Inhabited Virtual Heritage

Nadia Magnenat-Thalmann - *U. of Geneva*
Alan Chalmers - *U. of Bristol*
Daniel Thalmann - *EPFL*

Synopsis

- Inhabited Virtual Cultural Heritage is a novel way of conservation, preservation and interpretation of cultural history. By simulating an ancient community within the virtual reconstructions of a habitat, the public can better grasp and understand the culture of that community. The course will present the following concepts:
  - Reconstruction technology
  - Computer Animation technology
  - Interaction technology
- Three case studies will be shown: the simulation of the Xian Terra Cotta Army, the representation of Geneva in 1602 and the reconstruction of Aya Sofia church in Turkey.

MIRALab Presentation University of Geneva

Professor
Nadia Magnenat-Thalmann

Generating Animatable 3D Virtual Humans from Photographs

Nadia Magnenat-Thalmann
Won-Sook Lee
Jin Gu

Introduction

- Two techniques depending on the interest
  - *accuracy and precision of the obtained object model shapes*,
    - CAD systems, medical application.
  - *visual realism and speed for animation of the reconstructed models,*
    - internet applications
    - Virtual Reality applications.

Virtual humans for real-time applications

- What’s the components to consider?
  - acquisition of human shape data
  - realistic high-resolution texture data
  - functional information for animation of the human (both face and body)

State of the Art - Face

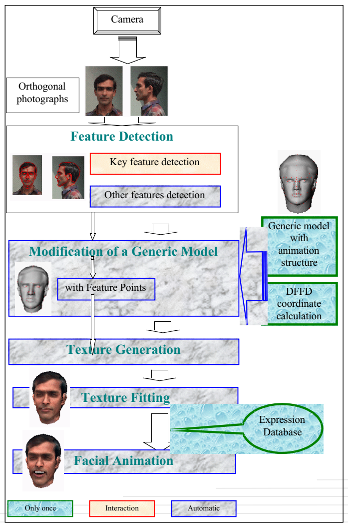
- Features on photographs (organized) and a generic model
  - Modeling used for getting the individualized face using a few points
    - [Kurihara 91] [Akimoto 93] [Ip 96]
  - Modeling used for expression database
    - [Pighin 98]

State of the Art - comparison

Photography	Laser Scanner
Cheaper	Expensive
Very general equipment	Special equipment
	Output: Numerous points
Usually high resolution of texture mapping	Usually low resolution of texture mapping
Easy to catch characteristic points	Often noisy to catch characteristic points
Difficult to catch non-characteristic points	Better to catch non-characteristic points
	Problems for hairy parts

Face Cloning

- Input
  - photograph
  - generic head & animation
- Method
  - Feature based
- Output
  - Animatable virtual human



Head shapes from photos

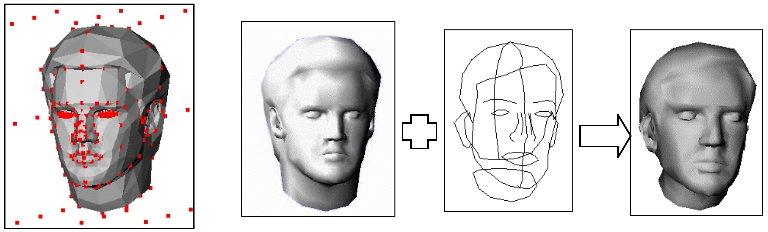
- Features on front (and side) view
  - eyes, nose, lips, hair and face outlines, etc.
- Semiautomatic structured feature detection
  - piecewise affine mapping
  - structured snake to keep structure of points

Head shapes from photos in 3D rather than in 2D

- Generation of (x, y, z) from (x, y_f) and (y_s, z)
  - criteria for giving more importance on the front view
  - robust even though the input photographs are not perfectly orthogonal
- Dirichlet FFD (DFFD)
  - the convex hull of a set of control points in general position

Head shapes from photos

- Feature points < control points

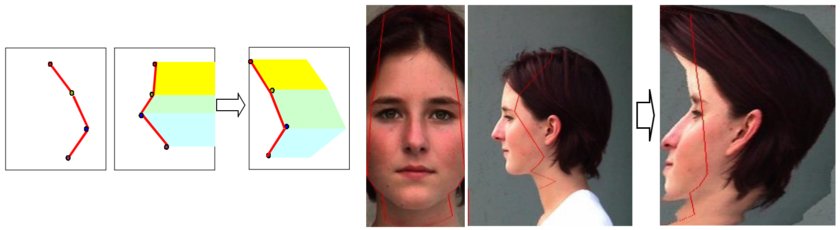


Texture mapping

- Texture Generation
  - One texture image from two images
    - Geometrical deformation
    - Multi-Resolution techniques
- Texture Mapping
  - Projection to three planes
  - Transformation to several spaces

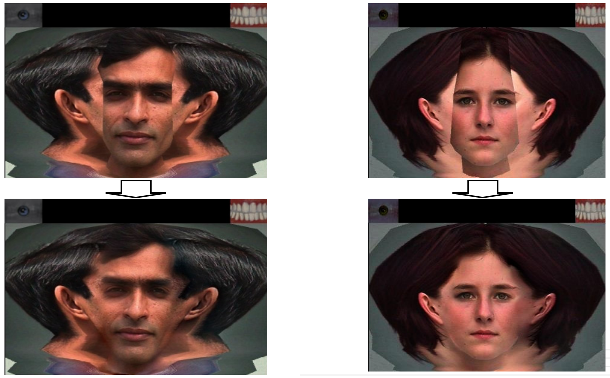
Seamless texture mapping

- Texture generation
  - Image deformation



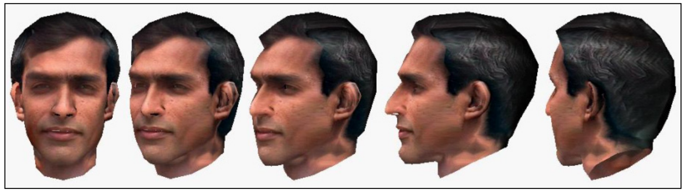
Seamless texture mapping

- Texture generation
  - Multiresolution image mosaic



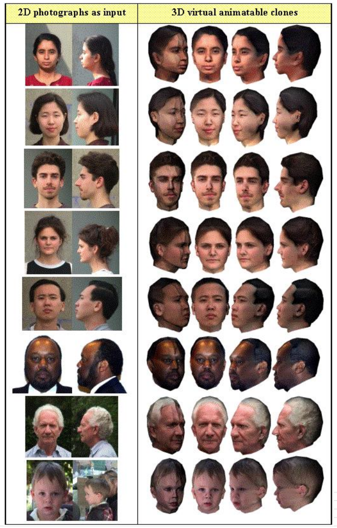
Results

- Rotation in 360 degree

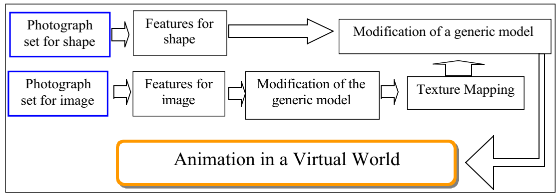


Results

- Several ethnic group from one generic model

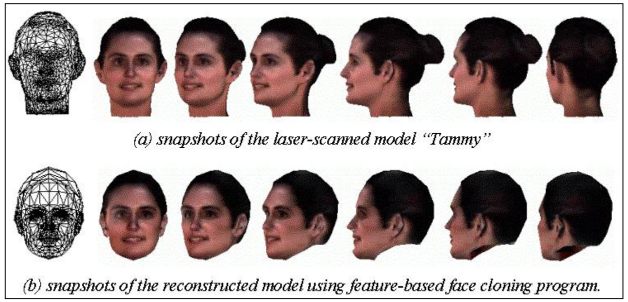


Results - shape texture separation



Results - Validation

- Visual comparison



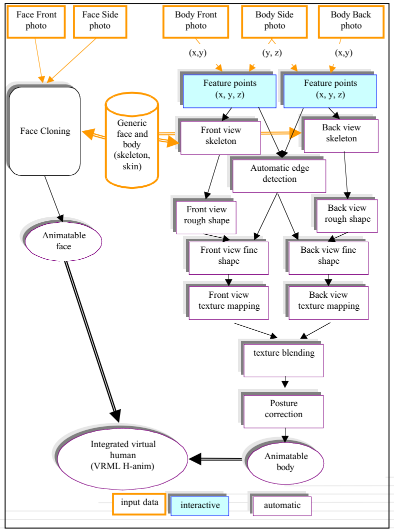
(a) snapshots of the laser-scanned model "Tammy"

(b) snapshots of the reconstructed model using feature-based face cloning program.

- 3D- distance measurement : 2.84306 %

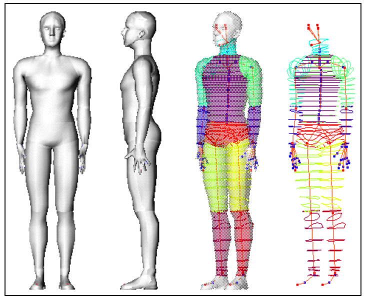
Body Cloning

- Input
  - three photographs
  - H-Anim 1.1 generic body
- Feature - edge based
- Output
  - animatable vitual human



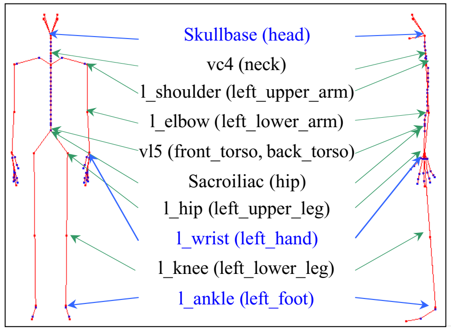
Body Cloning - Generic body

- Continuous mesh humanoids
  - MPEG-4 compatible H-Anim 1.1 formats [http:H-Anim]
  - 94 skeleton joints & 12 skin parts (different from the face with only skin)



Body Cloning - Generic body

- H-Anim joints related to skin parts
  - the local coordinates of the skin part *i* to global coordinate by 4x4 matrix M_i.



