive Parallel Acquisition

parallelized acquisition of regions which do not overlap in the camera image



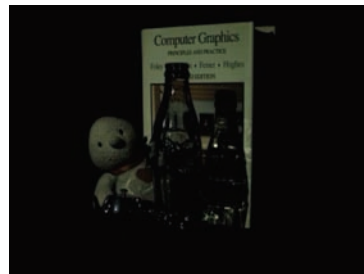
Adaptive Parallel Acquisition

parallelized acquisition of regions which do not overlap in the camera image



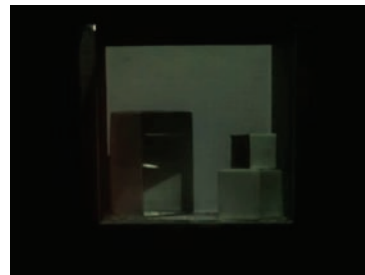
Relighting with Arbitrary Patterns

1.200 images, 2 hours → model - 220MB



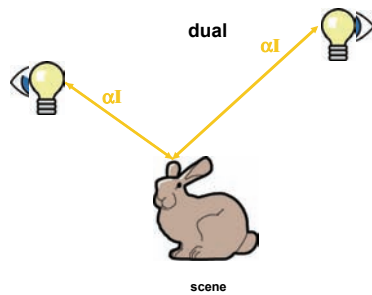
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Captured Global Light Transport



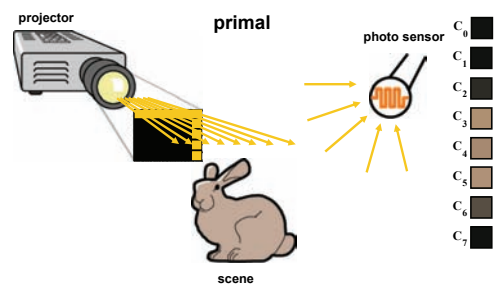
EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Helmholtz Reziprocität

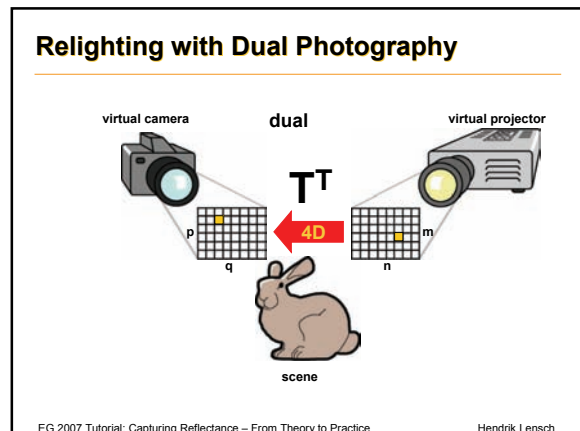
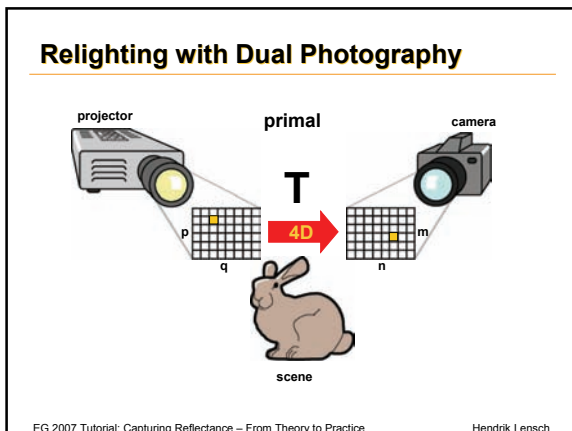
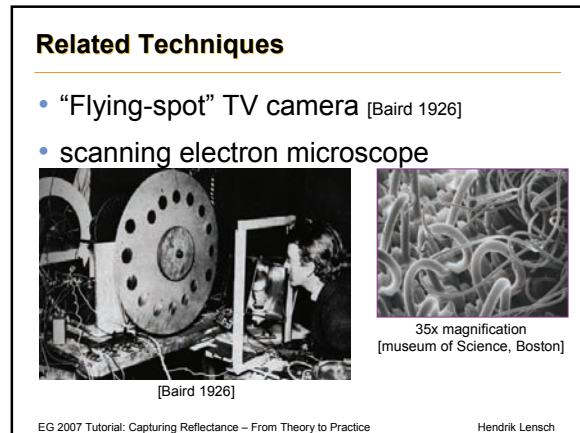
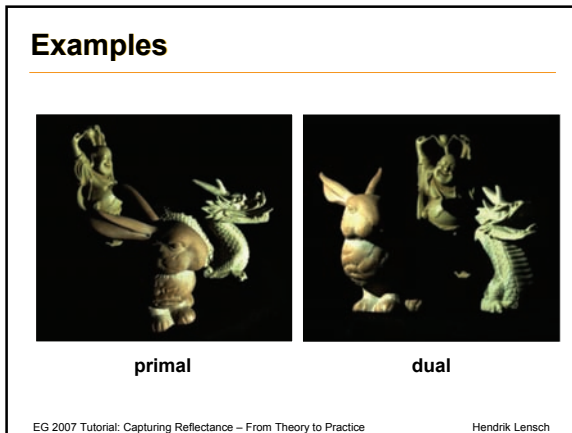
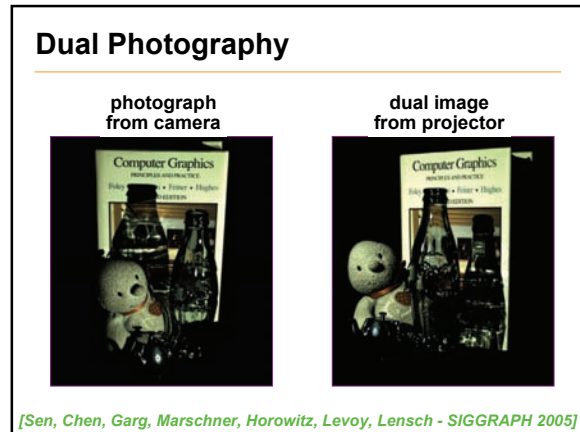
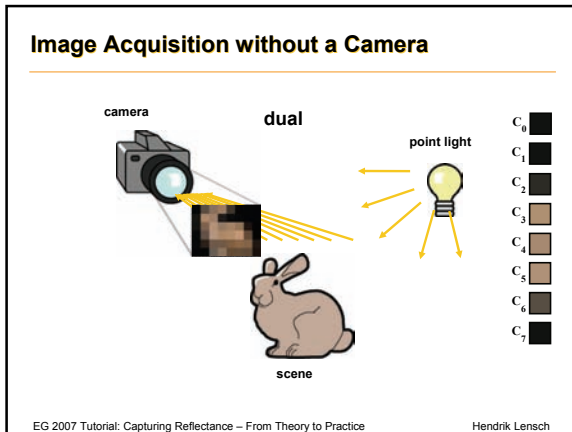


EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Image Acquisition without a Camera



EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch



Acquisition of 6D Reflectance Fields

active devices

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Dual Acquisition Process

parallel acquisition by passive devices

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Smooth Interpolation

100.000 images, 26 hours → model - 4.5GB

[Chen, Lensch - VMV2005]

8D Reflectance Fields

arbitrary view point + arbitrary illumination

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

\mathcal{H} -Matrices

[Hackbusch2000]

efficient representation of dense but **data-sparse** matrices

- subdivision hierarchy
- local low-rank approximation
- efficient evaluation

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Direct vs. Indirect Reflexions

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Direct vs. Indirect Reflexions

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Direct vs. Indirect Reflexions

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

2D Slices through a Reflectance Field

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Symmetric Acquisition

- symmetric 8th order tensor
- rank-1 approximation from two images only
- parallel acquisition of dense matrices

[Garg, Talvala, Levoy, Lensch – EGSR06]

Symmetric Exploration

B3 – row sums
B2 – rows+columns

B3 – column sums
B1 – rows+columns

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Symmetric Exploration

B3 – row sums
B2 – rows+columns

B3 – column sums
B1 – rows+columns

rank-1 approximation? $B3 \approx \begin{bmatrix} | \\ | \\ | \end{bmatrix} \cdot \begin{bmatrix} | \\ | \\ | \end{bmatrix}$

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Hierarchical Rank-1 Decomposition

$$\begin{bmatrix} B1 & R1 \\ R1 & B2 \end{bmatrix} = \begin{bmatrix} & B3 \\ B3^T & \end{bmatrix} + \begin{bmatrix} B1 & \\ & B2 \end{bmatrix} = \dots$$

already determined radiometrically independent

B1 and B2 are investigated in parallel.
 parallel acquisition even for dense matrices

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Dual vs. Symmetric Photography

- increased SNR because regions are determined at large block sizes

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

An 8D Reflectance Field

3.300 images, 6 hours → model – 1.4 GB

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Virtual Photography

- reflectance fields of arbitrarily complex scenes

novel illumination original acquisition pattern

[Garg, Talvala, Levoy, Lensch – EGSR 2006]

Application of Near-field Reflectance Fields

- getting rid of global effects

compare [Nayar2006]

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Application to 3D Scanning

photograph Minolta Vi910 w/o global effects

[Chen, Fuchs, Lensch, Seidel – CVPR 2007]

Card Experiment

primal

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Card Experiment

primal

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Card Experiment

primal

dual

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Near-Field Reflectance Fields

- Sequential Sampling
- Dual Photography
- Symmetric Photography based on \mathcal{H} -matrices
- first methods for acquiring the global light transport in arbitrary scenes

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Challenges

- densely sampled 8D reflectance fields
- upsampling / interpolation
- dynamic near-field reflectance fields
- interactive relighting
- global illumination with reflectance fields
- theory on the complexity of reflectance fields

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Thanks

- BMBF (FKC011MC01)
- DFG – Emmy Noether Program

<http://mpi-inf.mpg.de/~lensch>

EG 2007 Tutorial: Capturing Reflectance – From Theory to Practice Hendrik Lensch

Programming the Cell BE for High Performance Graphics

Eurographics 2007 Tutorial T9

Organizers & Speakers

Michael McCool (RapidMind Inc.)
Bruce D'Amora (IBM T.J. Watson Research Center)

EG:440

Programming the Cell BE for High Performance Graphics

Bruce D'Amora
IBM T.J. Watson Research Center

Michael McCool
RapidMind and University of Waterloo

Eurographics 2007
Tutorial Notes



Cell Broadband Engine *Architecture and Programming Environment*

Bruce D'Amora
Senior Technical Staff Member
Emerging Systems Software
IBM T.J. Watson Research Center
damora@us.ibm.com



IBM T.J. Watson Research Center

Agenda

- Architecture
- Programming Models
- Basic Programming
- Graphics Workloads
- Questions

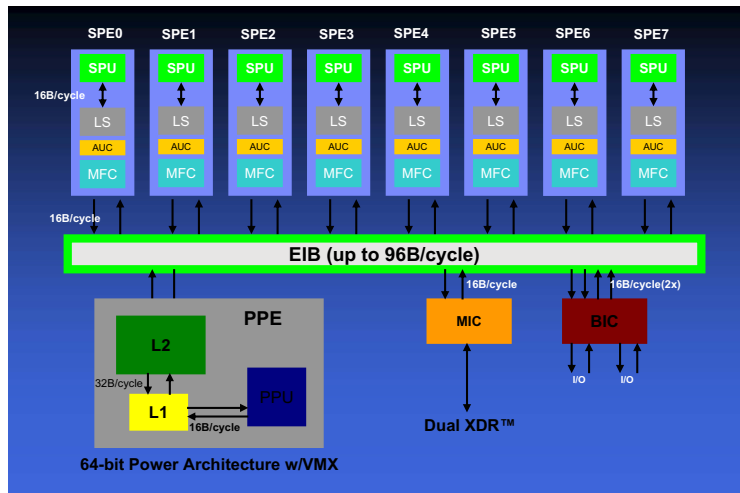
2 | 14 June 2007 | © 2007 IBM Corporation

IBM T.J. Watson Research Center

Architecture

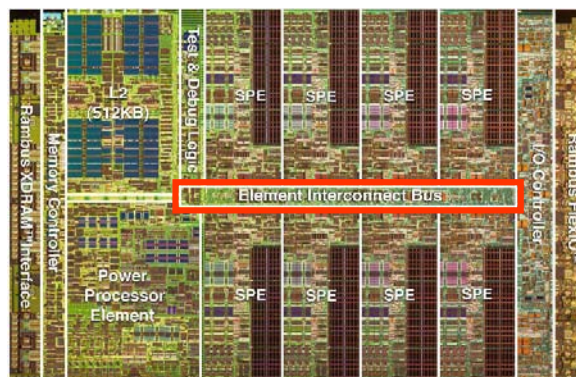
3 | 14 June 2007 | © 2007 IBM Corporation

Cell Broadband Engine Architecture



Element Interconnect Bus

- EIB data ring for internal communication
- Four 16 byte data rings, supporting multiple transfers
- 96B/cycle peak bandwidth
- Over 100 outstanding requests



IBM T.J. Watson Research Center

Power Processor Element

- PPE handles operating system and control tasks
- 64-bit Power Architecture™ with VMX
- In-order, 2-way hardware simultaneous multi-threading (SMT)
- Load/Store with 32KB L1 cache (I & D) and 512KB L2

6 | 14 June 2007 | © 2007 IBM Corporation

IBM T.J. Watson Research Center

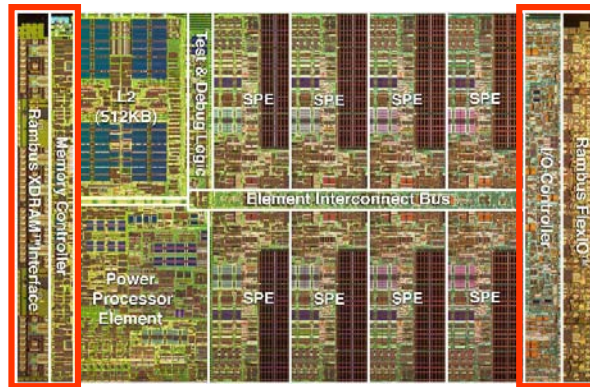
Synergistic Processor Element

- Dual issue, up to 16-way 128-bit SIMD
- Dedicated resources: 128 128-bit register file, 256KB Local Store
- Each can be dynamically configured to protect resources
- Dedicated DMA engine: Up to 16 outstanding requests per SPE

7 | 14 June 2007 | © 2007 IBM Corporation

I/O and Memory Interfaces

- Two configurable interfaces
- Up to 25.6 GB/s memory B/W
- Up to 70+ GB/s I/O B/W
 - Practical ~ 50GB/s

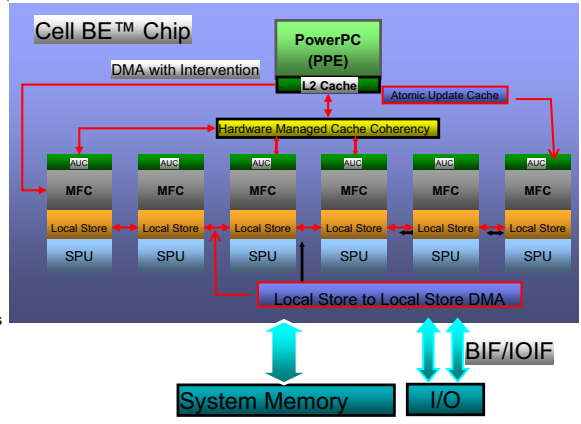


Programming

Cell BE Features Exploited by Software

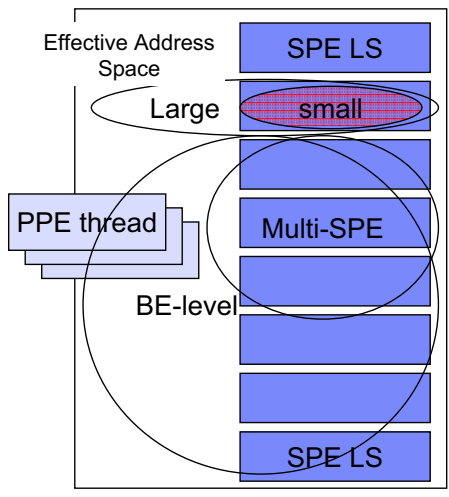
- Large register file
 - Keep intermediate and control data on chip
- DMA Engine – Memory Flow Controller
 - DMA between System Mem and LS
 - DMA from L2 cache-> LS
 - LS to LS DMA
 - Scatter->Gather support
- Atomic Update Cache
- Implement synchronization commands
- SPE Signalling Registers
- SPE <-> PPE Mailboxes

- Resource Reservation and Allocation
 - PPE can be shared across logical partitions
 - SPEs can be assigned to logical partitions
 - SPEs independently or Group Allocated



Common Cell programming models

- Single Cell environment:
- PPE programming models
 - SPE Programming models
 - Small single-SPE models
 - Large single-SPE models
 - Multi-SPE parallel programming models



Small single-SPE models

- **Single tasked environment**
- **Small enough to fit into a 256KB- local store**
- **Sufficient for many dedicated workloads**
- **Two address spaces – (SPE) LS & (SPE/PPE) EA**
- **Explicit input and output of the SPE program**
 - DMA
 - Mailboxes
 - System calls

Small single-SPE models – tools and environment

- **SPE compiler/linker compiles and links an SPE executable**
- **The SPE executable image is embedded as reference-able RO data in the PPE executable**
- **A Cell programmer controls an SPE program via a PPE controlling process and its SPE management library**
 - i.e. loads, initializes, starts/stops an SPE program
- **The PPE controlling process, OS(PPE), and runtime(PPE or SPE) together establish the SPE runtime environment, e.g. argument passing, memory mapping, system call service.**

IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
Tel: 914 938 3333
Fax: 914 938 3333
www.ibm.com
IBM T.J. Watson Research Center

Small single-SPE models – PPE controlling program

```

extern spe_program_handle spe_foo; /* the spe image handle from CESOF */

int main()
{
    int rc, status;
    speid_t spe_id;

    /* load & start the spe_foo program on an allocated spe */
    spe_id = spe_create_thread (0, &spe_foo, 0, NULL, -1, 0);

    /* wait for spe prog. to complete and return final status */
    rc = spe_wait (spe_id, &status, 0);

    return status;
}

```

14
© 2007 IBM Corporation
14 June 2007

IBM T.J. Watson Research Center
1101 Kitchawan Road
Yorktown Heights, NY 10598
Tel: 914 938 3333
Fax: 914 938 3333
www.ibm.com
IBM T.J. Watson Research Center

Small single-SPE models – SPE code

```

/* spe_foo.c: A C program to be compiled into an executable called "spe_foo" */
int main( int speid, addr64 argp, addr64 envp)
{
    char i;

    /* do something intelligent here */
    i = func_foo (argp);

    /* when the syscall is supported */
    printf( "Hello world! my result is %d \n", i);

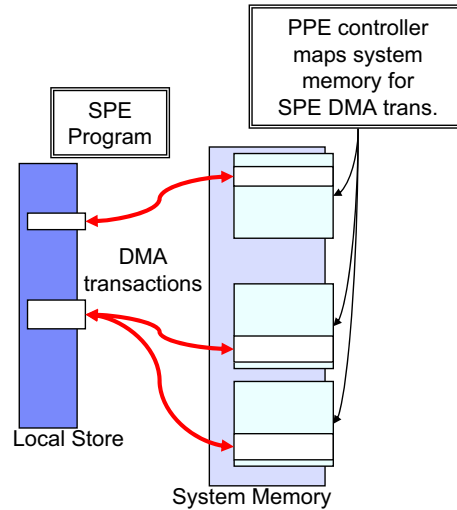
    return i;
}

```

15
© 2007 IBM Corporation
14 June 2007

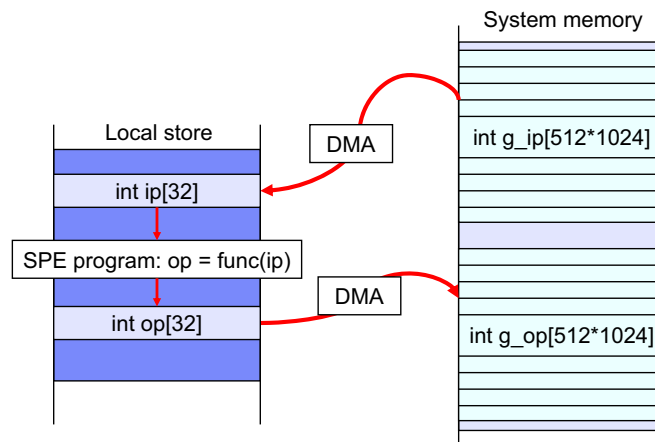
Large single-SPE programming models

- Data or code working set cannot fit completely into a local store
- The PPE controlling process, kernel, and libspe runtime set up the system memory mapping as SPE's secondary memory store
- The SPE program accesses the secondary memory store via its software-controlled SPE DMA engine - Memory Flow Controller (MFC)



Large single-SPE programming models – I/O data

- System memory for large size input / output data
 - e.g. Streaming model



IBM T.J. Watson Research Center

IBM, the IBM logo, and the Watson Research Center logo are trademarks of International Business Machines Corporation. © 2007 IBM Corporation. All rights reserved.

Large single-SPE programming models-SW Cache

- System memory as secondary memory store
 - Manual management of data buffers
 - Automatic software-managed data cache
 - Software cache framework libraries
 - Compiler runtime support

The diagram illustrates the memory architecture. On the left, a box labeled 'SPE program' has an arrow pointing to a 'Local store' structure. The 'Local store' is a vertical stack of memory blocks, with a blue-shaded section at the top containing 'SW cache entries'. On the right, a 'System memory' structure is shown as a vertical stack of horizontal lines, with a light blue-shaded section containing 'Global objects'. Two red arrows originate from the 'Global objects' section of the system memory: one points to the 'SW cache entries' section of the local store, and the other points to the 'SPE program' box, indicating data flow from system memory to the local cache and then to the program.

18
14 June 2007
© 2007 IBM Corporation

IBM T.J. Watson Research Center

IBM, the IBM logo, and the Watson Research Center logo are trademarks of International Business Machines Corporation. © 2007 IBM Corporation. All rights reserved.

Shared-memory Multiprocessor

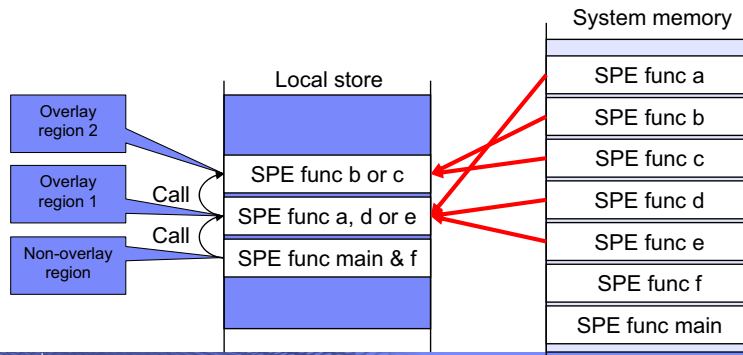
- Cell BE can be programmed as a shared-memory multiprocessor
 - PPE and SPE have different instruction sets and compilers
- SPEs and the PPE fully inter-operate in a cache-coherent model
- Cache-coherent DMA operations for SPEs
 - DMA operations use effective address common to all PPE and SPEs
 - SPE shared-memory **store** instructions are replaced
 - A store from the register file to the LS
 - DMA operation from LS to shared memory
 - SPE shared-memory **load** instructions are replaced
 - DMA operation from shared memory to LS
 - A load from LS to register file
- A compiler could manage part of the LS as a local cache for instructions and data obtained from shared memory.

19
14 June 2007
© 2007 IBM Corporation

Large single-SPE programming models- Overlays

- System memory as secondary memory store
 - Manual loading of plug-in into code buffer
 - Plug-in framework libraries
 - Automatic and manual software-managed code overlay
 - Compiler and Linker generated overlaying code

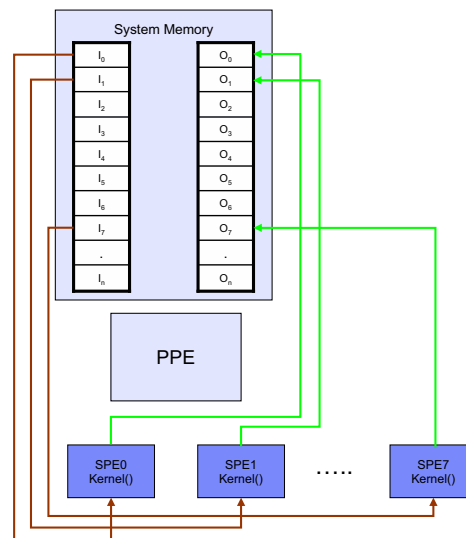
An overlay is SPU code that is dynamically loaded and executed by a running SPU program. It cannot be independently loaded or run on an SPE



Parallel programming models – Streaming

- **SPE initiated DMA**
- **Large array of data fed through a group of SPE programs**
- **A special case of job queue with regular data**
- **Each SPE program locks on the shared job queue to obtain next job**
- **For uneven jobs, workloads are self-balanced among available SPEs**

Data-parallel →



Basic Programming

“Hello World!” – SPE Only

- SPU Program

```

#include <stdio.h>

int main()
{
  printf("Hello world!\n");
  return 0;
}

```

- SPU Makefile

```

PROGRAM_spu := hello_spu
include $(CELL_TOP)/make.footer

```

**PROGRAM_spu tells make
to use SPE compiler**

IBM T.J. Watson Research Center

Synergistic PPE and SPE (SPE Embedded)

- Applications use software constructs called **SPE contexts** to manage and control SPEs.
- Linux schedules SPE contexts from all running applications onto the physical SPE resources in the system for execution according to the scheduling priorities and policies associated with the runnable SPE contexts.
- **libspe** provides API for communication and data transfer between PPE threads and SPEs.

26 | 14 June 2007 | © 2007 IBM Corporation

IBM T.J. Watson Research Center

How a PPE program embeds an SPE program?

4 basic steps must be done by the PPE program

1. Create an SPE context.
2. Load an SPE executable object into the SPE context local store.
3. Run the SPE context. This transfers control to the operating system, which requests the actual scheduling of the context onto a physical SPE in the system.
4. Destroy the SPE context.

27 | 14 June 2007 | © 2007 IBM Corporation

SPE context creation

- **spe_context_create - Create and initialize a new SPE context data structure.**

```
#include <libspe2.h>
spe_context_ptr_t spe_context_create(unsigned int flags,
spe_gang_context_ptr_t gang)
```

- *flags* - A bit-wise OR of modifiers that are applied when the SPE context is created.
- *gang* - Associate the new SPE context with this gang context. If NULL is specified, the new SPE context is not associated with any gang.
- On success, a pointer to the newly created SPE context is returned.

spe_program_load

- **spe_program_load - Load an SPE main program.**

```
#include <libspe2.h>
int spe_program_load (spe_context_ptr_t spe, spe_program_handle_t
*program)
```

- *spe* - A valid pointer to the SPE context for which an SPE program should be loaded.
- *program* - A valid address of a mapped SPE program.

IBM T.J. Watson Research Center

spe_context_run

- **spe_context_run - Request execution of an SPE context.**

```
#include <libspe2.h>
int spe_context_run(spe_context_ptr_t spe, unsigned int *entry, unsigned int runflags, void
*argp, void *envp, spe_stop_info_t *stopinfo)
- spe - A pointer to the SPE context that should be run.
- entry - Input: The entry point, that is, the initial value of the SPU instruction pointer, at which the SPE program should start
executing. If the value of entry is SPE_DEFAULT_ENTRY, the entry point for the SPU main program is obtained from the
loaded SPE image. This is usually the local store address of the initialization function crt0.
- runflags - A bit mask that can be used to request certain specific behavior for the execution of the SPE context. 0 indicates
default behavior.
- argp - An (optional) pointer to application specific data, and is passed as the second parameter to the SPE program,
- envp - An (optional) pointer to environment specific data, and is passed as the third parameter to the SPE program,
- stopinfo An (optional) pointer to a structure of type spe_stop_info_t
```

30
© 2007 IBM Corporation
14 June 2007

IBM T.J. Watson Research Center

spe_context_destroy

- **spe_context_destroy - Destroy the specified SPE context.**

```
#include <libspe2.h>
int spe_context_destroy (spe_context_ptr_t spe)
- spe - Specifies the SPE context to be destroyed
- On success, 0 (zero) is returned, else -1 is returned
```

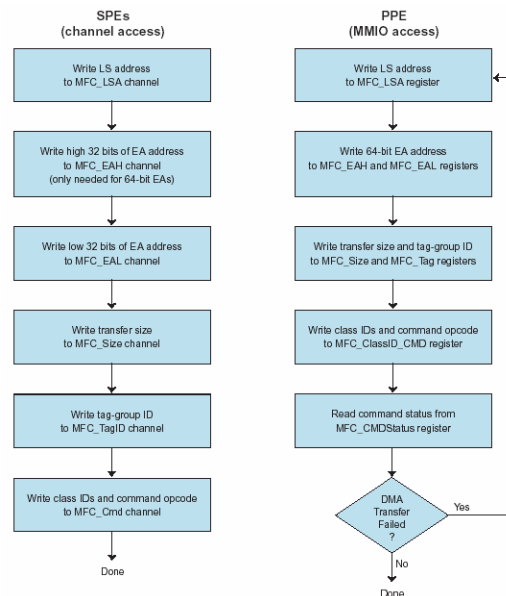
31
© 2007 IBM Corporation
14 June 2007

Memory Flow Controller (MFC) Commands

- **Main mechanism for SPUs to**
 - access main storage (DMA commands)
 - maintain **synchronization** with other processors and devices in the system (Synchronization commands)
- **Can be issued either by SPU via its MFC or by PPE or other device, as follows:**
 - Code running on the SPU issues an MFC command by executing a series of writes and/or reads using **channel instructions** - read channel (rdch), write channel (wrch), and read channel count (rchcnt).
 - Code running on the PPE or other devices issues an MFC command by performing a series of stores and/or loads to **memory-mapped I/O** (MMIO) registers in the MFC
- **MFC commands are queued in one of two independent MFC command queues:**
 - MFC SPU Command Queue — For channel-initiated commands by the associated SPU
 - MFC Proxy Command Queue — For MMIO-initiated commands by the PPE or other device

Sequences for Issuing MFC Commands

- All operations on a given channel are unidirectional
 - only read or write operations for a given channel, not bidirectional
- Accesses to channel-interface resources through MMIO addresses do not stall
- Channel operations are done in program order
- Channel read operations to reserved channels return '0's
- Channel write operations to reserved channels have no effect
- Reading of channel counts on reserved channels returns '0'
- Channel instructions use the 32-bit preferred slot in a 128-bit transfer



IBM T.J. Watson Research Center

DMA Overview

38 | 14 June 2007 | © 2007 IBM Corporation

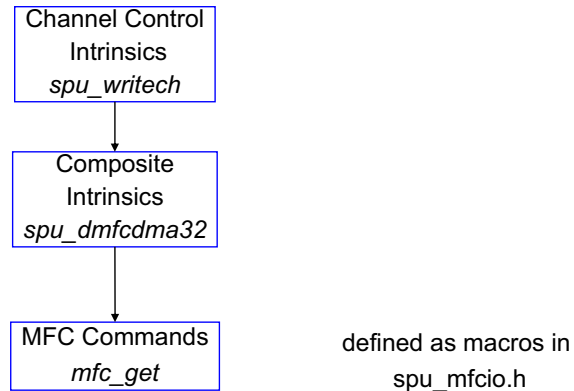
IBM T.J. Watson Research Center

DMA Commands

- MFC commands that transfer data are referred to as DMA commands
- Transfer direction for DMA commands referenced from the SPE
 - Into an SPE (from main storage to local store) → **get**
 - Out of an SPE (from local store to main storage) → **put**

39 | 14 June 2007 | © 2007 IBM Corporation

DMA Commands



For details see: SPU C/C++ Language Extensions

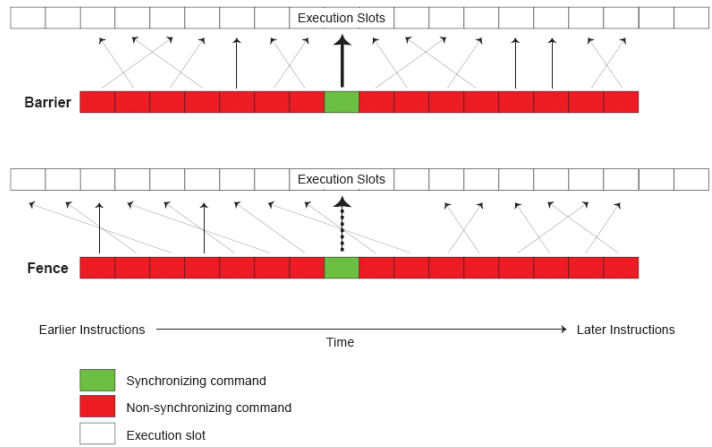
DMA Get and Put Command (SPU)

- DMA get from main memory into local store
(void) mfc_get(volatile void *ls, uint64_t ea, uint32_t size,
uint32_t tag, uint32_t tid, uint32_t rid)
- DMA put into main memory from local store
(void) mfc_put(volatile void *ls, uint64_t ea, uint32_t size,
uint32_t tag, uint32_t tid, uint32_t rid)
- **To ensure order of DMA request execution:**
 - mfc_putf : **fenced** (all commands executed before within the same tag group must finish first, later ones could be before)
 - mfc_putb : **barrier** (the barrier command and all commands issued thereafter are not executed until all previously issued commands in the same tag group have been performed)

DMA-Command Tag Groups

- **5-bit DMA Tag for all DMA commands (except getllar, putllc, and putllc)**
- **Tag can be used to**
 - determine status for entire group or command
 - check or wait on the completion of all queued commands in one or more tag groups
- **Tagging is optional but can be useful when using barriers to control the ordering of MFC commands within a single command queue.**
- **Synchronization of DMA commands within a tag group: fence and barrier**
 - Execution of a fenced command option is delayed until all previously issued commands within the same tag group have been performed.
 - Execution of a barrier command option and all subsequent commands is delayed until all previously issued commands in the same tag group have been performed.

Barriers and Fences



DMA Characteristics

- DMA transfers
 - transfer sizes can be 1, 2, 4, 8, and $n \times 16$ bytes (n integer)
 - maximum is 16KB per DMA transfer
 - 128B alignment is preferable (cache-line)
- DMA command queues per SPU
 - 16-element queue for SPU-initiated requests
 - 8-element queue for PPE-initiated requests
 - **SPU-initiated DMA is always preferable**
- DMA tags
 - each DMA command is tagged with a 5-bit identifier
 - same identifier can be used for multiple commands
 - tags used for polling status or waiting on completion of DMA commands
- DMA lists
 - a single DMA command can cause execution of a list of transfer requests (in LS)
 - lists implement scatter-gather functions
 - a list can contain up to 2K transfer requests

PPE – SPE DMA Transfer

Transfer from PPE (Main Memory) to SPE

- DMA get from main memory
 - `mfc_get(lsaddr, ea, size, tag_id, tid, rid);`
 - lsaddr = target address in SPU local store for fetched data (SPU local address)
 - ea = effective address from which data is fetched (global address)
 - size = transfer size in bytes
 - tag_id = tag-group identifier
 - tid = transfer-class id
 - rid = replacement-class id
- Also available via “composite intrinsic”:
 - `spu_mfcdma64(lsaddr, eahi, ealow, size, tag_id, cmd);`

DMA Command Status (SPE)

- DMA read and write commands are non-blocking
- Tags, tag groups, and tag masks used for:
 - checking status of DMA commands
 - waiting for completion of DMA commands
- Each DMA command has a 5-bit tag
 - commands with same tag value form a “tag group”
- Tag mask is used to identify tag groups for status checks
 - tag mask is a 32-bit word
 - each bit in the tag mask corresponds to a specific tag id:
 - `tag_mask = (1 << tag_id)`

DMA Tag Status (SPE)

- Set tag mask
 - `unsigned int tag_mask;`
 - `mfc_write_tag_mask(tag_mask);`
 - tag mask remains set until changed
- Fetch tag status
 - `unsigned int result;`
 - `result = mfc_read_tag_status(); /* or mfc_stat_tag_status(); */`
 - tag status is logically ANDed with current tag mask
 - tag status bit of '1' indicates that no DMA requests tagged with the specific tag id (corresponding to the status bit location) are still either in progress or in the DMA queue

Waiting for DMA Completion (SPE)

- Wait for any tagged DMA:
 - `mfc_read_tag_status_any()`:
 - wait until **any** of the specified tagged DMA commands is completed
- Wait for all tagged DMA:
 - `mfc_read_tag_status_all()`:
 - wait until **all** of the specified tagged DMA commands are completed
- Specified tagged DMA commands = command specified by current tag mask setting

IBM T.J. Watson Research Center

IBM, the IBM logo, and the "T.J. Watson Research Center" logo are trademarks of International Business Machines Corporation.

DMA Example: Read into Local Store

```

inline void dma_mem_to_ls(unsigned int mem_addr,
    volatile void *ls_addr, unsigned int size)
{
    unsigned int tag = 0;
    unsigned int mask = 1;
    mfc_get(ls_addr, mem_addr, size, tag, 0, 0);
    mfc_write_tag_mask(mask);
    mfc_read_tag_status_all();
}
  
```

Read contents of mem_addr into ls_addr

Set tag mask

Wait for all tag DMA completed

50
© 2007 IBM Corporation

14 June 2007

IBM T.J. Watson Research Center

IBM, the IBM logo, and the "T.J. Watson Research Center" logo are trademarks of International Business Machines Corporation.

DMA Example: Write to Main Memory

```

inline void dma_ls_to_mem(unsigned int mem_addr, volatile
    void *ls_addr, unsigned int size)
{
    unsigned int tag = 0;
    unsigned int mask = 1;
    mfc_put(ls_addr, mem_addr, size, tag, 0, 0);
    mfc_write_tag_mask(mask);
    mfc_read_tag_status_all();
}
  
```

Write contents of mem_addr into ls_addr

Set tag mask

Set tag mask

51
© 2007 IBM Corporation

14 June 2007

SPE – SPE DMA Transfer

SPE – SPE DMA

- Address in the other SPE's local store is represented as a 32-bit effective address (global address)
- SPE issuing the DMA command needs a pointer to the other SPE's local store as a 32-bit effective address (global address)
- PPE code can obtain effective address of an SPE's local store:


```
#include <libspe2.h>
speid_t speid;
void *spe_ls_addr;
..
spe_ls_addr = spe_get_ls(speid);
```
- Effective address of an SPE's local store can then be made available to other SPEs (e.g. via DMA or mailbox)

IBM T.J. Watson Research Center

DMA support for Double Buffering

```

#include <spu_intrinsics.h>
#include "cbe_mfc.h"
#define BUFFER_SIZE 4096
volatile unsigned char B[2][BUFFER_SIZE] __attribute__((aligned(128)));
void double_buffer_example (unsigned int eahi, unsigned int ealow, int buffers)
{
    int next_idx, buf_idx = 0;
    // Initiate first DMA transfer using first buffer
    spu_mfcdma64(B[buf_idx], eahi, ealow, BUFFER_SIZE, buf_idx, MFC_GET_CMD);
    ealow += BUFFER_SIZE;
    while (--buffers) {
        next_idx = buf_idx ^ 1;
        // Initiate next DMA transfer
        spu_mfcdma64(B[next_idx], eahi, ealow, BUFFER_SIZE, next_idx, MFC_GET_CMD);
        ealow += BUFFER_SIZE;
        // Wait for previous transfer to complete
        spu_writch (MFC_WrTagMask, 1 << buf_idx);
        (void) spu_mfcstat(2);
        // Use the data from the previous transfer
        use_data (B[buf_idx]);
        buf_idx = next_idx;
    }
    // Wait for last transfer to complete
    spu_writch (MFC_WrTagMask, 1 << buf_idx);
    (void)spu_mfcstat(2);
    // Use the data from the last transfer
    use_data (B[buf_idx]);
}

```

54
14 June 2007
© 2007 IBM Corporation

IBM T.J. Watson Research Center

Tips to Achieve Peak Bandwidth for DMAs

- The performance of a DMA data transfer is best when the source and destination addresses have the same quadword offsets within a PPE cache line.
- Quadword-offset-aligned data transfers generate full cache-line bus requests for every unrolling, except possibly the first and last unrolling.
- Transfers that start or end in the middle of a cache line transfer a partial cache line (less than 8 quadwords) in the first or last bus request, respectively.