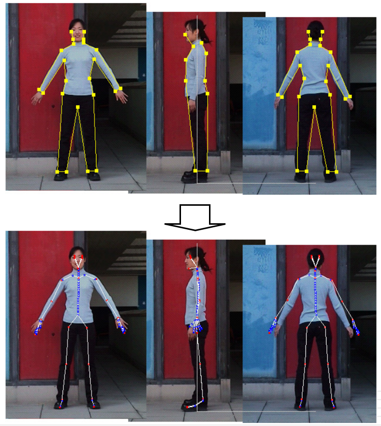
Body Cloning - Generic body

- Skin has grid structure
  - each skin part has several slices
  - each slice on the skin part has the same number of points
  - Share the same 3D coordinates between different skin part
- Resulting seamlessly continuous skin envelope

Body Features and skeleton

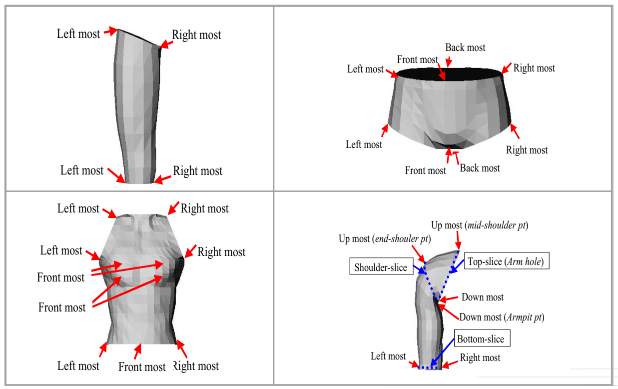
- Features and skeleton adjust

Feature points on images
⇩
Modify the *movable* skeleton joints
⇩
Modify other skeleton joints



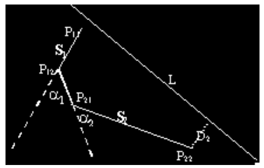
Body rough skin adjustment

- Feature points -> Control points -> skin modification



Body fine skin adjustment

- Feature driven edge extraction

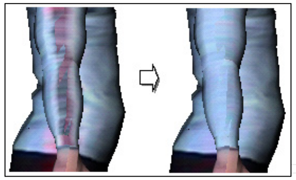


- Canny edge detector
- Each *feature segment* indicates the vicinity and approximate direction of the boundary to be found
- evaluate the “goodness” of the potential connection

Body Cloning - Texture mapping

- Front and side views are used
  - Deform body and texture for each side separately
- Texture blending
  - Problem caused by

    digitization and illumination
  - Linear blending following corresponding edges on the front and back views



MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Body and Face together

- Automatic connection with own face from face cloning system
  - use features on face and body
- Neck adjustment
  - bridge to connect the face and body smoothly and seamlessly

www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Body Results

- H-Anim 1.1 format
  - visualized by web browsers
  - Animatable

www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

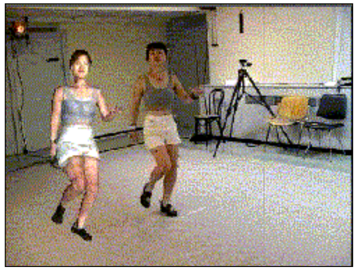
Body Cloning - Results

- Sometimes postcorrection needed
  - Skeleton correction from skin envolope
    - Elbow skeleton correction
  - H-Anim & Vicon (optical motion capture system) posture
    - length and angle coordinate
    - adjust angles for arms and legs

www.miralab.unige.ch
University of Geneva

Animation result with motion capture

- Animation with cloned body
  - Comparison with real motion



Conclusion

- Easy input like photographs is the first priority to build the system
- A complete integration of whole face and body parts from five photographs
- Continuous mesh for generic body
  - real-time animation without texture problems

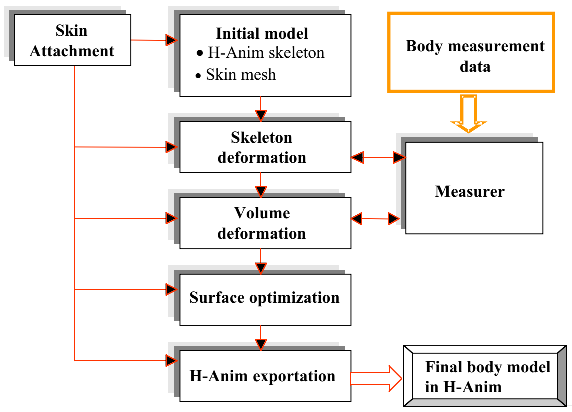
Conclusion

- Several problems are solved
  - efficient and robust semi-automatic feature detection method
  - 3D-deformation approaches rather than in 2D resulting error resistance for input images
  - more robust 3D deformation using DFFD
  - fully automatic generation of seamless texture mapping

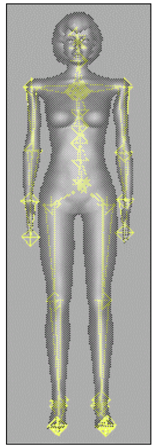
Measurement based body creation

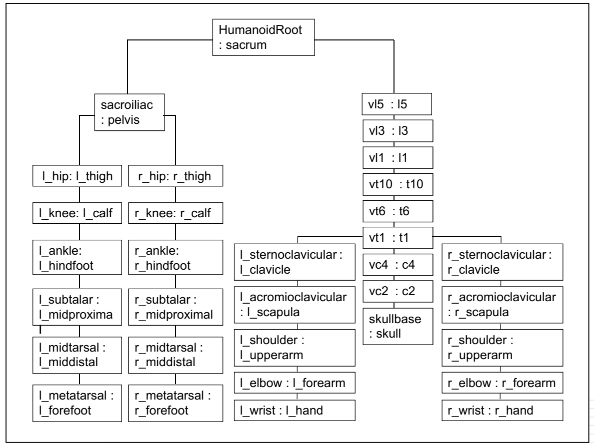
Nadia Magnenat-Thalmann
HyeWon Seo

Overview

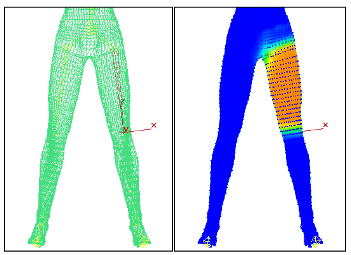


Initial model





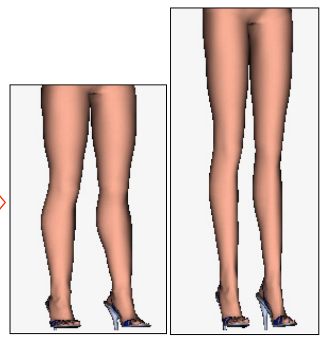
Skeletal deformation



Skin attachment to bones

$$s = \frac{T_{desired}}{T_{current}}$$

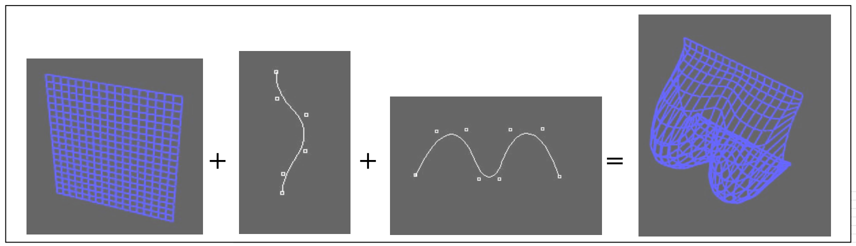
Scale factor



Skeletal deformation

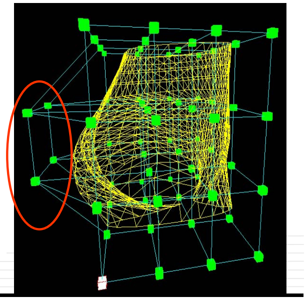
Volumetric deformation – breast example

- Breast
  - Grid structure (20 x 23).
  - Parametric curves for preserving the round aspect.



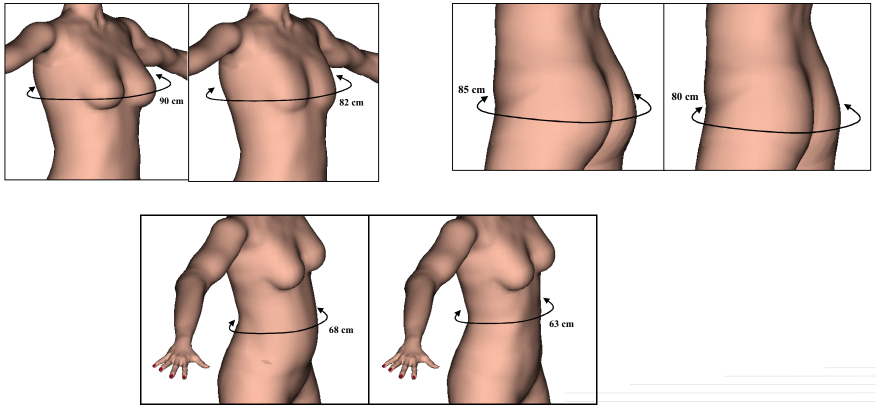
Volumetric deformation – other parts

- Waist
  - Similar to the breast but with the use of simpler(Bézier) curve.
- Hips
  - Deformation based on FFD(Free Form Deformation).



MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Volumetric deformation – results



www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Facial Animation

From Facial Mesh to Expressive Talking Faces

Nadia Magnenat-Thalmann
Sumedha Kshirsagar

www.miralab.unige.ch
University of Geneva

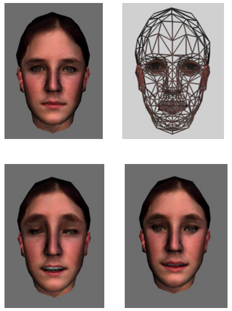
MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Overview

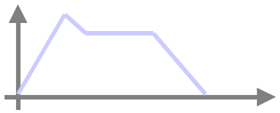
- Hierarchy in Facial Animation
- Definition of Static Expressions
- From Expressions to Animation
- Speech Animation Overview

www.miralab.unige.ch
University of Geneva

Hierarchy in Facial Animation

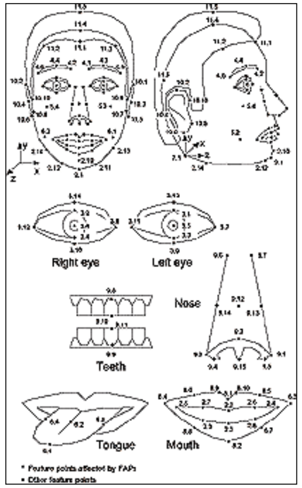


- Face Object : Collection of mesh vertices and topology
- Static Expressions : Deformation of this mesh controlled by parameters



- Animation : Varying the static expressions with time

Defining Static Expressions

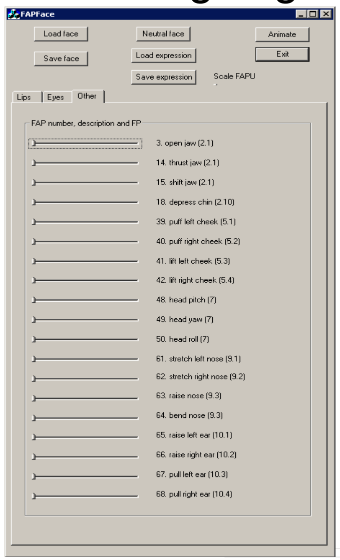


Need of Parameterization to define static expressions MPEG-4 Facial Animation Parameters

Feature Points defined on the Specific locations of the face

Animation defined by the displacements of these Feature Points

Designing Facial Expressions

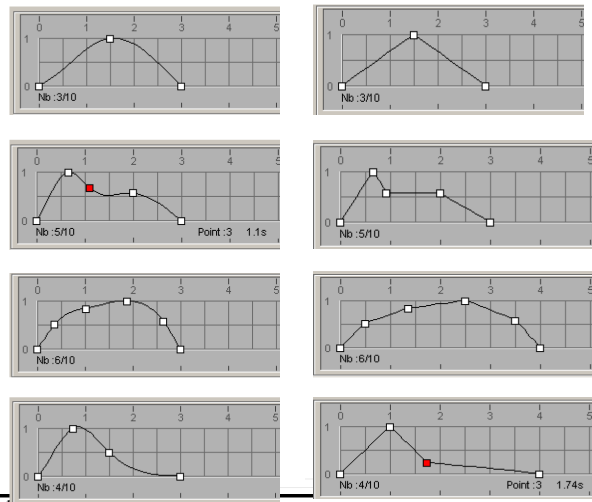


Facial Animation Parameters divided into three groups

- Lips : Lower inner midlip, Stretch corner lip etc.
- Eyes : Close right eyelid, Raise left eyebrow etc.
- Other : Puff right cheek, Roll head etc.

Different Time Envelopes for Expressions

Spline Interpolation | Linear Interpolation



Simple (Triangular)

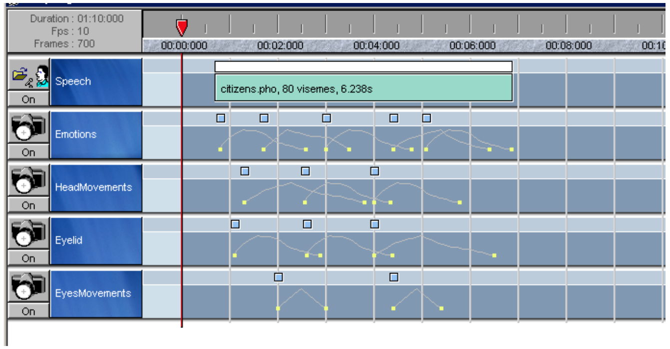
Attack-Decay-Sustain-Release

Multipoint Articulation

Quick Transition

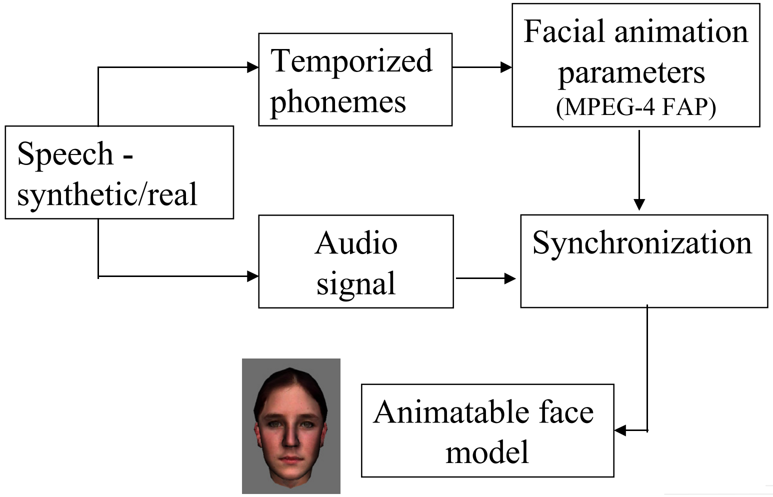
Building Animations

Possibility to add different expression envelopes at different time instants



Different animation tracks enables the designer to design head movements, facial expressions, eyebrow movements independently

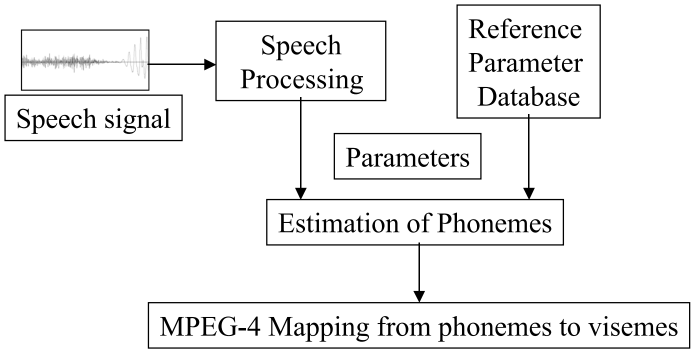
From Speech to Animation



From Natural Speech to Visemes

Extracting parameters from speech that are related to mouth shapes

Parameters : LPC, pitch, zero crossing



Mechanical Simulation of Deformable Surfaces for Animation of Synthetic Garments

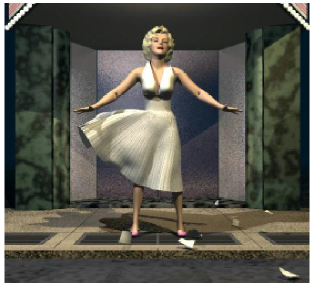
Nadia Magnenat-Thalmann
Pascal Volino
Marlène Arévalo

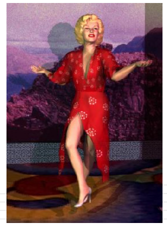
Cloth Simulation Techniques

- Geometrical Models
  - Reproduction of the geometrical deformations of the cloth.
- Mechanical Models
  - Simulation of the cloth deformations using equations derived from the mechanical behavior of fabrics.

MIRALab History

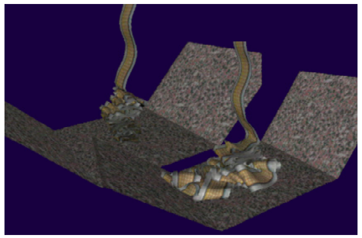
- Lafleur, Thalmann, 1991:
  - Simple viscoelastic surfaces using Lagrange equations.
- Carignan, Yang, Werner, Thalmann, 1991-92-93:
  - Modified Terzopoulos model with octree collision detection and avanced pattern-seaming garment design.





MIRALab History

- Volino, Courschesne, Thalmann, 1995-96:
  - Viscoelastic surfaces simulated with particle systems and constraint based collision response.
- Volino, Thalmann, 1997-98:
  - Fast and optimized spring mass model computed with Runge-Kutta integration and new design tools for creating garments.



MIRALab History

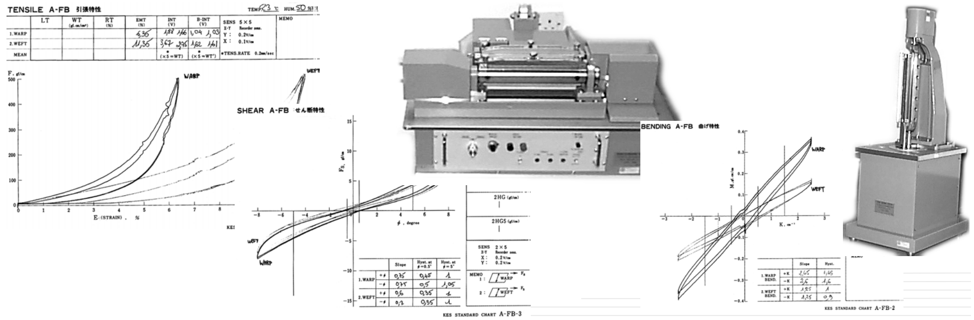
- Volino, Thalmann, 2000-01:
  - Fast and accurate model simulating dynamically complete viscoelasticity parameters using advanced implicit integration methods.

Mechanical Parameters

- Internal Forces (From surface deformations)
  - Elasticity (metric, curvature).
  - Viscosity (internal dissipation).
  - Plasticity (behavior curve hysteresis).
- External Forces (From environment interaction)
  - Gravity, Aerodynamic effects.
  - Contact reaction, Friction.
  - Miscellaneous external interactions.

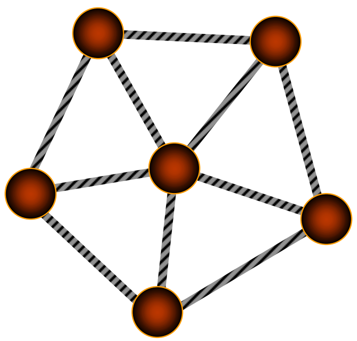
Parameter Measurements

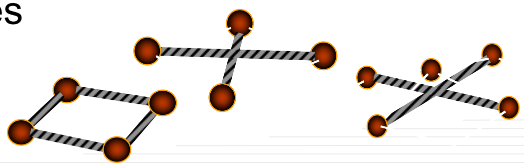
- Kawabata Evaluation System
  - Normalized procedure and equipment for measuring elasticity parameters.



Parameter Modeling

- Spring-Mass Systems
  - Discrete representation of the surface as a mesh of punctual masses, parameters represented as springs creating viscoelastic forces between them.





Parameter Modeling

- Spring-Mass Systems
  - Simple to implement.
  - Flexible for adaptation to geometrical constraints.
  - Inaccurate representation of parameters (surface anisotropy and bending).
  - Mainly used in fast simulation models for computer graphics.

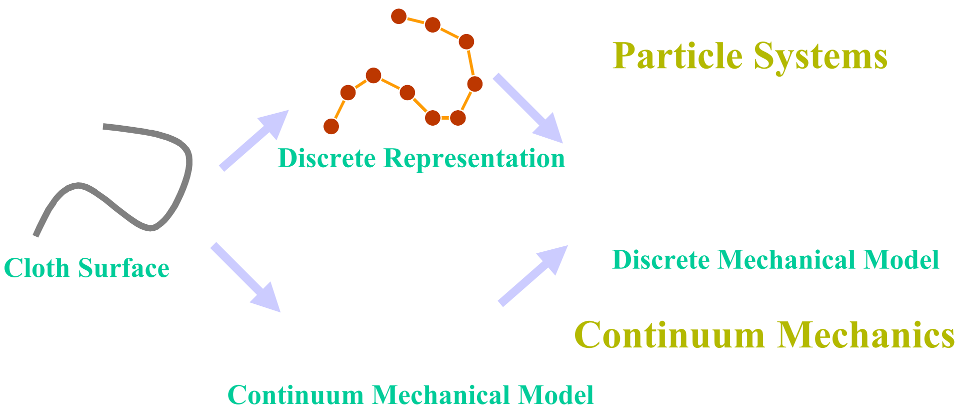
Parameter Modeling

- Continuum Mechanics
  - Expression of the surface energy and forces exerted on surface elements derived from surface deformation (Lagrange equations), and integration using finite difference discretization.

Parameter Modeling

- Continuum Mechanics
  - Accurate Modeling of material properties.
  - Complex implementation.
  - Slow computation.
  - Difficulties for integrating nonlinear models and geometrical constraints.
  - Mainly used for precise computation of simple and situations (Draping).

Parameter Modeling



Parameter Modeling

- Finite Elements
  - Particular formulation of continuum mechanics model where high-order elements are used to represent accurately deformations with adequate degrees of freedom and advanced energy minimization techniques compute the actual system evolution.

MIRALab Model

- First-Order Finite Element representation integrated using state-of the art particle systems methods.
  - Combines the advantage of accurate parameter representation with the flexibility of particle systems (choice of integration methods and collision response integration).

MIRALab Model

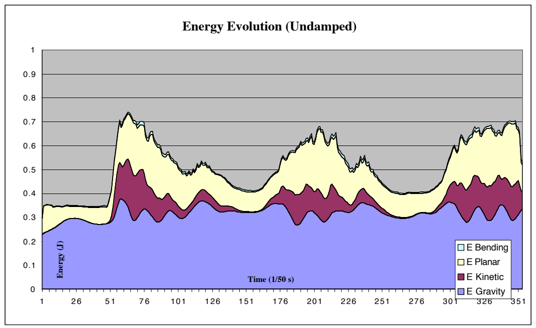
- First-Order Finite Elements
  - Degrees of freedom = Mesh vertex positions and speeds.
  - Accurate representation of metric elasticity (Anisotropic Weft-Warp and Shear elasticity curves, Poisson coefficient, viscosity curves) within elements.
  - Additional inter-element equations for modeling Weft-Warp bending forces.

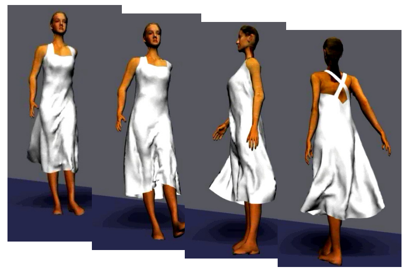
MIRALab Model

- Integration Methods
  - Explicit Runge-Kutta integration
    - Slow and precise high-order integration that ensures high accuracy level through controlled numerical error evaluation.
  - Implicit Euler and Midpoint integration
    - Fast and efficient integration that allows controlled approximations to highly speed up computations without instability problems related to explicit methods.

MIRALab Model

- Efficient and Accurate Simulations
  - Accurate evaluations of energy evolutions of the cloth during animations.



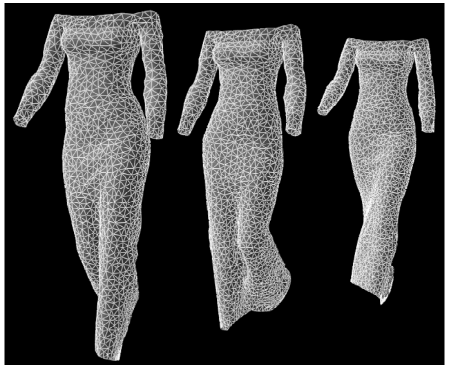


MIRALab Model

- Accurate representation of internal viscosity and damping parameters.
  - Important for producing realistic animations, not only draping on static bodies.
- Accurate representation of collision reaction and friction.
  - Allows garments to be maintained on the animated body mechanically through their own friction, without artificial “attachment points”.