55
14 June 2007
© 2007 IBM Corporation

Mailboxes Overview

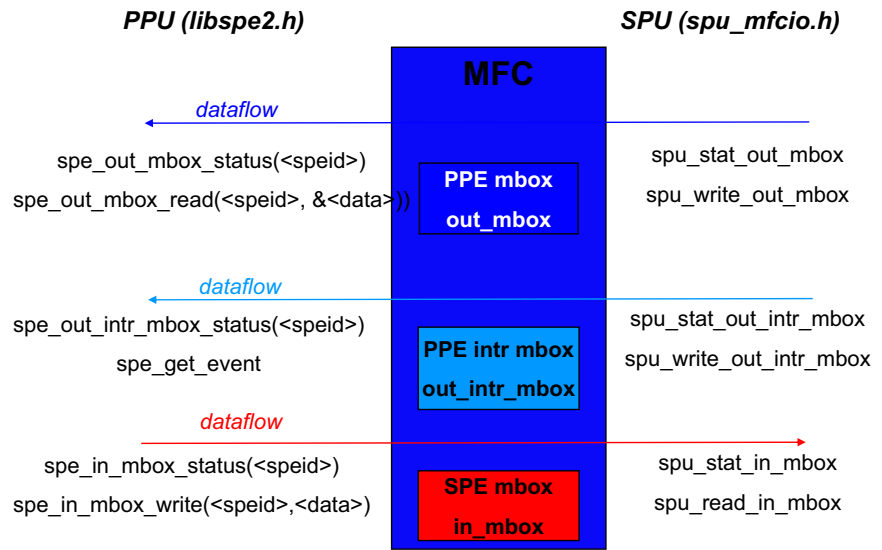
Uses of Mailboxes

- **To communicate messages up to 32 bits in length, such as buffer completion flags or program status**
 - e.g., When the SPE places computational results in main storage via DMA. After requesting the DMA transfer, the SPE waits for the DMA transfer to complete and then writes to an outbound mailbox to notify the PPE that its computation is complete
- **Can be used for any short-data transfer purpose, such as sending of storage addresses, function parameters, command parameters, and state-machine parameters**
- **Can also be used for communication between an SPE and other SPEs, processors, or devices**
 - Privileged software needs to allow one SPE to access the mailbox register in another SPE by mapping the target SPE's problem-state area into the EA space of the source SPE.
 - If software does not allow this, then only atomic operations and signal notifications are available for SPE-to-SPE communication.

Mailboxes - Characteristics

- Each MFC provides three mailbox queues of 32 bit each:
- PPE ("SPU write outbound") mailbox queue
 - SPE writes, PPE reads
 - 1 entry per queue
 - SPE stalls writing to full mailbox
 - PPE ("SPU write outbound") interrupt mailbox queue
 - like PPE mailbox queue, but an interrupt is posted to the PPE when the mailbox is written
 - SPU ("SPU read inbound") mailbox queue
 - PPE writes, SPE reads
 - 4 entries per queue
 - can be overwritten

Mailboxes API – libspe2



SPU Write Outbound Mailboxes

SPU Write Outbound Mailbox

- The value **written** to the SPU Write Outbound Mailbox channel SPU_WrOutMbox is entered into the outbound mailbox in the MFC if the mailbox has capacity to accept the value.
- If the mailbox can **accept** the value, the channel count for SPU_WrOutMbox is **decremented** by '1'.
- If the outbound mailbox is **full**, the channel count will read as '0'.
- If SPE software writes a value to SPU_WrOutMbox when the channel count is '0', the SPU will **stall** on the write.
- The SPU **remains stalled** until the PPE or other device reads a message from the outbound mailbox by reading the MMIO address of the mailbox.
- When the mailbox is **read** through the MMIO address, the channel count is incremented by '1'

SPU Write Outbound Interrupt Mailbox

- The value **written** to the SPU Write Outbound Interrupt Mailbox channel (SPU_WrOutIntrMbox) is entered into the outbound interrupt mailbox if the mailbox has capacity to accept the value.
- If the mailbox can **accept** the message, the channel count for SPU_WrOutIntrMbox is decremented by '1', and an **interrupt is raised** in the PPE or other device, depending on interrupt enabling and routing.
- There is no ordering of the interrupt and previously issued MFC commands.
- If the outbound interrupt mailbox is **full**, the channel count will read as '0'.
- If SPE software writes a value to SPU_WrOutIntrMbox when the channel count is '0', the SPU will **stall** on the write.
- The SPU **remains stalled** until the PPE or other device reads a mailbox message from the outbound interrupt mailbox by reading the MMIO address of the mailbox.
- When this is done, the channel count is **incremented** by '1'.

Waiting to Write SPU Write Outbound Mailbox Data

- To **avoid SPU stall**, SPU can use the read-channel-count instruction on the SPU Write Outbound Mailbox channel to determine if the queue is empty before writing to the channel.
 - If the read-channel-count instruction returns '0', the SPU Write Outbound Mailbox Queue is full.
 - If the read channel-count instruction returns a non-zero value, the value indicates the number of free entries in the SPU Write Outbound Mailbox Queue.
 - When the queue has free entries, the SPU can write to this channel without stalling the SPU.
- Polling SPU Write Outbound Mailbox or SPU Write Outbound Interrupt Mailbox.

```

/* To write the value 1 to the SPU Write Outbound Interrupt Mailbox instead
 * of the SPU Write Outbound Mailbox, simply replace SPU_WrOutMbox
 * with SPU_WrOutIntrMbox in the following example.*/
unsigned int mb_value;
do {
    /* Do other useful work while waiting.*/
} while (!spu_readchcnt(SPU_WrOutMbox)); // 0 → full, so something useful
spu_writetech(SPU_WrOutMbox, mb_value);

```

Polling for or Block on an SPU Write Outbound Mailbox Available Event

```

#define MBOX_AVAILABLE_EVENT 0x00000080
unsigned int event_status;
unsigned int mb_value;
spu_writetech(SPU_WrEventMask, MBOX_AVAILABLE_EVENT);
do {
    /*
     * Do other useful work while waiting.
     */
} while (!spu_readchcnt(SPU_RdEventStat));
event_status = spu_readch(SPU_RdEventStat); /* read status */
spu_writetech(SPU_WrEventAck, MBOX_AVAILABLE_EVENT); /* acknowledge event */
spu_writetech(SPU_WrOutMbox, mb_value); /* send mailbox message */
▪ NOTES: To block, instead of poll, simply delete the do-loop above.

```

PPU reads SPU Outbound Mailboxes

- **PPU must check Mailbox Status Register first**
 - check that unread data is available in the SPU Outbound Mailbox or SPU Outbound Interrupt Mailbox
 - otherwise, stale or undefined data may be returned
- **To determine that unread data is available**
 - PPE reads the Mailbox Status register
 - extracts the count value from the SPU_Out_Mbox_Count field
- **count is**
 - non-zero → at least one unread value is present
 - zero → PPE should not read but poll the Mailbox Status register

IBM T.J. Watson Research Center

SPU Read Inbound Mailbox

66 | 14 June 2007 | © 2007 IBM Corporation

IBM T.J. Watson Research Center

SPU Read Inbound Mailbox Channel

- **Mailbox is FIFO queue**
 - If the SPU Read Inbound Mailbox channel (SPU_RdInMbox) has a message, the value read from the mailbox is the oldest message written to the mailbox.
- **Mailbox Status (empty: channel count =0)**
 - If the inbound mailbox is empty, the SPU_RdInMbox channel count will read as '0'.
- **SPU stalls on reading empty mailbox**
 - If SPE software reads from SPU_RdInMbox when the channel count is '0', the SPU will stall on the read. The SPU remains stalled until the PPE or other device writes a message to the mailbox by writing to the MMIO address of the mailbox.
- **When the mailbox is written through the MMIO address, the channel count is incremented by '1'.**
- **When the mailbox is read by the SPU, the channel count is decremented by '1'.**

67 | 14 June 2007 | © 2007 IBM Corporation

SPU Read Inbound Mailbox Characteristics

- The SPU Read Inbound Mailbox can be **overrun** by a PPE in which case, mailbox message data will be lost.
- A **PPE writing** to the SPU Read Inbound Mailbox will **not stall** when this mailbox is full.

PPE Access to Mailboxes

- PPE can derive "addresses" of mailboxes from spe thread id
- First, create SPU thread, e.g.:


```
speid_t spe_id;
spe_id = spe_create_thread(0,spe_load_image,NULL,NULL,-1,0);
```

 - spe_id has type speid_t (normally an int)
- PPE mailbox calls use `spe_id` to identify desired SPE's mailbox
- Functions are in `libspe.a`

IBM T.J. Watson Research Center
1101 Kitchell Road
Yorktown Heights, NY 10593
Tel: 914 938 3333
Fax: 914 938 3333
www.ibm.com
IBM T.J. Watson Research Center

Read: PPE Mailbox Queue – PPE Calls (libspe.h)

- “SPU outbound” mailbox
- Check mailbox status:
 - `unsigned int count;`
 - `count = spe_stat_out_mbox(spe_id);`
 - count = 0 → no data in the mailbox
 - otherwise, count = number of incoming 32-bit words in the mailbox
- Get mailbox data:
 - `unsigned int data;`
 - `data = spe_read_out_inbox(spe_id);`
 - data contains next 32-bit word from mailbox
 - routine is non-blocking
 - routine returns MFC_ERROR (0xFFFFFFFF) if no data in mailbox

70
© 2007 IBM Corporation
14 June 2007

IBM T.J. Watson Research Center
1101 Kitchell Road
Yorktown Heights, NY 10593
Tel: 914 938 3333
Fax: 914 938 3333
www.ibm.com
IBM T.J. Watson Research Center

Write: PPE Mailbox Queues – SPU Calls (spu_mfcio.h)

- “SPU outbound” mailbox
- Check mailbox status:
 - `unsigned int count;`
 - `count = spu_stat_out_mbox();`
 - count = 0 → mailbox is full
 - otherwise, count = number of available 32-bit entries in the mailbox
- Put mailbox data:
 - `unsigned int data;`
 - `spu_write_out_mbox(data);`
 - data written to mailbox
 - routine blocks if mailbox contains unread data

71
© 2007 IBM Corporation
14 June 2007

PPE Interrupting Mailbox Queue – PPE Calls

- “SPU outbound” interrupting mailbox
- Check mailbox status:
 - `unsigned int count;`
 - `count = spe_stat_out_intr_mbox(spe_id);`
 - count = 0 → no data in the mailbox
 - otherwise, count = number of incoming 32-bit words in the mailbox
- Get mailbox data:
 - interrupting mailbox is a privileged register
 - user PPE applications read mailbox data via `spe_get_event`

PPE Interrupting Mailbox Queues – SPU Calls

- “SPU outbound” interrupting mailbox
- Put mailbox data:
 - `unsigned int data;`
 - `spe_write_out_intr_mbox(data);`
 - data written to interrupting mailbox
 - routine blocks if mailbox contains unread data
- defined in `spu_mfcio.h`

IBM T.J. Watson Research Center

Write: SPU Mailbox Queue – PPE Calls (libspe.h)

- “SPU inbound” mailbox
- Check mailbox status:
 - `unsigned int count;`
 - `count = spe_stat_in_mbox(spe_id);`
 - count = 0 → mailbox is full
 - otherwise, count = number of available 32-bit entries in the mailbox
- Put mailbox data:
 - `unsigned int data, result;`
 - `result = spe_write_in_mbox(spe_id,data);`
 - data written to next 32-bit word in mailbox
 - mailbox can overflow
 - routine returns 0xFFFFFFFF on failure

74
© 2007 IBM Corporation

IBM T.J. Watson Research Center

Read: SPU Mailbox Queue – SPU Calls (spu_mfcio.h)

- “SPU inbound” mailbox
- Check mailbox status:
 - `unsigned int count;`
 - `count = spu_stat_in_mbox();`
 - count = 0 → no data in the mailbox
 - otherwise, count = number of incoming 32-bit words in the mailbox
- Get mailbox data:
 - `unsigned int data;`
 - `data = spu_read_in_mbox();`
 - data contains next 32-bit word from mailbox
 - routine blocks if no data in mailbox

75
© 2007 IBM Corporation

Example using libspe2.x

The PPU program

```
#include <stdio.h>
#include <libspe.h>
#include <libmisc.h>
#include <string.h>
#include <libspe2.h>

//spu program
extern spe_program_handle_t getbuf_spu;
//local buffer
unsigned char buffer[128] __attribute__((aligned(128)));
//spe context
spe_context_ptr_t speid;
unsigned int flags = 0;
unsigned int entry = SPE_DEFAULT_ENTRY;
spe_stop_info_t stop_info;
int rc;
```

```
int main (void)
{
    strcpy (buffer, "Good morning!");
    printf("Original buffer is %s\n", buffer);
    speid = spe_context_create(flags, NULL);
    spe_program_load(speid, &getbuf_spu);
    rc = spe_context_run(speid, &entry, 0, buffer, NULL,
    &stop_info);
    spe_context_destroy(speid);
    printf("New modified buffer is %s\n", buffer);
    return 0;
}
```

```
DIRS = spu
PROGRAM_ppu = getbuf_dma
IMPORTS = -lspe2 -lpthread -lmisc \
        spu/getbuf_spu.a
include $(CELL_TOP)/make.footer
```

IBM T.J. Watson Research Center

IBM, the IBM logo, and the "T.J. Watson Research Center" logo are trademarks of International Business Machines Corporation.

The SPU program

```

#include <stdio.h>
#include <string.h>
#include <libmisc.h>
#include <spu_mfcio.h>
unsigned char buffer[128] __attribute__((aligned(128)));
int main(unsigned long long speid, unsigned long long argp, unsigned long long envp)
{
    int tag = 31, tag_mask = 1<<tag;
    // DMA in buffer from PPE
    mfc_get(buffer, (unsigned long long)argp, 128, tag, 0, 0);
    mfc_write_tag_mask(tag_mask);
    mfc_read_tag_status_any();
    printf("SPE received buffer \"%s\"\n", buffer);
    // modify buffer
    strcpy(buffer, "Good Morning!");
    printf("SPE sent to PPU buffer \"%s\"\n", buffer);
    // DMA out buffer to PPE
    mfc_put(buffer, (unsigned long long)argp, 128, tag, 0, 0);
    mfc_write_tag_mask(tag_mask);
    mfc_read_tag_status_any();
    return 0;
}
    
```

```

PROGRAM_spu      := getbuf_spu
LIBRARY_embed    := getbuf_spu.a
IMPORTS          = -lmisc
include $(CELL_TOP)/make.footer
    
```

78
14 June 2007
© 2007 IBM Corporation

IBM T.J. Watson Research Center

IBM, the IBM logo, and the "T.J. Watson Research Center" logo are trademarks of International Business Machines Corporation.

DMA Example: Read into Local Store

```

void dma_mem_to_ls(unsigned int mem_addr,
                  volatile void *ls_addr, unsigned int size)
{
    unsigned int tag = 0;
    unsigned int mask = 1;
    mfc_get(ls_addr, mem_addr, size, tag, 0, 0);
    mfc_write_tag_mask(mask);
    mfc_read_tag_status_all();
}
    
```

Read contents of mem_addr into ls_addr

Set tag mask

Wait for all tag DMA completed

79
14 June 2007
© 2007 IBM Corporation

Graphics Workloads

Cell Servers for Online Gaming



Motivation

- **Server side physics to enable next generation MMOGs**
- **Current video games perform limited amount of physical simulation**
 - Not enough client CPU resources

Rigid Body Dynamics

- **Objects in the game world are represented by one or more rigid bodies; a sparsely populated world will have about 1000 rigid bodies**
 - 6 degrees of freedom per rigid body
 - Linear position of the body's center of mass and linear velocity are represented by a 3 vector
 - Orientation representation is a unit quaternion
 - Angular velocity is a 3 vector
- **Forces and constraints define interactions between rigid bodies and allow joints, hinges, etc. to be implemented**
- **The physics engine provides real-time simulation of the interaction between the rigid bodies**

Sparse Matrix Data Structures on Cell

- **Matrix is block-sparse with 6x6 blocks**
 - diagonal blocks represent bodies and
 - off-diagonal blocks represent forces between bodies
- **Typical 65-body scene has ~200 nonzero blocks in a 65x65-block matrix**
- **Diagonal elements are assumed nonzero and are stored as a “block” vector for fast access**
- **Off-diagonal elements are stored in linked lists (one per block row) of block data and associated block column position**
- **6x6 float block data is currently stored in column-major form in a padded 8x6 block for ease of access**
- **Vectors used in sparse matrix multiplication are similarly stored with one unused float per three elements**

Numerical Integration

- **Game world is partitioned into non-interacting groups of 1 or more rigid bodies which can be simulated on a single SPU (maximum of about 120 bodies per group).**
- **SPU performs semi-implicit integration step for a second-order rigid body dynamics system using conjugate gradient squared algorithm;**
 - basic operation is multiplication of a 6x6-block-sparse matrix by a vector and multiplication of the matrix transpose by a second vector
- **Output of the integration step gives the change in velocity and angular velocity for each rigid body over one time step**
- **Integration algorithm:**
 1. Calculate the components of A and b. v_0 and W are trivial to extract. f_0 must be calculated. df_dx and df_dv both require considerable computational effort to calculate.
 2. Form A and b.
 3. solve $A \cdot \Delta v = b$ by a conjugate gradient method.
 4. step the system from Y_0 to Y_1 by Δv . This is nearly trivial except that integrating orientation is slightly ugly.

SPU Implementation: Rigid Body Structures

```

struct Rigid_Body {
    //state
    Vec3 position;
    Quaternion or Matrix33 orientation;
    Vec3 velocity;
    Vec3 angular_velocity;
    //mass params
    float inverse_mass;
    Matrix33 inverse_inertia;
    //other params:
    float coefficient_friction;
    float coefficient_damping;
    ...
} bodies[num_bodies];
The output is logically:
struct Rigid_Body_Step {
    Vec3 delta_velocity;
    Vec3 delta_angular_velocity;
} delta_v[num_bodies];

```

The forces can be global, unary, or binary. Here are examples of two common binary forces:

```

struct Point_To_Point_Constraint_Force {
    int index_body_a;
    int index_body_b;
    Vec3 point_body_space_a;
    Vec3 point_body_space_b;
};
struct Contact_Force {
    int index_body_a;
    int index_body_b;
    Vec3 point_world_space;
    Vec3 normal_world_space;
    float penetration;
};

```

IBM T.J. Watson Research Center
1101 Kitchell Road
Yorktown Heights, NY 10593
Tel: 914 937 5300
Fax: 914 937 5300
www.ibm.com
IBM T.J. Watson Research Center

Intermediate data structures

- Vec4 v0[2*num_bodies];
- Vec4 f0[2*num_bodies];
- Six component vectors are padded out to 8 components, with each one float of padding on each of the linear and angular components
 - If the SPE calculations were straightforward dense linear algebra, the padding could be dropped, but due to the sparse matrix block granularity, it is better to have the vector components aligned
- The most complicated data structure is the block sparse matrix:


```

      struct Block_Sparse_Matrix {
      struct Block {
          Matrix86 m;
          int column_index;
          Element* pointer_next;
      };
      Block* rows[NUM_BODIES];
      };
      
```
- The logically 6x6 blocks are padded to 8x8. The matrix is stored in a column major fashion, with padding on the 4th and 8th element to match padding in v0 and f0:


```

      Matrix43 linear_linear_linear_angular;
      Matrix43 angular_linear_angular_angular;
      
```
- Each row has a singly linked list to the elements. The list is maintained to be sorted by increasing column_index, so that find/insert operations can early out (given that there is never an insert without a find, there is no cost to maintaining this sort order):


```

      struct Block_Sparse_Matrix {
      struct Block {
          Matrix86 m;
          int column_index;
          Element* pointer_next;
      };
      Block* rows[NUM_BODIES];
      };
      
```

86
14 June 2007
© 2007 IBM Corporation

IBM T.J. Watson Research Center
1101 Kitchell Road
Yorktown Heights, NY 10593
Tel: 914 937 5300
Fax: 914 937 5300
www.ibm.com
IBM T.J. Watson Research Center

Numerical Integration Steps

Steps 1-4 are performed on the SPE.

1. Calculate the components of **A** and **b**. **v0** and **W** are trivial to extract. **f0** must be calculated. **df_dx** and **df_dv** both require considerable computational effort to calculate.
2. Form **A** and **b**
3. solve **A*delta_v = b** by a conjugate gradient method.
4. step the system from **Y0** to **Y1** by **delta_v**

The steps of the SPE implementation:

1. Initialize **A** and **b** to zero.
2. Construct **A**
 1. By looping over each global, unary, and binary force, and calculating its force contribution and its derivatives, multiplying by the appropriate factors and accumulating into **A** and **b**
 1. Example: for a binary force we accumulate **df_dv + h*df_dx** into **A** and **f0 + h*(df_dx*v0)** is accumulated into **b**
 2. For each binary force (between bodies of index **i** and **j**):
 1. Find/allocate the blocks **(i,i)**, **(j,j)**, **(i,j)** and **(j,i)** of **A**

87
14 June 2007
© 2007 IBM Corporation

Numerical Integration Steps (cont)

1. Calculate the force - the exact calculation of course depends on what type of binary force is required, but generally uses auxiliary force data (such as body space positions) and the two rigid body's kinematic state.
 3. **Calculate the derivatives. The force is logically two 6-vectors (one for each body), and its derivative with respect to a 6-vector body state (position or velocity) is logically a 6x6 matrix. A and b are finalized – this involves the $h^T W$ premultiply.**
 $A = I - h^T W A$
 $b = h^T W b$
 4. **Solve $A=b$ by a conjugate gradient method.**
- Why was conjugate gradient squared chosen?
- The preferred choice is bi-conjugate gradient, but this requires multiplies by A transpose
 - The sparse matrix transpose times vector can be written in a row-oriented fashion, but having the inner 6x6 logical block efficiently support both multiplication with a logical 6-vector and multiplication of its transpose with a logical 6-vector may be more expensive than the alternative – conjugate gradient squared.
 - Caching the transpose of the blocks would likely take too much memory

Conjugate Gradient Squared Method

- The conjugate gradient squared method only requires A times a vector – however, it has been found in practice to converge more slowly.
- Each iteration of the conjugate gradient performs two matrix vector products along with a handful of vector scales, adds, and inner products. The matrix product is the only non-trivial operation. It looks like this:

```
void mul(Vec8* res, const Block_Sparse_Matrix2& A, const Vec8* x)
{
    for (int i = 0; i < num_bodies; ++i) {
        Vec8 sum = 0;
        for (Block* b=A.rows[i]; b = b->pointer_next)
            sum += b->m * x[b->column_index];
        res[i] = sum;
    }
}
```

Where , $b \rightarrow m * x[b \rightarrow \text{column_index}]$ is pseudo code for `Column_Major_Matrix86 times Vec8` which is basically trivial SPE code.

IBM T.J. Watson Research Center
1101 Kitchell Road
Yorktown Heights, NY 10593
Tel: 914 937 5300
Fax: 914 937 5300
www.ibm.com
IBM T.J. Watson Research Center

SPU Sparse Matrix Multiply Code

```

void mul(vf4 d[], const SPU_Sparse_Matrix_Element* const A[], const vf4 x[])
{
    PROFILER(mul);
    int i;
    for (i=0; i < nv/2; ++i) {
        const SPU_Sparse_Matrix_Element* p = A[i];
        vf3 s0 = vf3_zero;
        vf3 s1 = vf3_zero;
        while (p) {
            int j = p->j;
            s0 = spu_add(s0, xform_vf3(&p->a.a[0][0], x[2*j+0]));
            s0 = spu_add(s0, xform_vf3(&p->a.a[0][1], x[2*j+1]));
            s1 = spu_add(s1, xform_vf3(&p->a.a[1][0], x[2*j+0]));
            s1 = spu_add(s1, xform_vf3(&p->a.a[1][1], x[2*j+1]));
            p = p->Pnext;
        }
        d[2*i+0] = s0;
        d[2*i+1] = s1;
    }
}

```

90
© 2007 IBM Corporation
14 June 2007

IBM T.J. Watson Research Center
1101 Kitchell Road
Yorktown Heights, NY 10593
Tel: 914 937 5300
Fax: 914 937 5300
www.ibm.com
IBM T.J. Watson Research Center

Memory constraints and workload size

- The number of matrix blocks required is less than $\text{num_bodies} + 2 * \text{num_binary_forces}$
- A typical 65 rigid body scene had approximately 400 contacts and 200 matrix block elements
- SPU memory usage for integrating this example scene follows:
 - Input:**
 - $\text{num_bodies} * \text{sizeof}(\text{Padded}(\text{Rigid_Body})) = 65 * 160\text{B} = 10400\text{B}$
 - $\text{num_contacts} * \text{sizeof}(\text{Padded}(\text{Contact_Force})) = 400 * 48\text{B} = 19200\text{B}$
 - TOTAL = 29600B**
 - Output:**
 - $\text{num_bodies} * \text{sizeof}(\text{Padded}(\text{Rigid_Body_Step})) = 65 * 32\text{B} = 2080\text{B}$
 - Intermediate:**
 - $\text{num_bodies} * \text{sizeof}(\text{Padded}(\text{W_Element})) = 65 * 64\text{B} = 4160\text{B}$
 - $\text{num_vectors} * \text{num_bodies} * \text{sizeof}(\text{Padded}(\text{VecB})) = 8 * 65 * 32\text{B} = 16640\text{B}$
 - $\text{num_bodies} * \text{sizeof}(\text{Block}^*) = 65 * 4\text{B} = 260\text{B}$
 - $\text{num_blocks} * \text{sizeof}(\text{Padded}(\text{Block})) = 200 * 208\text{B} = 41600\text{B}$
 - TOTAL = 62660B**
- Including double buffering the input and output areas, we use a **total of 126,020B**
- Maximum workload is probably less than 120 bodies
- [Demo](#)

91
© 2007 IBM Corporation
14 June 2007

Ray Tracing: Quaternion Julia Sets on the GPU



- Keenan Crane (University of Illinois) – GPU implementation
- Based on “Ray Tracing Deterministic 3-D Fractals” Computer Graphics, Volume 23, Number 3, July 1989
- “This kind of algorithm is pretty much ideal for the GPU - extremely high arithmetic intensity and almost zero bandwidth usage” – Keenan Crane

Optimal Data Organization: Array of Structures versus Structure of Arrays

(1) Array of Structures

Structure data organization for single triangle

Vertex a	x	y	z	w
Vertex b	x	y	z	w
Vertex c	x	y	z	w

```
typedef struct _Triangle {
    vector float a, b, c
} Triangles;
```

```
Triangles triangles[];
```

- AOS data-packing approach can produce small code sizes, but
 - Typically less than optimal for SIMD architectures
 - Generally requires significant loop-unrolling to improve its efficiency
 - Memory wasted
 - If the vertices contain fewer components than the SIMD vector can hold, e.g., 3 components instead of four

Optimal Data Organization: Array of Structures versus Structure of Arrays

(2) Structure of Arrays for 4 Triangles

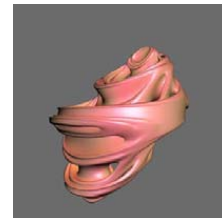
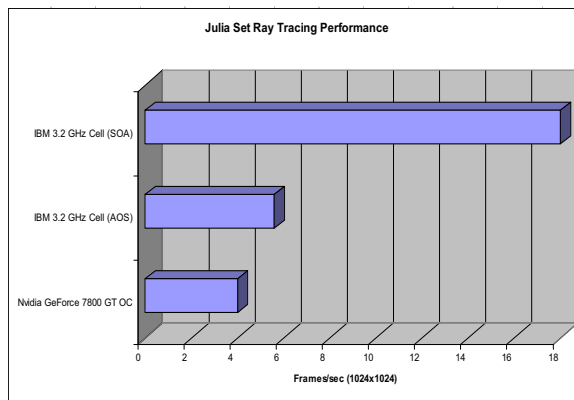
Structure data organization for 4 triangles

a[0]: x1,x2,x3,x4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
a[1]: y1,y2,y3,y4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
a[2]: z1,z2,z3,z4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
b[0]: x1,x2,x3,x4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
b[1]: y1,y2,y3,y4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
b[2]: z1,z2,z3,z4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
c[0]: x1,x2,x3,x4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
c[1]: y1,y2,y3,y4	Triangle 1	Triangle 2	Triangle 3	Triangle 4
c[2]: z1,z2,z3,z4	Triangle 1	Triangle 2	Triangle 3	Triangle 4

```
Struct Triangles {
    Vector float a[3], b[3], c[3];
}
```

- SOA data-packing approach can be more efficient for some algorithms
 - Typically executes well on SIMD architectures
 - Less memory wasted
 - Usually more complex code

Performance



7 SPEs used for rendering + 1 SPE reserved for image compression

Texture Mapping the Julia Set

- Texture references:
 - Difficult to set up (*predict*) DMAs in advance
 - Significant spatial & temporal locality
 - Small working set size (16-32 kb)
- Texture memory organization
 - Consistency with framebuffer rendering order
 - Tiled framebuffer memory → Tiled texture memory
- Cache layout organization
 - Use cache line size == texture tile size

* Findings from *The Design and Analysis of a Cache Architecture for Texture Mapping*, Ziyad S. Hakura, and Anoop Gupta [Stanford, 1997]

High Level API's

- Simplify programming
 - Hide details of DMA
- Common Operations
 - Cached data read, write
 - Pre-touch
 - Flush
 - Invalidate
 - etc.

```
#include <spe_cache.h>
#define LOAD1(addr)      \
  *((char *) spe_cache_rd(addr))
#define STORE1(addr, c) \
  *((char *) spe_cache_w(addr)) = c

void memcpy_ea(uint dst, uint src, uint size)
{
  while (size > 0) {
    char c = LOAD1(src);
    STORE1(dst, c);
    size--;
    src++;
    dst++;
  }
}
```

Low level Cache API

- Depend on cache type
- Programmer directly controls
 - Look up
 - Branch to miss handler
 - Wait for DMA completion
- Custom interfaces
 - Multiple lookups
 - Special data types
 - Cache locking

```
#include <spe_cache.h>
unsigned int __spe_cache_rd(unsigned int ea) {
    unsigned int ea_aligned = (ea) & ~SPE_CACHELINE_MASK;
    int set, line, byte, missing;
    unsigned int ret;

    missing = _spe_cache_dmap_lookup(ea_aligned, set);
    line = _spe_cacheline_num(set);
    byte = _spe_cacheline_byte_offset(ea);
    ret = *((unsigned int *) &spe_cache_mem[line + byte]);
    if (unlikely(missing)) {
        _spe_cache_miss(ea_aligned, set, 0, 1);
        spu_writect(22, SPE_CACHE_SET_TAGMASK(set));
        spu_mfcstat(MFC_TAG_UPDATE_ALL);
        ret = *((unsigned int *) &spe_cache_mem[line + byte]);
    }
    return ret;
}
```

Example: SPE Texture Mapping

- *Texturing maps images onto 3-D surfaces*
- *Cube environment mapping reflects image data from 1 of 6 surrounding texture maps*
- *Fresnel reflection & refraction increase realism, complexity of texture look up*
- [Animated 3-D Julia Set Fractal](#)



Interactive Ray-tracing

Renewed interest from Graphics Community

- Global Illumination
- Rendering time scales sub linearly with scene complexity
- Scales well on multi-core processors
- Mathematically elegant
- Algorithmically simple



Courtesy of Barry Minor, IBM Quasar Design Center

IBM iRT Interactive Ray-tracer

- Visualization of Huge Digital Models
- Powered by IBM QS20 Blades
- 720p and 1080p HDTV Output
- Seamless Scale Out
 - More Blades
 - More Cells
 - More performance
- Real-time Ambient Occlusion
- Server Side Rendering
 - Image Encode
 - IB or Network Image Delivery
- Dynamic Load Balancing
 - Across Multiple Blades, Cells, & SPEs

Courtesy of Barry Minor, IBM



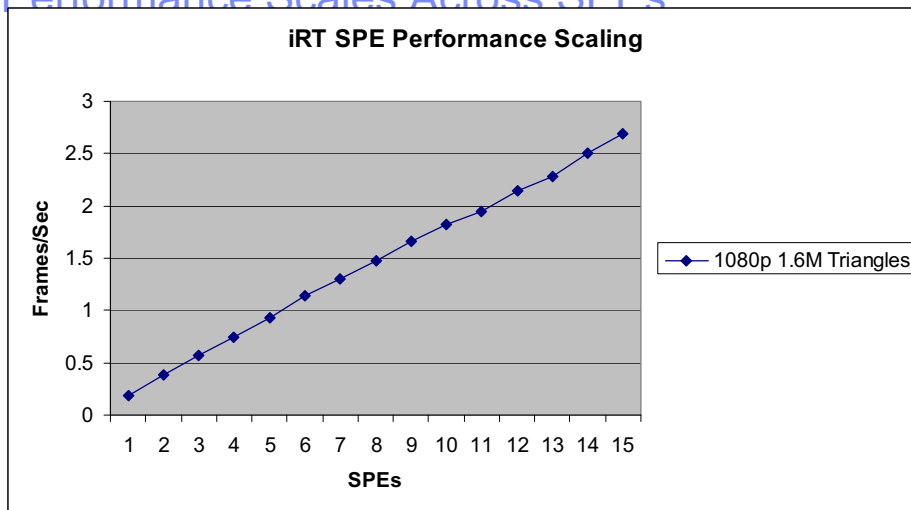
Courtesy of Barry Minor, IBM



IBM iRT Supported Rendering Features

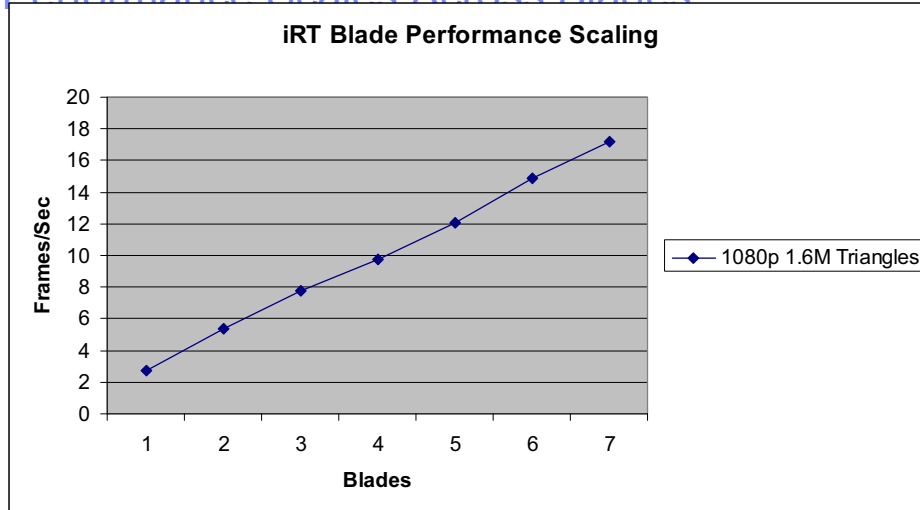
- **Texture Maps**
 - Bilinear Filtering
- **Bump Maps**
 - Blinn Style
- **Phong Lighting Model**
 - Phong Shading
- **Multi-Sampling**
 - 1, 4, 16 Samples per Pixel
 - Jitter Sampled
- **Ambient Occlusion**
 - 4, 16, 64 Random Samples per Primary
- **Optical Effects**
 - Reflection, Refraction

Performance Scales Across SPEs



QS20 Blades, FC5, Cell SDK 2.0

Performance Scales Across Blades



QS20 Blades, FC5, Cell SDK 2.0

Ray Tracing + Ambient Occlusion



Primary, Shadow, Secondary, Global illumination – 288 Rays per Pixel

IBM T.J. Watson Research Center
1000 Talbot Ave.
Yorktown Heights, NY 10598
Tel: 914.939.3333
Fax: 914.939.3333
www.ibm.com
IBM T.J. Watson Research Center

Ray-Triangle Intersection

```

static inline int isect_ray4_triangle (const struct ray4 *ray,
                                     const float4 p[3], hit_rec4 * hit, uint id)
{
    vec_uint4 vid = spu_splats (id);
    vec_float4 p0 = p[0].v;
    vec_float4 p1 = p[1].v;
    vec_float4 p2 = p[2].v;
    vec_float4 ro_x = ray->o.x;
    vec_float4 ro_y = ray->o.y;
    vec_float4 ro_z = ray->o.z;
    vec_float4 rd_x = ray->d.x;
    vec_float4 rd_y = ray->d.y;
    vec_float4 rd_z = ray->d.z;
    vec_float4 edge1 = spu_sub (p1, p0);
    vec_float4 edge2 = spu_sub (p2, p0);
    vec_float4 hit_t = hi->t;
    vec_float4 hit_u = hi->u;
    vec_float4 hit_v = hi->v;
    vec_uint4 hit_id = hi->id;
    vec_float4 one = spu_splats (1.0f);
    vec_float4 zero = spu_splats (0.0f);
    vec_float4 p0_x = spu_splats (spu_extract (p0, 0));
    vec_float4 p0_y = spu_splats (spu_extract (p0, 1));
    vec_float4 p0_z = spu_splats (spu_extract (p0, 2));
    vec_float4 edge1_x = spu_splats (spu_extract (edge1, 0));
    vec_float4 edge1_y = spu_splats (spu_extract (edge1, 1));
    vec_float4 edge1_z = spu_splats (spu_extract (edge1, 2));
    vec_float4 edge2_x = spu_splats (spu_extract (edge2, 0));
    vec_float4 edge2_y = spu_splats (spu_extract (edge2, 1));
    vec_float4 edge2_z = spu_splats (spu_extract (edge2, 2));
            
```

```

vec_float4 pvec_x, pvec_y, pvec_z;
vec_float4 tvec_x, tvec_y, tvec_z;
vec_float4 qvec_x, qvec_y, qvec_z;
vec_float4 u, v, t;
vec_float4 det, inv_det;
vec_uint4 u_geq_0, v_geq_0;
vec_uint4 uv_leq_1, t_lt_hit;
vec_uint4 t_geq_0, valid_hit;

_CROSS3_V (pvec, rd, edge2);
det = DOT3_V (edge1, pvec);
INVERSE (inv_det, det);
SUB3_V (tvec, ro, p0);
_CROSS3_V (qvec, tvec, edge1);
u = spu_mul (_DOT3_V (tvec, pvec), inv_det);
v = spu_mul (_DOT3_V (rd, qvec), inv_det);
t = spu_mul (_DOT3_V (edge2, qvec), inv_det);
u_geq_0 = spu_cmpge (u, zero);
v_geq_0 = spu_cmpge (v, zero);
uv_leq_1 = spu_cmple (spu_add (u, v), one);
t_lt_hit = spu_cmplt (t, hit_t);
t_geq_0 = spu_cmpge (t, zero);
valid_hit = spu_and (spu_and (spu_and (u_geq_0, v_geq_0),
                                     spu_and (uv_leq_1, t_lt_hit)), t_geq_0);

hit->t = spu_sel (hit_t, t, valid_hit);
hit->u = spu_sel (hit_u, u, valid_hit);
hit->v = spu_sel (hit_v, v, valid_hit);
hit->id = spu_sel (hit_id, vid, valid_hit);

return _any4 (valid_hit) ? 1 : 0;
}
            
```

106
14 June 2007
© 2007 IBM Corporation

IBM T.J. Watson Research Center
1000 Talbot Ave.
Yorktown Heights, NY 10598
Tel: 914.939.3333
Fax: 914.939.3333
www.ibm.com
IBM T.J. Watson Research Center

Demos



City



Lamborghini

107
14 June 2007
© 2007 IBM Corporation

IBM T.J. Watson Research Center

Thank you

108 14 June 2007 © 2007 IBM Corporation

IBM T.J. Watson Research Center

Questions?