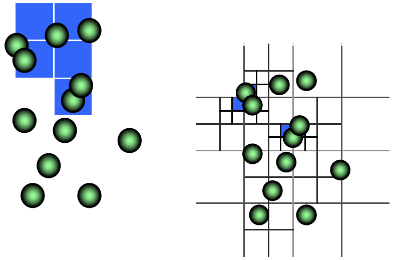
Collision Detection

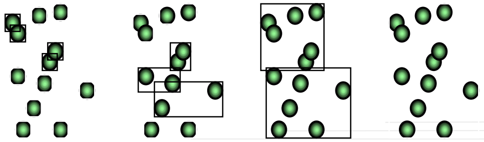
- Numerical Complexity
  - Arises from the high number of polygons that the object meshes have (cloth and body, several thousands of polygons), and how to extract the colliding polygons quickly.



Collision Detection

- Detection Techniques
  - Space subdivision: Only detect collision of objects sharing a same space region.
    - Grid subdivision (voxels).
    - Hierarchical subdivision (octree).
  - Object subdivision: Subdivide the object into geometrically localized sub-objects.
    - Hierarchical bounding-box subdivision.



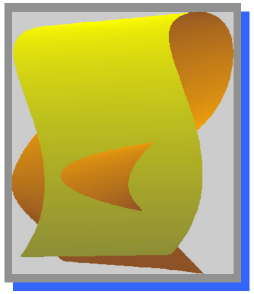


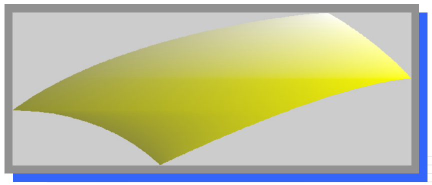
Collision Detection

- Detection Techniques
  - Space subdivision: Mostly used when dealing with numerous independent objects.
  - Object subdivision: Efficient when a constant structure can be identified between the colliding elements.
    - Adapted for the detecting collisions between mesh elements of a deformable cloth.

Self-Collision Detection

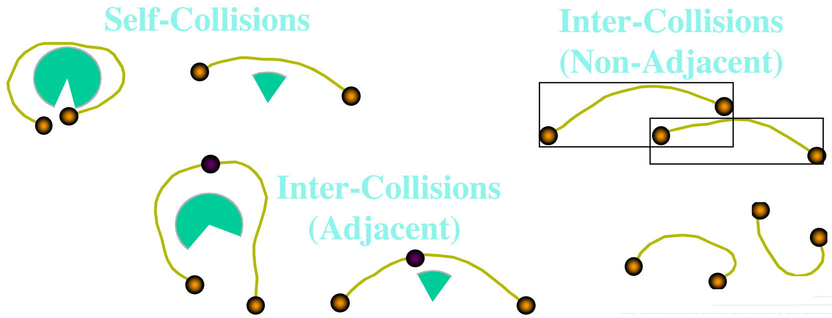
- Self-Collision Adjacency Problem
  - Avoid detection of “colliding” adjacent polygons though inclusion of curvature evaluation.
    - No self-collisions occur within a region with not enough curvature.





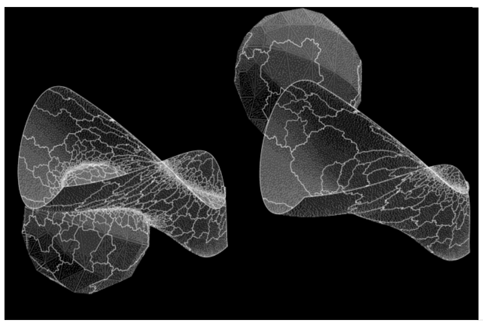
Self-Collision Detection

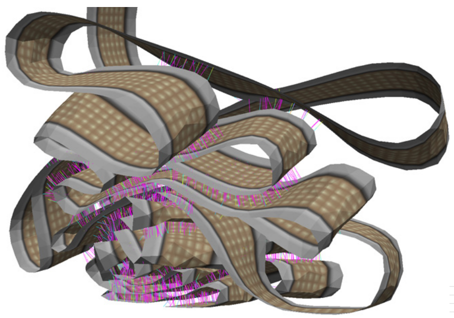
- Detection Within and Between Regions
  - Use of “curvature boxes” within regions and between adjacent regions, regular bounding boxes between non adjacent regions.



Self-Collision Detection

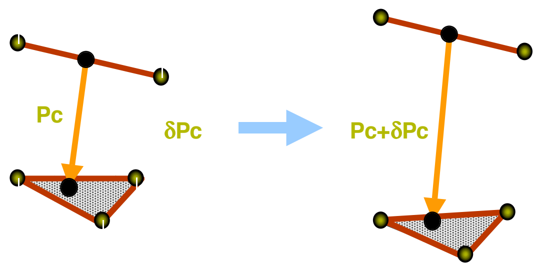
- Efficiency of self-collision detection is not the limiting factor of detection anymore.
  - Detection focused only in colliding regions.





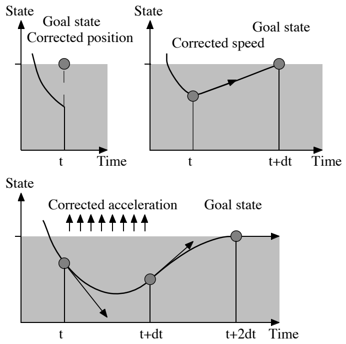
Collision Response

- Collision effect distributed on the vertices of the colliding mesh elements using mechanical momentum conservation laws.



Collision Response

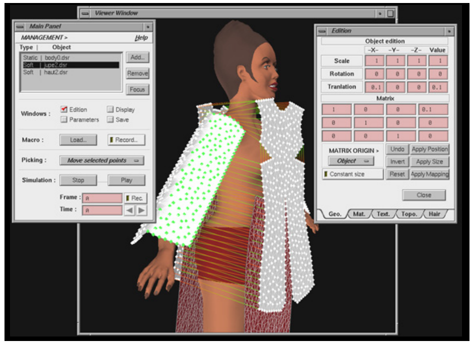
– Geometric constraint enforcement using combined correction of system state.
  - Position Correction: Obtaining desired position at current frame.
  - Speed Correction: Obtaining desired position at next frame.
  - Acceleration Correction: Obtaining desired position and speed at two next frames.

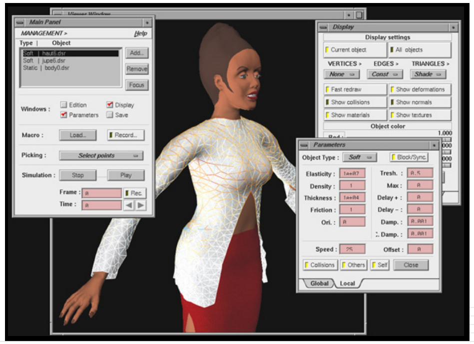


MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Designing Garments

- 3D Pattern Assembly Using Simulation



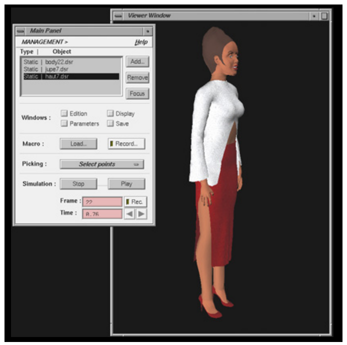


www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Animating Garments

- Mechanical Computation on Animated Body.



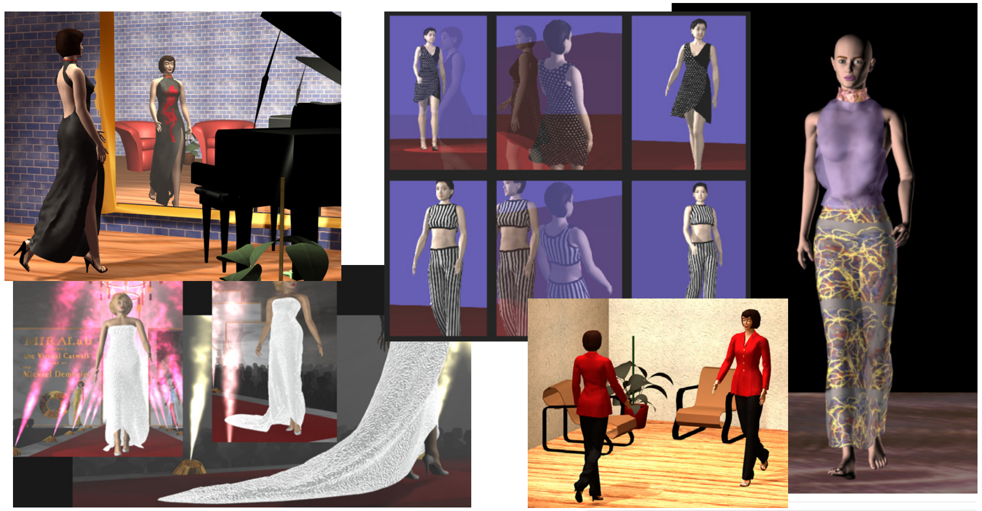
www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Virtual Fashion Design

www.miralab.unige.ch
University of Geneva

Creative Simulation



The Terra-Cotta Soldiers

Nadia Magnenat-Thalmann
Marlène Arévalo
Gaël Sannier

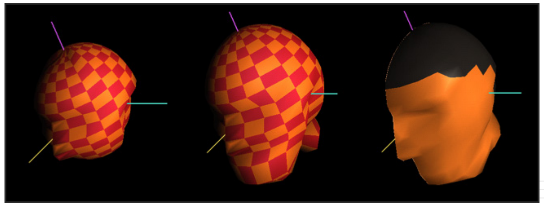
The Xian Project

- Excavation of the grave complex of the Ch'in emperor Shi Huang Ti in Xian in the 1970s has revealed a field of statues depicting soldiers, servants, and horses, estimated to total 6'000 pieces. The figures were modeled after the emperor's real army, and each face is different.
- The Xian project in 1997 is intended to recreate and give again life to this army using computer-generated techniques.

Discovery of the statues

Sculpting the Soldiers' Faces (I)

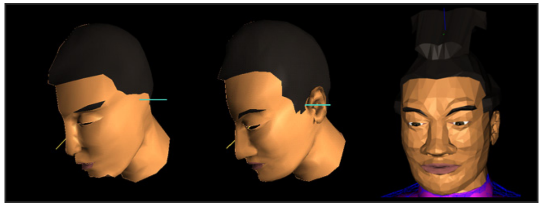
- The real soldier faces are all different and have details.
- We use a method similar to the modeling of clay; It consists of adding or eliminating parts of the material, and turning around the object.
- The steps of the first head modeling (I):
  - We apply scaling deformations on a sphere to obtain an egg shape aspect.
  - We move regions selected with triangles & also lift or move vertices.
  - We split in half in order to work more efficiently.



Creation of a soldier head from a sphere (I)

Sculpting the Soldiers' Faces (II)

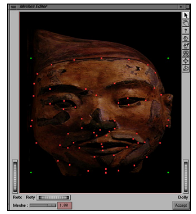
- The steps of the modeling (II):
  - We model specific regions (nose, jaws, eyes, etc) by sculpting and pushing back and forth vertices and regions.
  - We obtain an half face of the soldier to which we apply a reversed scaling on X axis to produce the other half.
  - The two sides are merged together which finally give us our first soldier's face.



Creation of a soldier head from a sphere (II)

Texture-fitting (I)

- To increase realism, we apply texture fitting to objects. We map a picture onto the object, in a way that allows the user to specify some matching points between the texture and the object:
  - We can see the texture while fitting it to the object.
  - Some interesting vertices are selected, suitable for circumscribe the area and fitting the texture to some specific features of the model. All these marked vertices are projected to the texture image.
  - We move each projected vertex to its right position on the 2D texture. The 3D object is mapped in real-time in the 3D window using the information given by the position of these marked vertices on the texture image.



Adjusting features upon the texture image

Result of the fitting in real-time in 3D

Texture-fitting (II)

- As we only have a single photo of each soldier face to model from, we create a global texture using this photo, so that this texture can be mapped around the whole head.

1- Photo of a real soldier 2- Texture image 3- 3D model

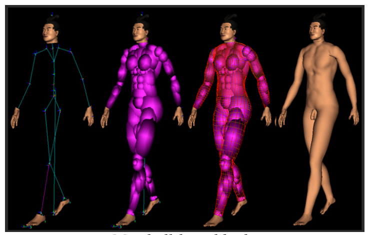
1- Photo of a real soldier 2- Texture image 3- 3D model

Final result of the whole 3D textured head

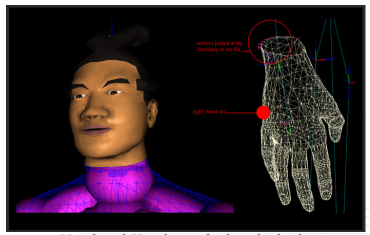
Final result of the whole 3D textured head

Creating the Soldier Bodies

- Our goal is to make realistic and efficient human modeling and deformation capabilities for many different bodies. So we use the metaball technique as it is inherent to interactive design.
- The metaballs hierarchy is taken from a standard model we have, we then modify the metaballs positions and shapes to fit soldiers anatomy.
- The head, hands and feet are attached to our body envelope.



Metaball-based body



Head and Hand attached to the body

The Film (I)

- Scenario:
  - We see first a scene with the 3D terra-cotta soldiers inside the earth.
  - It is dark with a starry sky.
  - The day is coming so more and more light is appearing. This suddenly awakes one terra cotta soldier. He is extremely astonished to see the scene around himself…

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

The Film (II)

- He notices the presence of a soldier near him and also his head which is on the ground. He took the head and put it on the next soldier's body…

- This latter start to live again. They look at each other, and all the army is slowly coming to life. They start to walk again, but the first soldiers decide to let them go...

www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

Flashback to the Future

Nadia Magnenat-Thalmann
Marlène Arévalo

www.miralab.unige.ch
University of Geneva

MIRALab
Where Research means Creativity
Eurographics2001
Inhabited Virtual Heritage

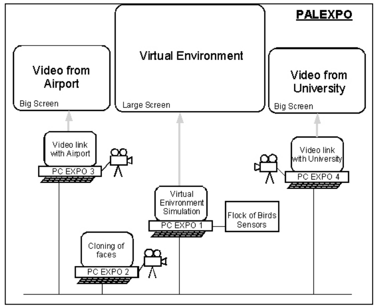
The Project

- A virtual reality experience developed in the MIRALab research laboratories of the University of Geneva. This real-time adventure, with 3D glasses, has been experienced at Palexpo in October l999, during Telecom'99.

www.miralab.unige.ch
University of Geneva

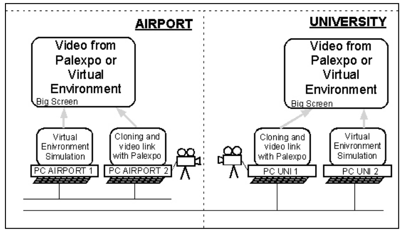
The Project

- To illustrate telecommunications, the show communicates in real time with three distant booths, one located in Palexpo, the second one in the Uni Dufour Hall and the third one at the Geneva Airport.



Booth at Palexpo

The Project



Booths at the University and at the Airport

The Project

- Real people are being cloned, and their virtual counterparts take part in 3D scenes from the past and the future.
- To do the virtual double of each person, we use a procedure based on two photographs, that can reconstruct the faces of individuals in 3D.

Face Cloning

The Project

- This world première illustrates the face-to-face interaction within the virtual scene of individuals who in reality are situated at a distance from each other, like you and I.
- It is also a first for the reconstruction of the Vieille Ville by computer and for the appearance of a virtual Mère Royaume.

The Vieille Ville of Geneva in real

The Vieille Ville of Geneva in virtual

1602

- Escalade: soldiers from Haute Savoie tried to invade Geneva and were stopped by the Geneva inhabitants and more particularly the “Mere Royaume”, who spilled the content of her cauldron over the invaders.

The Mère Royaume and 2 soldiers

1602: The Mère Royaume

The making of the SS. Sergius and Bacchus edifice

Nadia Magnenat-Thalmann
Allessandro Foni
Grégoire L’Hoste
Georgios Papagiannakis

The CAHRISMA project (I)

- Main objective of the CAHRISMA project (Conservation of the Acoustical Heritage by the Revival and Identification of the Sinans Mosques) is to innovate the concept of hybrid architectural heritage.
- Hybrid architectural heritage is a new way of identification that covers acoustical characteristics besides visual peculiarities.
- It states that, for the spaces, having acoustical importance, architectural heritage concept should be upgraded covering acoustical and visual properties. The effects of this improvement will reflect to actual implementation of conservation and restoration.

The CAHRISMA project (II)

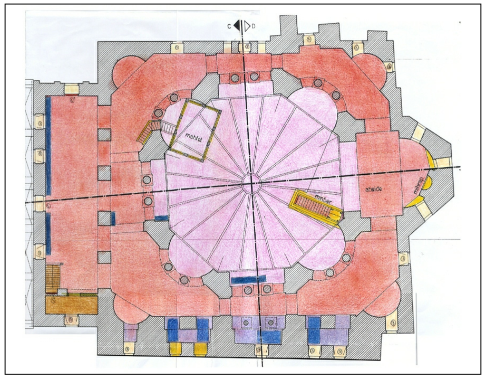
- MIRALab's involvement:
  - Real-time visualisation of selected spaces.
  - Creation of people (virtual bodies, faces and cloth textures).
  - Animation of virtual humans.
  - Integration of visual and acoustical models into a virtual 3D interactive system.
- One of the monuments selected for this project is SS. Sergius and Bacchus edifice in Istanbul.

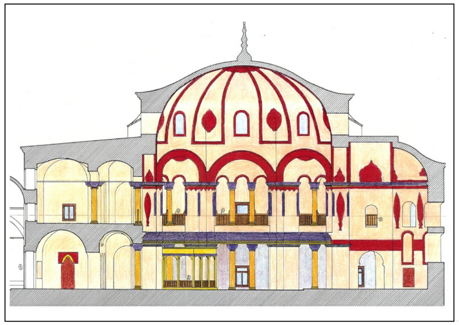
SS. Sergius and Bacchus church

- The church of the SS. Sergius and Bacchus, a landmark in Byzantine ecclesiastical architecture, was founded by Justinian probably in 527, the first year of his reign.
- The church of the SS. Sergius and Bacchus known to this day as "the Little Hagia Sophia", because the general principles of its architecture are comparable with those of the Great Church.
- Sometime between 1506 and 1512, the church of the SS. Sergius and Bacchus was converted into a mosque. The atrium was replaced by a peristyle, surviving to this day, and a courtyard where the medrese (religious school) stands today.

Reconstruction of the edifice 3D model (I)

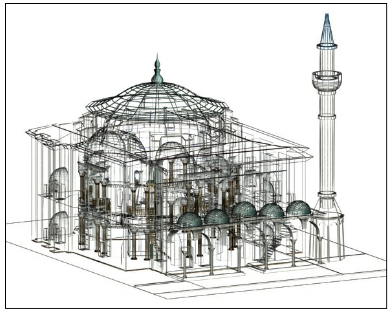
- The 3D model of the SS. Sergius and Bacchus edifice is reconstructed from the available architectural plans and the visual data resulted from the data collection process performed by UNIGE and EPFL teams.





Reconstruction of the edifice 3D model (II)

- The whole edifice is reconstructed in three dimensions using polygonal method of 3D Studio Max software.



View of the mesh model from 3D Studio Max

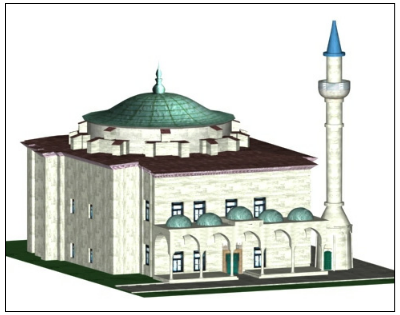
- During the modelling phase special consideration are taken to keep the number of polygons as low as possible, so that the final model would be optimised for real-time visualisation.

Texturing the 3D model

- The texture are created from 2D photographs, they are used as texture image maps to improve the visual details of the 3D model. A special care is taken to correct for the perspective of the picture and to enhance the aspect of the texture.

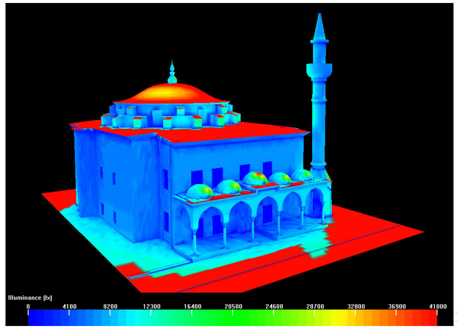

Actual picture


Texture extracted from the picture


Textured 3D model

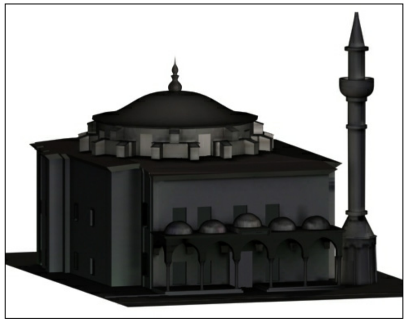
Lighting the 3D model

- The lighting of the 3D model is done with Lightscape software, as it allows for realistic lighting effects.
- The techniques used are physical based model of global illumination, such as radiosity and ray-tracing.


Distribution of light on the surfaces of the 3D model

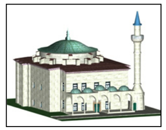
Use of light maps for realistic visualisation

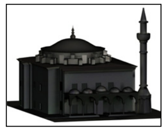
- The creation and use of light-maps, from the lights generated in Lightscape, allows the real-time visualisation of the realistic lighting.


Light-maps applied on the 3D model

Visualisation of the 3D model

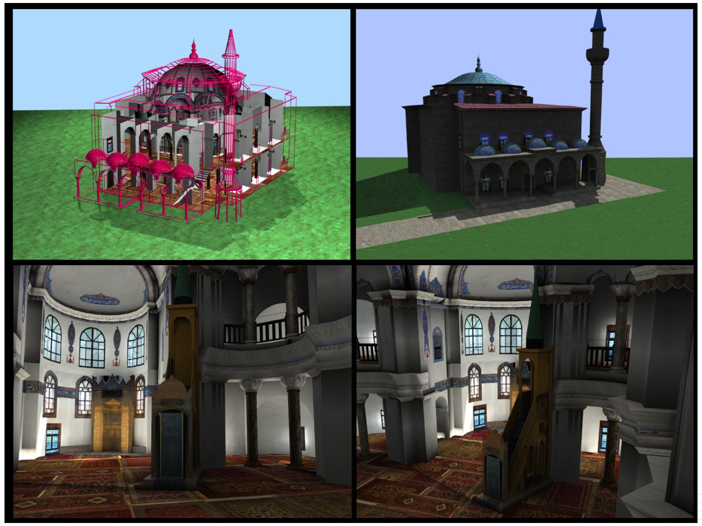
- Both textured and light-mapped models are exported in VRML and merged together for real-time visualisation on MIRALab's real-time rendering engine or on the World Wide Web.