109 14 June 2007 © 2007 IBM Corporation

IBM T.J. Watson Research Center
563 Route 9W
Yorktown Heights, NY 10598
Tel: 914.938.3333
Fax: 914.938.3333
www.ibm.comIBM T.J. Watson Research Center

(c) Copyright International Business Machines Corporation 2005.
All Rights Reserved. Printed in the United States April 2005.

The following are trademarks of International Business Machines Corporation in the United States, or other countries, or both.
IBM IBM Logo Power Architecture

Other company, product and service names may be trademarks or service marks of others.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in applications such as implantation, life support, or other hazardous uses where malfunction could result in death, bodily injury, or catastrophic property damage. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.


While the information contained herein is believed to be accurate, such information is preliminary, and should not be relied upon for accuracy or completeness, and no representations or warranties of accuracy or completeness are made.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page is <http://www.ibm.com>
The IBM Microelectronics Division home page is
<http://www.chips.ibm.com>

110© 2007 IBM Corporation14 June 2007



The RapidMind Development Platform

Michael McCool



RAPIDMIND

Outline

- **Basic Concepts**
 - Background
 - Performance
 - Architecture
 - Basic vocabulary
 - Defining program objects
 - Parallel programming model
 - Loop conversion example
- **Advanced Topics**
 - Accessors and copying semantics
 - Applications of dynamic code generation
 - Design patterns
 - Acceleration strategies
 - Program manipulation
- **Application Examples**
 - Crowd simulation, FFT and convolution, raytracing



RAPIDMIND

- **RapidMind Development Platform**
 - Single-source solution for portable parallel programming
 - Safe and deterministic data-parallel programming model
 - Scalable to arbitrary number of cores
 - Integrates with existing C++ compilers
- **Can be used for programming multiple targets**
 - *Unified programming model for both accelerators and CPUs*
 - Support for both GPUs and Cell BE generally available
 - Prototype backend demonstrated on multi-core CPU



- Programmability
 - Just an ISO standard C++ library
 - No new tools or workflow
 - No need for low-level understanding of the processor(s)
 - Expressive, **safe**, modular, and easy to learn
- Performance
 - Leverages all available computational resources
 - Encourages and supports scalable data parallelism
- Portability
 - Application programming independent of OS or target platform
 - New processors supported without change to application



- Use **existing** ISO standard C++ compiler:
 - Just include a header file, link to a library
 - Single-source solution, can be used with existing code bases
 - Does not require modification of debugging and build environments
- Allows specification of **arbitrary computation**:
 - **NOT** just a library of canned functions
 - Uses its own runtime optimizing code generator
 - User can specify arbitrary computational kernels
 - Staged compilation strategy avoids overhead of C++



RAPIDMIND

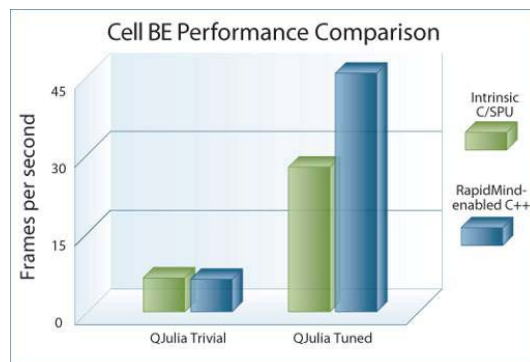
Portability

- Multiple hardware targets:
 - NVIDIA GPUs
 - AMD/ATI GPUs
 - Cell BE
 - Prototype for x86 multi-core demonstrated
- Independent of number of cores
- Independent of memory model
 - Shared or distributed
- If main processor does not change, can support new co-processor *without* even recompiling program



RAPIDMIND

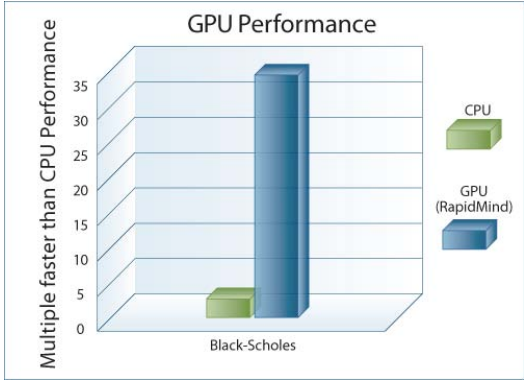
Cell BE Performance



- QJulia application
- Compared with IBM SDK implementation
- Comparable performance with same optimizations
- Additional optimizations possible with only a few lines of code that nearly doubled performance over IBM implementation
- Overall code size and complexity significantly lower than that of IBM SDK implementation



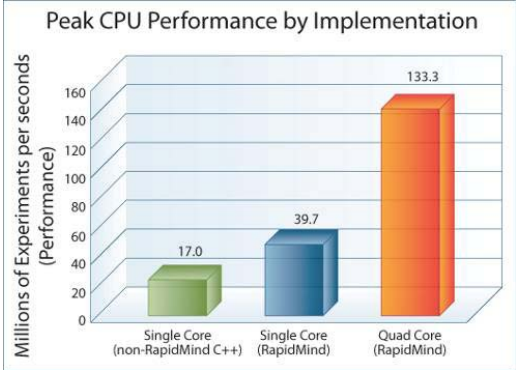
RAPIDMIND GPU Performance



- Financial quasi Monte-Carlo option-pricing benchmark done in “competition” with HP
- CPU code independently tuned by HP
- GPU implementation over 30x faster than single-core CPU implementation



RAPIDMIND CPU Performance



- Same financial quasi Monte-Carlo option-pricing benchmark as for GPU benchmark
- RapidMind implementation basically the same as the GPU implementation
- Prototype backend targeting four CPU cores
- RapidMind over 2x faster on one core, 8x faster on four cores



RAPIDMIND

Key Concepts

- **Vocabulary** for parallel programming
 - Set of nouns (types) and verbs (operations)
 - *Added* to existing standard language: ISO C++
- A *language* implemented as an *API*



RAPIDMIND

API == Language

- **API**
 - Issue a sequence of function calls
 - Manipulate state
 - Must issue calls in a certain order
 - Store sequences of calls in buffers (display lists)
 - Play back sequences of calls
- **Languages**
 - Issue a sequence of statements
 - Manipulate variables
 - Must have a certain syntax
 - Encapsulate sequences of statements in functions
 - Call functions to execute code

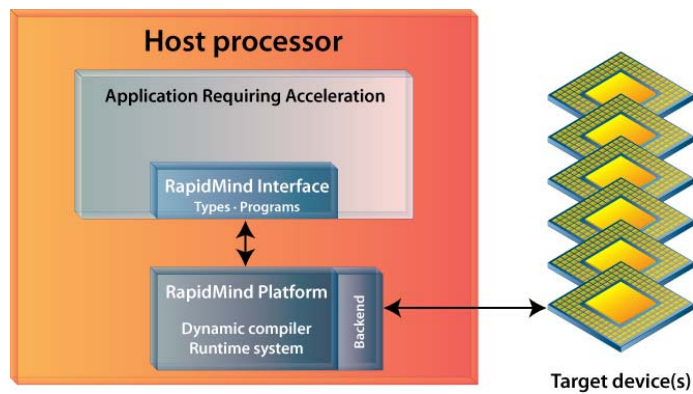


RAPIDMIND RapidMind Platform Interface

- A C++ API
 - for specifying data-parallel computation
- A data-parallel programming language
 - embedded inside C++



RAPIDMIND RapidMind Platform Architecture





RAPIDMIND

RapidMind Interface

Simple API:

- **Data Types:** Arrays and Values
- **Program Objects:** similar to C++ functions
- **Operations:** C++ and matrix-vector library
- **Collectives:** reductions, scatter, gather, etc.

To use:

- `#include <rapidmind/platform.hpp>`
- `using namespace rapidmind;`
- link to `rmplatform`

14



RAPIDMIND

Nouns: Basic Types

Purpose	Type
Container for fixed-length data	Value
Container for variable-sized multidimensional data	Array
Container for computations	Program



```
1 half  
2 double  
Value<3, float>  
4 int
```

Tuple
size

Element
type



```
1h  
2d  
Value3f  
4i
```

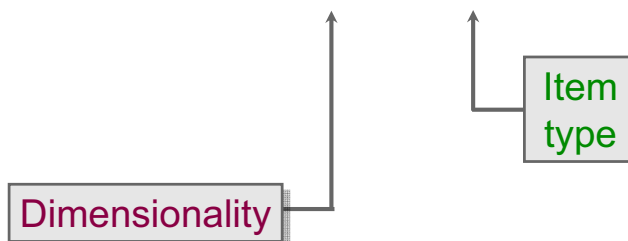
Tuple
size

Element
type



```

1 Value4d
Array<2, Value3f>
3 Value2i
  
```



- Operators act componentwise:
`+, -, *, /, %, &, |, ^, ~, <, ...`
- Swizzling and writemasking:
`Value4f c;`
`c(2,1,0)`
`c(0,0,0)`
`c(1,1,2,3)`
`c[3]`



RAPIDMIND

Verbs: Functions

- Can declare functions in the usual way:

```
Value3f
reflect (Value3f v, Value3f n) {
    return Value3f(2.0*dot(n,v)*n - v);
}
```

- Standard library

- Matrix operations
- Geometric operations
- Trigonometry
- Exponentials and logarithms
- Splines, interpolation, and polynomials
- etc.

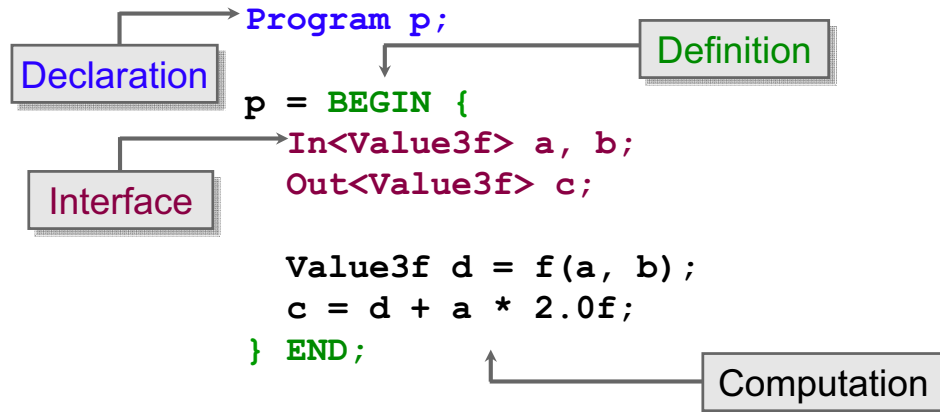


RAPIDMIND

Programs

- **Immediate mode:**
 - **Execute** operations on RapidMind types on host
 - Acts like a standard matrix-vector library
- **Retained mode:**
 - Enter retained mode with **BEGIN**, exit with **END**
 - **Record** operations on RapidMind types
 - Same operations that work in immediate mode
 - Store operations in **Program** object
 - Compile captured operations for coprocessor
 - Dynamic compilation

Dynamic construction of remote procedure call



- Apply programs to arrays, get new arrays

C = p(A,B) ;

Invokes parallel execution



RAPID MIND

Array Semantics

- Arrays use by-value semantics
 - Can assign arrays with $O(1)$ cost
 - Strong modularity
 - Simple and easy to understand
 - Consistent with value tuples
- Most data copies can be optimized away
 - Copies only required to complete partial updates
 - Parallel assignment means partial updates can be avoided
- By-reference semantics available:
 - Via the **ArrayAccessor** type



RAPID MIND

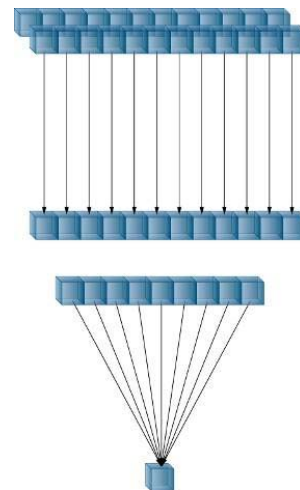
SPMD Data Parallel Programming Model

Apply functions to arrays:

- Application: $C = f(A, B)$
- May have control flow (SPMD model)
- May perform random reads from other arrays
- Can read and write to subarrays

Apply collective operations to arrays:

- Reduce: $a = \text{reduce}(p, A)$
- Gather: $A = B[U]$
- Scatter: $A[U] = B$
- Others...





RAPID MIND

Control Flow

Program p;

```

p = BEGIN {
  In<Value3f> a, b;
  Out<Value3f> c;

  Value3f d = f(a, b);
  IF (all(a > 0.0f)) {
    c = d + a * 2.0f;
  } ELSE {
    c = d - a * 2.0f;
  } ENDF;
} END;

```



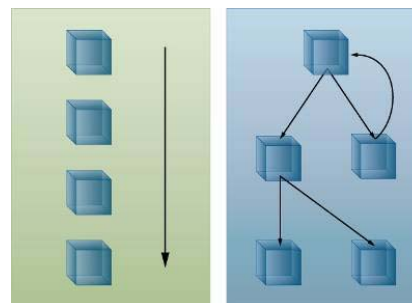
RAPID MIND

Control Flow:
SPMD vs. SIMD**SIMD:**

- *Single Instruction, Multiple Data*
- Kernels include sequences of simple instructions
- Take constant amount of time to execute

SPMD:

- *Single Program, Multiple Data*
- Kernels may include control flow (loops and conditionals)
- Can avoid unnecessary work



SIMD

SPMD

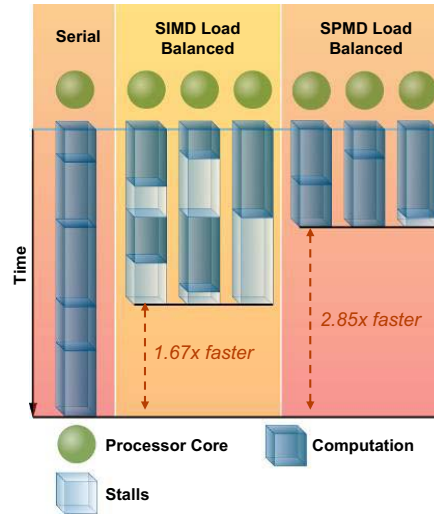
**SPMD includes but is
intrinsically more
powerful than SIMD**

SIMD scheduling

- Assumes constant time per kernel

SPMD scheduling

- Takes variable execution time into account
- Load balancing distributes workload evenly across cores



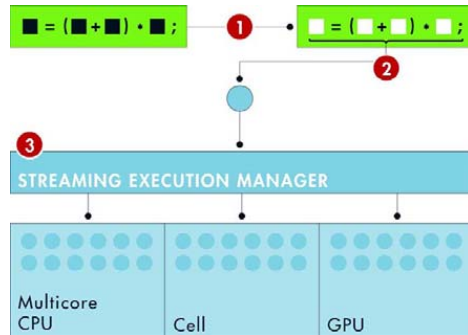
```

#include <cmath>

float f;
float a[512][512][3];
float b[512][512][3];

float func(
    float r, float s
) {
    return (r + s) * f;
}

void func_arrays() {
    for (int x = 0; x<512; x++) {
        for (int y = 0; y<512; y++) {
            for (int k = 0; k<3; k++) {
                a[y][x][k] =
                    func(a[y][x][k],b[y][x][k]);
            }
        }
    }
}
    
```





RAPIDMIND

0.

Access API

```
#include <cmath>

float f;
float a[512][512][3];
float b[512][512][3];

float func(
    float r, float s
) {
    return (r + s) * f;
}

void func_arrays() {
    for (int x = 0; x<512; x++) {
        for (int y = 0; y<512; y++) {
            for (int k = 0; k<3; k++) {
                a[y][x][k] =
                    func(a[y][x][k],b[y][x][k]);
            }
        }
    }
}
```

```
#include <rapidmind/platform.hpp>
#include <rapidmind/shortcuts.hpp>
using namespace rapidmind;
```



RAPIDMIND

1.

Replace Types

```
#include <cmath>

float f;
float a[512][512][3];
float b[512][512][3];

float func(
    float r, float s
) {
    return (r + s) * f;
}

void func_arrays() {
    for (int x = 0; x<512; x++) {
        for (int y = 0; y<512; y++) {
            for (int k = 0; k<3; k++) {
                a[y][x][k] =
                    func(a[y][x][k],b[y][x][k]);
            }
        }
    }
}
```

```
#include <rapidmind/platform.hpp>
#include <rapidmind/shortcuts.hpp>
using namespace rapidmind;
```

```
Value1f f;
Array<2,Value3f> a(512,512);
Array<2,Value3f> b(512,512);
```

```
Value3f func(
    Value3f r, Value3f s
) {
    return (r + s) * f;
}
```



RAPIDMIND

1b.

Replace
Types

```
#include <cmath>

float f;
float a[512][512][3];
float b[512][512][3];

float func(
    float r, float s
) {
    return (r + s) * f;
}

void func_arrays() {
    for (int x = 0; x<512; x++) {
        for (int y = 0; y<512; y++) {
            for (int k = 0; k<3; k++) {
                a[y][x][k] =
                    func(a[y][x][k],b[y][x][k]);
            }
        }
    }
}
```

```
#include <rapidmind/platform.hpp>
#include <rapidmind/shortcuts.hpp>
using namespace rapidmind;

Value1f f;
Array<2,Value3f> a(512,512);
Array<2,Value3f> b(512,512);

template <typename T>
T func(
    T r, T s
) {
    return (r + s) * f;
}
```



RAPIDMIND

2.

Capture
Computations

```
#include <cmath>

float f;
float a[512][512][3];
float b[512][512][3];

float func(
    float r, float s
) {
    return (r + s) * f;
}

void func_arrays() {
    for (int x = 0; x<512; x++) {
        for (int y = 0; y<512; y++) {
            for (int k = 0; k<3; k++) {
                a[y][x][k] =
                    func(a[y][x][k],b[y][x][k]);
            }
        }
    }
}
```

```
#include <rapidmind/platform.hpp>
#include <rapidmind/shortcuts.hpp>
using namespace rapidmind;

Value1f f;
Array<2,Value3f> a(512,512);
Array<2,Value3f> b(512,512);

Value3f func(
    Value3f r, Value3f s
) {
    return (r + s) * f;
}

void func_arrays() {
    Program func_prog = BEGIN {
        In<Value3f> r, s;
        Out<Value3f> q;
        q = func(r,s);
    } END;
    . . .
}
```



RAPIDMIND

3.

Parallel Execution

```
#include <cmath>

float f;
float a[512][512][3];
float b[512][512][3];

float func(
    float r, float s
) {
    return (r + s) * f;
}

void func_arrays() {
    for (int x = 0; x<512; x++)
        for (int y = 0; y<512; y++) {
            for (int k = 0; k<3; k++) {
                a[y][x][k] =
                    func(a[y][x][k],b[y][x][k]);
            }
        }
}
}
```

```
#include <rapidmind/platform.hpp>
#include <rapidmind/shortcuts.hpp>
using namespace rapidmind;

Value1f f;
Array<2,Value3f> a(512,512);
Array<2,Value3f> b(512,512);

Value3f func(
    Value3f r, Value3f s
) {
    return (r + s) * f;
}

void func_arrays() {
    Program func_prog = BEGIN {
        In<Value3f> r, s;
        Out<Value3f> q;
        q = func(r,s);
    } END;
    a = func_prog(a,b);
}
```



RAPIDMIND

Usage Summary

- Usage:
 - Include platform header
 - Link to runtime library
- Data:
 - Tuples
 - Arrays
 - *Remote data abstraction*
- Programs:
 - Defined dynamically
 - Execute on coprocessors
 - *Remote procedure abstraction*

```
#include <rapidmind/platform.hpp>
#include <rapidmind/shortcuts.hpp>
using namespace rapidmind;

Value1f f;
Array<2,Value3f> a(512,512);
Array<2,Value3f> b(512,512);

Value3f func(
    Value3f r, Value3f s
) {
    return (r + s) * f;
}

void func_arrays() {
    Program func_prog = BEGIN {
        In<Value3f> r, s;
        Out<Value3f> q;
        q = func(r,s);
    } END;
    a = func_prog(a,b);
}
```



RAPID MIND

Feature Summary

- Abstractions for *both* code *and* data
- Generate and manipulate code explicitly
 - C++ modularity
 - FORTRAN execution efficiency
- Can target GPU as well as Cell BE
- Simple, safe programming model
- Single-source ISO standard C++ program:
 - *No extensions needed*
 - *Use your existing compiler*



RAPID MIND

Advanced Topics

- Accessors
 - Extracting and accessing subarrays
 - Copying semantics
- Metaprogramming
 - Applications of dynamic code generation
- Design patterns
 - Processor pattern
 - Compiler pattern
- Acceleration strategies
 - Loop conversion
 - Interpreter conversion
 - Task conversion
- Program manipulation
 - Program algebra



RAPID MIND

Accessors

offset (A, n)

- Drop first n elements of A

shift (A, n)

- Translate index into array A by n

take (A, n)

- Drop all but first n elements of A

slice (A, i, j)

- Extract subarray from i to j, inclusive

stride (A, k)

- Extract every kth element

Return instance of **ArrayAccessor** type


- *References* subarray “view”, does not copy



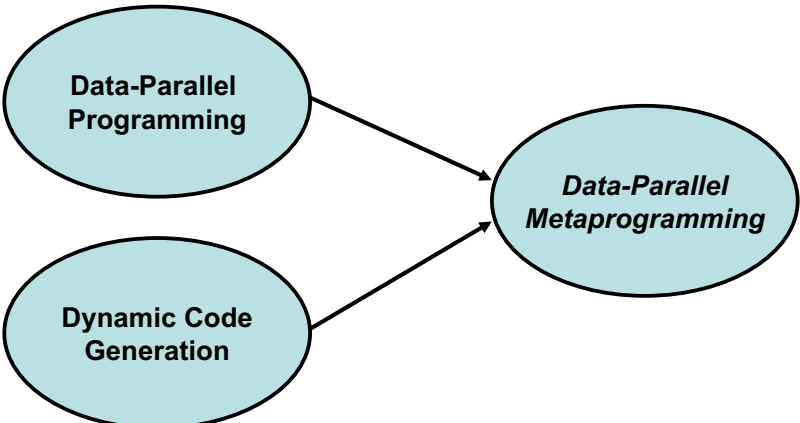
RAPID MIND

Copying Semantics


- Assignment to an **Array**:
 - by-value
 - assignment *replaces* destination
 - allocates new memory if needed
- Assignment to an **ArrayAccessor**:
 - by-value
 - assignment *copies* into destination
- Explicit copying can be forced with **copy** function
- Memory automatically freed if no longer referenced



Metaprogramming: Dynamic Code Generation



```
graph LR; A([Data-Parallel Programming]) --> C([Data-Parallel Metaprogramming]); B([Dynamic Code Generation]) --> C;
```



Advantages of Data Parallelism

- Efficient on a variety of computer architectures
 - Shared memory machines
 - Distributed memory machines
 - Vector/stream machines
- Predictable memory access patterns
- Scales to arbitrary number of processors
- Single thread of control
 - Simple extension of existing programming practice
 - No explicit synchronization needed
 - No deadlocks or non-determinism
 - Debugging simplified



RAPIDMIND

Advantages of Metaprogramming

- Object-oriented overhead of C++ avoided
 - Platform *only* compiles operations on RapidMind types
 - **Structure with C++:** templates, objects, namespaces, ...
 - **Run like FORTRAN (or better)**
- Metaprogramming can be used to build
 - Parameterized code, with possible automatic tuning
 - Code generated algorithmically
 - Code that adapts to hardware platform
 - Code that adapts to or is generated based on data
 - Compilers from interpreters
 - Higher order functions to parameterize operations



RAPIDMIND

Design Patterns

- Processor pattern
 - Manage code generation and initialization
 - Encapsulate parameterized code
- Compiler pattern
 - Remove overhead from computation specified at runtime



```

template <typename T, typename S>
class Processor {
protected:
    S m_f;

    T m_func(
        T r, T s
    ) {
        return (r + s) * m_f;
    }

    Program m_prog;

public:
    Processor(
        S f
    ): m_f(f) {
        m_prog = BEGIN {
            In<T> r, s;
            Out<T> q;
            q = m_func(r,s);
        } END;
    }

    Array<2,T>
    apply(
        const Array<2,T>& a,
        const Array<2,T>& b
    ) {
        return m_prog(a,b);
    }
};

// USAGE

// Initialize
ValueIf g;
Processor<Value3f, ValueIf> proc(g);

// Apply
Array<2, Value3f> p(512, 512);
Array<2, Value3f> q(512, 512);
p = proc.apply(p, q);

```



Problem:

- Need to evaluate some expression not known until runtime
- Example:
 - Image compositing
 - User may express sequence of operations in visual language

Solution 1: Interpreter Pattern

1. Encode computation in data structure (ex: operator dag)
2. Traverse data structure, *executing* operations
3. Return result

Solution 2: Compiler Pattern

1. Encode computation in data structure (ex: operator dag)
2. Traverse data structure, *recording* operations
3. Compile operations into program object
4. Execute program object on data
5. Return result



Approach 1: Loop Conversion

- Find hot spot
- Identify loop structures
- Convert loops to parallel operations



Approach 2: Interpreter Conversion

- Identify use of interpreter pattern
 - Convert to compiler pattern
- Advantages:**
- Can collect a significant amount of computation together even when there is no obvious hot spot
 - Can avoid memory and branching overhead of interpretation



Approach 3: Task Conversion

- Identify use or potential for task parallelism
- Convert to SPMD model
- Use arrays to communicate between tasks

Advantages:

- Simplified debugging
- Bulk synchronous model



- **Combination:**
 - Program “algebra” to combine programs into new programs
 - Can use to modify interfaces to existing programs
 - Can use to specialize existing programs
- **Partial evaluation:**
 - Can bind inputs one at a time
 - Can convert inputs to non-local variables and vice versa
- **Introspection:**
 - Can analyze program interface and performance at runtime
 - Use for self-tuning libraries



RAPID MIND

Program Algebra

- **Algebra:**
 - Set of objects
 - Set of operators
 - Closed
- **Objects:**
 - Programs
- **Operators:**
 - Functional composition:
 $p \ll q$
 - Concatenation:
 $\text{bundle}(p, q)$



RAPID MIND

Applications of the Program Algebra

- **Interface adaptation**
 - Reordering
 - Packing/unpacking
 - Input or output type conversion
- **Specialization**
 - Discard unneeded outputs
 - Eliminates unnecessary computation
- **Pipelining**
 - Combine producer/consumer programs into one:
 $A = (p \ll q \ll r)(B) ;$
 - Implement pipeline as single data-parallel task



- Can bind only some inputs of a program, not all
- Binding gives a new program with fewer inputs
 - If bind only 1 input of an n input program
 - Get back program with $n-1$ inputs
- Partial evaluation provides
 - Flexibility
 - Interface adaptation
 - Optimization opportunities
- Two kinds of binding:
 - Tight: uses `()`
 - Loose: uses `<<`; is invertible using `>>`



- Tight binding:
 - Program $q = p(A)$;**
- Execution can be deferred
- When eventually executes:
 - Uses value of **A** in effect at time of *binding*
 - Compiler can use actual value of **A** to optimize code



RAPID MIND

Loose Binding

- Loose binding:
`Program q = p << A;`
- Execution can be deferred
- When eventually executes:
 - Uses value of **A** in effect at time of *execution*
 - Value of **A** can be used to parameterize execution
- **A** acts like a non-local variable



RAPID MIND

Unbinding

- Convert input to non-local variable:
`q = p << A;`
- Convert non-local variable to input:
`q = p >> A;`



Examples

- *Crowd simulation (GDC)*
- *Ray tracing (w/ RTT)*
- *Fast Fourier transform*
- *Convolution*
- *Quasi Monte Carlo option pricing*
- *Matrix-matrix multiply (SGEMM)*
- *Transformation and lighting*
- *Color and gamma correction*
- *Object tracking*
- *Sorting*
- *Quaternion Julia set*
- *Deferred shading*
- *Vector textures*
- *Others...*





RAPIDMIND

Crowd Simulation

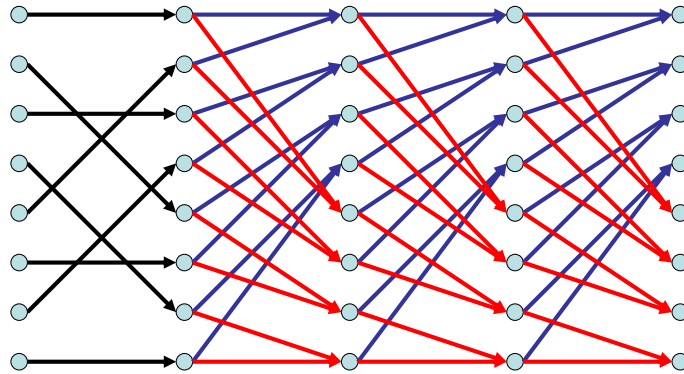
- Graphics on GPU
 - Shaders implemented using RapidMind platform
- Behavioral Simulation on Cell BE Blade
 - 16K autonomous characters (4K visible at once)
- Parallel Execution:
 - Rules to simulate social behavior and basic physics
- Global Communication:
 - Any character can interact with any other
 - Requires (approximate) solution to K-nearest-neighbor problem
 - Behavior depends on the environment
 - Random access to environmental parameter grid
 - Obstacles, ground cover and slope



RAPIDMIND

Fast Fourier Transform

- Fundamental signal processing operation
 - Image processing
 - Pattern matching
 - Solving differential equations
- Standard test case for parallel computation
- Involves both
 - Computation
 - Communication
- Many varieties and ways to implement
 - Will show radix-2 split-stream complex-to-complex 1D FFT



```

// Fast Fourier Transform
Array<1, Value2f>
FFT (Array<1, Value2f> data, int n) {
    int N = (1 << n);

    // define program objects
    ...

    // generate and scramble twiddle factors with gather
    ...

    // scramble input data using a gather
    ...

    // perform split-stream FFT using lg(N) passes
    ...
}

```



RAPID MIND

Fast Fourier Transform

```

// define program objects
Program butterfly_A = BEGIN {
    In<Value2f> a, b;
    Out<Value2f> c = a + b;
} END;

Program butterfly_B = BEGIN {
    In<Value2f> a, b, w;
    Value2f t = a - b;
    Out<Value2f> c;
    c[0] = t[0]*w[0] + t[1]*w[1];
    c[1] = t[1]*w[0] - t[0]*w[1];
} END;

```



RAPID MIND

Fast Fourier Transform

```

// generate and scramble twiddle factors with gather
Array<1, Value2f> w(N/2);
w = twiddle(n-1) [ bitreverse(n-1) ];

// allocate temporary storage
Array<1, Value2f> x[2];
x[0] = Array<1, Value2f>(N);
x[1] = Array<1, Value2f>(N);

// scramble input data using a gather
x[0] = data[ bitreverse(n) ];

// initialize source marker
int src = 0;

```



RAPID MIND

Fast Fourier Transform

```

// perform split-stream FFT using log(N) passes
for (int k=n-1; k>=0; k--) {
    // write into lower half of output array
    take(x[!src],N/2) = butterfly_A(
        stride(x[src],2),
        stride(offset(x[src],1),2)
    );
    // write into upper half of output array
    offset(x[!src],N/2) = butterfly_B(
        stride(x[src],2),
        stride(offset(x[src],1),2),
        take(w,1<<k)
    );
    // swap source and destination buffers
    src = !src;
}
// return final transform
return x[src];

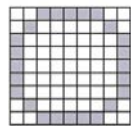
```



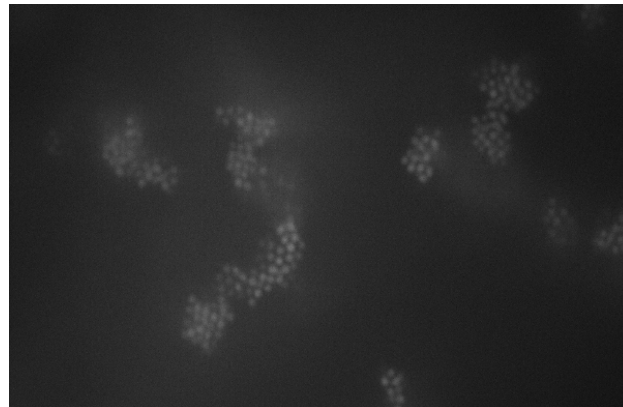
RAPID MIND

Convolution

- Fundamental signal processing operation
- For large filters, use FFT
 - FFT
 - Elementwise complex multiplication
 - Inverse FFT
- For small filters, do directly
 - Shift flipped filter to each pixel, multiply, sum
 - May process many images with one filter
 - Filters used in pattern matching may be sparse
 - Can exploit sparsity to get more efficient execution



*



Confocal microscopy image
courtesy of Peter J. Lu, Harvard



```

float filter[N0][N1];
Array<2, ValueIf> image(M0, M1);

Program convolve = BEGIN {
  In<Value2i> u;
  Out<ValueIf> result = ValueIf(0.0f);
  for (int i = 0; i < N0; i++) {
    for (int j = 0; j < N1; j++) {
      if (filter[i][j] != 0.0f) {
        Value2i tap = u - Value2i(i, j);
        result += filter[i][j] * image[tap];
      }
    }
  }
} END;

image = convolve << grid(M0, M1);

```



- **Real-time raytracing**
 - Supports reflection and refraction
 - Many recursive rays per pixel
 - Incoherent memory access
 - Accelerator data structure traversal
- **Commercial product:**
 - Developed by RTT AG, Germany
 - Used for automotive CAD visualization
- **Hardware:**
 - Released product runs on GPUs
 - Demonstrated on Cell BE at SIGGRAPH

Advanced Topics in Virtual Garment Simulation

Eurographics 2007 Tutorial T10

Organizers

Bernhard Thomaszewski (WSI/GRIS, University of Tübingen)
Markus Wacker (Computer graphics, University of Applied Sciences Dresden)
Wolfgang Straßer (WSI/GRIS, University of Tübingen)

Speakers

Bernhard Thomaszewski (WSI/GRIS, University of Tübingen)
Markus Wacker (Computer graphics, University of Applied Sciences Dresden)
Nadia Magnenat-Thalmann (MiraLab, Geneva)
Etienne Lyard (MiraLab, Geneva)

Advanced Topics in Virtual Garment Simulation

Part 1

B. Thomaszewski¹, M. Wacker^{1,2}, and W. Straßer¹

¹WSI/GRIS, University of Tübingen, Germany

²University of applied sciences Dresden, Germany

Abstract

For more than two decades, cloth simulation has been an active research area in computer graphics. In order to create efficient high-quality animations, techniques from many research fields have to be thoroughly combined. The ongoing interest in this field is also due to the multidisciplinary nature of cloth simulation which spurs development and progress in collision detection, numerical time integration, constrained dynamics, or motion control, to name just a few areas. Beyond the very basic approaches, the complexity of the material can be daunting if no guidance is given. It is therefore the goal of this tutorial to provide the reader with an introduction and a guideline to the relevant matter. In order to provide a concise review, we will focus on advanced topics in cloth simulation, shedding light on both theoretical and practical aspects. This will pave the ground for those willing to implement a contemporaneous cloth simulation system as well as researchers who consider to start working in this area.

1. Introduction

The physical simulation of deformable objects is a central research area in computer graphics. Problems emanating from this field are usually complex to model and pose significant demands on computational resources. This is particularly true for the more specific case of cloth simulation. Due to the thin and flexible nature of cloth, it produces detailed folds and wrinkles, which in turn can lead to complicated self-collisions. In order to tackle this challenge within the requirements imposed by computer graphics, specialised methods have to be designed for concrete applications. These can roughly be divided into two categories: applications for which quality is most important and those for which speed is the urgent demand. To clarify this distinction, the requirement for time-critical cloth simulation could read: *given a fixed timing (e.g. 25 frames per second) optimize the (visual) quality of the simulation*. An analogous formulation can be stated for the first case. Because every category stands in its own right, we will address both of these fields in this tutorial.

Since cloth simulation has been an active research field for quite a while now, there is a broad variety of different approaches. One objective of this course is therefore to give the reader an introduction to thoughtfully chosen matter and a guideline towards practical applications. Another is to introduce algorithms and tools necessary for creating

high quality animations. Finally, a further goal is to supply the audience with techniques for accelerating computations and eventually obtain fast simulations. The latter includes models specifically designed for computational efficiency. These methods can again be divided into two categories: algorithms which are optimized for sequential real-time computations and those which exploit the parallel potential of current hardware.



Figure 1: The Eurographics 2007 Phlegmatic Dragon covered by a sheet of cloth.

Because physically-based modelling has become the de

facto standard in cloth simulation, we will exclusively treat physically based methods in this tutorial and leave alternative approaches like geometry- or heuristics-based models aside.

1.1. Major Challenges in Cloth Simulation

Despite the comparably long history of cloth simulation this field can by no means be considered closed. For instance, the accurate modelling of real textile materials remains an open issue. Even if static properties can be measured quite accurately, textiles typically exhibit a high dynamical behaviour (e.g. hysteresis, damping, etc.) for which no accurate models are available yet. In the field of collision handling there is still room for improvement as well. Besides requirements like speed and robustness, error recovery is a point of particular practical relevance [VMT06a]. Furthermore, garment design and integration in CAD-like environments needs to be pushed forward if clothing simulation is to become a widely-used tool in industrial applications (see e.g. [CK05]). Likewise, electronic vending of clothing via the internet requires Virtual Try-on platforms specifically tailored to the individual needs of end customers. Finally, since high-quality animations can still be very time consuming, the need for designing faster techniques has not ceased. One way towards more efficient methods is to design, possibly from scratch, new algorithms that result in significant computational speed-up. Another is to develop or adapt algorithms which exploit parallelism on current hardware. In the course of this tutorial, we will address most of the aforementioned topics in great detail.

1.2. Overview

This tutorial assumes that the reader is already familiar with the basics of physically-based simulation. An overview of the state-of-the-art in this field is given in Sec. 2. The following sections of the first part are intended to supply the reader with the basics of contemporaneous cloth simulation. The third section gives an introduction to continuum mechanics and an example of numerical implementation. The way in which wrinkles and folds form depends mainly on the bending properties of the fabric. Because of its importance, a separate section (4) is devoted to this issue and practical ways of integrating bending into cloth simulation are discussed. In Sec. 5 we will describe how computations can be mapped onto parallel architectures. This includes generic modifications and extensions to existing algorithms in order to exploit potential parallelism. Both shared and distributed memory architectures will be considered, and we will address implementation related issues for both these settings.

The second part of these notes addresses the measurement of real world fabric parameters, including multi-layered textiles and seams, and their integration into cloth simulation. Furthermore, integrated Virtual Try-On applications are dis-

cussed, including models for parametrically deformable human bodies, body animation and motion retargeting as well as techniques for real-time cloth simulation.

2. State-of-the-Art in Virtual Clothing

In this section we will give an overview of the current state-of-the-art in virtual clothing. We will pay special attention to the topics covered in this tutorial. A more comprehensive summary of the evolution of this research can be found in [MTVW*05], Part 1.

2.1. Mechanical Models

For dynamically deformable surfaces, mass-spring systems [Pro95] and the more general particle systems [BHW94, VCMT95, EWS96] continue to be the most widely used simulation techniques in computer graphics. The popularity of mass spring systems is due to the ease of implementation and low computational costs. The accuracy offered by this method is, however, rather limited. As an example, simple homogeneous materials cannot be simulated consistently and the results highly depend on the specific mesh used in the simulation. If the reproduction of authentic material behaviour is desired (as, e.g., by the textile community), approaches based on continuum mechanics have to be used. Continuum-based approaches lead to a set of partial differential equations (PDEs), which have to be discretized in space and time. The spatial discretisation is usually carried out by means of finite differences (FDM) or finite element methods (FEM). Techniques based on finite elements and continuum mechanics (referred to as FE-approaches in the remainder) have not seen as much attention in cloth simulation as particle and mass spring systems. While we only mention the most relevant work here an extensive list can be found in [HB00]. Most of the existing FE-approaches are based on the geometrically exact thin shell formulation presented by Simo et al. [SFR89a]. Departing from the fully nonlinear theory, Eischen et al. [EDC96] proposed a cloth simulation method using quadrilateral, curvilinear elements. Because of the buckling behaviour of cloth, which can lead to divergence in the algorithm, an adaptive arc-length control is used. Etmuss et al. [EKS03] presented a linear FE-approach based on the plane-stress assumption. Bending is treated separately from in-plane deformations and a co-rotational strain formulation is used to account for arbitrary rigid body transformations. The work presented in [TWS06] extends the concept of Subdivision Finite-Elements by Cirak et al. [COS00] to dynamic cloth simulation using a co-rotational strain formulation. As a middle course between simple mass spring systems and finite elements, Volino et al. [VMT05] proposed an accurate particle system which draws on notions from continuum mechanics but replaces the numerical discretisation by a more direct geometric formulation.