Textures


Light-maps


Final model of the edifice

The results (I)

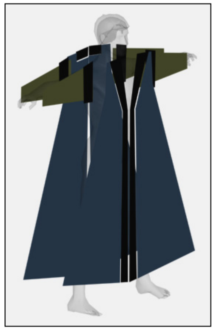


The results (II)

Creation of virtual humans

- Virtual humans are also modelled using the polygon method. The clothes of the model are realised with MIRALab cloth plug-in, according to picture of ancient time people.

Historical model



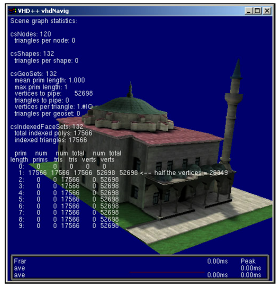
Patterns of the clothes

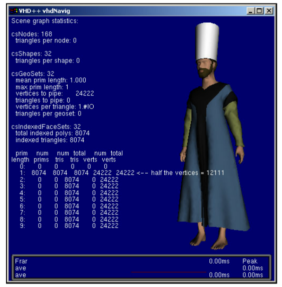
- Virtual human are then converted to h-anim standard format and animated with Vicon motion capture data.

3D dressed virtual human

Real-time visualisation of the 3D model

- Both model of the edifice and of the virtual human are loaded in MIRALab's real-time rendering engine. User can walk inside the 3D model and examine it interactively.





 Volume 19 (2000), Number 3

Generating Animatable 3D Virtual Humans from Photographs

WonSook. Lee, Jin Gu, and Nadia Magnenat-Thalmann

MIRALab, CUI, University of Geneva, Switzerland
Web: http://www.miralab.unige.ch
E-mail: {wslee, gu, thalmann}@cui.unige.ch

Abstract

We present an easy, practical and efficient full body cloning methodology. This system utilizes photos taken from the front, side and back of a person in any given imaging environment without requiring a special background or a controlled illuminating condition. A seamless generic body specified in the VRML H-Anim 1.1 format is used to generate an individualized virtual human. The system is composed of two major components: face-cloning and body-cloning. The face-cloning component uses feature points on front and side images and then applies DFFD for shape modification. Next a fully automatic seamless texture mapping is generated for 360^o *coloring on a 3D polygonal model. The body-cloning component has two steps: (i) feature points specification, which enables automatic silhouette detection in an arbitrary background (ii) two-stage body modification by using feature points and body silhouette respectively. The final integrated human model has photo-realistic animatable face, hands, feet and body. The result can be visualized in any VRML compliant browser.*

1. Introduction

In recent years, modeling virtual human body has attracted more and more attention from both the research and industrial community. It is no longer fantasy to imagine that one can see herself/himself in a virtual environment moving, talking and interacting with other virtual figures or even with real humans. By advances in algorithms and new developments in the supporting hardware this fantasy has become a reality.

The issues involved in modeling a virtual human model are as follows:

- acquisition of human face and body shape data
- realistic high-resolution texture
- functional information for animation of the human face and body

We address how to acquire an animatable human body with a realistic appearance. It is our goal to develop a technique that enables an easy acquisition of the avatar model having the ability to be animated well and produced at a low cost. There are two basic types of techniques for obtaining 3D object models, according to the different requirements for the models. The first type of technique focuses on the accuracy and precision of the obtained object models, such as those used in CAD systems and industrial applications. The second type of techniques concentrates on the shape and visual realism of the reconstructed models, such as those used in virtual reality applications.

When we have to place importance on the accuracy of the shape, there are various approaches to the reconstruction of a face either using a sculptor[8], a laser scanner[22], a stereoscopic camera[21], an active light stripper[24], video stream[16, 9]. In recent years, body cloning has also become an increasingly hot topic. Similarly, there are many methods that concern precision and accuracy[16, 11, 1, 7, 10, 13]. Generally, these systems are either expensive or require expertise knowledge in using them and need a special environment setting. Thus, most of them have limitations when compared practically to a commercial product (such as a camera) for the input of data for reconstruction and finally animation.

On the other hand, systems using the second type of techniques are much cheaper and easier to use. These techniques are usually model-based. There are several approaches to the reconstruction of either a face[2, 15, 17, 20] or a body[12] from photo date. These approaches concern mainly the individualized shape and visual realism using a high quality image input. For example, Hilton et al.[12] proposed a method for cloning virtual people. A generic human model is taken and information extracted from photos is used to modify the generic model. The approach is simple and efficient. However this method does not give a good reconstruction and animation for the face. In addition, their generic model is not seamless, which means the regions around certain skeleton joints do not have the smoothly connected surface, so that the final textured model has some mismatching problem when we animate the joints. It also lacks the flexibility in terms of the imaging environment since it requires a specially prepared background and properly controlled lighting when images are taken.

Our approach, which belongs to the second type, addresses the following questions and suggests the solutions.

What to produce from what?

We produce a realistic and animatable whole body including the face, hands and body from photo data. Every body parts are smoothly connected and textured. Photographs of the whole body cannot provide sufficient facial information in order to construct a good face model and further facial animation. Therefore, we take two additional photos that focus on the face only, besides the three whole body photos.

How easy is the environment to get the input?

We use simple snapshots with commercial cameras without any special environment. Instead of seeking a solution by using special environment, we provide a user-friendly interface, which allows non-expert user to interactively hint to the system certain important information about the human body. In this way, with a little amount of user interaction, we achieve more flexibility in using the system.

How automatic is the processing for users?

We provide an automatic system except for a few interactions at the beginning as shown in Figure 1.

How much can we animate?

The individualized virtual human inherites the functional structure from the generic human with animation capacity on the face and body.

How easy is it to visualize with other applications?

The VRML Humanoid Animation Working Group (H-Anim) exists for the main purpose of creating a standard VRML representation for humanoid. Our generic body is in VRML H-Anim 1.1 format[14] and the resulting body can be visualized by web browsers, such as Netscape and animated by a JAVA program.

The outline of the algorithm is shown in Figure 1. Section 2 is devoted to the face-cloning program while Section 3 explains the body cloning. The results are shown in Section 4 and are concluded in Section 5.

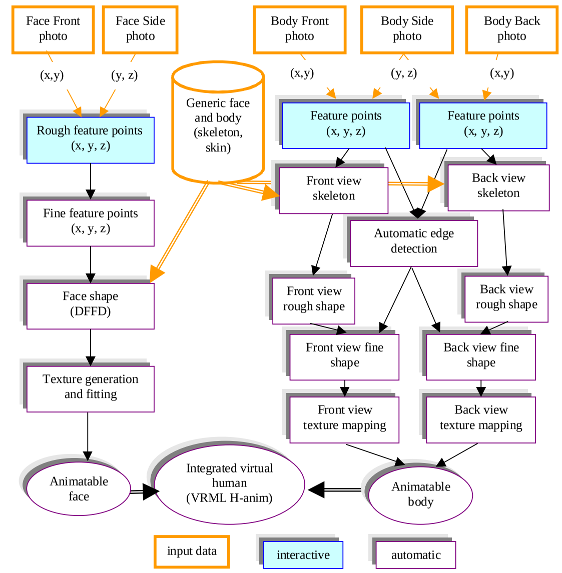


Figure 1: *Overflow of face and body cloning*

2. Face cloning

2.1. Shape modeling

In this section, we present a way to reconstruct a photo-realistic head for animation from orthogonal pictures. First, we prepare a generic head with an animation structure and two orthogonal pictures of the front and side views. The generic head has efficient triangulation, with finer triangles over the highly curved and/or highly articulated regions of the face and larger triangles elsewhere. It also includes eyeballs and teeth.

The main idea to get an individualized head, is to detect feature points (eyes, nose, lips, and so on) on the two images and then obtain the 3D position of the feature points to modify a generic head using a geometrical deformation. The feature detection is processed in a semi-automatic way. The user sets a very few feature points (key points) and the other feature points are fitted using a *piecewise affine transformation* first and then snake methods. The structure snake method with some anchor functionality is described in another paper[19]. Then, two 2D position coordinates in the front and side views, which are the *XY* and the *ZY* planes, are combined to be a 3D point. After using a global transformation to move the 3D feature points to the space for a generic head, Dirichlet Free Form Deformations (DFFD)[23] are used to get new geometrical coordinates of a generic head adapting to the detected feature points. The control points for the DFFD

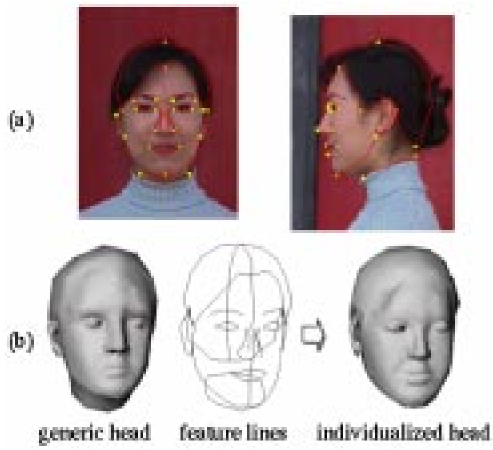


Figure 2: *(a) normalization and features. (b) Modification of a generic head with feature points*

are feature points detected from the images. Then the shapes of the eyes and teeth are separately adapted to the new head with translation and scaling from the generic model. Figure 2 shows the steps for head modification from photos.

2.2. Texture mapping

Texture mapping is useful not only to cover the rough matched shape, as here the shape is obtained only by feature point matching, but also to get a more realistic colorful face.

The main idea of texture mapping is to get an image by combining two orthogonal pictures in a proper way to get the highest resolution for the most detailed parts. The detected feature points data is used for automatic texture generation by combining two views (actually three views by creating the left view by flipping the right view). We first connect two pictures with a predefined index for feature lines using a geometrical deformation (see Figure 3 (a)) and a multi-resolution technique[6] for removing boundaries between different image source (see Figure 3(b)). The eyes and teeth images are added automatically on top of an image, and these are necessary for the animation of the eyes and mouth region.

To give a proper coordinate on a combined image for every point on a head, we first project an individualized 3D head onto three planes such as the front (XY), right (ZY) and left (ZY) directions. With the information of the predefined index for feature lines, which are used for image merging above, we decide on which plane a point on a 3D head is projected. Then projected points on one of three planes are transferred to either the front feature points space or the side feature points space in 2D. Finally, a transform on the image space is processed to obtain the texture coordinates. More details are found in the paper[20].

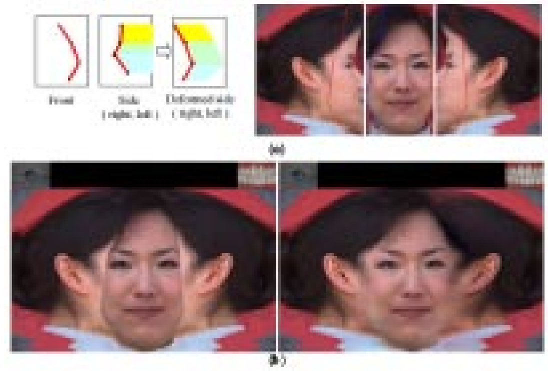


Figure 3: *(a) A geometrical deformation for the side views to connect to the front view (b) before and after multi-resolution techniques.*

Figure 4* shows several views of the final reconstructed head out of two pictures in Figure 2(a). When we connect this head with a body, we remove the neck (see the second last face in Figure 4*) since the neck is from the body due to the body skeleton animation for face rotation. The face animation is immediately possible as being inherited from the generic head as shown in the last face in Figure 4*.

Figure 4: *snapshots of a reconstructed head in several views and animation on the face*

3. Body cloning

Our body cloning is a model-based method. We use two main inputs. The first input is the generic body. The second is still photos of a person to be cloned. We assume the person wears trousers and not too loose clothes. We deform the generic body to adapt to the individualized body.