2.2. Numerical Time Integration

The mechanical model provides the means for computing internal forces due to fabric deformations. The dynamical evolution of the system (i.e., the trajectories of the nodes) is then determined by Newton's second law. In the discrete setting, the time dimension is decomposed into discrete time intervals and numerical integration methods are used to advance the system from a given state to its next state in time. Most generally, one distinguishes between two types of integration schemes: *explicit* methods compute the next state in time based on the current state derivatives, which are readily computed according to the mechanical model. Commonly used explicit integration methods are the second-order accurate Midpoint method used e.g. by Volino et al. [VCMT95] and the fourth-order accurate Runge-Kutta scheme used for instance by Eberhardt et al. [EWS96]. For computer graphics applications the numerical accuracy is usually less important than stability and robustness. As a well known fact, explicit schemes only provide conditional stability (see [HE01]). Since the differential equations resulting from cloth simulation are inherently stiff, explicit methods need small step sizes to guarantee a stable simulation. Consequently, computation times soon become excessive with increasing problem sizes. *Implicit* schemes do not suffer from this restriction since, in this case, stability is independent of the step size. Since the seminal work of Baraff et al. [BW98] implicit methods have therefore become predominant for physically-based simulations in computer graphics. Implicit methods include the unknown state of the system at the end of the time step in the update formula. Therefore, a system of (nonlinear) equations has to be solved in every time step. This process is one of the most time consuming parts in cloth simulation. Widely used representatives of this class are the first-order accurate backward Euler scheme [BW98] and the second-order BDF-2 scheme [HE01]. Recently, the Newmark family of integrators (including both explicit and implicit variants) found its way into computer graphics [BMF03, GH*03]. For a detailed overview and comparison of existing integration schemes and their efficiency for cloth simulation, the reader is referred to [VMT01] and [HE01].

2.3. Collision Handling

Besides the simulation of the mechanical properties of cloth the interaction with its environment has to be modeled as well. This involves the detection of any collisions and an adequate response to prevent the clothes from intersecting. The proper treatment of these two components (to which we refer as collision handling in the remainder) is a very complex task [THM*05]. While the physical simulation engine computes new states at distinct intervals only, collisions can occur at any instant in between such intervals. Algorithms based on continuous collision detection can handle these cases in a robust way, but are often very complex and time consuming. Therefore, the collision handling step is a major bottleneck

in the simulation pipeline. Basically, detecting interference between two arbitrarily shaped objects breaks down to determining the interference between all primitives (i.e. faces, edges, and vertices) of one mesh with every primitive of the mesh representing the other object. With complex objects comprising thousands of faces, this naïve approach soon becomes too expensive due to its quadratic average run time with respect to the number of faces. A common way to accelerate the interference tests is to structure the objects under consideration hierarchically with bounding volumes. A bounding volume hierarchy (BVH) is then constructed for each object in the scene (including deformable as well as rigid objects) in a preprocessing step, using, for example, a top-down approach. In this case, a bounding volume enclosing the entire object is set as the root node of the tree representing the hierarchy. This node is then subdivided recursively until a leaf criterion is reached. Usually, the leaves contain one single primitive. Common choices for bounding volumes in cloth simulation are axis aligned bounding boxes (AABB) [van97, LAM01] or the more general discrete oriented polytopes (k-DOPs) [KHM*98, MKE03]. For treating self-collisions efficiently, it is necessary to adapt the general BVH algorithms to this special case. Measures related to the curvature of the surface can be used to quickly rule out flat, non-intersecting parts of the surface [VMT94, Pro97]. As another useful extension for self-collisions, continuous collision detection based on BVHs can be used, in which the exact contact points between two successive time steps are detected [BFA02].

Once the intersecting parts of a garment have been determined, an appropriate response has to be computed in order to prevent the imminent intersections. A method well-suited for cloth simulation is the one presented by Bridson et al. [BFA02]. The essence of this method is to apply a stopping impulse to approaching triangles (i.e., to adjust their nodal velocities) whenever their distance falls below a certain threshold. However, even with such robust approaches there can be situations in which intersections cannot be prevented. An example for this are complicated multi-layer collisions or when cloth is pinched together due to character motion. In such cases special care is necessary in order to restore an intersection free state and to keep the simulation running [BWK03, VMT06a].

Despite the aforementioned acceleration techniques, collision handling remains a major bottleneck of cloth simulation. Recent developments aimed at further accelerating these algorithms by migrating computations to the graphics card [GKJ*05] or by exploiting parallel architectures [TB07, TPB07].

3. Mathematical and Physical Foundations

In this section, we will describe how a physically based model for deformable objects can be derived. As already pointed out in Sec. 2, mass-spring systems are still widely

used for cloth simulation in computer graphics. However, for authentic material mapping and hence realistic and reliable draping behaviour of cloth, as required e.g. by the textile community, one must necessarily resort to continuum mechanics. We will therefore restrict our considerations to methods based on continuum mechanics and begin with a brief review of the relevant foundations. As we will see, the central quantities in this theory are *strain* and *stress* which are related to each other via material or constitutive laws. We will only discuss some general material laws at this point and postpone more sophisticated models especially suited for textiles to the second part of these notes. Having established the governing equations, we will turn to an exemplary spatial discretisation.

3.1. Continuous Models

The structure of textiles or woven fabrics is clearly different from continuous media. However, modelling each single thread would certainly be an inefficient approach. We will instead approximate the garment geometrically with a polygonal mesh. If the regions of each polygon contains a sufficiently large number of threads (or weave periods) we can safely approximate the fabric as a continuous medium. Departing from a continuum model, we can derive a consistent spatial discretisation. Consistency here means that with increasing resolution the computed approximation converges to the actual solution of the continuous problem. This allows us to choose the spatial resolution in accordance to computational requirements without changing the properties of the cloth. In doing so, the dynamic motion of garments can be simulated efficiently and independently of discretisation throughout a broad range of resolutions. The first ingredient for such a continuum model is introduced in the next section.

3.2. Deformation Measures

In order to describe the equations that govern the (dynamic) behaviour of deformable objects, we consider the conditions that must hold in an equilibrium state. Generally speaking, a deformable object is in equilibrium if the internal forces due to deformation exactly cancel the external forces acting on its volume and boundary. The first step in this analysis is to compute deformation, which requires an appropriate measure.

A conceptual description of a deformable object embedded in three-dimensional is given by its *configuration mapping*

$$\varphi : \Omega \subset \mathbf{R}^3 \rightarrow \mathbf{R}^3, \quad (1)$$

where Ω is its parameter domain. In dynamic scenes, where the object undergoes translation, rotation, and deformation this mapping also depends on time

$$\varphi : \Omega \times [0, \infty] \rightarrow \mathbf{R}^3.$$

It is common to base descriptions of state and behaviour of deformable objects on an initial and a current configuration. For simplicity, the initial mapping is assumed to be the identity

$$\bar{\varphi}(x_1, x_2, x_3) = \varphi(x_1, x_2, x_3, 0) = \text{id}.$$

The configuration mapping can be given the interpretation of transforming positions of material points in the initial, undeformed state to corresponding positions in the current, deformed configuration (see Fig. 2).

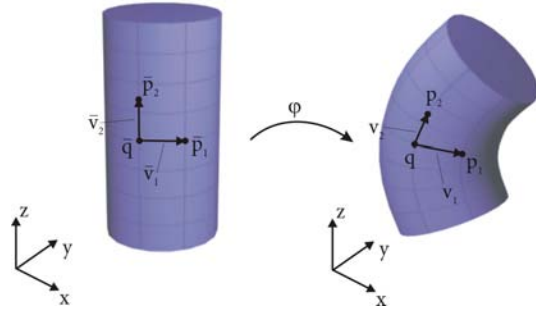


Figure 2: *Relative change of positions from undeformed to deformed configuration.*

For later derivations, it is convenient to express the current state in terms of a displacement field \mathbf{u} from the initial configuration as

$$\varphi = \bar{\varphi} + \mathbf{u} = \text{id} + \mathbf{u} : \Omega \rightarrow \mathbf{R}^3. \quad (2)$$

The displacement field itself is not an adequate measure of deformation, since invariance under rigid body motion such as translation is not given. A general deformation measure should capture the relative change between two elemental vectors in their initial and current configuration. Besides the obvious direct change in length, the angle formed by two vectors can change as well. One measure that takes both these characteristics into account is the scalar product. Consider the two vector pairs in the undeformed and deformed configuration in Fig. 2. The vectors are given by $\bar{\mathbf{v}}_i = \bar{\mathbf{p}}_i - \bar{\mathbf{q}}$, respectively $\mathbf{v}_i = \mathbf{p}_i - \mathbf{q}$. We can use a Taylor series expansion to express the vectors in the current configuration in an alternative form as

$$\begin{aligned} \mathbf{v}_i &= \varphi(\bar{\mathbf{q}} + \bar{\mathbf{v}}_i) - \varphi(\bar{\mathbf{q}}) \\ &= \varphi(\bar{\mathbf{q}}) + \nabla \varphi(\bar{\mathbf{q}}) \cdot \bar{\mathbf{v}}_i + O(\bar{\mathbf{v}}_i^2) - \varphi(\bar{\mathbf{q}}) \\ &\approx \nabla \varphi(\bar{\mathbf{q}}) \bar{\mathbf{v}}_i = (\nabla u(\bar{\mathbf{q}}) - \text{id}) \bar{\mathbf{v}}_i. \end{aligned}$$

For later use, we define the deformation gradient F as

$$F = \nabla \varphi, \quad (3)$$

which can be given the interpretation of mapping vectors in the initial configuration to vectors in the current configuration. A general deformation measure can now be derived as

the difference of scalar products in the rest and current state:

$$\mathbf{v}_1 \cdot \mathbf{v}_2 - \bar{\mathbf{v}}_1 \cdot \bar{\mathbf{v}}_2 = \bar{\mathbf{v}}_1 \cdot (\nabla \varphi^T \nabla \varphi - id) \cdot \bar{\mathbf{v}}_2. \quad (4)$$

Using Eq. (2) we can identify from Eq. (4) the symmetric *Green* strain tensor as

$$\boldsymbol{\varepsilon}_G = \frac{1}{2}(\nabla \varphi^T \nabla \varphi - id) = \frac{1}{2}(\nabla \mathbf{u}^T + \nabla \mathbf{u} + \nabla \mathbf{u}^T \nabla \mathbf{u}). \quad (5)$$

An alternative expression can be obtained using the deformation gradient

$$\boldsymbol{\varepsilon}_G = \frac{1}{2}(F^T F - id). \quad (6)$$

For practical purposes, Eq. (5) can be written in indicial notation as

$$(\boldsymbol{\varepsilon}_G)_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} + \sum_k \frac{\partial u_k}{\partial x_i} \frac{\partial u_k}{\partial x_j} \right). \quad (7)$$

In this form, the entries of the strain tensor can be interpreted geometrically: diagonal components ε_{ii} measure the change in length in the direction of the x_i -axis and off-diagonal entries ε_{ij} measure the shearing between two axes. For small displacements, the nonlinear terms are negligible. This gives rise to the linear *Cauchy* strain tensor

$$\boldsymbol{\varepsilon}_C = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u}^T), \quad (8)$$

which is used in small strain analysis. Because of its linearity, the Cauchy tensor is very common in computer graphics. A closer look reveals that while being invariant under translations, rotational invariance is not given. In dynamic simulations, where the objects usually undergo large rotations, this is a severe restriction. A possibility to circumvent this problem is to extract the rotational part from the deformation field. The key observation is that, using polar decomposition, F can be written as

$$F = RU, \quad (9)$$

where R is a rotation and U is a pure deformation. The rotation can then be determined by finding the eigenvectors of $F^T F$. From this together with Eq. (6) it can also be seen that $\boldsymbol{\varepsilon}_G$ is invariant under rotations since

$$F^T F = U^T R^T R U = U^T U \quad (10)$$

due to the orthogonality of R . Once the rotation field R is known, the rotated linear strain tensor can be computed as

$$\boldsymbol{\varepsilon}_{CR}(\varphi) = \boldsymbol{\varepsilon}_C(R^T \varphi). \quad (11)$$

In the expressions derived so far we have implicitly assumed a Cartesian reference frame. We may think of garments or more generally deformable surface as 2-manifolds embedded in three-dimensional space. As such, a two-dimensional parametrisation of the surface is necessary to correctly establish differential measures like deformation. For instance, approaches based on thin shell formulations (see Sec. 4.4) rely on (parametrised) curvilinear systems. In

such settings, all expressions will depend on partial derivatives of the parametrisation with respect to local coordinates. Although in a different way, common deformation measures are eventually recovered (see Sec. 4.4).

Strains in a deformable solid are accompanied by counter-acting forces, which are the subject of the following section.

3.3. Internal Stress and Equilibrium

To describe the distribution of internal forces, we will first consider a differential surface element with area dA on a cross section of a deformable body as depicted in Fig. 3. Let \mathbf{n} denote the normal of that element and let the resultant force acting on it be $d\mathbf{f}$.

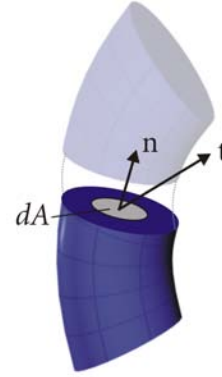


Figure 3: Deformable solid in cross-sectional view. Traction vector \mathbf{t} resulting from forces on area dA with normal \mathbf{n} .

Then, the traction vector \mathbf{t} is defined as

$$\mathbf{t} = \lim_{dA \rightarrow 0} \frac{d\mathbf{f}}{dA}.$$

Generalizing this expression for every normal direction leads to a mapping (or tensor) $\boldsymbol{\sigma}$ that maps every unit normal direction onto its traction vector,

$$\mathbf{t}(\mathbf{n}) = \boldsymbol{\sigma} \mathbf{n}. \quad (12)$$

The tensor $\boldsymbol{\sigma}$ is called the *Cauchy* stress tensor. This symmetric tensor will be the subject of the next step on our way to the equilibrium equations.

Consider now a volume element V of a deformable body and let \mathbf{f} be the body forces per unit volume acting on V . If we neglect forces due to inertia, translational equilibrium[†] implies that the sum of all forces acting on the volume V is

[†] as opposed to rotational equilibrium (see [BW97]p.103)

zero. Using equation (12), this can be expressed in integral form as

$$\int_{\partial V} \mathbf{t} da + \int_V \mathbf{f} dv = \int_{\partial V} \boldsymbol{\sigma} \mathbf{n} da + \int_V \mathbf{f} dv = 0, \quad (13)$$

where \mathbf{t} denotes the externally applied traction forces on the border ∂V . With the Gaussian divergence theorem, the surface integral of $\boldsymbol{\sigma} \mathbf{n}$ can be transformed into a volume integral,

$$\int_{\partial V} \boldsymbol{\sigma} \mathbf{n} da + \int_V \mathbf{f} dv = \int_V (\operatorname{div} \boldsymbol{\sigma} + \mathbf{f}) dv = 0. \quad (14)$$

Since this must hold for any enclosed volume, the (point wise) equilibrium equation of a deformable solid follows as

$$\operatorname{div} \boldsymbol{\sigma}(\mathbf{x}) + \mathbf{f}(\mathbf{x}) = 0. \quad (15)$$

For dynamic simulation this equation has to be augmented by terms accounting for inertial forces. Furthermore, $\boldsymbol{\sigma}$ will also include viscous stress contributions due to material damping, leading to

$$\operatorname{div} \boldsymbol{\sigma}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{f}(\mathbf{x}) = \rho \ddot{\mathbf{x}}. \quad (16)$$

Here, ρ is the mass density and \mathbf{x} denotes the vector field of positions. This continuous formulation is the starting point for numerical treatment. In Sec. 3.6 we will look at an exemplary spatial numerical discretisation. Before we continue with constitutive laws in Sec. 3.5 we will take a look at a hitherto neglected aspect of deformation.

3.4. Bending

As we have seen in the previous sections, the deformation modes in general (three-dimensional) continuum mechanics can be separated into stretching and shearing (cf. Eq. 7). These two deformation modes are *orthogonal* to each other: pure stretching does not lead to shear deformation and vice versa. In the special case of thin deformable surfaces, we intuitively identify *bending* as a further deformation mode. The question arises as to whether for bending deformation an analogous orthogonality relation holds. We will pursue this issue in the following.

The concept of bending cannot be derived from the three-dimensional, point wise view of continuum mechanics, since this standpoint is indifferent of *shape*. If we consider for example an elastic ball it is immediately clear that we cannot *bend* such an object. The same holds for other objects that do not exhibit a direction with significantly smaller size. If there is, however, such a direction like in the case of a thin plate or an elastic rod, bending deformations become possible.

As we can see from these examples, the ability to bend an object is closely related to its shape, or more precisely, to the proportion of its extents in the different dimensions. The ball is perfectly isotropic and has no intrinsic orientation. Therefore, the choice of a reference frame is arbitrary. While the geometry of the cube furnishes an intrinsic coordinate system, none of the dimensions is accentuated above

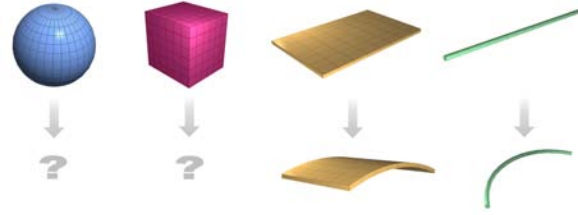


Figure 4: Whether an object can be bent depends mainly on its shape. It is not clear how to bend a sphere or a cube. For a thin plate or a rod the notion of bending is intuitively clear.

the others. In the example of the plate however, one direction can be distinguished, in which the lengths are clearly inferior to those in the orthogonal directions. The same holds for the case of a thin flexible rod. Note that the intuitive bending deformation of thin objects like the plate in the example is such that it causes as little in-plane deformation as possible. Hence, we may assume a bending mode which is orthogonal to the deformation modes of stretching and shearing. For the moment, we can most generally identify bending deformation of an embedded manifold as a change in curvature. We will return to this issue in a more specific discussion in the following paragraphs. In analogy to strain associated with stretching and shearing (to which we collectively refer as *membrane*-strains) we define the bending strain $\gamma_{\alpha\beta}$, $\alpha, \beta \in \{1, 2\}$, where the components $\gamma_{\alpha\alpha}$ are normal curvatures in the directions of surface coordinates and $\gamma_{\alpha\beta}$, $\alpha \neq \beta$, is the torsional component.

3.5. Constitutive Relations in Linear Elasticity

The previous section led to the formulation of equilibrium equations involving the stress tensor $\boldsymbol{\sigma}$. In general, the relationship between stress and strain can be of high complexity. Here, we will focus on situations where stresses in a body depend only on its current state of deformation. Materials that fulfil this requirement are called *hyperelastic*. Generally, for two different types of elastic material, the same deformation will result in different stresses in terms of magnitude and possibly direction. This relationship between stress and strain is described by the elasticity tensor \mathcal{C} .

In the simplest case, the relationship between stress and strain is linear. Note that the term *linear* appears in two different contexts. First, there is a geometric relationship between displacement and strain. A linear strain-displacement assumption results in the Cauchy strain tensor, as described above. Second, the dependence between strain and stress itself can be either linear or nonlinear. We will start with a linear material law, for which the stress tensor can be written as

$$\boldsymbol{\sigma} = \mathcal{C} : \boldsymbol{\varepsilon}, \quad (17)$$

where tensorial notation was used. This formulation accounts for the fact that a component ϵ_{ij} can potentially have influence on every entry σ_{kl} of the stress tensor. Using the summation convention, the stress tensor can conveniently be expressed as

$$\sigma_{ij} = C_{ijkl} \epsilon_{kl} . \quad (18)$$

Therefore, the entry C_{ijkl} can be interpreted as linking σ_{ij} to ϵ_{kl} .

As the simplest model, a linear-elastic isotropic material is governed by only two independent constants. In this case, the elasticity tensor[‡] C reads

$$C_{ijkl} = \lambda \delta_{ij} \delta_{kl} + 2\mu \delta_{ik} \delta_{jl} , \quad (19)$$

where λ and μ are the *Lamé* constants (cf. [BW97]). These constants can be used to express the well-known Young modulus E and the Poisson ratio ν as

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu} , \quad \nu = \frac{\lambda}{2(\lambda + \mu)} . \quad (20)$$

Fig. 5 illustrates the meaning of these constants. In this example, a simple elastic beam is subjected to a longitudinal loading along the x -axis. This loading leads to a deformation $\epsilon_{xx} = \frac{l-l_0}{l_0}$ counteracted by the stress σ_{xx} in the same direction. In linear small strain elasticity, these quantities are related by

$$\frac{\sigma_{xx}}{\epsilon_{xx}} = E .$$

In addition to the extensional strain in the direction of the

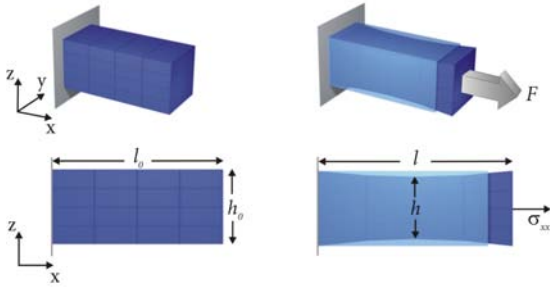


Figure 5: An elastic beam is subjected to a horizontal loading in x direction. Longitudinal as well as transverse deformation can be observed.

loading, we can usually also observe deformations $\epsilon_{yy} = \epsilon_{zz} = \frac{h-h_0}{h_0}$ in the orthogonal directions. Since this transverse contraction can be completely attributed to the axial loading, we can write

$$\epsilon_{yy} = \epsilon_{zz} = \frac{\nu}{E} \sigma_{xx} ,$$

[‡] In this context, the formulation in terms of tensors is common but basically indicial, tensorial, and matrix notations are equivalent (see also [Bel00]).

(see also [Hau04]). For physical realism, ν has to be such that $0 \leq \nu \leq 0.5$. In the case of textiles, ν is usually smaller than 0.3.

The material model described above corresponds to Hooke's law in three dimensions. Because of its linearity and easy implementation, it is widely used in computer graphics. In order to model the basic in-plane properties of textiles, a more general material law is needed. In two dimensions, stress and strain are described by 2×2 tensors and the elasticity tensor consists of 16 components. Symmetry considerations[§] imply that

$$C_{ijkl} = C_{klij} ,$$

as well as

$$C_{ijkl} = C_{jikl} , \quad C_{ijkl} = C_{ijlk} .$$

This results in an elasticity tensor of the form

$$C = \begin{bmatrix} C_{1111} & C_{1112} & C_{1122} \\ \text{sym.} & C_{1212} & C_{1222} \\ & & C_{2222} \end{bmatrix} ,$$

where components determined by symmetry have been omitted. This tensor is minimal in the sense that, assuming complete anisotropy, it cannot be further reduced. Hence, we can identify at most six independent constants describing an anisotropic, linear elastic material in two dimensions. For implementation purposes, vectorial notation is more convenient. As an example, the symmetric 2×2 stress tensor can be written as a 3-vector,

$$\sigma = [\sigma_{xx} \quad \sigma_{yy} \quad \sigma_{xy}]^T .$$

An analogous elasticity matrix can be derived from the elasticity tensor (see e.g. [Bel00]) as

$$C = \begin{bmatrix} C_{1111} & C_{1122} & C_{1112} \\ \text{sym.} & C_{2222} & C_{1212} \\ & & C_{1212} \end{bmatrix} . \quad (21)$$

As an example for textile simulation, one could use four of these constants: C_{1111} and C_{2222} are readily related to Young moduli E_u and E_v in the fabric's yarn directions *weft* and *warp*, which do not necessarily coincide with the coordinate directions x and y . Furthermore, C_{1212} is related to a shear modulus G and C_{1122} to transverse contraction coefficients ν_u and ν_v . With this material model, the stress-strain relationship can be written in matrix form as

$$\begin{bmatrix} \sigma_u \\ \sigma_v \\ \sigma_{uv} \end{bmatrix} = \frac{1}{1 - \nu_u \nu_v} \begin{bmatrix} E_u & \nu_u E_v & 0 \\ \nu_v E_u & E_v & 0 \\ 0 & 0 & G(1 - \nu_u \nu_v) \end{bmatrix} . \quad (22)$$

[§] The elasticity tensor can mathematically be derived in terms of second order partial derivatives of a stored strain energy function (see [BW97]). The symmetry follows from the commutability of the mixed partial derivatives.

In a similar way to Eq. (22), the bending stress τ can be related to the bending strain γ as

$$\tau = \mathbf{C}_{bend} \gamma, \quad (23)$$

where \mathbf{C}_{bend} models the bending properties of a specific material (see e.g. [VMT05]).

Because of its linearity and easy implementation, the material model described above is widely used for garment simulation. Beyond the small deformation range, the behaviour of most materials cannot reasonably be approximated by a linear model. In part 2 of the tutorial advanced constitutive laws are discussed, which capture fabric behaviour more accurately.

3.6. Spatial Discretisation with Linear Finite Elements

Having discussed constitutive models, we can now proceed with the discretisation of the governing equilibrium equations (14), respectively (15). We will use a formulation based on linear finite elements as an illustrative example [EKS03].

The finite element method is a procedure for the numerical solution of partial differential equations. The origins of this method can be found in structural mechanics, where the distribution of strains and stresses throughout a body in static equilibrium is sought. For the sake of brevity, we will not describe in detail how the final discrete equations used for implementation are derived. Instead, we will depart directly from the discrete setting. The reader interested in detailed derivations is referred to the standard text books [ZT00a] and [Bat96].

Assuming a decomposition of the domain into disjoint elements, the problem under consideration is formulated in terms of a displacement approach. This means, that for every node defined by the geometric decomposition, a displacement vector from its initial position is to be determined, such that the resulting final configuration is a solution to the elasticity problem. This process derives from a *variational principle*, the so called virtual work equation, which is mathematically rooted in variational calculus. In such a principle, an energy functional $\Pi = \Pi(\mathbf{u})$ is sought to be rendered stationary, in this case with respect to nodal displacements \mathbf{u} . In static elasticity problems, this functional is the sum of potential and strain energy[¶]. The functional is stationary if

$$\delta \Pi(\mathbf{u}) = 0, \quad \text{for all variations } \delta \mathbf{u}. \quad (24)$$

From this, the (dynamic) virtual work equation can be derived as

$$\int_{\Omega} \delta \boldsymbol{\varepsilon} : \mathbf{C}(\boldsymbol{\varepsilon}) d\Omega - \int_{\Omega} \delta \mathbf{u} \cdot \mathbf{f} d\Omega + \int_{\Omega} \delta \mathbf{u} \cdot \rho \ddot{\mathbf{u}} d\Omega = 0. \quad (25)$$

Here, the first term is due to the internal strain energy, the second one accounts for body forces per volume, and the last

term represents inertia forces. This equation forms the basis for the subsequent finite element discretisation. An important observation on the way towards the discrete formulation is that, using further mathematical transformations, the virtual work equation per element can be expressed alternatively in terms of equivalent internal and external nodal forces. If we assume a linear relationship between displacement and strain, as well as between strain and stress, the problem can finally be expressed in the form of

$$\mathbf{K} \mathbf{u} = \mathbf{f}. \quad (26)$$

The matrix \mathbf{K} is called the *stiffness matrix* and \mathbf{f} is the vector of external forces per node. Visually, this equation states that in an equilibrium state the displacement of the nodes is such that the equivalent internal nodal forces (resulting from internal stress) exactly cancel the external forces. For dynamic problems, this equation has to be augmented by terms accounting for inertia and viscous forces. This results in a second order initial value problem which reads

$$\mathbf{M} \ddot{\mathbf{u}}(t) + \mathbf{D} \dot{\mathbf{u}}(t) + \mathbf{K} \mathbf{u}(t) + \mathbf{f} = 0, \quad (27)$$

where \mathbf{M} is the mass matrix and \mathbf{D} is the viscosity matrix. In this equation, the initial conditions $\mathbf{u}(0) = \mathbf{u}_0$ and $\dot{\mathbf{u}}(0) = \mathbf{v}_0$ are assumed, where \mathbf{v}_0 denotes the initial velocity. Note that the vector of nodal displacements $\mathbf{u} = \mathbf{u}(t)$ now depends on time. In the following, we will explain how to set up the stiffness matrix for a practical example, namely the plane stress analysis of two-dimensional elasticity. This particular case can be used as a basis for a continuum mechanics-based cloth simulator.

3.6.1. Plane Stress Analysis

Two of the most compelling advantages of the finite element method are its modularity and versatility. Once a general framework for the method has been laid out it can be applied to a broad range of problems. The specialisation on the actual problem follows through decisions on additional properties like material laws and, what is most important, the choice of an actual element type. As a concrete example, we will investigate the special case of plane stress analysis in this section. Being the simplest candidate, the linear triangular element with nodal displacement as only degrees of freedom will be used.

Plane stress analysis can be applied to elasticity problems that are inherently two-dimensional, i.e. where the in-plane deformation is predominant. This restricts the range of problems to settings which are essentially *flat* or which can at least be reasonably approximated with flat elements. Nevertheless, it is possible to use the plane stress assumption as a basis for cloth simulation (see [EKS03]). In the following, we will, for simplicity, assume that the object under consideration lies in the xy -plane. As the name already indicates, in the case of plane stress, the out-of-plane components of the stress tensor are zero, i.e. $\sigma_{iz} = \sigma_{zj} = 0$.

The geometry of the linear triangular element is given by

[¶] The extension to kinetic energy is omitted here for simplicity.

the coordinates of its three points \mathbf{p}_1 , \mathbf{p}_2 and \mathbf{p}_3 (cf. Fig. 6). The three linear nodal shape functions are completely

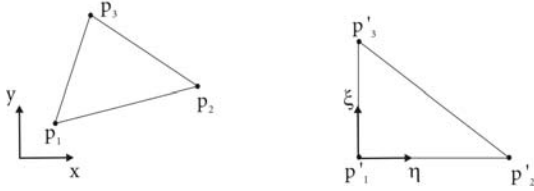


Figure 6: Geometry of a triangular element (left) and its corresponding generic element (right).

defined by the requirement

$$N_i(\mathbf{p}_j) = \delta_{ij} .$$

Further, the approximation of the displacement field over the element is uniquely defined by the nodal in-plane displacements $\bar{\mathbf{u}}_x$ and $\bar{\mathbf{u}}_y$ in the x and y -direction as

$$\mathbf{u} = \sum_i N_i \bar{\mathbf{u}}_i .$$

The definition of the shape functions, which are depicted in Fig. 7, automatically ensures displacement continuity across elements. For this simple element, an explicit expression for

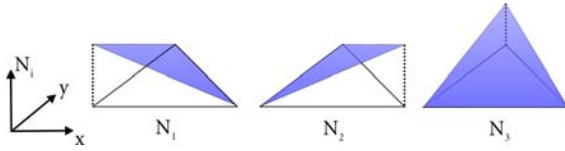


Figure 7: The three linear shape functions of a triangular element.

the shape functions in terms of the Cartesian coordinate can readily be derived. Nevertheless, it is instructive for the general case to take another approach. Notice that every triangular element can be transformed to a generic element as shown in Fig. 6. In this local space with coordinates ξ and η , the shape functions are trivially given by

$$N_1 = 1 - \xi - \eta, \quad N_2 = \xi, \quad N_3 = \eta .$$

Likewise, it can easily be verified that the shape function derivatives with respect to the local coordinates follow as

$$\frac{\partial N_1}{\partial \vartheta} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \frac{\partial N_2}{\partial \vartheta} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \frac{\partial N_3}{\partial \vartheta} = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

where $\vartheta = [\xi \quad \eta]^T$ denotes the vector of local coordinates. For the subsequent formulations, the shape function derivatives with respect to the Cartesian coordinates are required. These are obtained by use of the chain rule as

$$\frac{\partial N_i}{\partial x_j} = \frac{\partial N_i}{\partial \xi} \frac{\partial \xi}{\partial x_j} + \frac{\partial N_i}{\partial \eta} \frac{\partial \eta}{\partial x_j} .$$

In practice, the necessary computation of

$$\left(\frac{\partial x}{\partial \vartheta} \right)^{-1}$$

can be accomplished without difficulty. As can be seen in the following, the shape function derivatives are indeed the only quantities actually needed in computation. Hence, there is no need for explicitly deriving the shape function expressions.

The next stage consists in formulating strain. In the case of moderately small deformations, a linear strain definition in terms of the displacement field is common. Over a single triangular element, the Cauchy strain is defined as

$$\boldsymbol{\varepsilon} = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \mathbf{u} = \sum_{i=0}^3 \begin{bmatrix} \frac{\partial N_i}{\partial x} & 0 \\ 0 & \frac{\partial N_i}{\partial y} \\ \frac{\partial N_i}{\partial y} & \frac{\partial N_i}{\partial x} \end{bmatrix} \bar{\mathbf{u}}_i = \sum_{i=0}^3 \mathbf{B}_i \bar{\mathbf{u}}_i \quad (28)$$

Once the strain is computed for an element, the stress follows by a simple linear relation (see Sec. 3.5).

Stiffness Matrix The point of departure for setting up the stiffness matrix is the elemental virtual work equation. Using some basic transformations, the integral term leading to the global stiffness matrix can be written in terms of elemental contributions as

$$\int_{\Omega} \mathbf{B}^T \mathbf{C} \mathbf{B} \mathbf{u} \, d\Omega = \sum_i \int_{\Omega_i} \mathbf{B}^T \mathbf{C} \mathbf{B} \mathbf{u}_i \, d\Omega_i ,$$

provided that $\Omega = \cup_i \Omega_i$ and $\Omega_i \cap \Omega_j = \emptyset$ for all $i \neq j$ holds. Hence, the stiffness matrix can be assembled in an element-wise manner

$$\mathbf{K}_{ij} = \sum_e \mathbf{K}_{ij}^e$$

where \mathbf{K}_{ij}^e is the contribution of element e to the the entry of the global stiffness matrix linking nodes i and j . In the geometrically linear approach, the involved matrices \mathbf{B}_i from equation (28) are constant over one element. Together with a linear elastic material law the above integral reduces to

$$\mathbf{K} = \sum_i \int_{\Omega_i} \mathbf{B}_i^T \mathbf{C} \mathbf{B}_i \, d\Omega = \sum_i \mathbf{B}_i^T \mathbf{C} \mathbf{B}_i t A_i$$

where t is the constant material thickness and A_i is the area of a triangular element.

Practical Considerations With the methods presented so far only static analysis is possible. However, the extension to dynamic simulation is not difficult as it consists mainly of the addition of inertia and viscous forces already mentioned in Eq. (16). The actual way in which this is accomplished depends on the actual numerical time integration scheme. It is worth noting that the stiffness matrix automatically supplies a means of evaluating internal forces in the current configuration. In this case a simple matrix-vector multiplication is sufficient. However, if elastic forces are not integrated implicitly (as in explicit schemes or certain variants

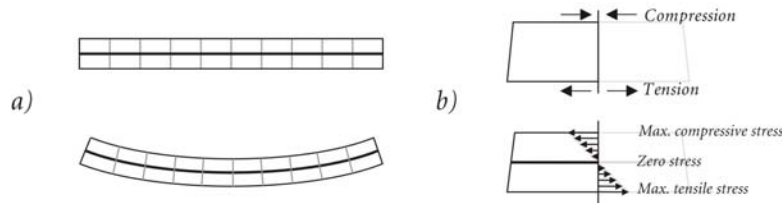


Figure 8: Cross-sectional view of bending deformation with finite thickness. a) Bending leads to a combination of compressive (top layer) and tensile (bottom layer) deformations. The neutral axis (shown in bold) remains unstretched. b) Linearly varying stresses through the thickness imply the existence of a zero-stress axis, the neutral axis.

of the Newmark algorithm [BMF03]), they can be computed directly without having to assemble the matrix itself.

4. The Importance of Bending in Cloth Simulation

Wrinkles and folds play an important role in the appearance of real textiles. The way in which they form depends mainly on the bending properties of the specific material type. While a remarkable amount of effort has been spent on precisely reproducing the in-plane forces, few existing models are concerned with an accurate and consistent way of modelling bending energy. Nevertheless, the characteristic folding and buckling behaviour of cloth highly depends on bending properties. The range of existing approaches to modelling bending is broad and we will investigate the most relevant methods in this section. From the field of engineering, the thin plate or thin shell equations, which will also be introduced in the following, are known to be an adequate approach to this problem. The associated minimisation problem includes fourth order derivatives with respect to the displacements, which in most of the frameworks are not readily computable. Therefore, a direct implementation of such energy-based methods is often avoided. However, many methods draw their inspiration from the theory of elastic beams or plates and we will therefore start with a review of the relevant matter.

4.1. Bending with Finite Thickness

Let us consider a thin plate and assume, for the moment, that the plate is cylindrically bent. In this case, we can – without loss of generality – restrict our investigations to a thin slice of the plate. Thus, we arrive at a geometry corresponding to the classical beam element (see e.g. [ZT00a]). In the cross-sectional view shown in Fig. 8 it can be seen that the bottom layer is stretched while the top layer has been compressed (cf. [Kee99]). We can reasonably assume that the maximum values of tension and compression occur on the boundary layers. If we further assume that the induced stresses vary monotonically between these maxima we arrive at an axis with zero stress, the so called *neutral axis* (see Fig. 8), is the primary parametrisation domain. In the following sections

we will see that the theory of elastic beams and plates is essentially based on this dimension reduction.

4.2. Linear Elasticity of Beams and Plates

The simplest model corresponding to our interests is the one-dimensional, linear elastic beam. As we will see in section 4.3, many existing approaches to bending in cloth simulation rely on this model. In the corresponding elasticity problem, the central unknown is the lateral deflection w of the neutral axis. We begin the description by introducing the kinematic constraints which lead to the common model of the *Euler-Bernoulli* beam. In this model, the *Kirchhoff Assumptions* are used, i.e. lines that are initially normal to the neutral axis are supposed to remain straight (i.e. they do not bend), normal to the neutral axis, and unstretched. The deformed state of the beam can be described by the displacements u_0 and w_0 of the neutral axis and a rotation θ of the normal (see Fig. 9).

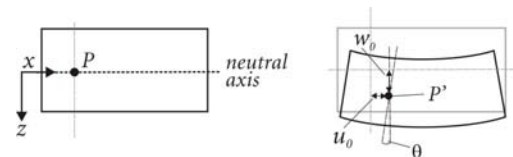


Figure 9: Displacements u_0 and w_0 of the neutral axis and cross-sectional rotation θ for a deformed beam element.

The horizontal and vertical displacements of any material point in the beam are given by

$$u(x, z) = u_0(x) - z\theta(x), \quad w(x, z) = w_0(x),$$

with the strains

$$\epsilon_x = \frac{\partial u}{\partial x} = \frac{\partial u_0}{\partial x} - z \frac{\partial \theta}{\partial x}.$$

Because normal lines are assumed to remain unstretched, the strain ϵ_z in this direction can be neglected. Further, the requirement that normal lines remain perpendicular to the neutral axis implies

$$\theta = \frac{\partial w_0}{\partial x}. \tag{29}$$

Hence, we have for the transverse shear strain

$$\varepsilon_{xz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} = -\theta + \frac{\partial w_0}{\partial x} = 0. \quad (30)$$

With the strains defined, the stresses now follow with an appropriate constitutive law. If we assume a linear elastic material law we have

$$\sigma_x = \frac{E}{1-\nu^2} \varepsilon_x, \quad (31)$$

where E is Young's modulus and ν is Poisson's ratio. The bending moment around the horizontal axis is obtained as

$$M = D \frac{\partial \theta}{\partial x} = \frac{Eh^3}{12(1-\nu^2)} \frac{\partial^2 w_0}{\partial x^2}, \quad (32)$$

where the cubic thickness term h^3 is due to the moment of inertia. Note that for small deflections w_0 , the term $\frac{\partial^2 w_0}{\partial x^2}$ is actually the curvature κ of the beam. This allows us to write the generalised stress-strain relationship of the Euler-Bernoulli beam in a clearer manner as $M = D\kappa$. The linear moment-curvature relationship is exploited by some approaches in cloth simulation to directly model bending forces (e.g. [VCMT95]).

In order to establish the governing equilibrium equations, we consider the forces acting on a differential beam element. In Fig. 10 loads and stress resultants on the beam are shown.

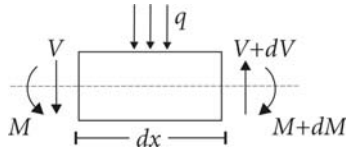


Figure 10: Distributed lateral forces q , transverse shear force V , and bending moment M acting on a differential beam element of length dx .

The beam is in equilibrium if the transverse internal force V (or shear resultant) and the external distributed load q are in balance. This leads to the equilibrium equation of the Euler-Bernoulli beam

$$\frac{Eh^3}{12(1-\nu^2)} \frac{\partial^4 w}{\partial x^4} = -q. \quad (33)$$

The above formulations directly carry over to cylindrically deformed plates. They can as well be translated to the general theory of (doubly curved) thin plates (see [ZT00b]). In engineering, thin plate elements are used to support lateral loads. Because curvature now occurs in both transverse directions, one speaks of the *neutral surface*, or simply *mid-surface*, in analogy to the neutral plane. Again, it is assumed that the stretch deformations of the mid-surface are negligible. Hence, the primary unknown is again the lateral deflection w . However, the deflection now varies in both x and y

direction which renders the problem two-dimensional. For thin plates, the equilibrium equation is

$$\frac{Eh^3}{12(1-\nu^2)} \left(\frac{\partial^4 w}{\partial x^4} + 2 \frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4} \right) = -p. \quad (34)$$

This is a biharmonic equation involving fourth order partial derivatives. Investigating the corresponding strains it can be seen, that second order derivatives of the (lateral) displacement field are required. The thin plate equations have been used in computer graphics, too. For instance, they appear in a common minimisation problem from variational design (see e.g. [WW98]). Despite its demands on continuity, the thin plate approach can be used in physically based simulation. Since this theory does not take into account in-plane deformations it has to be augmented by an appropriate membrane model. This conjunction can be found in the class of *Kirchhoff-Love* thin shell theories (see Sec. 4.4).

4.3. Existing Approaches: From Springs to Shells

In this section we will report on bending models used in physically based simulation. In the seminal work of Terzopoulos et al. [TPBF87] a model for the animation of elastically deformable surfaces based on continuum mechanics is presented. The authors derive an elastic strain energy depending on the nonlinear metric and curvature tensors. The associated partial differential equations are discretised in space using finite differences on a regular quadrilateral grid. Although this approach is based on a physically sound theory, it was not widely adopted in the computer graphics community, due to its significant computational complexity. In the following years, approaches for the simulation of deformable surfaces like cloth mainly relied on particle and mass-spring systems. For modelling forces due to bending deformation, most of these methods use some kind of angular measure to approximate curvature. Breen et al. [BHW94] were among the first to use a coupled particle system in cloth simulation. The authors present an approach based on energy potentials for modelling the static drape of cloth. Departing from linear beam theory (see section 4.2), they first derive the bending energy between two successive edges in a rectangular discretisation. Curvature is approximated by fitting a circle through the three points involved in such a bending element. Using a biphasic curvature expression, Breen et al. model bending energy by approximating the nonlinear curves obtained from measurements with the *Kawabata Evaluation System* [Kaw80] numerically with quadratic fits. Once the energies are set up at the nodes, the gradients have to be computed to obtain nodal forces. The approach presented by Eberhardt et al. [EWS96] extends this work to the dynamic range. Computation times are greatly reduced through the use of sophisticated integration schemes. However, Eberhardt et al. do not approximate curvature but directly use the angle as a deformation measure.

Volino et al. [VCMT95] use a mass-spring system in-

spired by continuum mechanics. The basic bending element is formed by two adjacent triangles from the underlying unstructured grid (see Fig. 11). To determine curvature, a circle fitting inside the two triangles is found using the dihedral angle. The curvature over an element is then obtained as the inverse of the circle's radius. The authors point out that the curvature has to be limited to a certain maximum to prevent bending forces from reaching infinity. Using linear beam theory, the actual forces are deduced from the geometry of the involved triangles.

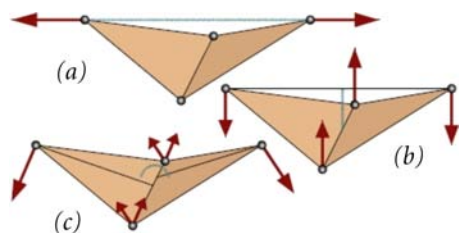


Figure 11: *Different ways of modelling bending forces on triangle meshes. Forces can be computed according to tension crossover springs (a), based on the dihedral angle (b), or based on a weighted sum of vertex positions (c). (Fig. taken from [VMT06b])*

Baraff et al. [BW98] use the same basic bending element as in [VCMT95]. Following their proposed computational framework, a constraint expression for bending energy is derived. This essentially corresponds to an energy term which depends quadratically on the dihedral angle.

A different approach to cloth simulation was proposed by Eischen et al [EDC96, EB00]. Their method is based on the nonlinear shell theory derived by Simo et al. [SFR89a, SFR89b]. A four node bilinear element with nodal displacements and director rotations as the primary unknowns is used for discretisation (see [SFR89a]). In the context of shell theory, curvature is directly accessible through bending strains and does not have to be approximated otherwise. Like in [BHW94] Eischen et al. use measured data obtained from the Kawabata Evaluation System and use a 5th-order polynomial fit to approximate the curves. In sum, the approach leads to highly nonlinear equilibrium equations which have to be solved, for example, with the Newton-Raphson procedure. This solver is coupled with an adaptive arc length control to account for limit or bifurcation points in the solution due to buckling instabilities. The proposed method is limited to the static case and does not account for dynamic effects. For subsequent comparison, Eischen et al. [EB00] present a particle-based approach inspired from continuum mechanics. Like Breen et al. they use a regular quadrilateral discretisation but derive forces directly without explicit resort to energy potentials. However, their constructions are based on linear elasticity theory. Bending forces are derived using linear beam theory which again results in a linear moment-curvature relationship. The angle

formed by two consecutive edges is taken as a direct measure for curvature. The authors state that the outcome of the two methods cannot be visually distinguished on the scale of the images they produced.

More recently, Choi et al. [CK02] proposed a bending model based on assumptions on the buckling behaviour of fabric. Starting from a standard quadrilateral mass-spring system, the basic bending element consists of an interleaved spring. The authors advocate that compressive in-plane forces on textiles lead to large out-of-plane deflections once a critical loading is reached. For the notoriously unstable post-buckling state the buckled shape is predicted as a circular arc of constant length and curvature. With this assumption, the curvature can be computed analytically without any angle appearing. Hence, linear beam theory can be applied to derive the bending energy. Lastly, the authors derive expressions for force vectors and Jacobians at the nodes which allows the use of an implicit time integration scheme.

Bridson et al. [BMF03] proposed another derivation of bending forces for cloth simulation. Again, two adjacent triangles form the basic bending element. With the assumption that bending forces should neither cause in-plane deformation of the fabric nor lead to rigid body motions, they derive the directions and relative magnitudes for the four bending force vectors of an element. These vectors are then scaled with a bending stiffness constant and the sine of the dihedral angle (see Fig. 11, b). An additional scaling factor accounts for anisotropy of the mesh. For the numerical time integration, Bridson et al. suggest to use a mixed implicit-explicit integration scheme in which the (comparably small) bending forces can be handled in an explicit manner while viscous damping forces are treated implicitly. In this way computing the complicated derivatives of the bending forces is avoided.

In the bending model used by Eitzmuß et al. [EKS03] forces are computed according to an approximation of the surface Laplacian. This idea is motivated by the fact that discrete mean curvature is closely related to the discrete surface Laplacian (see [MDSB03]). Using linear finite elements, the approximate Laplacian is computed for each element and projected onto the corresponding vertex normals. As usual, the element contributions are summed up to give the point wise Laplacian for every vertex.

While in most of the previous approaches curvature is approximated rather inaccurately, Grinspun et al. [GH*03] presented a method which is based on a sound curvature derivation. Their work extends existing cloth simulators to the range of objects for which the bending resistance is predominant. To this end, a discrete flexural energy potential is established using differential geometry. Again, the basic bending element consists of two adjacent triangles. The energy derives from an approximation to the squared difference of mean curvature in the current and initial configuration. The derivatives of the bending energy are intricate to com-

pute and hence the authors suggest the use of an automatic differentiation system.

Recently, Volino et al. [VMT06b] presented an approach which combines fairly good accuracy for representing quantitative bending stiffness with a simple and efficient computational procedure. In this method, a *bending vector* is computed that represents the bending of the surface through a simple linear combination of particle positions (see Fig. 11, c). This vector is then redistributed as particle forces according to the bending stiffness of the surface. It can be shown that this scheme preserves total translational and rotational momentum without the need of recomputing the distribution coefficients according to the current position of the particles. This leads to a very simple computation process which is entirely linear, and thus very well adapted to implicit numerical integration. In terms of computational simplicity and speed, this approach competes well with simpler spring models, since it requires only linear operations, which can conveniently be cast into matrix form. Additionally, the Jacobian of the bending forces is constant throughout the simulation, which is a considerable computational advantage when using implicit numerical integration methods. The authors demonstrated that the accuracy of their model is close to the more accurate normal-based method as used in, e.g. [BMF03]. Nevertheless, it does not require expensive operations like trigonometric functions or square root evaluations.

Assuming an inextensible surface and, thus, isometric deformations, Bergou et al. [BWH*06] derived a bending energy which depends quadratically on nodal positions. Similarly to [VMT06b], this leads to a computationally economic model with bending forces linear in positions and a constant associated Jacobian. An extensive analysis of this and other isometric bending models based on discrete curvature energies can be found in [WBH*07].

4.4. SubdivisionFE

As pointed out above, the thin shell equations are a good choice when physical accuracy is important. A corresponding finite element approach requires a C^1 -continuous displacement field (to be exact the shape functions have to be in H^2). The main problem with this requirement is guaranteeing continuity across elements which usually necessitates the use of additional variables (e.g. nodal rotations and slopes). An elegant and convenient way to avoid this additional complexity is to use subdivision finite elements as a basis (see [COS00]). The following section gives a brief account of the approach presented in [TWS06], which utilizes this type of finite elements .

4.4.1. Thin Shell Mechanics

In the Kirchhoff-Love theory of thin shells the configuration mapping (2) is expressed in terms of the mid-surface

parametrisation $\mathbf{x}(\theta^1, \theta^2)$ (see Fig. 12) as

$$\varphi(\theta^1, \theta^2, \theta^3) = \mathbf{x}(\theta^1, \theta^2) + \theta^3 \mathbf{a}_3(\theta^1, \theta^2), \quad (35)$$

where θ^i denote curvilinear coordinates and \mathbf{a}_3 is the director field normal to the surface. In analogy to Eq. (2) we write

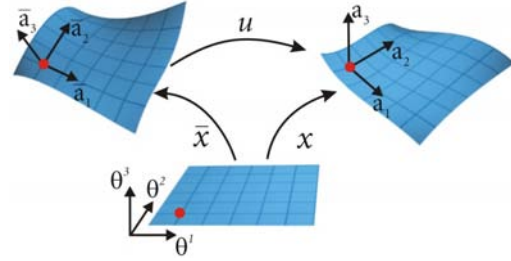


Figure 12: A material point (red) on the shell's mid-surface with basis vector frame in the initial, reference, and current configuration (from left to right).

$$\mathbf{x}(\theta^1, \theta^2) = \bar{\mathbf{x}}(\theta^1, \theta^2) + \mathbf{u}(\theta^1, \theta^2). \quad (36)$$

From this, tangential surface basis vectors can be defined as

$$\mathbf{a}_\alpha = \mathbf{x}_{,\alpha}, \quad (37)$$

where the comma denotes partial differentiation. From hereon, we use Greek indices to denote variables, which can take on the values $\{1, 2\}$ in order to distinguish from Latin indices, which can take on the values $\{1, 2, 3\}$. Moreover, the covariant tangent base vectors are given through differentiation of the configuration mapping as

$$\mathbf{g}_\alpha = \varphi_{,\alpha} = \mathbf{a}_\alpha + \theta^3 \mathbf{a}_{3,\alpha} \quad (38)$$

from which the surface *metric tensor* is derived as

$$g_{ij} = \mathbf{g}_i \cdot \mathbf{g}_j. \quad (39)$$

Following Eq. (5) this leads to the definition of the Green strain

$$\varepsilon_{ij}^G = \frac{1}{2}(g_{ij} - \bar{g}_{ij}) = \alpha_{ij} + \theta^3 \beta_{ij}, \quad (40)$$

where α and β are membrane and bending strains, respectively. In the Kirchhoff-Love theory (cf. Sec. 4.2), the director \mathbf{a}_3 is assumed to stay normal to the surface, straight and unstretched,

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|}. \quad (41)$$

Consequently, we have $\alpha_{3\beta} = \alpha_{\alpha 3} = 0$. The strains then simplify to

$$\alpha_{\alpha\beta} = \frac{1}{2}(\mathbf{a}_\alpha \cdot \mathbf{a}_\beta - \bar{\mathbf{a}}_\alpha \cdot \bar{\mathbf{a}}_\beta), \quad \beta_{\alpha\beta} = (\bar{\mathbf{a}}_{\alpha,\beta} \cdot \bar{\mathbf{a}}_3 - \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3).$$

Departing from $\mathbf{a}_\alpha = \bar{\mathbf{x}}_{,\alpha} + \mathbf{u}_{,\alpha}$ and neglecting nonlinear terms, this can be recast to an expression which is linear in

displacements [COS00]. Resultant membrane and bending stresses follow as

$$n^{\alpha\beta} = \frac{\partial\Psi}{\partial\alpha_{\alpha\beta}}, \quad m^{\alpha\beta} = \frac{\partial\Psi}{\partial\beta_{\alpha\beta}}, \quad (42)$$

where Ψ is the strain energy density. The particular form of Ψ depends again on the specific material law used (see Sec. 3.5). As for the plane stress case, it is possible to use the corotational strain measure of Eq. (11), although the rotation field is determined in a slightly different way (see [TWS06]).

4.4.2. Subdivision-Based Finite Elements

Subdivision is a process for constructing smooth limit surfaces through successive refinement of an initial control mesh. As one of the most prominent examples Loop's subdivision scheme fulfils all prerequisites to serve as a basis for the thin shell discretisation. Besides the usual C^1 -continuity inherent to subdivision surfaces, an important feature of this schemes is that the curvature of the resulting limit surface is L_2 - or square integrable [RS01]. Due to this property, the subdivision basis functions can be used as shape functions for the FE-solution of the thin shell equations. In each step of this subdivision method, the positions of newly inserted nodes as well as those of old nodes are computed through a linear combination of vertices from the coarse mesh determined by the so called subdivision mask.

Only the immediate neighbours (i.e. the 1-ring) of a vertex have influence on this computation which gives rise to an efficient implementation. The process of subdivision itself can be considered as a linear operation and consequently be written in matrix form. It is therefore possible to directly derive properties like derivatives of the limit surface using an Eigenanalysis of the subdivision matrix. This yields simple expressions that can be computed efficiently. The resulting limit surface (or more generally, the limit field) can be evaluated at an arbitrary point in the interior of a patch, which is an important property for numerical integration. The key observation is that in regular settings (i.e. when all involved vertices have valence 6) Loop's scheme leads to generalised quartic box splines. In this case surface properties in one patch are completely defined by the 12 nodal values in the 1-neighbourhood (see Fig. 14) and the associated box spline basis functions N_i .

For instance, if we denote the local patch coordinates by θ^α , the limit surface can be expressed as

$$\mathbf{x}(\theta^1, \theta^2) = \sum_{i=1}^{12} N_i(\theta^1, \theta^2) \mathbf{x}_i, \quad (43)$$

where \mathbf{x}_i are the nodal positions of the underlying mesh. In the same way, the displacement field interpolation is obtained from the nodal values. Additionally, differential quantities can be determined as

$$\mathbf{x}_{,\alpha}(\theta^1, \theta^2) = \sum_{i=1}^{12} N_{i,\alpha}(\theta^1, \theta^2) \mathbf{x}_i. \quad (44)$$

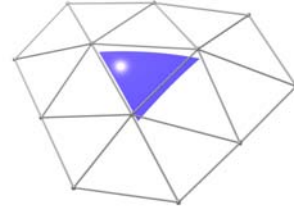


Figure 14: 1-ring neighbourhood of a regular patch consisting of 12 nodes.

If the patch has an irregular vertex, the box spline assumption no longer holds and thus interior parameter points cannot be evaluated. For the following finite element discretisation, however, only quantities at the barycenter of the triangles are needed for integral evaluation. Hence, Cirak et al. required the initial mesh to have at most one irregular vertex per triangle. Then, after one subdivision step the barycenter lies again inside a regular patch (see Fig. 14). This process of subdivision and evaluation of the newly generated patch can again be expressed as a sequence of matrix multiplications.

Spatial Discretisation With the definition of the membrane and bending strains and assuming a linear elastic material (Eq. (17)) the internal energy from the virtual work equation (25) can be rewritten as

$$\int_{\Omega} \delta\epsilon : C(\epsilon) d\Omega = \int_{\Omega} \left(\delta\alpha^T \mathbf{H}_m \alpha + \delta\beta^T \mathbf{H}_b \beta \right) d\Omega, \quad (45)$$

where \mathbf{H}_m and \mathbf{H}_b are matrices corresponding to the membrane and bending part of the material law (see [COS00] for a complete derivation). Due to the linear strain interpolation, we have

$$\alpha(\theta^1, \theta^2) = \sum_i^N \mathbf{M}_i(\theta^1, \theta^2) \mathbf{u}_i, \quad \beta(\theta^1, \theta^2) = \sum_i^N \mathbf{B}_i(\theta^1, \theta^2) \mathbf{u}_i$$

for matrices \mathbf{M}_i and \mathbf{B}_i relating nodal displacements \mathbf{u}_i to membrane and bending strain. This gives rise to a formulation of the complete system in the classical form of

$$\mathbf{K} \mathbf{u} = \mathbf{f} \quad (46)$$

with vectors of nodal displacement \mathbf{u} and forces \mathbf{f} . The stiffness matrix \mathbf{K} can be assembled in the usual element-wise fashion

$$\mathbf{K}_{ij} = \sum_e \int_{\Omega_e} \left(\mathbf{M}_i^T \mathbf{H}_m \mathbf{M}_j + \mathbf{B}_i^T \mathbf{H}_b \mathbf{B}_j \right) d\Omega = \sum_e \mathbf{K}_{ij}^e. \quad (47)$$

The integral in this equation can be evaluated using numerical quadrature. In this form, the above equations are only valid on regular patches. However, as mentioned above, in irregular settings one subdivision step is sufficient for evaluations at the barycenters. For a patch with an irregular vertex of valence N let \mathbf{S} denote the the subdivision operator

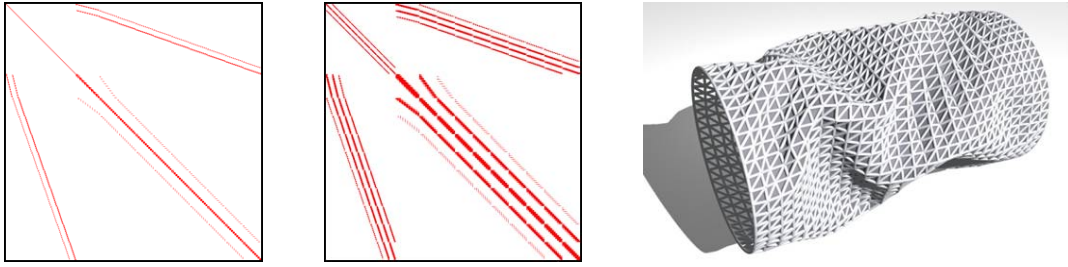


Figure 13: A comparison of the sparsity structures resulting from an ordinary finite element approach (left) and the subdivision FE-based method (middle) for a regularly tessellated mesh comprised of 1448 vertices (right).

(see [COS00]). Further, let \mathbf{P} be the projection operator extracting the 12 vertices corresponding to the central regular subpatch (Fig. 14, right). Then we can write

$$\mathbf{K}_{ij}^e = \int_{\Omega_e} \left[\mathbf{S}^T \mathbf{P}^T \left(\mathbf{M}_i^T \mathbf{H}_m \mathbf{M}_j + \mathbf{B}_i^T \mathbf{H}_b \mathbf{B}_j \right) \mathbf{P} \mathbf{S} \right] d\Omega \quad (48)$$

and thus simply include the conceptual subdivision step into the stiffness matrix.

To cover dynamic effects of moving and deforming objects inertia as well as viscous forces have to be included. This leads to a second order ODE in time analogous to Eq. (27). After transformation into a set of coupled first order ODEs, standard numerical time integration schemes can be applied (see [HE01]).

Discussion The setup process for the matrix \mathbf{K} in Eq. (48) is very similar to a common FE-formulation. The evaluation cost for an element matrix in this approach, however, is slightly higher than for usual methods. This is mainly due to the increased connectivity of the elements: a regular patch contains a neighbourhood of 12 nodes where for standard approaches there are only three. This means that more entries have to be computed and written into the system matrix (see Fig. 13). As a consequence, the setup and resolution of the linear system is slower than for standard approaches like the one described in Sec. 3.6. To put this into the right context, processing times for collision detection and response can still be much larger. A striking advantage of this approach is that it alleviates the modelling of a broad range of different materials (see Fig. 15). For example, because the full curvature tensor is available (cf. Eq. (25)), bending properties can directly be controlled, allowing for complex anisotropic models. Since this approach is entirely based on the sound mathematical foundation of continuum mechanics, the simulation is largely independent of discretisation throughout a broad range of resolutions.

5. Parallel Techniques for Cloth Simulation

Up to this point we have mostly dealt with methods that were designed to exhibit as much visual realism as possible. Moreover, techniques for measuring and reproducing



Figure 15: Top: Different types of folds on a garment's sleeve generated using a typical textile material. Bottom: Sequence taken from an animation of axial compression of a cylinder with a metal-like material.

real world materials in an accurate way were discussed. This realism and accuracy, however, comes at a price: the computational costs can be very high and run times for realistic scenarios are often excessive. Most of the computation time is spent on two stages, time integration and collision handling. In the following, we will therefore examine these two major bottlenecks, which are present in every physically-based cloth simulation system.

5.1. Implicit time integration

The ordinary differential equation resulting from the temporal discretisation of Eq. (27) are notoriously stiff. Due to stability reasons implicit schemes are widely accepted as the method of choice for numerical time integration (cf. [BW98]). Implicit schemes require the solution of a usually nonlinear system at each time step. As a result of the spatial discretisation, the matrix of this system is usually very sparse. There are essentially two alternatives for the solution of the system. One is to use an iterative method like the popular conjugate gradients (cg) algorithm [She94]. Another is to use direct solvers based on factorization. The cg-method is more popular in computer graphics as it offers much sim-

pler user interaction, alleviates the integration of arbitrary boundary conditions and allows balancing accuracy against speed. We will therefore focus on the cg-method in the following.

5.2. Collision Handling

For realistic scenes, the interaction of deformable objects with their virtual environment has to be modelled. This involves the detection of proximities (collision detection) and the reaction necessary to keep an intersection-free state (collision response). In the remainder, we refer to these two components collectively as *collision handling*. We usually distinguish between external collisions (with other objects in the scene) and self-collisions. For each of these types different variants of algorithms are usually used. Even with common acceleration structures, these algorithms are still computationally expensive. For complex scenarios with complicated self-collisions the collision handling can easily make up more than half of the overall computation time. It is therefore a second bottleneck for the physical simulation and hence deserves special attention.

Target Architecture The distinctive characteristic of parallel platforms is their memory architecture. In the following we will consider both distributed and shared address spaces. As exemplary representatives we choose clusters built from commodity components for the distributed memory setting and multi-core, multi-processor systems for the shared memory case. Although the basic ideas remain the same, the specific form and implementation of the actual algorithms depend on the target architecture. This is due to the fact that for different platforms, optimal performance is achieved in quite different ways as well. For example, one of the most important objectives on distributed memory architectures (DMA) is to minimize communication between the nodes of the cluster. While (inter-process) communication on shared address space architectures is not a costly aspect, care has to be taken for synchronization (including e.g. data access by multiple threads) as well. We will point out important differences at the appropriate places.

Programming Paradigms and Implementation Another distinguishing property of a given architecture is how it supports and favours different programming paradigms. On DMA machines a widely adopted strategy is that of single program multiple data (SPMD) where every node executes the same code but has different data to operate on. The communication in this case is usually message passing style, for example an implementation of the message passing interface (MPI). As a concrete example, the MPI-based numerical toolkit PETSc offers extensive support for SPMD style programming in scientific computation. Although it supports other approaches and can be run on many architectures, PETSC is especially well suited for large scale applications on DMAs. It can also be used in the shared memory setting

although in this case, message passing is actually not necessary and synchronization can often be implemented more efficiently. Additionally, it is often desirable on these platforms to use a multi-threaded programming approach which is not supported by PETSc. However, most of the operating systems directly support this paradigm and there are numerous toolkits and frameworks offering convenient interfaces for thread-based programming. Another interesting alternative with growing acceptance is the OpenMP interface. This interface offers extensive support for exploiting loop-level parallelism. It is entirely based on compiler directives and is therefore highly portable. Moreover, the user does not have to care about thread accounting (i.e. creation, synchronization, termination) which makes this interface easy to use.

5.3. Parallel Solution of Sparse Linear Systems

As was stated above, one of the major computational burdens is due to the solution of a sparse linear equation system related which derives from implicit time integration. We assume that a sparse linear system of the form $\mathbf{Ax} = \mathbf{b}$ is to be solved up to some residual tolerance using the cg-method. The number of necessary iterations and therefore the speed of convergence depends on the condition number of the matrix \mathbf{A} . Usually, this condition number is improved using a preconditioning matrix \mathbf{M} leading to a modified system

$$\mathbf{M}^{-1}\mathbf{Ax} = \mathbf{M}^{-1}\mathbf{b},$$

where $\mathbf{M}^{-1}\mathbf{A}$ is supposed to have a better condition number and \mathbf{M}^{-1} is fairly easy to compute. The choice of an appropriate preconditioner is crucial because it can reduce the number of iterations substantially. The setup and solution of the linear system now breaks down to a sequence of operations in which (due to their computational complexity) the sparse matrix vector multiplication (SpMV) and the application of the preconditioner are most important. As a basis for the actual parallelization we will consider problem decomposition approaches subsequently.

5.4. Problem Decomposition

In the following explanations we use the compressed row storage (CRS) format for sparse matrices in which nonzero entries are stored in an array along with a row pointer and a column index array (see [Saa03]). The most intuitive (and abstract) way to decompose the SpMV operation into a number of smaller sub-problems is to simply partition the matrix into sets of contiguous rows. The multiplication can then be carried out in parallel among the sets. This simple approach has several disadvantages. First, the matrices we deal with are always symmetric (due to the underlying PDE). Hence, only the upper triangular part, including the diagonal, has to be stored. This leads to smaller memory requirements for the data as well as for the index structure. In its sequential version, the resulting numerical kernel is more efficient (cf.

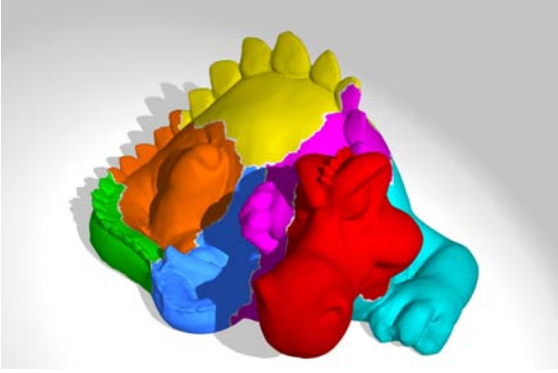


Figure 16: The Phlegmatic Dragon model is decomposed into 8 disjoint regions (indicated by different colours). This partitioning serves as a basis for parallel processing.

[LVDY04]): it visits every matrix entry only once, performing both dot products and vector scalar products. However, the access pattern to the solution vector is not as local as for the non-symmetric case, i.e., entries from different sets need to be written by a single processor. The required synchronization would make a direct parallel implementation of the symmetric SpMV kernel inefficient. Another reason why the above decomposition is inadequate is that it does not take into account two other important components of linear system solution: matrix assembly and preconditioning. Methods based on domain decomposition are better suited for this case. They divide the input data geometrically into disjoint regions. Here, we will only consider non-overlapping vertex decompositions, which result in a partitioning P of the domain Ω into subdomains Ω_i such that $\Omega = \cup_i \Omega_i$ and $\Omega_i \cap \Omega_j = \emptyset$, for $i \neq j$. Decompositions can be obtained using graph partitioning methods such as Metis [KK96] in our case. An example of this can be seen in Fig. 16 and Fig. 17, which also shows a special vertex classification. This will be explained in the next section.

5.5. Parallel Sparse Matrix Vector Multiplication

Let $n_{i,loc}$ be the number of local vertices belonging to partition i and let V_i be the set of corresponding indices. These vertices can be decomposed into n_{int} internal vertices and n_{bnd} interface or boundary vertices, which are adjacent to n_{ext} vertices from other partitions (see Fig. 17). If we reorder the vertices globally such that vertices in one partition are enumerated sequentially we obtain again a partitioning of the matrix into a set of contiguous rows. The rows $a_{i,0}$ to $a_{i,n}$ of matrix A where $i \in V_i$ have the following special structure: the block $A_{i,loc}$ defined by $\{a_{lm} | l \in V_i, m \in V_i\}$ and lying on the diagonal of A is symmetric. The nonzero entries in this block describe the interaction between the local nodes of partition i . More specifically, this means that when

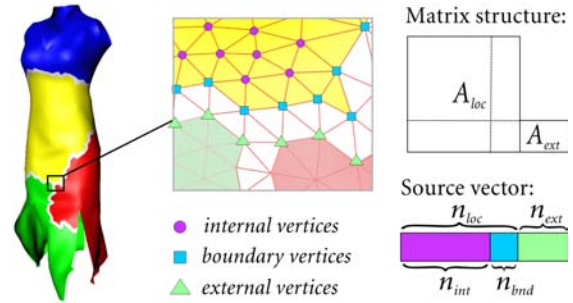


Figure 17: Decomposition of a mesh into four disjoint partitions indicated by different colours. The associated vertex ordering leads to a special structure of the matrices and the source vector.

nodes l and m are connected by an edge in the mesh, there is a nonzero entry a_{lm} in the corresponding submatrix of A . Besides this symmetric block on the diagonal there are further nonzero entries a_{le} where $l \in V_i$ is an interface node and $e \notin V_i$. These entries describe the coupling between the local interface nodes and neighbouring external nodes. The multiplication can be carried out efficiently in parallel if we adopt the following local vertex numbering scheme (cf. [Saa03]). The local vertices are reordered such that all internal nodes come before the interface nodes. For further performance enhancement, a numbering scheme that exploits locality (such as a self avoiding walk [OBHL02]) can be used to sort the local vertices. Then, external interface nodes from neighbouring partitions are locally renumbered as well. Let A_{ext} be the matrix which describes the coupling between internal and external interface nodes for a given partition. Notice that A_{ext} is a sparse rectangular matrix with n_{bnd} rows. With this setup the multiplication proceeds as follows

1. $y(0, n_{loc}) = A_{loc} \cdot x(0, n_{loc})$
2. $y(n_{int}, n_{loc}) += A_{ext} \cdot x_{ext}(0, n_{ext})$

The first operation is a symmetric SpMV, the second one is a non-symmetric SpMV followed by an addition. Both these operations can be carried out in parallel among all partitions. This decomposition is not only used for the SpMV kernel but also as a basis for the parallel matrix assembly, preconditioner setup and preconditioner application. Additionally, it can be implemented efficiently on both distributed and shared memory architectures.

5.6. Parallel Preconditioning

In order to make the cg-method fast, it is indispensable to use an efficient preconditioner. There is a broad variety of different preconditioners ranging from simple diagonal scaling (Jacobi preconditioning) to sophisticated multilevel variants. For the actual choice one has to weigh the time saved from the reduced iteration count against the cost for setup

and repeated application of the preconditioner. Additionally, one has to take into account how well a specific preconditioner can be parallelized. Unfortunately, designing efficient preconditioners is usually the most difficult part in the parallel cg-method [DHv93]. As an example, the Jacobi preconditioner is very simple to set up and apply even in parallel but the reduction of necessary iterations is rather limited. Preconditioners based on (usually incomplete) factorization of the matrix itself or an approximation of it are more promising. One example from this class is the SSOR preconditioner. It is fairly cheap to set up and leads to the solution of two triangular systems. For the sequential case, this preconditioner has proved to be a good choice in terms of efficiency [HE01]. However, parallelizing the solution of the triangular systems is very hard. Even if it is not possible to decouple the solution of the original triangular systems into independent problems we can devise an approximation with the desired properties. Let \bar{A} be the block diagonal matrix with block entries $A_{ii} = A_{i,loc}$, i.e. the external matrices A_{ext} are dropped from A to give \bar{A} . Setting up the SSOR-preconditioner on this modified matrix leads again to the solution of two triangular systems. However, solving these systems breaks down to the solution of decoupled triangular systems corresponding to the $A_{i,loc}$ blocks on the diagonal. This means that they can be carried out in parallel for every partition. For reasons of data locality we use n smaller SSOR preconditioners constructed directly from the $A_{i,loc}$ -blocks. Approximating A with \bar{A} means a comparably small loss of information which in turn leads to a slightly increased iteration count. However, this increase is usually small compared to the speedup obtained through parallelization.

5.7. Optimizations

Besides the topics that were treated above a further aspect restricts the efficiency of a parallel implementation of the cg-method. Dense matrix multiplications usually scale very well since they have regular access patterns to memory and a high computational intensity. This is not true for the SpMV case. A typical SpMV algorithm using a matrix in the CRS format looks as follows:

Algorithm 1 Symmetric Spmv

```

1: for  $i = 1$  to  $n_{rows}$  do
2:    $start = ptr[i], end = ptr[i + 1];$ 
3:   for  $j = start$  to  $end$  do
4:      $y[i] += dat[j] * x[ind[j]];$ 
5:   end for
6: end for

```

It can be seen that the actual matrix data A_{dat} as well as the index structure ind is traversed linearly (with unit stride) while accesses to the vector's data x_{dat} occur totally arbitrary, i.e. the locality of these data accesses cannot be assumed. It can also be observed that the computational intensity per data element is rather modest. The performance

of the SpMV algorithm is therefore mostly limited by memory bandwidth and cache performance. Because of this, it is important to improve data locality and thus cache performance. A good way to achieve this is to exploit the natural block layout of the matrix as determined by the underlying PDE: the coupling between two vertices is described by a 3x3 block – therefore nonzero entries in the matrix occur always in blocks. This blocked data layout already compensates for a lot of the inefficiency. An additional benefit can be achieved using single precision floating point data instead of double precision. This reduces the necessary matrix data (not including index structure) transferred from memory by a factor of two.

5.8. Parallel Collision Handling

From the parallelization point of view, the problem of collision handling differs substantially from the physical model. Collisions can be distributed very unevenly in the scene and their (typically changing) locations cannot be determined statically. This is why the naive approach of letting each processor care for the collisions of its own partition can lead to considerable processor idling, which seriously affects the overall parallel efficiency. Therefore, a dynamic problem decomposition is mandatory. Because bounding volume hierarchies (BVHs) are widely used for collision detection between deformable objects, it is reasonable to use this approach as a basis for a parallel implementation. As we have seen in a previous section, we can generally distinguish between two different types of collisions: external collisions and self collisions. To detect the first type we have to test our deformable object against every other (rigid or deformable) object in the scene. For the latter case, the deformable object has to be tested against itself. In this section, we will first lay down the basic algorithm for parallel collision handling. Subsequently, necessary adaptation for both distributed and shared memory architectures will be addressed.

5.8.1. Basic Problem Decomposition

The recursive collision test of two BVHs can be considered as a depth-first tree traversal. For inducing parallelism, this procedure is implemented using a stack which holds individual tests of two BVs.

In the BVH testing procedure, expansion of a tree node results in two additional tree nodes each representing a refined BVH test. As in the sequential procedure, the first test is carried out by starting a new recursion level. However, before entering the recursion, the second test is pushed onto the stack. The traversal goes on downwards until a leaf is reached. Upward traversal begins by processing elements from the stack. In this way, all of the nodes in the tree are visited. Fig. 18 shows a snapshot of a BVH testing process along with the corresponding state of the stack. Note that, although we assume a binary tree in this example, an extension to general n -ary trees is possible (see [TPB07] for

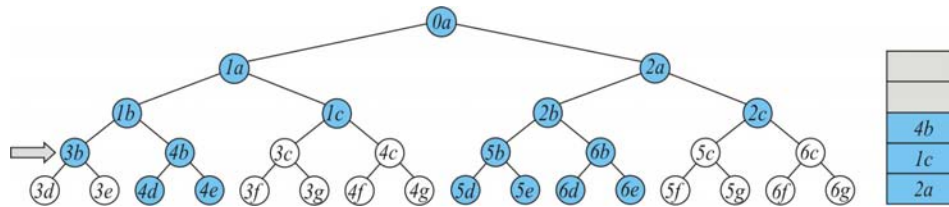


Figure 18: Dynamic problem decomposition. The arrow indicates the current state of the BVH testing procedure. The stack on the right stores the root of untried branches.

details). With a proper indexation of the hierarchies a single test can be represented using only a few indices and can thus be stored economically in appropriate data structures.

Tests which are recorded on the stack can be executed in one of the following ways:

- A test can be removed from the top of the stack and executed sequentially when the recursion gets back to the current level. Conceptually, this case is very similar to the procedure of the original algorithm.
- One or more tests can be removed from the bottom of the stack and executed by a newly generated (sub-)task. For each assigned test, this task executes a BVH testing procedure for which the considered test defines the root of a BVH testing tree.

In order to prevent that tasks of too fine a granularity are generated, several tests can be removed at once from the stack and assigned to a single task. The actual decomposition and load balancing strategy depends on the specific platform and is discussed in subsequent paragraphs.

5.8.2. Distributed Memory

The first step for embedding the above algorithm into the SPMD framework is to construct a BVH. The physical modelling phase provides a decomposition of the input mesh into *local* meshes owned by nodes of the cluster. On each node, a BVH hierarchy is set up on this mesh using a standard top-down approach. Once this is done, the root nodes of the different processors can be combined to form a global hierarchy of the mesh. Since the structure of this hierarchy is kept constant throughout the simulation, this global hierarchy can safely be replicated once on every node. This replication later enables the location-independent execution of sub-tasks. While external collision can now be treated in the usual way, care has to be taken for self-collisions. Two different types of self-collisions have to be distinguished: collisions between sub-meshes of different processors and those that are real self-collisions on the processor-local mesh. For the latter case, existing techniques can be used since this corresponds to the usual self collision problem. For the case of inter-processor self-collisions, the corresponding BVHs are tested against each other, similar to the way standard collisions are treated.