3.1. Generic body structure

The generic body is in MPEG-4 compatible H-Anim 1.1 formats[14]. The skeleton and several skin parts displayed with several colors are shown in Figure 5 where the skin parts are smoothly connected. It has 94 skeleton joints and 15 skin parts including *head*, *right_hand*, *left_hand*, *right_foot* and *left_foot*. The first version of generic body we are using is collected from a public domain[3] and modified for our usage. Each skin part is saved in local coordinates and is related to a skeleton joint as shown in Figure 6, where the skeleton location is indicated by arrows and related skin parts are

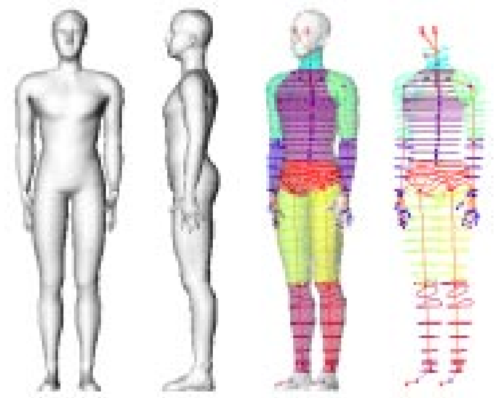


Figure 5: *The seamless generic body with H-Anim 1.1 skeleton and several skin parts*

written inside (). The right side skin parts are not shown, but it is easy to guess from the left side. Each skin part is transformed into global coordinates by a 4x4 matrix which connects to the corresponding skeleton joints.

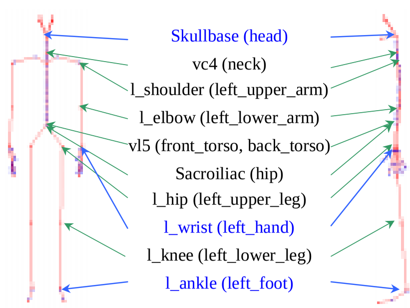


Figure 6: *The H-Anim joints related to skin parts*

The 12 skin parts beside *head*, *hands* and *feet* are designed to have real-time deformation for animation[3]. The good points of these skin parts are that first they compose a seamless skin envelope, so that the texture mapping will be smoothly connected with animation. Secondly the way how to organize the points is specially designed such that each skin part is composed of several slices and each slice in the skin part has the same number of points. For example the *hip* has 6 slices and 26 points on each slice (the total point number on *hip* is 6x26 = 156). We call it as the *grid structure*, which makes the *piecewise affine transformation* possible in later sections.

The *head*, *hands* and *feet* are separate objects with different structures from *grid structure*. They are also animated in different ways.

3.2. Taking photographs and initialization

We focus on the simplest environment to take photos with only one camera. We take three photos, from the front, the side and back. In this case, the front and back views are not exact reflections of each other since we asked the person to rotate for the back view after taking the front view.

We input the height of the person and the image body heights are checked on the three images for normalization. Figure 7 shows normalized images. Since we use arbitrary background to take photos and the size of the person is not fixed, we use interactive feature points localization on images as shown as small points in Figure 7. This simple feature points localization is used for skeleton modification, rough skin deformation and automatic edge detection in the next sections.

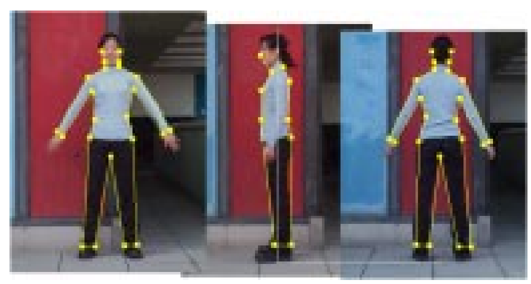


Figure 7: *Feature points on three images*

3.3. Skeleton modification

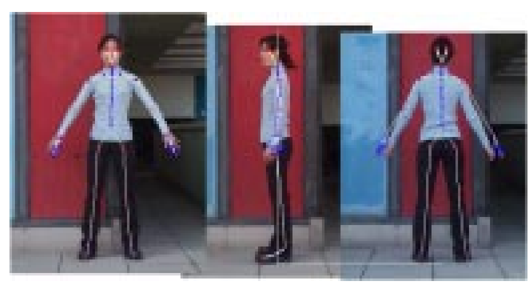


Figure 8: *Automatic Skeleton fitting with feature points.*

When we have feature points for the skin envelope (even though the person put on clothes, we assume that the clothes outlines are close to the skin outlines), we can have an estimation of the skeleton. For example, for the *r_elbow* must be located around middle position between the right end shoulder point and outer end point of the right wrist. Here we apply affine transformations and Barycentric interpolation to find the typical skeleton joints from feature points. Since there are 94 skeleton joints, we make a subset of joints as key joints, which are modified by feature points while others are modified by *piecewise affine transformations* defined by key joints and the skeleton hierarchy.

The skeleton joints of the *head*, *hands* and *feet* are simply scaled and translated with the transformation between the generic body's skeleton and the person's skeleton, for example *vt1* and *HumanoidRoot* can be used to find the transformation. Figure 8 shows the skeleton modified by the front and the side views. Since the front and back views do not have the exactly same pose, the skeleton modified by the front view does not match on the back view.

3.4. Rough skin modification

As we mentioned in the section 3.1, each skin part is connected to a skeleton joint by a 4x4 matrix to be in global coordinates. We update the matrix by scaling, translation and rotation defined by the corresponding skeleton joint and the child skeleton joints. As we see in Figure 9, adjacent skin parts are not guaranteed to be continuous. The shoulder parts are overlapping with torso parts.

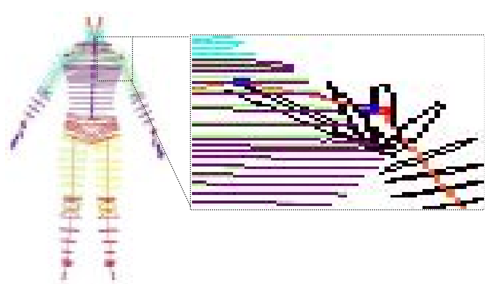


Figure 9: *Updated skin parts by skeleton joints and related matrix*

A simple linear transformation by 4x4 matrix does not solve the overlapping or separating problems. So we need a special transformation to solve the overlapping (or separating) problem and integrate the skin parts properly with an approximated shape to the person.

Here we define a freeform deformation to make a rough matched continuous body with feature points information. The control points are placed at certain required positions to represent the shape characteristics. Hence the skin model can be deformed by moving these control points, which is designed such as they have corresponding points on the front (or back) and the side view images, or the locations are found with certain relations. For example the boundary between the *front_torso* and *right_upper_arm* can be found from the feature points on the bottom-right corner point of the neck and on the right end shoulder point on the images. Furthermore, several control points are located at the boundaries between two parts, so that surface continuity is preserved when the posture of the generic body is changed. These control points are used for the *piecewise affine transformation*.

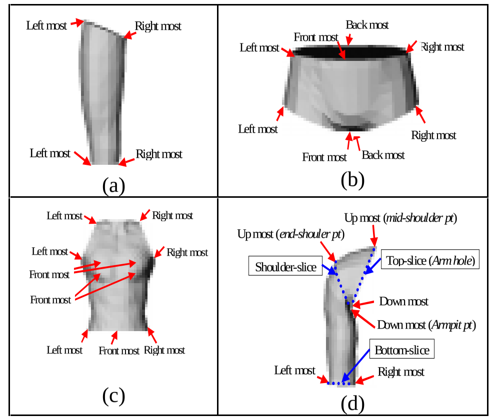


Figure 10: *The control points on right_upper_leg, hip, front_torso and right_upper_arm parts*

We apply separate deformations for each skin part. We show some examples of the control points in Figure 10.

We apply first *piecewise affine transformation* where each affine transformation is defined on each segment on a curve. Thanks to the *grid structure* of skin parts, we apply the *piecewise affine transformation* horizontally first on slices with control points and then vertically on points which have the same index on each slice. For example for the *hip* part, we define the first affine transformation with the left-most point on the top-slice and the front-most point on the top-slice and corresponding control points on images. Also we apply the affine transformation on points between the left-most point and the front-most point. Then we define the next affine transformation for front-most point and right-most point and apply it to points in between. We define and apply the third and forth affine transformations on the other two pieces on the top-slice. Then we get a new top-slice from the generic body's slice, which fits to the person's top-slice now. Then we apply the similar process for the bottom-slice that has control points too. After getting the new bottom-slice, we define an affine transformation in the vertical direction using two points with the same index on the top-slice and the bottom-slice. Figure 11 shows the steps for the *hip* part. For the *_upper_arm parts, we apply *piecewise affine transformation* for three slices such as the top-slice, the shoulder-slice, and the bottom-slice as shown in in Figure 10 (d).

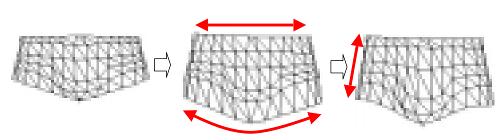


Figure 11: *The process of the affine transformations for the hip*

After applying the *piecewise affine transformation*, we make one more process to stick adjacent skin parts properly. When the generic body is loaded, the connection database is built automatically and it is used for the skin parts connection. The database contains which point on which skin part is connected to which point on which skin part. The *neck* is not deformed using the feature points from images since it is too small on the images. The *piecewise affine transformations* are defined from adjacent points on the *head* with the top-slice on *neck* and from adjacent points on the *front_torso* and *back_torso* with the bottom-slice on *neck*.

It is a rough deformation since we deform the generic body only with a few feature points. So it does not match exactly for the outfit on the images. However it serves as an initialization of skin for the shape and catches a proper functional approximation for animation such as having a proper skeleton and skin parts localization.

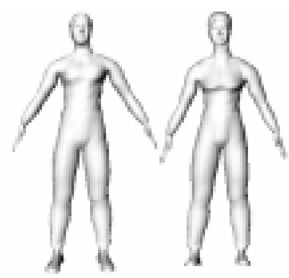


Figure 12: *two bodies (front+side/back+side) with rough skin deformation*

Since the front and back views were taken in different positions, we produce two bodies as seen in Figure 12. The left one obtained from the front and the side images and the right body from the back and the side images. The reason to produce two bodies is because of two sources for texture mapping, which will be described later.

3.5. Heuristic based silhouette extraction

There are plenty of literatures available about boundary extraction or edge linking[5, 18]. It can be treated as a graph-searching problem, as an optimization problem, or as an energy minimization and regularization procedure. However, these algorithms are usually inefficient due to the need for backtracking or exhaustive search. Or the algorithms need time to reach convergence or stable result, such as the snake algorithm. We design a simple algorithm by making use of the feature points on the body. In this section these feature points serves as the heuristics for the body silhouette extraction.

First, Canny edge detector is applied to every image. Then a coloring-like linking algorithm is used to link the edgels (edge pixels) into connected segments. Due to possible noise caused by the background, the edgels generated by the background sometimes are connected to the body edgels. To avoid this potential wrong connection from occurring, we split the segments into short line segments.

To make the following discussion easier, let us call the line segments formed by consecutive feature points as *feature segments*, while the short line segments formed by linking edge pixels, as *edge segments*. Each feature segment indicates the vicinity and approximate direction of the boundary to be found. From the Canny edge detector and linking step, we obtain the *edge segments* generated by the object as well as by the background. We first throw away those lying outside the vicinity of any *feature segments*. The goal now is, for each feature segment, to find a path that is formed by an ordered set of *edge segments* within its vicinity.