All discussed BVH tests form a set of *top-level tasks* of the parallel collision handling method. For scenes where collisions are not uniformly distributed, the number of top-level tasks and also the amount of time to execute individual top-level tasks can differ considerably among the processors. Hence, a load balancing scheme which distributes the load among the processors evenly is mandatory.

Dynamic Problem Decomposition and Load Balancing

As described above, every processor maintains a stack data structure where dynamically generating tasks are stored. For dynamic load balancing tasks must be transferred between processors at run-time. In distributed memory architectures, task transfers require explicit communication operations. Depending on the information associated with a task, task transfers can significantly contribute to the overall parallel overhead, especially when the computation to communication ratio of the tasks is poor. Finding a compact description of tasks is therefore crucial to minimize communication overhead. In our case, the cost for transferring a task is rather low.

Because the global BVH is replicated on every node, individual tests can be represented simply as an array of integers of the form $(obj1, dop1, obj2, dop2)$. The two BV hierarchies to be tested are identified by $obj1$ and $obj2$, and the root of the test is specified by $dop1$ and $dop2$.

The additional costs for building the copies of the BV hierarchies at the initialization phase are negligible since storage requirements are comparably low (say, only a few megabytes at max) even for complex scenes. The overhead during the course of the simulation is also insignificant: Updating hierarchies (at the beginning of every collision handling phase) requires one all-to-all broadcast operation to provide all processors the complete positions vector.

The dynamic load balancing process is responsible for triggering and coordinating task creation and task transfer operations in order to prevent that processors run idle. In order to achieve high scalability, a fully distributed scheme should be used. The distributed task pool model is a specifically well-suited basis. In this scheme every processor maintains a local task pool. Upon creation, a task is first placed in the local task pool. Subsequently, it can be instantiated

and executed locally, when the processor gets idle. It also might be transferred to a remote task pool for load balancing purposes. As discussed subsequently, task creation and task transfer operations are initiated autonomously by the processors.

To achieve a high degree of efficiency an (active) processor must be able to satisfy task requests from other processors immediately. Additionally, we must ensure that tasks with sufficient granularity are generated. In our case, this means that when a task is to be created, the stack must contain a minimum number of BVH tests which can be assigned to the task. In order to meet both requirements, we generate tasks in a proactive fashion, i.e. independently of incoming tasks requests. Tasks are generated (and placed in the local task pool) when the size of the stack exceeds a threshold value τ . Since task generation generally imposes an overhead (even when the new task is subsequently executed on the same processor) we increase τ linearly with the current size of the task pool σ : $\tau = a\sigma + b$. This simple heuristic enables us on the one hand to provide in a timely fashion tasks with a minimum granularity (determined by b) and ensures on the other hand that the overhead of additional task decomposition operations is compensated by an increased granularity of the resulting tasks. The parameter values a and b largely depend on the parallel architecture used and should be determined experimentally. Usually, choosing $a = 2$ and $b = 8$ delivers the best results.

A new task gets $\tau/2$ tests, where the tests are taken from the bottom of the stack. Tests are taken from the bottom of the stack for creating new tasks because such tests have a higher potential of representing a large testing tree since they originate closer to the root of the current testing tree.

For transferring tasks between task pools, a receiver initiated scheme is employed. When a processor runs idle and the local task pool is empty, it tries to steal tasks from remote pools. First, a victim node is chosen to which a request for tasks is sent. For selecting victims we apply a round robin scheme. If available, the victim transfers a task from its pool to the local pool. Otherwise the request is rejected by sending a corresponding message. In the latter case another victim node is chosen and a new request is issued.

For the actual implementation of the previously described methods, tools supporting distributed multithreading are needed. An efficient and convenient variant can be found in the parallel system platform DOTS [BKLwW99]. DOTS provides extensive support for the multithreading parallel programming model (not to be confused with the shared-memory model) which is particularly well suited for task-parallel applications that employ fully dynamic problem decomposition. In this way, the user does not have to care for task transfer and thread accounting explicitly, which is a great alleviation. Combining PETSc and DOTS, the collision handling algorithm can be implemented efficiently on

DMA. For a thorough performance analysis the interested reader is referred to [TB06, TB07].

5.8.3. Shared Memory

Compared to the approach for DMA the basic parallelization strategy remains the same. Because communication is cheaper and thus dynamic collaboration between threads is less expensive, the shared memory setting enables us to set up heuristics exploiting temporal and spatial coherence. In this way, thread creation overhead can be controlled effectively.

Unlike in the distributed memory setting we do not have to care for load balancing explicitly. As long as there are enough threads ready for execution the scheduler will keep all cores busy. However, for problems with high irregularity, like parallel collision handling, it is generally impossible to precisely adjust the amount of logical parallelism to be exploited to the amount of available parallelism (i.e., idle processors). Especially on shared memory architectures, thread creation overhead can considerably contribute to the overall parallel overhead. Therefore, an over-saturation with threads has to be avoided as well.

In [TPB07] thread creation overhead is minimized on two levels. On the algorithmic level, a heuristics-based approach is used which prevents threads with too fine a granularity from being generated. On the implementation level, the process of thread creation and thread execution is decoupled. The next two paragraphs explain these optimizations in more detail.

Controlling Task Granularity For effectively controlling the granularity of a task, we need a good estimate of how much work corresponds to a certain task. The computational cost for carrying out a test in the collision tree is determined by the number of nodes in its subtree. Generally, this number is not known in advance. Because of the inherent temporal locality due to the dynamic simulation we can, however, exploit coherence between two successive time steps. After each collision detection pass we compute the number of tests in the respective subtree for every node in the collision tree using back propagation. This information is then used as a work estimate for tasks in the subsequent collision handling phase.

In this way, creating tasks with too small an amount of work can be avoided. Additionally, this information can be used to determine which tests should be carried out immediately. The error involved in the work estimation is usually very small. This can also be seen in Fig. 20 which shows a comparison of the estimated and the actual work load for 5.5 seconds of simulation for a representative test scene (see Fig. 19). The new task splitting scheme performs robustly and always keeps track with usual approaches. In most applications, this scheme even results in a significant performance

B. Thomaszewski, M. Wacker, W. Straßer / *Advanced Topics in Virtual Garment Simulation - Part I*

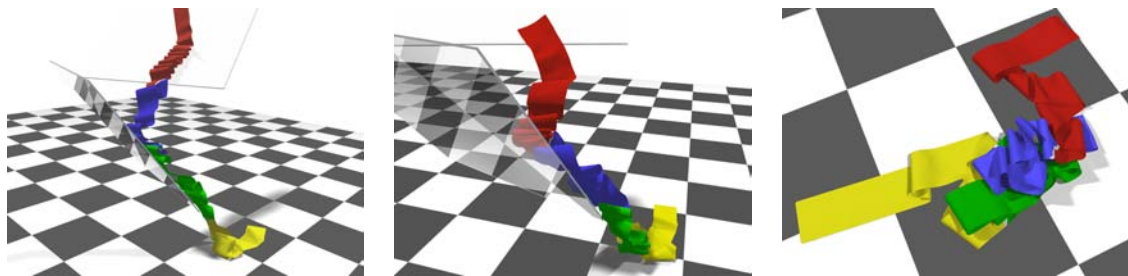


Figure 19: Three shots from a representative test scene. A long (00.5m x 2.00m) ribbon consisting of 4141 vertices falls on two slightly inclined planes and slides onto the floor. Due to surface friction complex folds are formed as it slides over the planes. This again leads to complicated self-collisions which are reliably handled by the parallel collision handling algorithm.

gain [TPB07]. This attests to the fact that temporal coherence in dynamic collision detection is a valuable source for performance improvements.

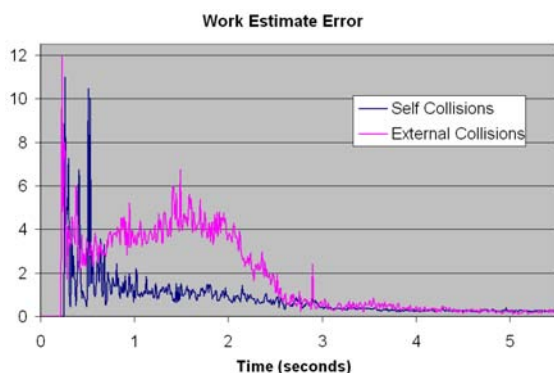


Figure 20: Work estimate error for a typical test scene. The diagram shows the deviation from the actual amount of work over time in percent. Even in this very dynamic scene, the temporal coherence is high.

Implementation For an implementation of the above algorithm, methods supporting the multithreaded programming are needed. Again, DOTS is an attractive candidate for this purpose. In order to ensure high performance on shared memory architectures, DOTS employs lightweight mechanisms for manipulating threads. Forking a thread results in the creation of a passive object, which can later be instantiated for execution. Thread objects can either be executed by a pre-forked OS native worker thread or can be executed as continuation of a thread which would otherwise be blocked, e.g., a thread reaching a synchronization primitive.

The above algorithm performs good and scales well even in challenging scenes (see Fig. 19). The actual speed-up that can be obtained depends, of course, on the specific scene and is in general the better the more work there is to be done in collision handling (see [TPB07]). A further aspect to note is

the performance of the different task creation strategies. The naive stationary approach does not scale well when compared to the randomized version, which already shows quite a good performance. The work estimate approach performs very good and can, depending on the actual scenario, outperform the randomized version.

References

- [Bat96] BATHE K.-J.: *Finite Element Procedures*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1996.
- [Bel00] BELYTSCHKO T.: *Nonlinear Finite Elements for Continua and Structures*. John Wiley, 2000.
- [BFA02] BRIDSON R., FEDKIW R. P., ANDERSON J.: Robust Treatment of Collisions, Contact, and Friction for Cloth Animation. In *Proceedings of ACM SIGGRAPH '02* (2002), ACM Press, pp. 594–603.
- [BHW94] BREEN D., HOUSE D., WOZNY M.: Predicting the drape of woven cloth using interacting particles. In *Proceedings of ACM SIGGRAPH '94* (1994), ACM Press, pp. 365–372.
- [BKLwW99] BLOCHINGER W., KÜCHLIN W., LUDWIG C., WEBER A.: An object-oriented platform for distributed high-performance Symbolic Computation. vol. 49, Elsevier, pp. 161–178.
- [BMF03] BRIDSON R., MARINO S., FEDKIW R.: Simulation of clothing with folds and wrinkles. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)* (2003), ACM Press, pp. 28–36.
- [BW97] BONET J., WOOD R. D.: *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.
- [BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of ACM SIGGRAPH '98* (1998), ACM Press, pp. 43–54.
- [BWH*06] BERGOU M., WARDETZKY M., HARMON

- D., ZORIN D., GRINSPUN E.: A Quadratic Bending Model for Inextensible Surfaces. In *Fourth Eurographics Symposium on Geometry Processing* (Jun 2006), pp. 227–230.
- [BWK03] BARAFF D., WITKIN A., KASS M.: Untangling cloth. In *Proceeding of ACM SIGGRAPH '03* (2003), ACM Press, pp. 862–870.
- [CK02] CHOI K.-J., KO H.-S.: Stable but responsive cloth. In *Proceedings of ACM SIGGRAPH '03* (2002), vol. 21, pp. 604–611.
- [CK05] CHOI K.-J., KO H.-S.: Research problems in clothing simulation. *Computer-Aided Design* 37 (2005), 585–592.
- [COS00] CIRAK F., ORTIZ M., SCHRÖDER P.: Sub-division surfaces: A new paradigm for thin-shell finite-element analysis. *Journal for Numerical Methods in Engineering* 47 (2000), 2039–2072.
- [DHv93] DEMMEL J., HEATH M., VAN DER VORST H.: Parallel numerical linear algebra. In *Acta Numerica 1993*. Cambridge University Press, 1993, pp. 111–198.
- [EB00] EISCHEN J., BIGLIANI R.: Continuum versus particle representations. In *Cloth Modeling and Animation*, House D., Breen D., (Eds.). A. K. Peters, 2000, pp. 79–122.
- [EDC96] EISCHEN J., DENG S., CLAPP T.: Finite-element modeling and control of flexible fabric parts. *IEEE Computer Graphics and Applications* 16, 5 (Sept. 1996), 71–80.
- [EKS03] ETZMUSS O., KECKEISEN M., STRASSER W.: A Fast Finite Element Solution for Cloth Modelling. In *Proceedings of Pacific Graphics* (2003), pp. 244–251.
- [EWS96] EBERHARDT B., WEBER A., STRASSER W.: A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications* 16, 5 (Sept. 1996), 52–59.
- [GH*03] GRINSPUN E., HIRANI A., DESBRUN M., SCHRÖDER P.: Discrete shells. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2003)* (2003), ACM Press, pp. 62–67.
- [GKJ*05] GOVINDARAJU N., KNOTT D., JAIN N., KABUL I., TAMSTORF R., GAYLE R., LIN M., MANOCHA D.: Interactive collision detection between deformable models using chromatic decomposition. In *Proceedings of ACM SIGGRAPH '05* (2005), ACM Press.
- [Hau04] HAUTH M.: *Visual Simulation of Deformable Models*. Phd thesis, Wilhelm-Schickard-Institut für Informatik, University of Tübingen, Germany, July 2004.
- [HB00] HOUSE D. H., BREEN D. E. (Eds.): *Cloth Modeling and Animation*. A K Peters, 2000.
- [HE01] HAUTH M., ETZMUSS O.: A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods. In *Computer Graphics Forum* (2001), pp. 319–328.
- [Kaw80] KAWABATA S.: The standardization and analysis of hand evaluation. *The Textile Machinery Society of Japan* (1980).
- [Kee99] KEELER S.: The science of forming: A look at bending. *Metal Forming Magazine* (October 1999), 27–29.
- [KHM*98] KLOSOWSKI J. T., HELD M., MITCHELL J. S. B., SOWIZRAL H., ZIKAN K.: Efficient collision detection using bounding volume hierarchies of k -DOPs. *IEEE Transactions on Visualization and Computer Graphics* 4, 1 (1998), 21–36.
- [KK96] KARYPIS G., KUMAR V.: *Parallel Multilevel k -way Partitioning Schemes for Irregular Graphs*. Tech. Rep. 036, Minneapolis, MN 55454, May 1996.
- [LAM01] LARSSON T., AKENINE-MÖLLER T.: Collision detection for continuously deforming bodies. *Proceedings of Eurographics '01* (2001), 325–333.
- [LV DY04] LEE B. C., VUDUC R. W., DEMMEL J. W., YELICK K. A.: Performance models for evaluation and automatic tuning of symmetric sparse matrix-vector multiply. In *ICPP '04: Proceedings of the 2004 International Conference on Parallel Processing (ICPP'04)* (2004), IEEE Computer Society, pp. 169–176.
- [MDSB03] MEYER M., DESBRUN M., SCHRÖDER P., BARR A. H.: Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, Hege H.-C., Polthier K., (Eds.). Springer-Verlag, 2003, pp. 35–57.
- [MKE03] MEZGER J., KIMMERLE S., ETZMUSS O.: Hierarchical Techniques in Collision Detection for Cloth Animation. *Journal of WSCG* 11, 2 (2003), 322–329.
- [MTVW*05] MAGNENAT-THALMANN N., VOLINO P., WACKER M., THOMASZEWSKI B., KECKEISEN M.: Key Techniques for Interactive Virtual Garment Simulation. In *Proc. of Eurographics 2005, Tutorial 4* (2005).
- [OBHL02] OLIKER L., BISWAS R., HUSBANDS P., LI X.: Effects of ordering strategies and programming paradigms on sparse matrix computations. *Siam Review* 44:3 (2002).
- [Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Proceedings of Graphics Interface (GI 1995)* (1995), Canadian Computer-Human Communications Society, pp. 147–154.
- [Pro97] PROVOT X.: Collision and self-collision handling in cloth model dedicated to design garments. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation (CAS 1997)* (1997), Springer-Verlag, pp. 177–189.

- [RS01] REIF U., SCHRÖDER P.: Curvature integrability of subdivision surfaces. *Advances in Computational Mathematics* 14, 2 (2001), 157–174.
- [Saa03] SAAD Y.: *Iterative Methods for Sparse Linear Systems*, 2nd ed. SIAM, 2003.
- [SFR89a] SIMO J. C., FOX D. D., RIFAI M. S.: On a stress resultant geometrically exact shell model. part i: Formulation and optimal parametrization. In *Computational Methods in Applied Mechanics and Engineering* (1989), vol. 72, pp. 267–302.
- [SFR89b] SIMO J. C., FOX D. D., RIFAI M. S.: On a stress resultant geometrically exact shell model. part ii: The linear theory; computational aspects. In *Computational Methods in Applied Mechanics and Engineering* (1989), vol. 73, pp. 53–92.
- [She94] SHEWCHUCK J. R.: An Introduction to the Conjugate Gradient Method Without the Agonizing Pain, 1994.
- [TB06] THOMASZEWSKI B., BLOCHINGER W.: Parallel simulation of cloth on distributed memory architectures. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '06)* (2006).
- [TB07] THOMASZEWSKI B., BLOCHINGER W.: Physically based simulation of cloth on distributed memory architectures. *Parallel Computing* 33 (2007), 377–390.
- [THM*05] TESCHNER M., HEIDELBERGER B., MANOCHA D., GOVINDARAJU N., ZACHMANN G., KIMMERLE S., MEZGER J., FUHRMANN A.: Collision Handling in Dynamic Simulation Environments. In *Eurographics Tutorials* (2005), pp. 79–185.
- [TPB07] THOMASZEWSKI B., PABST S., BLOCHINGER W.: Exploiting parallelism in physically-based simulations on multi-core processor architectures. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV '07)* (2007).
- [TPBF87] TERZOPOULOS D., PLATT J., BARR A., FLEISCHER K.: Elastically deformable models. In *Proceedings of ACM SIGGRAPH '87* (July 1987), ACM Press, pp. 205–214.
- [TWS06] THOMASZEWSKI B., WACKER M., STRASSER W.: A consistent bending model for cloth simulation with corotational subdivision finite elements. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)* (2006), Eurographics Association, pp. 107–116.
- [van97] VAN DEN BERGEN G.: Efficient collision detection of complex deformable models using AABB trees. *Journal of Graphics Tools: JGT* 2, 4 (1997), 1–14.
- [VCMT95] VOLINO P., COURCHESNE M., MAGNENAT-THALMANN N.: Versatile and efficient techniques for simulating cloth and other deformable objects. In *Proceedings of ACM SIGGRAPH '95* (1995), ACM Press, pp. 137–144.
- [VMT94] VOLINO P., MAGNENAT-THALMANN N.: Efficient self-collision detection on smoothly discretized surface animations using geometrical shape regularity. In *Proceedings of Eurographics '94* (1994), Computer Graphics Forum, pp. 155–166.
- [VMT01] VOLINO P., MAGNENAT-THALMANN N.: Comparing efficiency of integration methods for cloth simulation. In *Proceedings of Computer Graphics International 2001 (CGI 2001)* (2001), IEEE Computer Society, pp. 265–274.
- [VMT05] VOLINO P., MAGNENAT-THALMANN N.: Accurate garment prototyping and simulation. *Computer-Aided Design & Applications* 2, 5 (2005), 645–654.
- [VMT06a] VOLINO P., MAGNENAT-THALMANN N.: Resolving surface collisions through intersection contour minimization. In *Proceedings of ACM SIGGRAPH '06* (2006), vol. 25, ACM Press, pp. 1154–1159.
- [VMT06b] VOLINO P., MAGNENAT-THALMANN N.: Simple linear bending stiffness in particle systems. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 2006)* (2006), Eurographics Association, pp. 101–105.
- [WBH*07] WARDETZKY M., BERGOU M., HARMON D., ZORIN D., GRINSPUN E.: Discrete Quadratic Curvature Energies. *Computer Aided Geometric Design (to appear)* (2007).
- [WW98] WEIMER H., WARREN J.: Subdivision schemes for thin plate splines. *Computer Graphics Forum* 17, 3 (1998), 303–314. ISSN 1067-7055.
- [ZT00a] ZIENKIEWICZ O. C., TAYLOR R. L.: *The Finite Element Method. Volume 1: The Basis*, 5th ed. Butterworth Heinemann, 2000.
- [ZT00b] ZIENKIEWICZ O. C., TAYLOR R. L.: *The Finite Element Method. Volume 2: Solid Mechanics*, 5th ed. Butterworth Heinemann, 2000.

Advanced Topics in Virtual Garment Simulation

Part 2

Real world fabrics and Virtual Try On

E. Lyard¹, C. Luible¹, P. Volino¹, M. Kasap¹, V. Muggeo¹ and N. Magnenat-Thalmann¹

¹MIRALab, University of Geneva, Switzerland

1	Real World Fabrics (Accuracy of cloth simulations)	3
1.1	Mechanical properties of cloth.....	3
1.2	Measurement of fabric properties	4
1.3	Mapping to Computational Models	5
1.4	Multi-Layered Fabrics and Seams	7
1.4.1	Seams	8
1.4.2	Multi-layered fabrics.....	9
2	Towards an Integrated Virtual Try On Application.....	9
2.1	Parametrically Deformable Human Bodies	9
2.1.1	Geometric Deformation	10
2.1.2	Physics based deformation.....	13
2.1.3	Example Based Deformation	14
2.2	Body Animation and motion retargeting	17
2.2.1	Skeleton Animation	17
2.2.2	Body Animation.....	18
2.2.3	Motion retargeting	19
2.3	Real-time cloth simulation.....	22
2.3.1	Introduction.....	22
2.3.2	Techniques for Real-Time Garment Simulation.....	22
2.4	An integrated Virtual Try On application.....	26
2.4.1	A common platform.....	28
2.4.2	Web data management.....	29
	Web Interface.....	29
3	References.....	31

1 Real World Fabrics (Accuracy of cloth simulations)

Fabrics are complex visco-elastic materials. They must have sufficient strength and at the same time they have to be flexible, elastic and easy to pleat and shape. Their simulation is not easy, as their behaviour is difficult to describe and predict. Nevertheless, computation algorithms have been developed over many years and evolved to such a level so that today we are able to not only simulate simplified, static clothes, but also complex dynamically moving garments, in the time frame, expected by the clothing industry[VM05]. But, not only advanced computational models are responsible for precise virtual garment simulations. Also exact input parameters play an important role for a correct reproduction of the fabrics mechanical behaviour. For instance, newly-developed computation systems finally allow the simulation of the non-linear fabric behaviour; but in order to truly reflect these characteristics in the virtual computation, we have to be able first of all to measure them appropriately. Experimental values for the main mechanical and physical parameters can be derived from standard fabric characterization experiments such as the “Kawabata Evaluation System for fabrics” (KES) [KAW80] and the “Fabric Assurance by simple Testing” (FAST) [MIN95]. However, both characterisation methods have not been designed for the purpose of deriving parameters for virtual simulations.

1.1 Mechanical properties of cloth

The mechanical behaviour of fabric reflects the nature and molecular structure of the fiber material constituting the cloth. The arrangement and orientation of the fibres in the fabric structure has its influence as well. Fabric fibres can be organized in several ways. The main structures are:

- Woven Fabrics: threads are orthogonally aligned and interlaced in an alternating way using different patterns (such as plain or twirl).
- Knitted fabrics: threads are curled along a given pattern, and the curls are interlaced on successive rows.
- Non-woven fabrics: there are no threads, and the fibres are arranged in an unstructured way, such as paper fibres.

Woven fabrics are the most common type of fabric used in garments. They are relatively stiff though thin, easy to produce, and may be used in a variety of ways in clothing design. In contrast, knitted fabrics are loose and very elastic. They are usually employed in woollens or in underwear. This structure greatly influences the mechanical behavior of the fabric material, which is mainly determined by:

- The nature of the fibre: wool, cotton, synthetic, etc.
- The thread structure: diameter, internal fibre and yarn structure, etc.
- The thread arrangement: woven or knitted, and particular pattern variation.
- The pattern properties: tight or loose.

These properties determine the stiffness of a material, its ability to bend, and its visual appearance. The mechanical properties of deformable surfaces can be grouped into four main families:

- Elasticity, which characterises the internal forces resulting from a given geometrical deformation.
- Viscosity, which includes the internal forces resulting from a given deformation rate.
- Plasticity, which describes at which point of deformation irreversible material changes occur.
- Resilience, which defines the limits at which the structure will break.

Most important are the elastic properties, which are the main contributor of mechanical effects in the usual contexts where cloth objects are used. Deformations are often small and slow enough to make the effect of viscosity, plasticity and resilience insignificant. One major hypothesis is that quasistatic models in the domain of elastic deformations will be sufficient for models intended to simulate the rest position of the garment on an immobile mannequin (draping). However, when a realistic animation is needed, the parameters relating energy dissipation to the evolution of deformation are also needed, and complete dynamic models including viscosity and plasticity should be used. Depending on the amplitude of the mechanical phenomena under consideration, the curves expressing mechanical properties exhibit shapes of

varying complexity. If the amplitude is small enough, these shapes may be approximated by straight lines. This linearity hypothesis is a common way to simplify the characterisation and modeling of mechanical phenomena.

While the linear laws described in the Section 3.5, Part 1, are valid for small deformations of the cloth, large deformations usually lead to nonlinear response of the cloth. In this case the dependence between stress and strain is no more linear. This effect is usually manifested as a stiffening of the cloth as the deformation increases. This can possibly be followed by rupture (resilience) or persistent deformations when the constraints are released (plasticity). A common way to deal with such nonlinear models is to assume weft and warp deformation modes as still being independent, and replace each linear parameter by nonlinear strain-stress functions.

For the practical measurement of mechanical properties of fabrics there are many different standards and instruments. This subject is discussed in detail in the next section.

1.2 Measurement of fabric properties

Each textile possesses typical characteristics, influenced by the textiles raw material (natural fibres, synthetics, etc.), yarn structures (degree of twist), planar structure (weave, knit) and finishing treatment, which are advantageous for some types of garments, but can be unfavourable for others, regarding garment comfort. For an optimal usage of each type of fabric, the garment and textile industry invented the concept of “fabric hand”, what is an assessment process, where each textile is evaluated regarding its quality and suitability. The fabric hand attributes can be obtained through subjective assessment or objective measurement. Objective fabric characterization methods measure and relate the major mechanical properties, in order to obtain comparable information about textiles. The applied physical tests analyse and reflect the sensations felt during the subjective fabric assessment, where the textile is touched, squeezed, rubbed or otherwise handled and describe them with a numerical value[AAT02]. Important fabric hand properties are flexibility, compressibility, elasticity, resilience, density, surface contour (roughness, smoothness), surface friction and thermal attributes, which are the result of a broad fundamental research on fabric properties[PIE30][LD61]. In virtual simulations, the main imitated mechanical properties are elasticity, shear, bending, density and friction. The KES system was developed in the 1970’s by Kawabata and constituted the first standardization of objective fabric assessments. Since, this method is widely used for the objective characterisation of fabrics, as well as for studies of fabric mechanical properties. In the late 1980’s the CSIRO Association in Australia realised the importance of a commercial measurement for wool fabrics and tried to offer a simpler and cheaper alternative to KES, the FAST method. Both, KES and FAST measure the same parameters; however different measurement principles are applied. The FAST method uses simpler procedures than KES and permits only a linear interpretation of the measured data, whereas KES provides a complete stress-strain profile for all measured characteristics. The measurements of both systems are conducted in the low force area, what corresponds to loads which a fabric is likely to undergo during garment manufacturing. Alternative, more flexible measurement devices exist for the measurement of tensile and hysteresis properties.

Elasticity tests are designed in a way to return the correlation between applied forces and corresponding fabric elongations. The FAST method measures the elasticity property at one load of 100 N/m along warp and weft direction. KES tests the tensile behaviour with an increasing force of up to 500 N/m also along weft and warp direction. After the tensile load attains the maximum force, the recovery process is recorded.

During shear tests, the required forces to change the angle between the orthogonal intersecting threads of textiles to certain extend are assessed. Whereas the tensile property is more influenced by the fabrics fibre composition and the yarn structure, the shear characteristic is mainly influenced by the fabric structure. Different measurement principles can be applied. The main standard method fixes the fabric between two clamps and applies opposite forces until a maximum angle (KES [KAW80]) or maximum force is attained. Other methods measure the fabrics extension-compression in the bias direction (FAST [MIN95]) [BEH61].

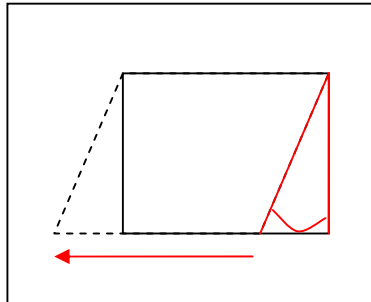


Figure 1: Angle-force method

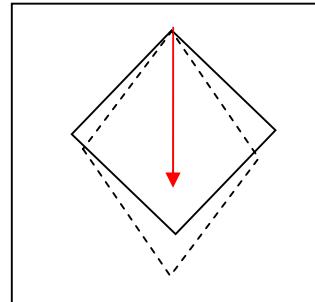


Figure 2: Extension in bias direction

Regarding fabric bending tests, there are two main categories. The first category measures the bending deformation under the fabrics own weight. The most well known method within this category is the Cantilever test, which uses the engineering principles of beam theory. A fabric is moved forward to project as a cantilever from a horizontal platform. As soon as the leading edge of the fabric reaches an angle of 41.5° to the horizontal platform the bending length is measured (FAST). Another method of this category consists in the folded loop method, where the fabric is fold back on itself and the height of the loop measured. The second category of bending tests is designed to return the moment-curvature relationship by measuring forces or moments (KES). Therefore, a fabric is fixed between two clamps and bent in an arc of constant curvature, where the curvature changes continuously and applied moments recorded.

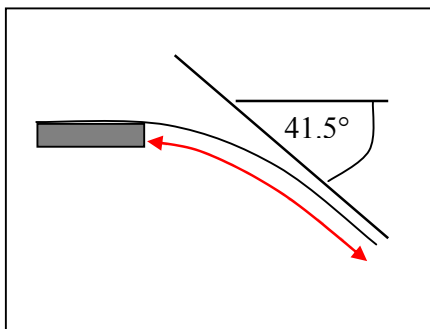


Figure 3: Cantilever method

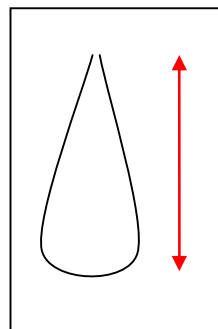


Figure 4: Folded loop method

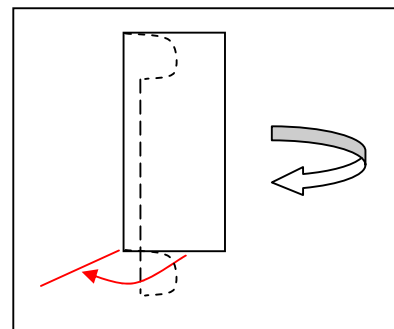


Figure 5: Moment-curvature method

In contrast to elasticity, shear and bending, the friction property is not an internal but external mechanical force, varying with each other contact material. Distinctions are made between static and dynamic friction. The static friction is related to the initial force, what is needed to overcome to start moving a material against another object or surface. The dynamic friction however occurs during the movement itself and is therefore generally lower than the initial static friction. There are several methods to assess friction. Within the standard fabric characterisation experiments (KES), the friction is measured by moving a piano-wire over the fabric at a constant force frequency. Friction is not measured by the FAST system.

1.3 Mapping to Computational Models

For virtual garment simulations not the calculated standard fabric hand values, but the actual measured empirical data is of interest, as therefrom important mechanical input parameters can be derived. For the actual derivation of fabric input parameters, a mathematical description of the measured data is needed. Depending on the complexity of the implemented computational model and the available amount of measured data, this mathematical interpretation can be linear (from FAST) or non-linear (from KES and alternative devices) derived.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

The tensile behaviour of fabrics is strongly non-linear for most textiles. Strain-stress profiles of very elastic materials are particularly characterized by a curved envelope. Versatile computational models are able to simulate the nonlinear tensile behaviour and therefore ask for an adequate input data. Linear parameters derived from the measurement data from FAST, are correct in the low force area, occurring for example during static simulations, where the fabric is basically stressed by its own weight. However, linear parameters are incorrect for the simulation of higher stresses, as referring to them much lower loads are sufficient to achieve larger fabric elongations. Moreover, used in virtual garment fitting processes, linear parameters return a wrong feedback about the garment comfort.

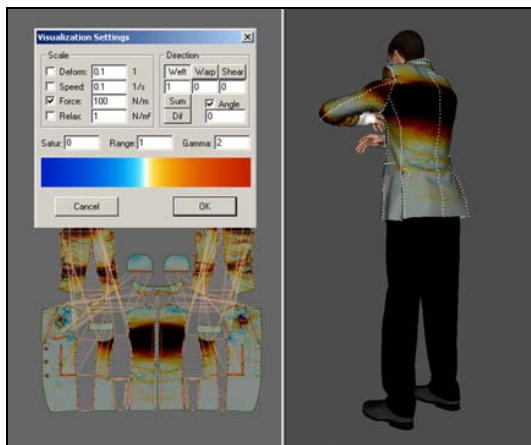


Figure 6: Simulation using KES tensile data

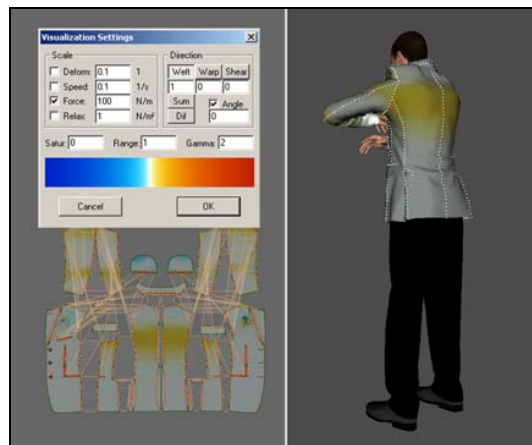


Figure 7: Simulation using FAST tensile data

At a first glance, the non-linear fabric parameters derived from the KES strain-stress envelopes seem to be better suited for the derivation of nonlinear tensile parameters. However the KES method is limited as well, as it returns the strain-stress profile only up to one maximum load. Dynamic cloth simulation is a much more complex issue. When the garment follows the movement of the mannequin, the fabric undergoes not only one but many deformations and relaxations of various low and high loads in different temporal distances. For that reason, derived parameters from KES are accurate for the one specific measured force (500 N/m), but not for various loads. Hence, for dynamic simulations, multiple load/unload experiments with different applied forces, which reflect what actually happens during the wear of a garment, are needed. Also aspects which are related to the simulation history such as plasticity and properties which are time related such as viscosity, become important input characteristics for dynamic garment simulations. Until today, the viscosity of textiles is a little investigated field of research and no standard measurement exists for the characterisation.

Depending on the type of fabric material, the shear behaviour varies from linear to non-linear. State of the art simulation systems use a nonlinear shear computation model. Therefore, depending on the fabric material, a linear or nonlinear mathematical description is needed (Figure).

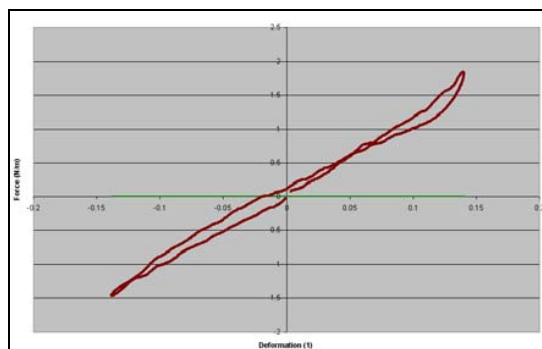


Figure 8: Linear shear strain-stress envelop

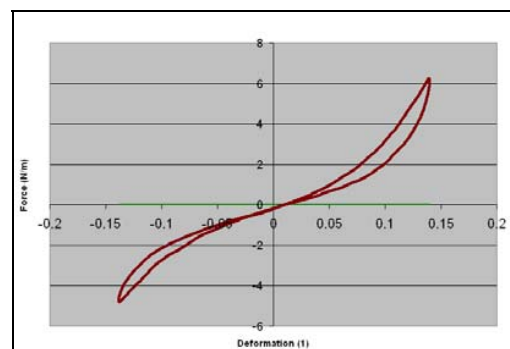


Figure 9: Non-linear shear-strain-stress envelope

Similar to the tensile property, the nonlinearity of the shear behaviour can be more accurately derived from complete strain-stress envelopes, whereas the more linear shear compartment can be accurately interpreted from single measured forces as well. However, in contrast to the tensile property, the error in the comfort feedback for simplified nonlinear parameter is smaller. This is related to the fact that regarding shear, generally lower forces are concerned.

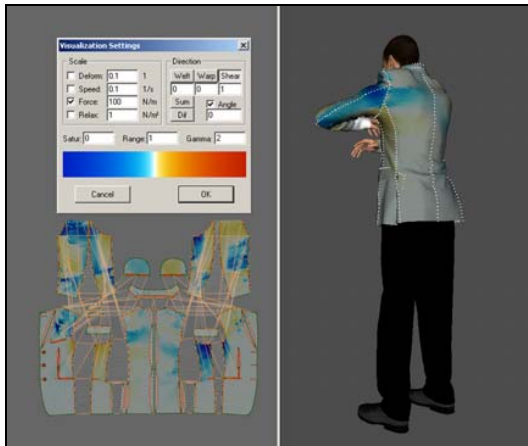


Figure 10: Accurate nonlinear shear parameter

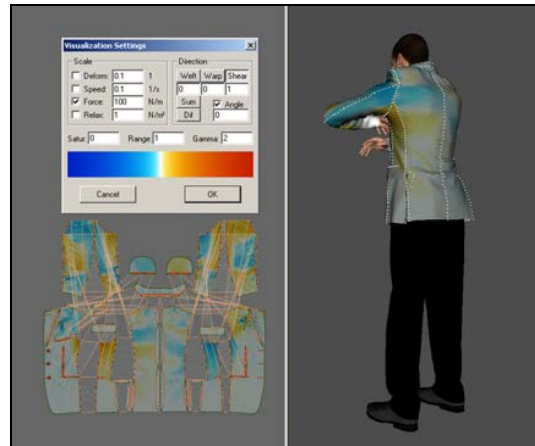


Figure 11: Simple linear shear parameter

Even in versatile computation system, the complex bending property is still linear modelled. Thus, a simple (linear) mathematical interpretation for the bending behaviour is precise enough. The bending rigidity is returned by standard measurement methods as characteristic value. As this measure is a description of the slope between two major points of the measured data, it is suited to be directly taken as linear bending characteristic. The comparison of the bending rigidity obtained from the FAST and the KES measurement systems shows a good correlation for the bending rigidity value.

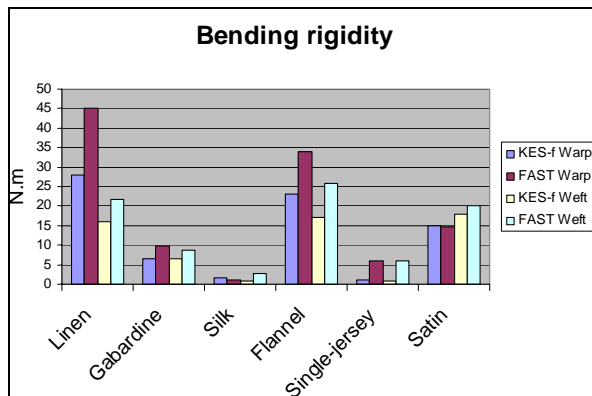


Figure 12: Bending rigidity FAST and KES-f



Figure 13: Virtual drape of the flannel fabric

1.4 Multi-Layered Fabrics and Seams

Regarding complete garments, not only the fabric characteristics are important for a mechanical accuracy and a good visual appearance. Additional clothing aspects, such as seams, interlinings and fabric fusing become important influencing factors for virtual computations and demand a separate examination.



Figure 14: men suits



Figure 15: fitting men suit



Figure 16: virtual simulated men suit

1.4.1 Seams

Conventional garments are generally not composed out of only one, but many different pieces of fabrics. On the one hand, this is due to the complexity of the shape of the human body with its curves and bulges, which need to be covered by a fitted garment. On the other hand, this is related to changing tendencies and trends, which constantly ask for new silhouettes. Hence, as a garment is composed out of multiple single surfaces and they have to be somehow connected subsequently, in order to form a complete garment. The mechanical behaviour of a single textile and the behaviour of a tailored garment out of the same fabric are different as the characteristics of the junctions of the single surface pieces influence the general comportment as well. For the combination of two fabric pieces there several different methods:

The traditional way of combining two fabric pieces is by applying sewing techniques, where two or more surfaces are linked together with threads. Hereby, distinction can be made between different types of seams, such as the plain seams, the welt seams, the welding seams, decorative seams, etc. Their mechanical characteristics vary according their amount of fabric layers and their amount stitches and topstitches. The plain seam consists of two fabric layers, the outer fabric and the seam allowance (Figure) and is mainly used for standard fabric assemblies. The welt seam is composed out of three or more fabric layers, the outer fabric and two times or more the folded seam allowance, completed by one or two topstitches (Figure). It is mainly used for parts of the garment, where a lot of abrasion is expected and also where additional stability is needed, as for example at the inner side of a pair of jeans. Modern high tech textiles, especially water proof garments are welded instead of sewed, as the stitches would impact their performance. Decorative seams are stitching in various patterns on top of the outer fabric. Whereas the decorative seams influence less the mechanical behaviour, the seams containing multiple fabric layers can change the fabric comportment significantly. Therefore, if we want to accurately imitate those parts of the garment, the seam mechanics need to be measured and accurately simulated as well. The stiffness of seams is an additional parameter, which can be specified inside the simulation application for each seam. Their characteristics can be obtained with the above described fabric measurement methods.

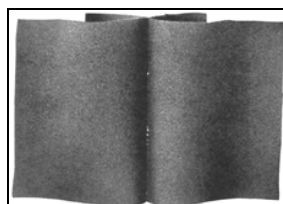


Figure 17: Plain seam



Figure 18: Welt seam with top-stitch

Another important aspect regarding accurate seaming simulations is the problematic of seam pucker. Seam pucker is the occurrence of unwanted small fabric wrinkles at a garment's seam, due to fabric gatherings caused by the sewing thread. Depending on the fabrics thickness and stiffness, this shrinkage can be up to 5% of the length of a seam. For an accurate virtual simulation, it is important to consider this parameter, as it influences the fit and especially the quality of the garment.



Figure 19: Effect of seam pucker

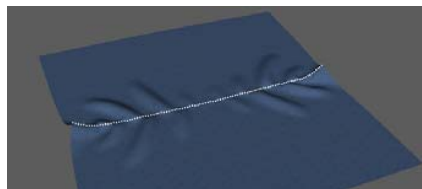


Figure 20: Virtual simulation of seam shrinkage

1.4.2 Multi-layered fabrics

In order to give to a garment more stability in some areas for functional or aesthetical reasons, a second fabric layers can be added. This second fabric layer is either permanently fixed to the outer fabric (fusing) or it is a loose additional textile (interlining). Regarding the permanently fixed fabric layer it is clear that the mechanical characteristics of the bonded combination of the two materials should be measured for their accurate virtual re-creation. However this can be easily accomplished by measuring not the single but the fused fabrics.

For the non-fixed fabric layers this is however a more difficult task. On the one hand the combination of both characteristics is needed, but on the other hand, the fabrics are single layers and for an accurate prototyping they should be treated separately. A simulation of two textile layers would cause two main problems. The multi-layer garment simulation asks for an important feature, inevitable for those challenging calculations, such as a powerful collision response method with stability and robustness. However, the high amount of polygons is visibly slowing down the simulation.

Today's collision algorithms also do not allow the simulation of various layers in small distances, such as the thickness of interlining fabric. Thus, because of the fairly unnatural distance of multilayer fabrics, the simulation looks unrealistic. Because of these two aspects it is suggested to also simulate the non-fixed fabric layers with only one virtual surface, using mechanical characteristics which are a combination of materials, even the simulation of an endless amount of fabric-layers is possible with today's simulation systems from a technical point of view.

We've just seen how to efficiently measure the physical properties of a cloth in order to produce high fidelity virtual simulations. This allows to estimate accurately the fitting of a given outfit on someone's avatar. However, if one wants to remain high fidelity, then the avatar must correspond exactly to the person's dimensions, thus the need to an efficient avatar deformation techniques, which is the topic of the next section.

2 Towards an Integrated Virtual Try On Application

2.1 Parametrically Deformable Human Bodies

3D human body models are the most important accessories of the computer graphics environments such as games, virtual modelling tools and virtual reality applications. Evolution of the computer graphic techniques and hardware technology let it possible to use more realistic models in those environments which use muscle and fat tissue deformation effects during animation. The primary characters of those environments, human body models, require specific techniques for real-time animation and realism. Because of their importance, specialized research areas focused on face, hand, skin, muscle modelling and skeleton attachment to generate realistic models which will improve the quality and realism of the environments. Human body models which are subject to all those techniques and environments are initially generated by 3D model designers or acquired by 3D body scanners like the ones from Human Solutions [HS:07]. These initial models are the template ones for the future deformation methods.

In recent 3 decades, human body deformation techniques evolved under the following three main headings: geometric, example-based, and physics-based. The geometric deformation techniques are not only the fastest but also the simplest ones. On the other hand result of the geometric deformation technique is not satisfactory to generate realistic models. Because of their computational simplicity and less requirements, it

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

is still the most preferable technique in real time animation. Example-based approaches are the most challenging field after effective use of the 3D body scanners. Based on a model database, the requested model interpolated from the database with appropriate parameters. Main constraint of this approach is that it requires number of post processed complex data as an input. Lastly, physics-based deformation is computationally the most complex method but it generates the most realistic results.

Initially, Lasseter [Las87] mentioned about the multi-layered approach for animation, which reduces the modelling complexity but increases the reality. Lately, to achieve higher degree of realism, multi-layered approach is used with the combination of deformation techniques to generate a body model with skeleton, muscle and fat tissue layers. From this point of view, Chadwick [CHP89] is one of the early initiator in this area with modelling muscle layer for deformation during animation. Recently, realistic body modelling techniques are based on multi-layered approach integrated with hardware acceleration for real-time computation.

2.1.1 Geometric Deformation

History of the geometric deformation starts with Barr [Bar81], where he applied basic transformations on super-quadrics. Recently these super-quadric ellipsoids are used to simulate the muscle effects on body model. After his successive work on super-quadrics, Barr [Bar84] developed hierarchical solid modelling operations that simulate twisting, bending, tapering like transformations on geometric objects. Blanc [Bla94] proposed the procedural generic implementation of these global deformation techniques.

Based on Barr's work, Sederberg and Parry[SP86] announced the Free Form Deformation (FFD) technique also today which is commonly used and have lots of extensions to solve specific problems related with deformation. As stated by the author "A good physical analogy for FFD is to consider a parallelepiped of clear, flexible plastic in which is embedded an object, or several objects, which we wish to deform. The object is imagined to also be flexible, so that it deforms along with the plastic that surrounds it."

In the first row, a set of original geometric models enclosed with a cube and the deformation of these models along with the enclosed cube could be observed. In the second row just the enclosing cube and the transformation over it illustrated. Using this technique with its basic form, people tried to generate complex (for that day) models.

Sederberg and Parry demonstrated a possible application of FFD method by modelling a handset from a stick. This approach used in many graphic applications for deforming complex models. One of the most interesting applications for that day is developed by Chadwick[CHP89]. Using robotics skeleton approach for animation of the human body model, he added muscle effect on top of the skeleton with FFD representation. For this animation frames, Chadwick used basic adjoining FFDs to achieve the real-time deformation. Parametric muscle deformation generated according to Denevit and Hartenberg parameters [HD55] related with the joint angles in robotics.

In the same time period, Thalmanns[MTT87] developed joint-dependent local deformation (JLD) operators to deform body model surface for animation. Each JLD operator is affecting its uniquely defined domain and its value is determined as a function of angular values of the joints under the operator.



Figure 21: JLD based animation of body models, Thalmanns[MTT87].

Since FFDs are based on a parallel piped cubic volume covering the model, they had some limitations. First, it is not possible to cover a complex model without sparse sub-volumes. Second but not the least, deformation with more freedom is not possible. Coquillart[Coq90] come with the extended version of the standard FFD method that is called ExtendedFFD. This new method uses non-parallel piped 3D lattices to include the shape of the deformation.

To have more flexibility on the deformation, Lamousin[LWN94] logically extends the FFD by mapping it on a non-uniform rational B_Splines(NURBS) and called this technique NurbsFFD. NFFD offers much more control on the model which is not achieved in the prior implementations. An interesting application of NFFD is demonstrated by the author with human leg model as an input.

Moccozet[MT97] even extends the EFFD by presenting a generalized method with techniques of scattered data interpolation based on Delaunay and Dirichlet/Varonoi diagrams. This is why it is called DirichletFFD. One of the advantages of this technique is the control of local deformations where it is crucial for 3D modelling and animation. Author implemented a multi-layer deformable hand model to simulate the intermediate layer between the skeleton and the skin.

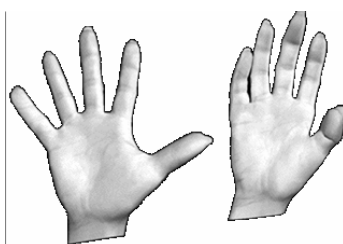


Figure 22: DFFD applied hand model deformation, Moccozet[MT97].

Originally human body model animation is generated from the point of view as for robots. This non-realistic motion implementation method improved with new deformation techniques. Early initiators of realistic motion generation, Thalmanns[MTT91], applied two methods for improving deformation during animation. For the body parts contacting with other objects, finite element method(FEM) is used, for the parts that are not contacting JLD[MTT87] used for simulating the natural behaviour of the human body model.

One of the earlier researches about anthropometric modelling of the human body model is introduced by Azuola [ABH*94]. First the human body model is segmented into groups according to the synovial [BPW93] joints. Deformation on the corresponding joint is constrained with its degree of freedom (DOF). Joints with one to three DOF selected with their movement limitations. According to the anthropometric measurement database, anthropometrically segmented body model deformed with FFD methods.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

Jianhua[JTT94a] used a new approach for body model representation. Instead of polygonal representation, model divided into slices. Each slice is defined with parametric curves namely b-splines. Depending on the neighbouring joints distance and the normal vectors of the slices, collision prevented. Radius of each contour is scaled to achieve the muscular deformation effect.

Jianhua[JTT94b] extended contour based representation of body model with metaballs[HMT*85]. Like cylindrical representation, metaballs are used for smooth and detailed modelling.

One of the pioneering work in the anatomical modelling field is recognized by Wilhelms[WG97] work. Apart from body mesh, underlying muscle, skeleton and generalized tissues are also parametrically modelled according to the anatomical principals. Though detailed modelling of underlying layers of the skin, animation became more realistic compared to its preceding approaches. By his complete multi layered modelling work, Wilhelms make a big impact on the future body modelling techniques.

Similarly Scheepers [SPCM97] developed multilayered anatomically deformable models in the same time period. Tubular shaped bi-cubic patch meshes capped with elliptic hemispheres attached on both end of the corresponding skeleton. Depending on the corresponding joint angle, underlying muscle and fat tissue structures deformed the skin surface.

Douglas [DMS98] developed a system based on the anthropometric statistic of human face measurements in a population to model the face. Statistical results are used as geometric constraints on the specific part of the parametric surface. Because of its parametric property, variational modelling [GC95] technique is selected to deform the surface which satisfies the underlying constraints.

Singh [SF98] introduced a new deformation technique based on free form curve. Author inspired by armatures used by sculptures. In this approach with a single wire, direct manipulation for deformation on the model is possible. Also interacting multiple wires with accumulation is possible to generate more complex deformations.

While most of the geometric techniques inspired from FFD, Singh[SK00] in contrast proposed a surface oriented FFD. He implemented a control surface defined by a distance function around the surface. This new approach allows localization of the control lattice complexity for detailed deformation. Furthermore, this make the approach ideally suited to the automatic skinning.

Marinov [MBK07] present an efficient technique based on multi-resolution deformation on a high resolution meshes. Proposed deformation process handled in the GPU with the help of pre-computed deformation operators and the gradient information. By this way dynamic 3D model deformation succeeded with several times faster computation. After the deformation, for calculation of the new normal field, he provided neighbouring vertex and other required information as vertex attributes. For deformation process, basically affine transformation operators applied on the control points.

In contrast with the previous work on muscle modeling, Dong et al[DCKY02] take a further step on realistic muscle modelling. Using Visible Human Dataset[VH:95], horizontal human body slice images are post-processed to extract main muscle contours. After contour extraction, cubic B-spline surfaces are built to fit on the contours. Dong et al used a new deformation procedure for dynamic muscle shape forming. Previously generated muscle geometries attached on the skeleton, then according to the action lines and the joint angles new position of the muscle endpoints calculated. With the new muscle endpoints, direction of the tangent vectors reoriented to change the shape of the geometry.

One another approach for human body deformation is applying the sweep based method on the limbs [HYKJ03]. In this approach each limb is approximated by swept based ellipsoid which changes its size as it moves through the limb. Transition from each joint, the ellipsoid changes its orientation through the new

one. All ellipsoids interpolated to fit in the original model. Resulting approximated model processed with displacement map to reflect the original shape because of its smooth behaviour.

Recent geometric muscle deformation technique presented by Pratscher [PCLS05] by using multi-shell structured ellipsoids. Each shell has its own level of hardness for deforming the attached skin. Using number of heuristics, body mesh is partitioned into segments to determine the location of the muscles. User has the opportunity to customize the muscle connections, size etc. and those parameters saved under a musculoskeletal template then which can be applied on different bodies. Muscle mapping on the body with single or multiple pose is also considered.

Kavan et al [KZ05] proposed a skinning approach which is alternative to the standard linear blend skinning. They use spherical deformation methods to improve the side effects of the previous method. Main advantage of the method is to spherically deform the joint parts to overcome the collapsing-joint artifacts. Theoretical properties of the rotational interpolation are the basics of the spherical blend skinning and computationally require the similar system resources as the previous method.

Hyun [EHW*07] extended his previous sweep based approach for body deformation by adding GPU assisted collision detection for limbs during deformation. Given polygonal mesh approximated by control sweep surfaces and according to the joint angle changes, sweep surfaces deformed and overlapping parts are blended. Some anatomical features like elbow-protrusion, skin folding etc. are emulated in the GPU.

Cross sectional representation of the body model used by Zhaoqi et al [ZTS06] to generate skeleton-driven deformations. Even this approach seems to be very similar to the one proposed by Shen [ST95], Zhaoqi et al generalizes the cross sectional contours to preserve the original details of the body.

Park et al [PH06] developed a novel approach for skin deformation with visually realistic results. They attached very large number of markers (~350 markers) on a human body. With those markers they motion captured the body. Resulting data applied on a virtual model to reflect visual details such as muscle deformation. Since it requires huge number of markers it is not practical and easily repeatable but the good point would be that once the data captured it can be applied to other models.

Li [LW07] presented an approach for automatic creation of the human body skeleton with controllable parametric structures. Anatomical features of the body determined according to the contour based model data. According to the extracted features, appropriate size skeleton model is constructed. Over the skeleton model, multi resolution parametric sweep based surface is generated for controllable body model.

One of the earlier surveys on human body model CAESAR is also the subject to human body deformation field. Recently Wang [WR07] used this database for animating the static scanned data. This database is not only consisting of body models but also the corresponding landmarks and feature points of individual models. From this point of view, Wang et al attached h-anim skeleton on the models by using the landmark information. They developed a web based system for parametric manipulation of the models in this database.

Recent work for body animation is proposed by Aguiar et al [dATSS07]. Using 3D human body scanner, they rapidly attach a skeleton on the scanned model for animation. Their system tested with markers and marker free scanned data. They employ a Laplacian mesh deformation schema which is based on the marker information to compute the poses of the model.

2.1.2 Physics based deformation

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

One another approach for anatomically-based muscle modelling is coming from Nadel[NT98a]. In his approach muscles are attached on to the skeleton as usual but the muscles are represented by two layers called the action lines and the volumetric forms. Modifications on the action lines are automatically captured and used for further muscle deformation. Also mass spring systems are used for muscle modeling.

In his latter work, Nadel[NT98b] extended his previous approach by modelling the muscles with physical methods. To simulate physical deformation, he used mass-spring system with new kind of springs called “angular springs”. By this way he achieved more control over the muscle volume to generate realistic visual effects.

Aubel[AT] proposed a new automated skin deformation method based on multi-layered human model. Deformations applied by considering the physiological and anatomical structure of the model’s skin. 1D mass-spring systems used to achieve the muscle motion and deformation. Also the muscle layer covered with viscoelastic fat layer to generate dynamic effects during the animation. Based on this implementation Aubel[AT01] extended his method to let the designers have full interaction on the model by dynamically changing the muscle shape.

Capell [CGC*02] at al implemented dynamic skeleton driven deformation by using equations of motion of elastic solids. Volumetric finite element mesh is used to perform the deformation with few constraints. i.e. line constraints for skeleton representation and the edges of the volumetric mesh coincident with the bones. They linearize the non-linear motion equations to be solved over volumetric regions associated with each bone.

One another physics based muscle modelling approach presented by Teran at al [TSB*05]. It is one of the most detailed modelling approaches where muscle material heterogeneity also considered. Segmented visible human data set [VH:95] used to extract the real muscle shape format which will be the base envelope for the proposed model. Once the muscle shape constructed, it is filled with tetrahedrons to apply physical behaviour by means of FEM. Because the tendons and the muscle belly behave differently in its density, different parts of the model simulated with appropriate tetrahedron material properties. Result of the simulations became much more realistic then the previous developments.

Application of physical phenomena to improve the flesh parts of the body not only constrained to the muscles. Larboulette[LCA05] proposed a fast and simple technique to enhance the standard character skinning method by adding dynamic skin effects through the underlying skeleton. Method called rigid skinning is applied on the existing skinning information without modifying its kinematic deformations or other post processed data. Efficiency of this technique is coming from its real-time applicability of visco-elastic properties on the body parts.

One of the recent attempts for realistic muscle modelling is represented by Zhou at al [ZL05]. Geometric and physical modelling techniques are combined to simulate real muscle effect by using NURBS based Galerkin method. From visible human dataset[VH:95], 3D reconstruction of the muscle shape achieved by fitting the result into a Nurbs form. With simple FEM properties and a few control points it is possible to deform the muscle model during the animation.

While most of the researches are focused on the muscle modelling for realistic skin deformation, Capell at al [CBC*05] showed that the character rigging is possible to animate the model with high level parameters. Dynamic elastic bodies are subject to the application area of this method. Using force based rigging to deform the character, underlying collisions handled with a novel approach proposed by them.

2.1.3 Example Based Deformation

Since the human body scanners became widely used, it is possible to generate visually realistic models. Recent body scanner systems have the capability of capturing high resolution data along with the texture information. Apart from such benefits, these systems generate static models which require post-processing stages to let them available for further deformation and animation. In the last decade we observe key attempts to solve such problems and benefit from scanner systems high resolution data generation capabilities. An early research on this field carried out by Seo [SMT03]. She proposed a method consisting of 3 main steps for parametrically synthesize visually realistic human body models: Pre-processing the 3D scan data, function approximation of the parameterized modeller and the runtime evaluator. Together with set of scanned human body data, template model with appropriate skeleton attachment designed. For parametric deformation, specific landmarks over the template model determined. Same landmarks also specified on the scanned data set. Regarding the user specified model parameters appropriate scanned data found from the database. Finally template body model mapped on the resulting scanned data with skeleton adjustment and displacement mapping. Resulting mesh processed with refinement operator to handle irregular deformation of the template one while mapping stage. Seo's extended and complete version of the work can be found in Seo[SMT04].

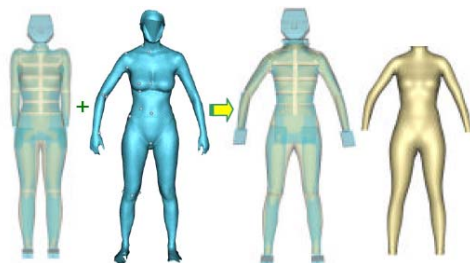


Figure 23: Mapping template model on the scanned data, Seo[33].

Similarly Allen et al [ACP03] developed an example based approach to transfer template body model on to a scanned data. They used 250 scanned data to demonstrate parameterization and reconstruction. With this approach it becomes possible to analyze the human body modelling applications like texture transfer, skinning transfer, shape analyze etc.

One novel extension to Seo[SMT04]'s method is attaching the real human skeleton model. Magnenat-Thalmann[MTYCS04] extracted the skeleton information from the visible human dataset[VH:95] where the proceedings extracted just the muscle tissue from the same dataset. With this approach instead of attaching and representing chopstick like skeleton models, author achieved to attach the skeleton with real characteristics. For different size models, skeletons scaled with the similar operations explained in their previous work.

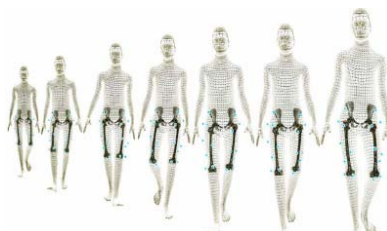


Figure 24: Real like skeleton attachment, -Thalmann[MTYCS04].

Anguelov et al[ASK*05] presents a pose space deformation of the body model by using body scanner. Scanning the same person with different poses, deformation space generated. Two different body deformation models learned from the generated data: rigid and non-rigid. Proposed framework generates the desired body shape according to the parameters like angles of the joints and the eigen-coefficients of the shape. Different pose deformations captured from an individual is also applicable to others.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

Since the hardware accelerated computation becomes more popular, researches shifted on the GPU based deformation techniques. Rhee et al [RLN06] is one of the earlier researchers who proposed a real-time weighted pose space deformation technique. From a sufficient set of example of an articulated model they interpolate the displacements. Regarding this information skinning deformations parallelized on GPU to take the real-time efficiency.

After that section, we know how to deform a 3D character so that its dimensions match several criterions, for instance match someone's sizes. It is very likely that this person, who wants to try out garments, will want to see his/her avatar in motion. This isn't a trivial stage because as the avatar was deformed, its animation must be adapted accordingly. This problem is addressed in the next section, which focus on body animation and retargeting.

2.2 Body Animation and motion retargeting

Before dressing up a virtual character it must first be animated appropriately. This must be done in such a way that makes the cloth simulation possible. For instance, time derivatives of the body motion must be available so that the cloth can behave accordingly. The geometry of the body must be deformed smoothly so that the collision detection remains consistent, and the tessellation has to be uniform in order to maximize performances.

The most widely adopted way to create a virtual human goes through the modeling of a skeleton. This skeleton is the actual animated entity, and the character shape is deformed according to the skeleton pose.

2.2.1 Skeleton Animation

Skeletons that are used in character animation were conceptually borrowed from robotics. They are modeled through a collection of rigid links (or bones) connected together by joints (Figure 25). Each joint can have up to 6 degrees of freedom (DoF): 3 translations and 3 rotations, but they usually have between 1 and 3 rotational DoF in order to mimic the human skeleton. Skeletons are a rooted hierarchy, i.e. it starts from one node which has 6 DoF and goes hierarchically through the skeleton until an end effector is reached. This means that changing the orientation of a joint will modify the configuration of all the bones which are placed further down in the hierarchy. This root node is meant to place the character in the 3D scene, while the subsequent nodes will adapt its posture.

Such a model makes possible to manipulate limbs by simply tweaking 1 parameter value. For example, modifying the orientation of the shoulder joint will move the entire arm.

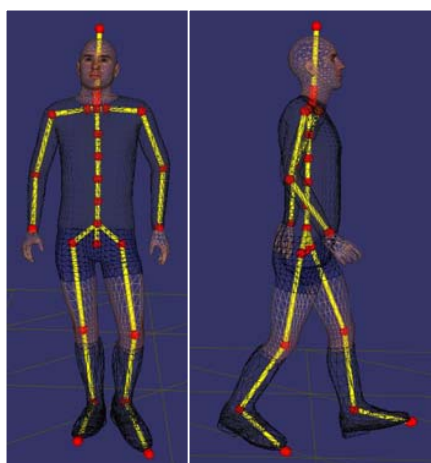


Figure 25: A virtual character in wire frame and the skeleton used to animate it. In yellow are the links (or bones) and in red the joints

There are various ways to create a skeletal animation. The most basic one, which is still widely used, is hand animation [Las87][Lass94]. A skilled animator will animate the skeleton using key frames, by carefully tuning the poses of the skeleton over time. This is a time consuming process, but it provides a total control over the final result.

Another way to generate animations is to use dynamic simulations. It works very well for motions without a specific goal, such as ragdolls [Kok04]. However the results are quite robot looking when addressing motions with a specific goal [HWBO95][YLS04]. The reason why these motions have a robotic look is because they all rely on finite state machine in order to convert the original goal to dynamic impulses, very much like the robotic community does. One way to overcome this issue is to add physics on top of a very basic motion. For instance, Liu and Popovic [LP02] proposed a framework which takes a collection of basic postures to be achieved by a character, and outputs a high quality motion by taking the physics into account. These methods have the advantage of relying solely on software in order to generate

the motion. It saves time compared to traditional animation, however such systems are very complex to develop, which makes their use quite limited.

The last way of creating motions is to record it directly on a real subject. This process is called motion capture (mocap), and it is probably the most widely used for commercial productions. Indeed, because the motion comes from a real performance, it ensures the highest level of realism for the resulting animations. There are several class of systems, and even though the most common systems are optical [Vic][MoAn] (Figure 26), several other approaches exist, such as magnetic trackers [Pol][Ass], exoskeleton [Meta] and more recently markerless captures [Orga] became commercially available.



Figure 26: A Vicon MX camera, with infrared emitters and filter

All these systems deliver different kind of data, and a post-processing stage must take place between the capture and character animation in order to convert the captured data to a suitable format. Optical mocap only records the location in 3D of reflective markers placed on the subject's body. These markers do not exactly reflect the bones motion, thus it must be estimated from the markers data only. This is done via an optimization process which places the skeleton in the configuration that better match the markers positions. In order to do so, several commercial packages are available, would they be delivered with the mocap hardware [VIQ][MoCal] or purchased separately [MoBuild].

2.2.2 Body Animation

Once the skeleton is animated, the actual character's shape must be deformed accordingly. There exist a wide variety of approaches that address this problem with various deformation results and performances. The most widely used, even though quite old, remains linear blend skinning [Magn88]. It simply blends together the displacement of a vertex, as if it would be rigidly attached to a bone. It requires a bind pose, which gives the relative displacement between the character mesh and the skeleton, along with some attachment data that defines which bones must be taken into account to deform a vertex. It is formulated as follow:

$$v' = \sum_i w_i M_i B_i^{-1} v$$

With v the initial vertex position in bind pose, B_i the bind transformation matrix of joint i , M_i the current transformation matrix of joint i , and w_i the weight of joint i ($\sum_i w_i = 1$). It has the advantage of being very simple to formulate and implement, however it generates artifacts when the deformations become too large. Indeed, $\sum_i w_i M_i B_i^{-1}$ no longer is a rigid transform matrix, thus artifacts such as collapsing elbow and candy wrapper may appear.

The hardest part of this method is to define the blending weights so that the deformations are smooth and purposeful. Various CG packages [Max][Digi][Maya] provide interfaces to define the weights, which still must be tuned by an experienced designer.

In order to address the artifacts caused by linear blend skinning, various approaches were proposed. Wang et al. [Wang03] puts variable weights depending on the current bones transformations, Kavan et al. [Kavan05] interpolate the transformations instead of the final vertex positions, and Mohr et al. [Mohr03] introduces pseudo joints which aim at refining the skeleton motion.

More recently, Kavan et al. [KCZO07] proposed to use dual quaternions [Cliff82] in order to represent translations, which makes possible to efficiently blend translations and rotations without creating a degenerated transform.

Dual quaternions are similar to regular quaternions, i.e.

$$\hat{q} = \hat{w} + i\hat{x} + j\hat{y} + k\hat{z}$$

only the coefficients w, x, y and z are dual numbers, i.e. they're of the form

$$\hat{a} = a_0 + \varepsilon a_\varepsilon \text{ with } \varepsilon^2 = 0$$

A translation (t0, t1, t2) can be represented by the unit dual quaternion

$$\hat{q} = 1 + \frac{\varepsilon}{2}(t_0i + t_1j + t_2k)$$

and a rigid transform by the product $\hat{q}q_0$, q_0 being a unit quaternion representing a rotation. The linear blending thus simply becomes:

$$\frac{w_1\hat{q}_1 + \dots + w_n\hat{q}_n}{\|w_1\hat{q}_1 + \dots + w_n\hat{q}_n\|}$$

\hat{q}_i being the dual quaternion representing the transformations to be taken into account, and w_i the corresponding blending weights.

This method has the advantage of taking care of the artifacts usually introduced by linear blend skinning, while remaining fast due to the use of quaternions for the interpolation.

Other trends for skin deformation exist, such as physically based [Hua06][Cap05][Guo05] or example based [Kry02][Sloa01][Rhe06][Jam05][Par06], however due to their complexity and computational cost they aren't well suited for real-time animation, and will not be reviewed here.

2.2.3 Motion retargeting

Within the context of a virtual try on application, a given motion will probably be used in order to animate various sizes of body. Because each body is meant to reflect the physiological features of the user, it is subject to non uniform scaling of its limbs, along with a drastic change of the limbs shapes. This calls for the use of a motion retargeting algorithm, because one given motion can correctly be applied to only one body which resembles the subject onto whom it was captured.

In most applications requiring motion retargeting, the virtual character must evolve in a surrounding environment, and possibly interact with it. Dedicated solutions, making use of either inverse kinematics [TGB00][BCHB03] or global optimization [Glei98][LS99] were proposed. All these methods make use of constraints, which are to be enforced while remaining as close from the original motion as possible.

For a virtual try on application, the goal here isn't to modify the motion so that it complies with a given set of constraints, but rather to change it in such a way that it remains as close as possible from the original, while removing ugly artifacts like foot skating. As this problem is included in the more general approaches cited above, they still can be used. What they do is to minimize a set of parameters while enforcing a set of constraints (foot planting, interaction with the environment...). This goal can be formulated as follow:

$$\begin{aligned} & \text{Minimize} && f(x) = \frac{1}{2} x^T M x \\ & \text{Subject to} && c_i(x) = 0, i \in N_e \\ & && c_i(x) > 0, i \in N_i \end{aligned}$$

With M a diagonal matrix defining a weight on each parameter: the bigger the weight of a parameter, the harder it is to modify it. x the parameter vector, c(x) a set of constraints to be satisfied.

All the high frequency features of a motion clip should be kept by this optimization process because they are what makes a motion natural looking. Thus, the vector x isn't composed of the actual parameter values of the joints, but rather of spline control points which are used to defined the changes from the original parameter curves. Thus, the control points spacing will determine the minimal frequency that is to be added to the motion signal. If they're too far apart, then the retargeting goal might not be feasible, while if they're too close they may add high frequency features and degrade the quality of the animation. A good spacing for the control point is between 2 and 10 frames.

Constraint optimization algorithm are quite tricky to implement, but fortunately it is possible to convert this problem into an unconstrained one by using the penalty method to handle the constraints. The above objective function thus becomes

$$g(x) = f(x) + \sum_{i \in N_e} w_i c_i(x)^2 + \sum_{i \in N_i} w_i (\min(c_i(x), 0))^2$$

With w_i constraints weight used to put more importance on such or such constraints. This can be minimized using a regular conjugate gradient method.

Besides regular kinematics constraints, one may wish to enforce dynamic constraints as well, so that the resulting motion better matches the body onto which it's being applied. Several approaches were proposed so far [Pop99][ALP04] which rely on global optimization for performing the adaptation. A recent work from Tak et al.[TK05] used a Kalman filter in order to modify the motion and enforce physical consideration. They used a measurement model H composed of a collection of functions taking into account the constraints imposed by the user:

$$Z = \begin{bmatrix} H_K(q, \dot{q}, \ddot{q}) \\ H_B(q, \dot{q}, \ddot{q}) \\ H_T(q, \dot{q}, \ddot{q}) \\ H_M(q, \dot{q}, \ddot{q}) \end{bmatrix}$$

With HK the kinematic constraints (e.g. specify an end effector location), HB the balance constraint (i.e. make the zero momentum point lie within the supporting area), HT the torque constraints (i.e. this muscle can exert at most this force), HM the momentum constraints (for flying phases).

As the entire motion clip is already available, the process model simply is the value of the motion parameter at time t_k , and it doesn't depend on the previous state:

$$\hat{x}_k^- = [q_k \dot{q}_k \ddot{q}_k]$$

The prediction consist of taking a measurement \hat{Z}_k^- calculated by taking samples around \hat{X}_k^- and making them go through the measurement model H.

The final update is eventually given by :

$$\hat{X}_k = \hat{X}_k^- + K_k (Z_k - \hat{Z}_k^-)$$

With K_k the kalman gain, and $Z_k = H(\hat{X}_k^-)$.

This formula quite differs from the regular kalman update, and the details of the calculation can of course be retrieved from the original paper.

Such framework is quite complex to develop, and a fair amount of user interaction is required for each motion clip in order to clearly define the constraints. Moreover, unless the character drastically changes from the original captured subject, the physically related artifacts aren't much visible. Thus, the very first problem that must be taken care of is foot skating because it is most visible to the casual eye. Kovar et al.[KSG02] used once again global optimization for addressing this issue while Glardon et al.[GBT06] used inverse kinematics in order to deal with it. However, these approaches only take the skeleton into account and with our virtual try on application in mind, it seems important to also consider the skin motion [LM07]. The goal here isn't to enforce the feet at a particular location, but rather to keep the motion as close from the original one as possible while getting rid of the foot skating. In order to do to, and assuming that the part of the foot sole that must remain planted is known for the entire timeline, the appropriate displacement ΔR_t of the root node of the skeleton can be calculated as follow:

$$\Delta R_t = o(v_i, t) - o(v_i, t + \delta) + o(v_j, t + \delta) - o(v_j, t + 1)$$

With $o(v,t)$ the offset at time t of vertex v from the root, in world coordinates, v_i the vertex of the character's mesh that must remain static until time $t + \delta$, and v_j the one to remain planted after that time.

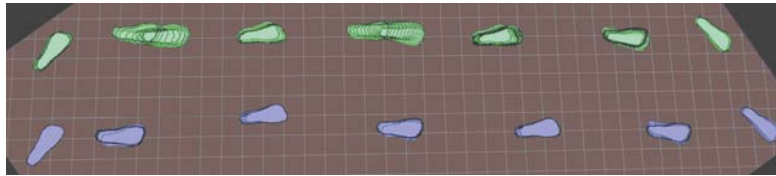


Figure 27: Footprints left by a virtual character before (top, in green) and after (bottom, in blue) foot skate cleanup.

This formula works very well for the horizontal translation of the character, as it introduces a drift according to the amount of foot skate that is present in the animation (Figure 27). However, in the vertical direction it also introduces a drift, due to the inaccuracy of the animation. In that direction, and as the motion of the character isn't to be modified, we can minimize the penetration of the feet into the ground and if the animation of the character is correct no penetration will remain, regardless of the character scaling.

Considering the heights h_t of each static point throughout the animation, the offset to be applied to the root

translation so that their mean becomes zero simply is
$$\Delta H = \sum \frac{h_t}{N}$$
, N being the total number of samples.

Now considering \bar{H} the average root height over the animation, its actual height H_t over the animation can be written as $H_t = \bar{H} + r_t$.

The scaling factor α which applied to the offsets r_t minimizes the variance σ^2 of the static points around the ground floor is given by:

$$\alpha = -\frac{\sum r_t \cdot (\bar{H} + l_t)}{\sum r_t^2}$$

Now that the avatars are able to walk around, they must be actually dressed up. High quality simulations of garment are very heavy in terms of computations thus they do not perform in real time. The next section outline how it is possible to adapt existing algorithms for garment animation, so that they perform with less accuracy, but in real-time.

2.3 Real-time cloth simulation

2.3.1 Introduction

One of the most challenging areas in research is in the development of a robust methodology for simulating clothes in real-time. In order to define a cloth simulation system that is able to simulate complex garments realistically, whilst maintaining a reasonable computation time, a deeper study of the cloth model and the identification of its behaviour at different levels are necessary.

This study is not intended to integrate yet another more precise physical model of garment behaviour, but rather focus on the real-time constraints for the simulation and the visual cloth motion features to which an observer is sensitive. Most of the existing approaches use a general-purpose simulation method using collision detection and physical simulation for the whole garment. Unfortunately, simulations that simply calculate all potentially colliding vertices may generate a highly realistic movement, but do not provide a guaranteed frame time. A new simulation model should be implemented that avoids heavy calculation of the collision detection and particle system wherever possible.

Indeed, the whole cloth does not need to be simulated with a general-purpose simulation method; instead many optimizations can be made. For example, the trouser will never collide with the arms. Collision detection may be simplified by restricting the collision detection to only potentially colliding surfaces. Also, stretched garments do not need to be simulated with a complex physical method. It can be simply simulated by keeping an offset between the garment and the underlying skin surface. Indeed, the computation cost can be greatly reduced by making use of predetermined conditions between the cloth and the body model, avoiding complex collision detection and physical deformations wherever possible.

2.3.2 Techniques for Real-Time Garment Simulation

Fast Cloth Simulation

The first approach of real-time garment simulation is to optimize usual cloth simulation methods for better performance. This is usually carried out through the use of particle systems, which allow simple computation of approximate mechanical models.

Spring-mass systems are typically used in this context. When high accuracy is still needed, some particle systems allow the expression of viscoelastic materials with the accuracy of continuum mechanics [ETZ 03] [VOL 05], as described as follows.

Fast and Accurate Particle Systems

The goal of this model is to simulate the nonlinear and anisotropic behavior of cloth materials, which are typically described as strain-stress curves measured along the weft, warp and shear deformation modes. The major challenge is to find the best compromise between the high requirement for mechanical accuracy (quantitative accuracy with anisotropic nonlinear strain-stress behavior) and the drastic performance requirements of real-time and interactive applications.

One of the most efficient solutions is to take advantage of the particle system described in [VOL 05]. The mechanical model takes advantage is indeed based on continuum mechanics, but is still a particle system. Hence, it follows many of the properties initially found in finite elements [IRV 04]. Basically, the system evaluates the strain of each triangle element according to the position and speed of the particles, then uses the mechanical properties of the material for computing the stress of the elements, and converts back the stresses into equivalent particle forces.

While this scheme has strong analogies to first-order finite elements, we have however carried out some developments aimed at vastly improving the computation speed without too many sacrifices in the accuracy. Among these developments, computational simplifications are obtained by avoiding the computations required for the linearization of the strain and stress tensors [ETZ 03]. Furthermore, an adapted accurate computation of the Jacobian is implemented for ensuring numerical stability even in very severe deformations [CHO 02].

The main interest of this computational process is to offer very good computational performance while handling materials that possibly have nonlinear and anisotropic strain-stress laws, with accuracy in par with finite-element models. Yet, our system offers all the performance and flexibility related to particle systems, particularly through the possibility of handling directly geometrical constraints such as collisions.

Bending stiffness should also be considered. However, bending forces are quite low in actual cloth materials, and through the use of large elements, it becomes quite useless to waste computation time evaluating forces that have almost no effect in the simulation. Still, when stiff bending forces are to be considered, new fast linear bending simulation schemes [VOL 06] may offer a very good computation compromise.



Figure 28: Accurate models of nonlinear anisotropic viscoelasticity are required to applications such as interactive prototyping.

Efficient Numerical Solvers

Numerical integration is another key issue to fast and efficient particle systems. While explicit methods guarantee dynamical accuracy at a very high computational time usually incompatible with real-time applications, implicit methods allow moving the trade line toward much higher computation performance at the expense of accuracy.

High-performance solvers [EBE 00] [HAU 01] [VOL 01] now focus on implicit methods for attaining the performance required for interactive models [COT 99] [JAM99] [HAU 03] [MEY 01] [MUL 00] [MUL 02]. However, their computational speed is obtained either through large compromises on the accuracy which, for low-order methods, affect the dynamic cloth motion realism through numerical damping, and for high-order methods, compromise the stability.

Since real-time application might deal with erratic animation behaviors (noisy motion tracking, skipped frames, etc), the BDF-2 method [HAU 01] might not be the best candidate, as this 2nd-order method might exhibit stability issues under severe deformation conditions particularly when dealing with highly nonlinear mechanical behaviors (collisions). Instead, adaptations of the simple Backward Euler method [VOL 01] seem to offer the best unconditional stability that is really necessary in the context of real-time applications. Meanwhile, accuracy can be significantly improved through the use of the Backward Midpoint variant [VOL 05].

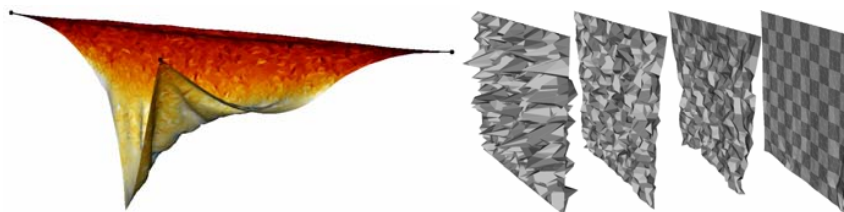


Figure 29: Stability tests on extremely deformed objects, and numerical convergence tests.

Collision Handling

Collision detection is usually one of the bottlenecks in real-time animation. The problem is particularly acute in the case of clothes because these objects are highly deformable. The most appropriate general solution is to use Bounding Volume Hierarchies, which take advantage of small-scale motion consistency of the objects during motion [MEY 00]. Specific acceleration techniques might approximate collisions and use graphics hardware to compute collisions on bump maps [VAS 01].

However, it is quite unlikely that general collision detection schemes offer adequate performance in the context of real-time simulation. Therefore, context-specific optimisations are usually required, for instance

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

by restricting collision detection only to the surface regions that have large probability to collide, and ignoring self-collision detection.

Integrating Body and Garment Animation

The main idea of hybrid approaches is to employ fast specific simulation techniques depending on the simulation context of particular regions of the cloth surface. A typical example, described in [COR 02], is described in the following.

The Hybrid Approach

When observing a garment worn on a moving character, we notice that the movement of the garment can be classified into several categories depending on how the garment is laid on and whether it sticks to, or flows on, the body surface. For instance, a tight pair of trousers will mainly follow the movement of the legs, whilst a skirt will float around the legs. The first part of the study is to identify all the possible categories:

- * Garment regions that stick to the body with a constant offset. In this case, the cloth follows exactly the movement of the underlying skin surface.

- * Garment regions that flow around the body. The movement of the cloth does not follow exactly the movement of the body. In case of a long skirt, the left side of the skirt can collide with the right legs.

- * Garment regions that move within a certain distance to the body surface are placed in another category. The best examples are shirtsleeves. The assumption in this case is that the cloth surface always collides with the same skin surface and its movement is mainly perpendicular to the body surface.

These three categories are animated with three different cloth layers. The idea behind the proposed method is to avoid the heavy calculation of physical deformation and of collision detection wherever possible, i.e. where collision detection is not necessary. The main interest of our approach is to pre-process the target cloth and body model so that they are efficiently computable during runtime. The skin and the garment are divided into a set of segments and the associated simulation method is defined for each. For each layer, we propose solutions and explain why they have been chosen.

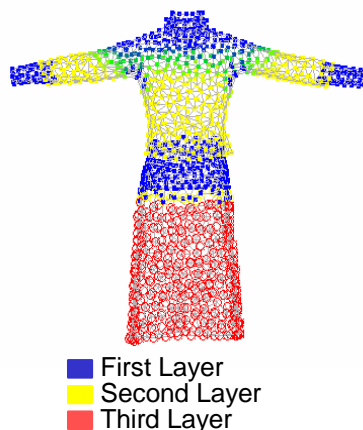


Figure 30: Garment segmentation

The Augmented Skinning Approach

Another approach is to base garment animation on directly on skinning animation. Hence, the garment is simply considered as a skin, uniformly animated like the rest of the body. However, this animation is complemented by a method which animates the skin according to mechanics, using the original skinned position for approximate collision processing.

In this approach, the garment has to be skinned, as would be the rest of the body surface. This is typically done through the use of automatic algorithms that extrapolate the skinning weights of the garment from the skinning weights of the underlying body surface. The automatic skinning extrapolation required an algorithm which tracks the relevant features of the body shape ruling the animation of any vertex of the garment surface. This algorithm can be designed by extending a proximity map (nearest mesh feature

algorithm) with additional visibility considerations for pinpointing the actual geometrical dependencies between the surfaces of the body and the cloth. Additional smoothness criteria should also be embedded so as to prevent any jaggy deformation over the garment surface. Further optimizations, such as the reduction of bone dependency count, are also performed for reducing the computational time of skinning animation. Collision data is obtained from the nearest-feature algorithm used in the skinning extrapolation scheme, and optimized with specific distance and visibility considerations.

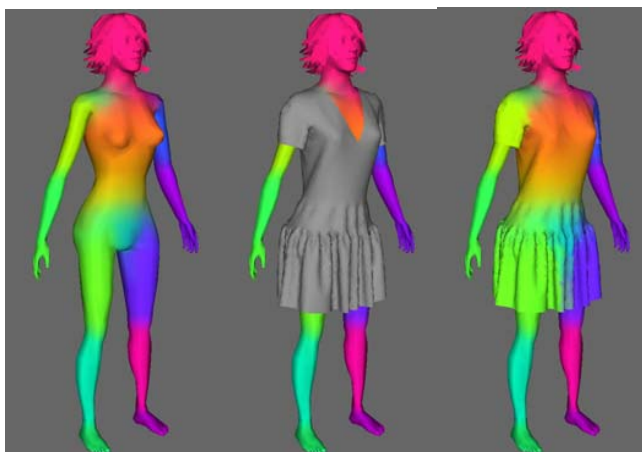


Figure 31: Extrapolating skinning information from the body surface to the garment surface.

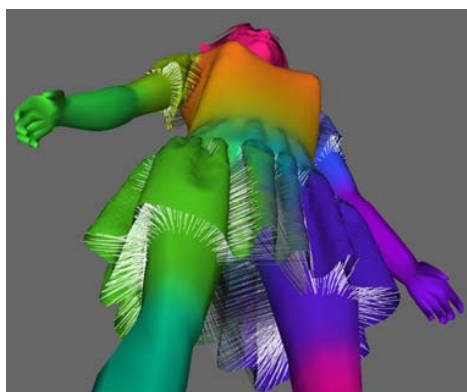


Figure 32: Collision data between the garment and the body extracted from proximity maps.

Data-Driven Animation

The main idea of this approach consists of developing a cloth simulator that can learn the cloth behavior through a sequence of pre-computed cloth simulation. This approach allows simulating the garment features such as wrinkles or gathers in real-time.

In [COR 04] is presented a data-driven method for simulating clothes worn by 3D characters in real-time. It divides the problem as simulating the garments in two phases. The first phase, a rough mesh reproduces the dynamic behavior of the garments. Its physical properties are defined by the pre-calculated sequence. In the second phase, the fine mesh simulates the details of the garments.

To effectively optimize the physics-based deformation, which is the bottleneck of the simulation, a coarse representation of the cloth mesh is used to drive the gross behavior in simulation. It considers that the gross cloth behavior is driven mainly by two separable contributions: the skeleton-driven movement of the

character and the mechanical properties of the cloth. This consideration is partly inspired by the hybrid real-time simulation method proposed in the previous section [COR 02], where a hybrid deformation method is used to combine dynamic surfaces with skeleton-driven deformation (SDD). Unlike that method, however, our method exhibits significantly more efficient and realistic behavior. This effect is achieved by focusing on the analysis of cloth movements in relation to its associated skin surface, and adopting a learning strategy. The idea is to use the analysis of the pre-simulated sequence to identify the region largely explained by joint movement and to replace the physics based simulation with geometric methods wherever possible.

In this approach, the key ingredients of the new technique are associated with different facets of cloth simulation: First, our novel collision detection prunes out unnecessary collision tests by tightly localizing potentially colliding regions through the analysis of the cloth movement in relation to the skeleton. Second, we use the pre-simulated sequence to approximate the dynamic behavior of the coarse mesh geometrically wherever possible. Finally, fine details such as wrinkles are also simulated in a data-driven manner, by using the pre-simulated cloth sequence as examples. Subsequently, real-time animation of fully dressed human could be generated, which would be suitable for applications such as games where visual plausibility is more important than accuracy.

Due to the computational expenses of solving the full numerical system of the physics-based deformation, we seek simplifications by constructing a coarse mesh representation of the garment. The coarse mesh is used to deduce the large-scale behavior of the cloth in a data-driven manner, based on the input pre-simulated sequence. A number of optimization strategies are adopted: The two following sections describe a pre-processing that constructs and segments a coarse mesh representation into different region types. We then describe in the next two sections the spring-mass system and collision handling of the coarse mesh at each frame of the simulation. Also described is the runtime process.

Now we have all the pieces required for a Virtual Try On application: Cloth animation, body sizing and animation. Putting them together in a single, web based application can turn out to be very difficult if not well thought out in the first place. The next –and last– section gives guidelines on how to proceed.

2.4 An integrated Virtual Try On application

Internet is a field of interest which evolves perpetually, especially in the fashion industry. Its recent developments have made possible to distribute complex 3D data through the internet. Thus, streaming and 3D technologies can now be combined in order to create a Virtual Try On application (VTO). The VTO is a window on a virtual dressing room where users can define measurements of a 3D body, put clothes on it and make it walk around. Due to the limitations of the real time aspect and the web constraints, the integration platform needs to be optimized and well designed as presented in [CLSM01] [PLAM02].

In this section we present how to integrate all these elements in a web based application. Even though the web 2.0 allows for rich content and an increased bandwidth for data transfer, the existing stand alone applications must be redesigned in order to take into account the server-client architecture. Also, the data handling must be optimized so that the downloading time is reduced to a minimum. For instance, instead of adding many small details to the 3D models (Figure 33), state of the art rendering techniques such as CG shaders can be used in order to improve the visual quality of the whole application (Figure 34).



Figure 33: Real time dressed and animated virtual body.

2.4.1 A common platform

Several components must be put together in order to create a VTO application. A body animation system has to be created, along with a body deformer and real-time cloth animator module. Moreover, the system has to be fed by cloth designers, thus it must be possible to create content with the mainstream 3D modellers such as 3DS Max or Maya.

With this outlook, Collada (Collaborative Design Activity) seems to be a good candidate for data exchange between the content creators and run-time engine. Collada is an open file standard for interactive 3D applications currently being promoted by several major player of the industry, such as Sony or Khronos. The standard covers most of the features required for our VTO application: Scene graph hierarchy, materials and textures, animation, skinning and shaders. It has free plugins available for most of the 3D creation packages which makes it usable by the designers. The current features of Collada are well suited for body animation and rendering, but it lacks very specific schemes for the data related to the body customization and cloth animation. Fortunately, it is extensible with new specialized tags, which makes it possible to include also the data specific to a VTO application into one single Collada file.

Because the VTO is a web based application, it must be embedded into a web browser, so that a user can simply browse the internet in order to access it. This can be done through the ActiveX controls of internet explorer, which make possible to put any application (would it be 3D or not) into a web browser. Thus the core VTO application will be encapsulated into an activeX control so that no cumbersome installation process is required for the user.

There exist plenty of 3D engine, onto which a VTO application could be built: OpenSceneGraph, OpenSG, OpenInventor, openRM, OGRE... We choose to use OpenSceneGraph for several reasons: first of all, it is completely open source and free of charge, that way the final application can be distributed freely, and the maintenance is made easier by having a full access to the source code. Second, it has an off-the-shelf activeX control and thus the embedding into a web browser isn't a problem any longer. Last, the Collada plugin that converts Collada files into OSG data structure is available so that we can still benefit from the fancy shader effects as they were exported from the modelling tools.



Figure 34: Real time cloth animation (with FX shaders).

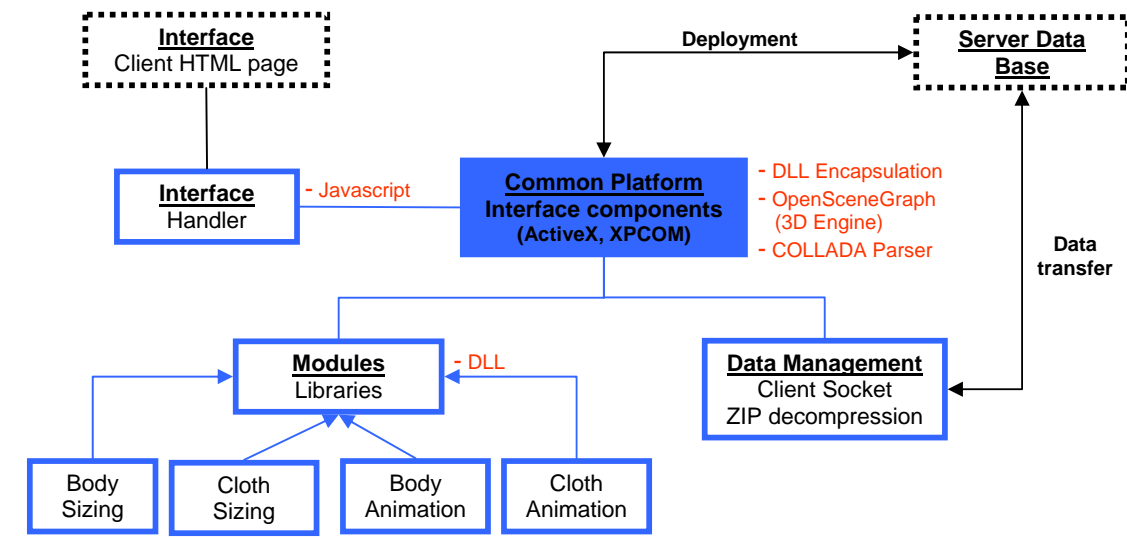


Figure 35: Relation between each component of the VTO application.

These three elements (COLLADA file format structure, Interface components (ActiveX and XPCOM) and OpenSceneGraph 3D engine) create our common integration web platform in term of structure and coding language. These elements constitute the core of our common platform (see Figure 35).

2.4.2 Web data management

Even though the internet bandwidth has drastically increased over the past decade, the content being transferred continues to become bigger and bigger. Thus, a data management strategy is essential for the VTO. As Collada is a text format only (i.e. no binary format was defined), these files will have to be compressed in some way before to be transmitted through the web. Because the text files themselves aren't too big (a few megabytes, without the textures), a simple zip compression is sufficient in order to ensure a quick download of the 3D content. For instance, the compression ratio obtained with WinZip 8.1 is more than 70% for a standard Collada file. That way, a 2 Mb file will be less than 600Kb after compression and the download time will be divided by 3.

In practice, the data is decompressed directly by the VTO which embeds a Zip decompression module. The decompression time is less than 0.5 seconds which is negligible for the whole downloading process.

On other hand, the textures employed are not easily compressible like a simple text. This represents the main difficulty for the web data management. The current best solution is to integrate a streaming module for the textures. It permits to display the 3D shape before to display the whole object. As COLLADA integrates the standard XML format, we could use it to stream textures from URL paths. Our application will stream the requested textures and according to textures ID it will subsequently process texturing on the object. This approach seems to be a good compromise between high resolution model and downloading time.

In this tutorial, the client/server management side will not be developed. We can only say that a data base is the usual solution to manage data but the best interest is how to broadcast it. Currently, a solution could be the peer to peer broadcasting method. This method increases the data speed deployment but it is totally dependent on the number of connected peers.

Web Interface

The web interface of the Virtual Try On platform provides the user with all the functionalities needed to define his/her own avatar. The user can modify measurements of a generic virtual body to fit

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

his/her general physical aspect. Choose its garments (including size and texture) and visualize the virtual body walking in a fashion show room with the garments animated in real time.

The web interface needs to be simple and user-friendly. All users should be able to handle it easily and intuitively. It follows that the main functionalities of the Virtual Try On are divided in three modules. In the first one, the user provides his measurements in order to get a 3D model with his profile. In the second one, he can choose between different cloth categories, select a garment to get information about it and try it on via his 3D model. Finally, in the third module the user can see his model walking around and can interact with it by zooming on it or changing the view angle.

The interface content should be well-ordered. Indeed the web interface elements which are the most important of the Virtual Try On web based application should be highlighted. In our prototype we show up the fact that we can choose the exact measurements for the 3D model. Indeed this element is very important in this interface and should appear earlier. This is the reason why we put this measurements selection with the common sizes selection in order to create a submenu called Measurement. Naturally, this measurements submenu will interact directly with the web based application and with our body sizing module. Each slider represents a sizable body part of the virtual body (see previous paragraph on Parametrically Deformable Human Bodies method). Next, the users have to select the cloth itself and to be able to get relevant information on it. Regarding to our goal, this submenu permits to get a global view of the cloth data bank, to follow the user choices and to view the result on the virtual body. The third submenu is called Show Room and permits to show the virtual body deformed and the chosen cloth, through an animation. It is composed by two buttons which are used to play or to stop the walk animation. Finally, the proposed submenus sequence follows the natural logic proposed as to know the measurements following by the dressing choice and the show room. This corresponds to our modules sequence i.e. body sizing following by cloth sizing following by body and cloth animation.

The current interface is developed using Flash tool. The Flash Player developed and distributed by Adobe Systems (www.adobe.com) is a client application available for most common web browsers. It features support for vector and raster graphics, a scripting language called Action Script and bi-directional streaming of audio and video. This tool provides compatibilities with ActiveX and XPCOM components and can communicate through JavaScript by FSCommand which are Action Script methods.

To conclude, the Virtual Try On regroups two main elements which are the web based application and the web interface. Our web based application is constructed on a common structure which is composed by the Collada file format, the OpenSceneGraph 3D engine, interface components and our in-house modules. Currently, this development is not finished yet, but currently follows the presented integration strategy above.

3 References

- [AAT02] AATCC Technical Manual 2002, www.aatcc.org
- [ABH*94] Azuola F., Badler N., Ho P., Kakadiaris I., Metaxas D., Ting B.: Building anthropometry-based virtual human models. Proc. IMAGE VII Conference (June 1994).
- [ACP03] Allen B., Curless B., Popovic Z.: The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graph.* 22, 3 (2003), 587-594.
- [ALP04] ABE Y., LIU K., POPOVIC Z.: Momentum-based parameterization of dynamic character motion, SCA 2004.
- [ASK*05] Anguelov D., Srinivasan P., Koller D., Thrun S., Rodgers J., Davis J.: Scape: shape completion and animation of people. *ACM Trans. Graph.* 24, 3 (2005), 408-416.
- [Ass] <http://www.ascension-tech.com/> the Ascension technologies website, accessed May2007
- [AT] Aubel A., Thalmann D.: Realistic deformation of human body shapes. In *Computer Animation and Simulation '00*, pp. 125-135.
- [AT01] Aubel A., Thalmann D.: Interactive modeling of human musculature. *Interactive Modeling of Human Musculature. Computer Animation 2001*, Seoul (2001).
- [Bar81] Barr A.: Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications* 1, 1 (1981), 11-23.
- [Bar84] Barr A.: Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1984), ACM Press, pp. 21-30.
- [BAR 98] D. BARAFF, A. WITKIN, 1998, Large Steps in Cloth Simulation, *Computer Graphics (SIGGRAPH'98 proceedings)*, ACM Press, pp 43-54.
- [BCHB03] BOULIC R., LE CALLENNEC B., HERREN M., BAY H.: Experimenting prioritized IK for motion editing. *Proceedings of Eurographics 2003*.
- [BEH61] B. Behre, "Mechanical Properties of Textile Fabrics Part I: Shearing", *Textile Research Journal*, Vol. 31, No.2, 1961, pp. 87-93
- [Bla94] Blanc C.: Superquadrics a generic implementation of axial procedural deformation techniques. In *Graphics Gems* (1994), vol. 5, Academic Press, pp. 249-256.
- [BPW93] Badler N. I., Phillips C. B., Webber B. L.: *Simulating Humans: Computer Graphics Animation and Control*. Oxford University Press, New York, NY, Oxford, 1993.
- [CHO 02] K.J. CHOI, H.S. KO, 2002, Stable but Responsive Cloth, *Computer Graphics (SIGGRAPH'02 proceedings)*, Addison Wesley, pp 604-611.
- [Cap05] CAPPEL S., BURKHART M., CURLESS B., DUCHAMP T., POPOVIC A.: Physically Based Rigging for Deformable Characters, *Symposium on Computer Animation 2005*.
- [CBC*05] Capell S., Burkhart M., Curless B., Duchamp T., Popovic Z.: Physically based rigging for deformable characters. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2005), ACM Press, pp. 301-310.
- [CGC*02] Capell S., Green S., Curless B., Duchamp T., Popovic Z.: Interactive skeleton-driven dynamic deformations. In *SIGGRAPH'02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 586-593.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

- [CHP89] Chadwick J. E., Haumann D. R., Parent R. E.: Layered construction for deformable animated characters. In SIGGRAPH'89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1989), ACM Press, pp. 243-252.
- [Cliff82] CLIFFORD W.: Mathematical Papers, London, Macmillan, 1882.
- [CLSM01] Frédéric Cordier, WonSook Lee, HyeWon Seo, Nadia Magnenat-Thalmann, Virtual-Try-On on the Web, VRIC, Virtual Reality International Conference, Laval Virtual 2001, May 16-18.
- [Coq90] Coquillart S.: Extended free-form deformation: a sculpturing tool for 3d geometric modeling. In SIGGRAPH '90: Proceedings of the 17th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1990), ACM Press, pp. 187-196.
- [COR 02] F. CORDIER, N. MAGNENAT-THALMANN, 2002, Real-Time Animation of Dressed Virtual Humans. Eurographics Conference Proceedings, Blackwell.
- [COR 04] F. CORDIER, N. MAGNENAT-THALMANN, 2004, A Data-driven Approach for Real-Time Clothes Simulation. 12th Pacific Conference on Computer Graphics and Applications, IEEE Computer Society, pp. 257-266.
- [COT 99] S. COTIN, H. DELINGETTE, N. AYACHE, 1999, Real-Time Elastic Deformations of Soft Tissues for Surgery Simulation, IEEE Transactions on Visualization and Computer Graphics, 5(1), pp 62-73.
- [dATSS07] de Aguiar E., Theobalt C., Stoll C., Seidel H.-P.: Rapid animation of laser-scanned humans. In IEEE Virtual Reality 2007 (Charlotte, USA, 2007), IEEE, pp. 223-226.
- [DCKY02] Dong F., Clapworthy G. J., Krokos M. A., Yao J.: An anatomy-based approach to human muscle modeling and deformation. IEEE Transactions on Visualization and Computer Graphics 8, 2 (2002), 154-170.
- [DES 99] M. DESBRUN, P.SCHRÖDER, A.H. BARR, 1999, Interactive Animation of Structured Deformable Objects, Proceedings of Graphics Interface, A K Peters, pp 1-8.
- [DES 01] G. DESBRUNNE, M. DESBRUN, M.P. CANI, A.H. BARR, 2001, Dynamic Real-Time Deformations Using Space & Time Adaptive Sampling, Computer Graphics (SIGGRAPH'01 proceedings), Addison Wesley, pp 31-36.
- [Digi] <http://www.digimation.com/home/> Digimation, provider of Bones Pro 3, accessed May 2007
- [DMS98] DeCarlo D., Metaxas D., Stone M.: An anthropometric face model using variational techniques. In SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1998), ACM Press, pp. 67-74.
- [EBE 00] B. EBERHARDT, O. ETZMUSS, M. HAUTH, 2000, Implicit-Explicit Schemes for Fast Animation with Particles Systems, Proceedings of the Eurographics workshop on Computer Animation and Simulation, Springer-Verlag, pp 137-151.
- [EHW*07] E. H. D., H. Y. S., W. C. J., K. S. J., S. K. M., B. J.: Sweep-based human deformation. The Visual Computer 21 (2007), 542-550.
- [ETZ 03] O. ETZMUSS, J. GROSS, W. STRASSER, 2003, Deriving a Particle System from Continuum Mechanics for the Animation of Deformable Objects, IEEE Transactions on Visualization and Computer Graphics, IEEE Press, pp 538-550.
- [ETZ 03] O. ETZMUSS, M. KECKEISEN, W. STRASSER, 2003, A Fast Finite-Element Solution for Cloth Modeling, Proceedings of the 11th Pacific Conference on -Computer Graphics and Applications, pp 244-251.
- [GBT06] GLARDON P., BOULIC R., THALMANN D.: Robust on-line adaptive footplant detection and enforcement. The Visual Computer 22(3), 194-209, 2006.

- [GC95] Gortler S. J., Cohen M. F.: Hierarchical and variational geometric modeling with wavelets. In *SI3D '95: Proceedings of the 1995 symposium on Interactive 3D graphics* (New York, NY, USA, 1995), ACM Press, pp. 35-*
- [Glei98] GLEICHER M.: Retargeting motion to new characters. *SIGGRAPH 98*.
- [GRI 03] E. GRIMSPUN, A.H. HIRANI, M. DESBRUN, P. SCHRÖDER, 2003, *Discrete Shells*, Eurographics Symposium on Computer Animation, pp 62-68.
- [Guo05] GUO Z., WONG K.: *Skinning with Deformable Chunks*, Computer Graphics Forum 2005.
- [HAU 01] M. HAUTH, O. ETZMUSS, 2001, *A High Performance Solver for the Animation of Deformable Objects using Advanced Numerical Methods*, Eurographics 2001 proceedings, pp 137-151.
- [HAU 03] M. HAUTH, J. GROSS, W. STRASSER, 2003, *Interactive Physically-Based Solid Dynamics*, Eurographics Symposium on Computer Animation, pp 17-27.
- [HD55] Hartenberg R. S., Denavit J.: A kinematic notation for lower pair mechanisms based on matrices. *Journal of Applied Mechanics* 77 (1955), 215-221.
- [HMT*85] H. N., M. H., T. K., T. K., I. S., K O.: Object modelling by distribution function and a method for image generation. *Transaction for the Institute of Electronics and Communication for Engineers of Japan J68-D* (1985), 718-725.
- [HS:07] Human solutions, <http://www.human-solutions.com>, May 2007.
- [Hua06] HUANG J., SHI X., LIU X., ZHOU K. : *Subspace gradient domain mesh deformation*, ACM Transactions on Graphics 2006
- [HWBO95] HODGINS J., WOOTEN W., BROGAN D., O'BRIEN J.: *Animating Human Athletics*, SIGGRAPH 1995.
- [HYKJ03] Hyun D.-E., Yoon S.-H., Kim M.-S., Jittler B.: Modeling and deformation of arms and legs based on ellipsoidal sweeping. In *PG '03: Proceedings of the 11th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2003), IEEE Computer Society, p. 204.
- [IRV 04] G. IRVING, J. TERAN, R. FEDKIW, 2004, *Invertible Finite Elements for Robust Simulation of Large Deformation*, Eurographics Symposium on Computer Animation, pp 131-140.
- [JAM 99] D. JAMES, D. PAI, 1999, *ArtDefo - Accurate Real-Time Deformable Objects*, Computer Graphics (SIGGRAPH'99 proceedings), ACM Press, pp 65-72.
- [Jam05] JAMES D., TWIGG C.: *Skinning Mesh Animations*, ACM Transactions on Graphics 2005.
- [JTT94a] Jianhua S., Thalmann N. M., Thalmann D.: Human skin deformation from cross sections. *Proc. Computer Graphics International '94* (1994).
- [JTT94b] Jianhua S., Thalmann N. M., Thalmann D.: Human skin deformation from cross-sections. In *Computer Graphics Int. '94* (27-1 1994).
- [Kavan05] KAVAN L., ZARA J.: *Spherical blend skinning: a real-time deformation of articulated models*, Interactive 3D Graphics 2005
- [KAW80] Kawabata S., "The standardization and analysis of hand evaluation (2nd edition). The hand evaluation and standardization committee", The Textile Machinery Society of Japan, Osaka, 1980
- [KCZO07] KAVAN L, COLLINS S., ZARA J., O'SULLIVAN C.: *Skinning with dual quaternions*, Symposium on Interactive 3D Graphics. 2007.
- [Kok04] KOKKEVIS E.: *Practical Physics for Articulated Characters*, Game Developers Conference, 2004
- [Kry02] KRY P., JAMES D., PAI D.: *EigenSkin: Real Time Large Deformation Character Skinning in Hardware*, Symposium on Computer Animation 2002.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

- [KSG02] KOVAR L., SCHREINER J., GLEICHER M.: Footskate cleanup for motion capture editing. SCA2002.
- [KZ05] Kavan L., Zara J.: Spherical blend skinning: a real-time deformation of articulated models. In I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games (New York, NY, USA, 2005), ACM Press, pp. 9-16.
- [Las87] Lasseter J.: Principles of traditional animation applied to 3D computer animation. In SIGGRAPH '87: Proceedings of the 14th Annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1987), ACM Press, pp. 35-44.
- [Lass94] LASSETER J.: Tracks to Animating Characters with a Computer, SIGGRAPH94 Course
- [LCA05] Larboulette C., Cani M.-P., Arnaldi B.: Dynamic skinning: adding real-time dynamic effects to an existing character animation. In SCCG '05: Proceedings of the 21st spring conference on Computer graphics (New York, NY, USA, 2005), ACM Press, pp. 87-93.
- [LD61] J. Lindberg and B. Dahlberg, "Mechanical Properties of Textile Fabrics, Part III: Shearing and Buckling of Various Commercial Fabrics," Textile Research Journal, Vol. 31, 1961
- [LM07] LYARD E., MAGNENAT-THALMANN N.: A simple footskate removal method for virtual reality applications, The visual computer, to appear.
- [LP02] LIU K., POPOVIC Z.: Synthesis of Complex Dynamic Character Motion from Simple Animations, SIGGRAPH 2002.
- [LS99] LEE J., SHIN S.: A hierarchical approach to interactive motion editing for human –like figures. Proceedings of SIGGRAPH 1999.
- [LW07] Li J., Wang Y.: Automatically construct skeletons and parametric structures for polygonal human bodies. Computer Graphics International (2007).
- [LWN94] Lamousin H. J., W. N. W.: Nurbs-based free-form deformations. IEEE Computer Graphics and Applications 14, 9 (1994), 59-65.
- [Magn88] MAGNENAT-THALMANN N., LAPERRIERE R., THALMANN D.: Joint-dependant local deformations for hand animation and object grasping, Graphics Interface, 1988.
- [Max] <http://usa.autodesk.com/adsk/servlet/index?id=5659302&siteID=123112> Autodesk 3DSMax website, accessed May 2007
- [Maya] <http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7635018> Autodesk Maya, accessed May 2007
- [MBK07] Marinov M., Botsch M., Kobbelt L.: Gpu-based multiresolution deformation using approximate normal field reconstruction. Journal of graphics tools 12, 1 (2007), 27-46.
- [Meta07] <http://www.metamotion.com/gypsy/gypsy-motion-capture-system-mocap.htm>
- [MEY 01] M. MEYER, G. DEBUNNE, M. DESBRUN, A. H. BARR, 2001, Interactive Animation of Cloth-like Objects in Virtual Reality, Journal of Visualization and Computer Animation, Wiley, 12(1), pp 1-12.
- [MIN95] Minazio P. G., "FAST – Fabric Assurance by Simple Testing", International Journal of Clothing Science and Technology, Vol. 7, No. 2/3, 1995
- [MoAn] <http://www.motionanalysis.com/> the Motion Analysis website, accessed May 2007
- [MoBuild] www.autodesk.com/motionbuilder Autodesk post processing software, accessed May 2007
- [MoCal] <http://www.motionanalysis.com/applications/animation/games/calcium.html>, Motion Analysis Calcium post processing software, accessed May 2007
- [Mohr03] MOHR A., GLEICHER M.: Building Efficient, Accurate Character Skins From Examples, ACM Transactions on Graphics 2003.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

[MT97] Moccozet L., Thalmann N. M.: Dirichlet free-form deformations and their application to hand simulation. In CA '97: Proceedings of the Computer Animation (Washington, DC, USA, 1997), IEEE Computer Society, p. 93.

[MTT87] Magnenat-Thalmann N., Thalmann D.: The direction of synthetic actors in the film rendez-vous Montreal. IEEE Computer Graphics and Applications 7, 12 (Dec. 1987), 9-19.

[MTT91] Magnenat-Thalmann N., Thalmann D.: Human body deformations using joint-dependent local operators and finite-element theory. 243-262.

[MTYCS04] Magnenat-Thalmann N., Yahia-Cherif L., Seo H.: Modeling anatomical-based humans. In ICIG '04: Proceedings of the Third International Conference on Image and Graphics (ICIG'04) (Washington, DC, USA, 2004), IEEE Computer Society, pp. 476-480.

[MUL 02] M. MULLER, J. DORSEY, L. MCMILLAN, R. JAGNOW, B. CUTLER, 2002, Stable Real-Time Deformations, Proceedings of the Eurographics Symposium on Computer Animation, pp 49-54.

[MUL 00] M. MULLER, M. GROSS, 2000, Interactive Virtual Materials, Proceedings of Graphics Interface, Canadian Human-Computer Communications Society, pp 239-246.

[NT98a] Nedel L., Thalmann D.: Modeling and deformation of the human body using an anatomically-based approach. In CA '98: Proceedings of the Computer Animation (Washington, DC, USA, 1998), IEEE Computer Society, p. 34.

[NT98b] Nedel L. P., Thalmann D.: Real time muscle deformations using mass-spring systems. In CGI '98: Proceedings of the Computer Graphics International 1998 (Washington, DC, USA, 1998), IEEE Computer Society, p. 156.

[OBR 99] J. O'BRIEN, J. HODGINS, 1999, Graphical Modeling and Animation of Brittle Fracture, Computer Graphics (SIGGRAPH'99 proceedings), ACM Press, pp 137-146.

[Orga] <http://www.organicmotion.com>, the Organic Motion website, accessed May 2007

[Par06] PARK S. HODGINS J.: Capturing and animating skin deformation in human motion, ACM Transactions on Graphics 2006.

[PCLS05] Pratscher M., Coleman P., Laszlo J., Singh K.: Outside-in anatomy based character rigging. In SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation (New York, NY, USA, 2005), ACM Press, pp. 329-338.

[PH06] Park S. I., Hodgins J. K.: Capturing and animating skin deformation in human motion. ACM Trans. Graph. 25, 3 (2006), 881-889.

[PIE30] Peirce F.T., "The Handle of Cloth as a Measurable Quantity", Journal of the Textile Institute, Vol. 21, 1930

[PLAM02] Dimitris Protosaltou, Christiane Luible, Marlene Arevalo, Nadia Magnenat-Thalmann, A body and garment creation method for an Internet based virtual fitting room. Computer Graphics International Conference Proceedings, Springer Verlag, pp. 105 -122. July, 2002.

[Pol] <http://www.polhemus.com/> the Polhemus website, accessed May 2007

[Pop99] POPOVIC Z.: Physically based motion transformation, SIGGRAPH99

[Rhe06] RHEE T., LEWIS J.P. NEUMANN U.: Real-Time Weighted Pose-Space Deformation on the GPU, Computer Graphics Forum 2006.

[RLN06] Rhee T., Lewis J., Neumann U.: Real-time weighted pose-space deformation on the GPU. Computer Graphics Forum 25, 3 (2006), 439-448.

[SF98] Singh K., Fiume E.: Wires: a geometric deformation technique. In SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1998), ACM Press, pp. 405-414.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

- [SK00] Singh K., Kokkevis E.: Skinning characters using surface oriented free-form deformations. In Graphics Interface (2000), pp. 35-42.
- [Sloa01] SLOAN P., ROSE F., COHEN M.: Shape by Example, Interactive 3D Graphics 2001.
- [SMT03] Seo H., Magnenat-Thalmann N.: An automatic modeling of human bodies from sizing parameters. In I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics (New York, NY, USA, 2003), ACM Press, pp. 19-26.
- [SMT04] Seo H., Magnenat-Thalmann N.: An example-based approach to human body manipulation. Graph. Models 66, 1 (2004), 1-23.
- [SP86] Sederberg T. W., Parry S. R.: Free-form deformation of solid geometric models. In SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1986), ACM Press, pp. 151-160.
- [SPCM97] Scheepers F., Parent R. E., Carlson W. E., May S. F.: Anatomy-based modeling of the human musculature. In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 163-172.
- [ST95] Shen J., Thalmann D.: Interactive shape design using metaballs and splines. In Implicit Surfaces'95 (Grenoble, France, 1995), pp. 187-196.
- [TBB07] <http://www.thinkbroadband.com/news/3049-world-broadband-statistics.html> published originally by Point-Topic.com Tuesday 10 April 2007
- [TGB00] TOLANI D., GOSWAMI A., BADLER N.: Real-time inverse kinematics techniques for the anthropomorphic limbs. Graphical Models 62, 353-388 (2000)
- [THA 03] N. MAGNENAT-THALMANN, H. SEO, F. CORDIER, 2003, Automatic Modeling of Virtual Humans and Body clothing. Proc. 3-D Digital Imaging and Modeling, IEEE Computer Society Press, pp. 2-10.
- [TK05] TAK S., KO H.: A physically-based motion retargeting filter, ACM Transactions on Graphics 24(1), 2005.
- [TSB*05] Teran J., Sifakis E., Blemker S. S., Ng-Thow-Hing V., Lau C., Fedkiw R.: Creating and simulating skeletal muscle from the visible human data set. IEEE Transactions on Visualization and Computer Graphics 11, 3 (2005), 317-328.
- [VH:95] Visible human project, <http://www.nlm.nih.gov/research/visible>, 1995.
- [Vic] www.vicon.com, the vicon website, accessed May 2007
- [VIQ] <http://www.vicon.com/products/viconiq.html> , Vicon IQ post processing software, accessed May 2007
- [VM05] Volino P., Magnenat-Thalmann N., "Accurate Garment Prototyping and Simulation", Computer-Aided Design & Applications, Vol. 2, No. 1-4, 2005
- [VOL 00] P. VOLINO, N. MAGNENAT-THALMANN, 2000, Virtual Clothing: Theory and Practice, Springer.
- [VOL 05] P. VOLINO, N. MAGNENAT-THALMANN, 2005, Accurate Garment Prototyping and Simulation, Computer-Aided Design & Applications, CAD Solutions, 2(5), pp 645-654.
- [VOL 05] P. VOLINO, N. MAGNENAT-THALMANN, 2005, Implicit Midpoint Integration and Adaptive Damping for Efficient Cloth Simulation, Computer Animation and Virtual Worlds, Wiley, 16(3-4), pp 163-175.
- [VOL 06] P. VOLINO, N. MAGNENAT-THALMANN, 2006, Simple Linear Bending Stiffness in Particle Systems, Proceedings of the Symposium on Computer Animation 2006, pp 101-105.

E. Lyard, C. Luible, P. Volino, M. Kasap, V. Muggeo and N. Magnenat-Thalmann / Advanced Topics in Virtual Garment Simulation – Part 2.

[Wang03] Wang et al., Multi-weight enveloping: least-squares approximation techniques for skin animation, SCA2003

[WG97] Wilhelms J., Gelder A. V.: Anatomically based modeling. In SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 173-180.

[WR07] Wang Q., Ressler S.: Generation and manipulation of h-anim caesar scan bodies. In Web3D '07: Proceedings of the twelfth international conference on 3D web technology (New York, NY, USA, 2007), ACM Press, pp. 191-194.

[YLS04] YANG P., LASZLO J., SINGH K.: Layered Dynamic Control for Interactive Character Swimming, SCA 2004.

[ZL05] Zhou X., Lu J.: Nurbs-based galerkin method and application to skeletal muscle modeling. In SPM '05: Proceedings of the 2005 ACM symposium on Solid and physical modeling (New York, NY, USA, 2005), ACM Press, pp. 71-78.

[ZTS06] Zhaoqi W., Tianlu M., Shihong X.: A fast and handy method for skeleton-driven body deformation. Computer Entertainment 4, 4 (2006), 6.