To link the *edge segments* into meaningful boundaries, we first look for the admissible connection for each edge segment. See Figure 13. We define a connection between edge segment $S_1 = P_{12} - P_{11}$ and $S_2 = P_{22} - P_{21}$ as admissible if:

$$S_1 \cdot S_2 \geq 0.0, \;\; \alpha_1 < T_{sm}, \;\; \alpha_2 < T_{sm}$$

where P_{11}, P_{12}, P_{21} and P_{22} are the ends of the two segments under consideration, α_1 is the angle between S_1 and the potential connecting segment $C_{12} = P_{21} - P_{12}$, α_2 is the angle between C_{12} and S_2, T_{sm} is the maximum angle allowed for the connection. Next in order to select the most desired connection, we define a function G^L to evaluate the "goodness" of the potential connection :

$$G^L(S_1, S_2) = cos(\alpha_1)cos(\alpha_2) \frac{(1 + cos(\alpha_1)cos(\alpha_2)log(M_2))}{(wD_1 + (1-w)D_2)}$$

where D_1 is the length of the C_{12} and D_2 is the distance from the P_{22} to the feature segment L, M_2 is the edge magnitude of edge segment S_2, w is the weight of D_1 taken as a constant in our experiments. The rationale behind this definition is that we always favor a connection that contributes to a smooth path running along L, formed by segments with strong edge magnitude.

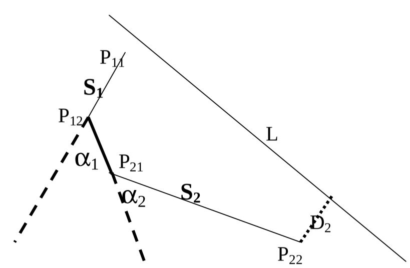


Figure 13: *Link two edge segments*

Thus based on G^L, we find, for the two ends of each edge segment, its best connection that maximizes G^L among all of its neighboring *edge segments*. Now we are ready to build the path. Starting from each edge segment, we connect it to its two best connections computed by G^L to form a partial path. For the *edge segments* sitting on the head and tail of this partial path, we connect their open ends to their respective best connections. This procedure is repeated until there is no further connection possible. So a path $\mathbf{P}$ consists of a sequence of *edge segments* S_i, $i = 1,2,...,N$. Now we need another evaluation function to assess the "goodness" of a path. We define a function G^P as:

$$G^P(P) = \Sigma M_i / (w_1 DT_1 + w_2 DT_2 + w_3 \Sigma D_i (1 + sin(\alpha_{i,1}) + sin(\alpha_{i,2})))$$

where ΣM_i is the summation of the edge magnitudes, DT_1 is the distance between the path ends to the ends of the feature segment, and DT_2 is the average distance from the pixels on the path to the feature segment. D_i, $\alpha_{i,1}$ and $\alpha_{i,2}$ correspond to D_1, α_1 and α_2 in Figure 13, respectively. w_1, w_2, w_3 are the weights of each measurement (here we take the value 0.2, 0.2 and 0.8 respectively). Among all the paths found, the one $\mathbf{P}^*$ maximizing G^P is selected, i.e. $\mathbf{P}^* = arg\max_P(G^P(\mathbf{P}))$. We show a few examples in Figure 14.

3.6. Fine skin modification with silhouette information

After the skeleton adjustment and rough skin modification in the previous sections, the skin parts have been adjusted into proper orientation and rough size. Now we discuss how the image silhouettes are used to further modify the body so that the final body will produce the same silhouettes as those extracted from the images.

To build the 2D-3D association, we back project the 2D into 3D space. The silhouettes from the front or back view are mapped onto the XY plane and also the side view is mapped onto the ZY plane. For each view v, every slice S_i has two points, V^v_{i1} and V^v_{i2} on the occluding slice. These two points correspond to two pixels on the silhouette. Denote these two corresponding pixels as P^v_{j1} and P^v_{j2}. Generally the number of pixels is much larger than the number of slices, so we use the following formula to select the proper pixel pair:

$$P^v_{jk} = P^v_1 + (MC_i - MC_1)/(MC_K - MC_1)(P^v_N - P^v_1)$$

where $k = 1,2$, and P^v_1 and P^v_N are the first and last pixel on the silhouette, $MC_i, i = 1,2,...,K$ is the mass center of slice i. All the parts except the arms have silhouettes from two views, i.e. front/back and side views. The arms only have silhouettes extracted from the front/back view. Now the problem is reduced to how to modify the model slice given 2 or 4 data points that are associated to points on the slice. We first apply a global translation to all the points on the slice so that the mass center of the slice coincides with the midpoint of the two back-projected pixels. Then we do a global scaling

Figure 14: *Images with super-imposed silhouette*

to the slice. The scaling factor is estimated as the ratio of the distance between the two associated pixels and that between the two points. The silhouette from front/back image is used to scale the slice on XY plane and that from side view is used to scale in ZY plane. In order to ensure the two points V^v_{i1} and V^v_{i2} that sit on the occluding slice to produce the pixels same as the two associated pixels P^v_{i1} and P^v_{i2}, we apply a translation $T_{i,m}$ to each point $V_{i,m}$ of the slice as follows:

$$T_{i,m} = w(P^v_{i1} - V^v_{i1}) + (1-w)(P^v_{i2} - V^v_{i2})$$

where $w = ArcL(V_{i,m}, V^v_{i2})/ArcL(V^v_{i1}, V^v_{i2})$, $m = 1,2,...,N_i, N_i$ is the number of points on slice S_i, and $ArcL(.)$ is a function computing the arc length of the slice curve. This makes sure the modified slice will generate the exactly same image pixel points under the same projection while keeping the curve smoothness. Special care is needed for the shoulder-upper arms joining. The generic body has slices that are used to smoothly transit the shoulder to upper arm (see Figure 10 (d)). However there is actually no explicit corresponding information in the image. Fortunately, we can rely on the association of the upper arm with the torso to adjust these few slices. First of all, the orientation and the size in XY plane of these slices are adjusted by making use of the silhouette generated by the *middle shoulder point* and the estimated *armpit point*. Secondly, we have to enforce the upper arm to be seamed to the armhole, which is associated with the torso. In such a way, we can estimate a rough scaling in YZ plane since the torso has been modified by its side view silhouettes.

3.7. Texture mapping

We use only the front and back views for texture mapping since the two views are enough to cover the whole body except for the head. The head texture mapping is done with the front and the side views as shown in the section 3.2.

For the texture mapping, we have to give texture coordinates to points on skin envelope. Since there are two images used, we have to make a partition of the skin envelope polygons, either to the front view or to the back view by checking the cross product of the vertex normal with the viewing vector. If the point belongs to a front (back) view polygon (i.e. visible to front/back view point), we define the point as having front (back) view. If the vertex belongs to both a front view polygon and a back view polygon, we define the vertex as having front+back view. Since the vertex with front+back view is located on the boundary of the front and side views, we set two texture coordinates, one in the front view image and the other in the back view image. For the other vertices, it is straightforward to set the texture coordinate either in the front view image or in the back view image.

To get the texture coordinates, we use a projection onto the XY plane in the image space. Here we have to pay attention since there are two bodies either from the front and the side views or the back and side views. We follow the process such as:

1. deform the body with the back and side views;
2. project back/front+back viewpoints onto the back view image plane to get the texture coordinates;
3. deform the body with the front and side views;
4. project front/front+back viewpoints onto the front view image plane to get the texture coordinates.

Then the final individualized body has the proper texture mapping on both the front view and the back view. However there are still some problems on the boundaries as shown in the left side images in Figure 15. There are mainly two reasons causing this problem. First, due to the size of polygons on the virtual body, which is much bigger than the pixel size of images, the projected boundaries of the triangles on the frontal and back view boundary do not match to the detected boundaries based on pixel size. In addition due to the limited digitization resolution of the camera, the pixel colors on the boundary of foreground and background are usually the smeared combination of the boundary and foreground color. Furthermore the noisy effect of the texture is magnified by the 3D triangles on the boundaries. Secondly although the two images used for texture mapping are usually taken under same illumination condition, they are not necessarily to have the same visual intensities or colors. So when they are mapped onto the 3D body, the difference in the color and intensity can be easily perceived.

In order to remove the first cause by digitization process, we modify the pixel colors within the neighborhood of each edge pixel. Since from the previous edge detection processing, we have already known which side of the boundary is background, we search along the perpendicular direction to the edge for a foreground pixel and take its color as the color for the edge neighborhood pixels. As result shows, this simple processing removes the noisy effect of the digitization process.

Next, we need to smooth the frontal and back texture to remove the difference between the two images. From the feature points given in the previous processing, we can recognize semantically the various body parts, hence establish the part correspondences between the two images. We further find the pixel correspondence according to the boundary lengths. Within the neighborhood of the two pixels from frontal and back view images respectively, we use a linear blending. Let C^F and C^B are edge pixels in correspondence from the frontal and back view images respectively. Then for any pixel p^F and p^B in the linear neighborhood of length L_ε defined to be perpendicular to the local boundary at C^F and C^B respectively, we compute its color using the following blending function:

$$F^{bld}(p^F) = \alpha_1^F F(p^F) + (1.0 - \alpha_1^F) B(p^B)$$

$$B^{bld}(p^B) = \alpha_1^B F(p^F) + (1.0 - \alpha_1^B) B(p^B)$$

where $\alpha_1^F = 0.5(1.0 + d(C^F, p^F)/L_\varepsilon)$, $\alpha_1^B = 0.5(1.0 + d(C^B, p^B)/L_\varepsilon)$, $F(.)$ and $B(.)$ denote the original color of the original p^F and p^B while $F^{bld}(.)$ and $B^{bld}(.)$ denote the blended color for the pixels under consideration.

Figure 15 shows the texture mapping results before and after texture blending.

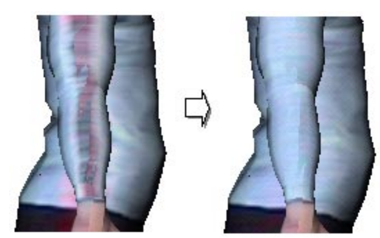


Figure 15: *The effect of texture blending*

4. Connection between body and face and results

We processed face cloning and body cloning separately. Even though the size of the face is much smaller, we have

Figure 16: *Final H-Anim 1.1 bodies with detailed individualized faces connected to the bodies. They are modified from one generic model*

to keep a detailed structure and high resolution for a face since we often zoom in the face to see facial animation and communication. So we use separated images for face cloning and body cloning and these two cloning methods use different texture mapping schemes. The body texture comes from the front and back view while the face texture comes from the front and side view due to the sphere like shape, which needs a side view for proper texture for ear parts. When we have a face and a body reconstructed separately, we have to connect them properly to make a smooth envelope for a perfectly smoothed body. We check the face size and location of the face on the front and side view body images using feature points as shown in Figure 7. Then simple translation and scaling locate the individualized face on the individualized body. We use an automatic sticking between them by finding the nearest points on the neck and the head to ensure the connection.

Figure 17: *Animation in a virtual environment*

The final bodies are shown in Figure 16* with the input images in Figure 14. Since the generic body had H-Anim 1.1 structure, the individualized bodies keep the same structure in VRML H-Anim 1.1 format, which means we can animate the bodies immediately. The final bodies are exported into VRML format and can be loaded in public web browsers. Figure 17 shows an animation example in a virtual environment[4]. Since we are using seamless skin structure for a real-time animation, the skin and textures are smoothly connected during animation.

5. Conclusion

In this paper, we introduce a model-based approach to photo-realistic animatable virtual human cloning from several pictures. The method takes as input two photos of the face of the subject and three photos of her/his body. Two different cloning methods are employed to face and body respectively. The efficient and robust face cloning method shows the processes of modifying a generic head for shape acquisition and producing texture images by combining orthogonal image pair smoothly. An H-Anim 1.1 compliant generic body is taken to serve as our reference model for a body. Unlike other existing systems, which require special environment to obtain input data, we seek the solution through a friendly user interface for an accurate localization of feature points, which are used both for automatic edge extraction and for modifying the generic body into individualized ones. A simple but effective heuristics-based boundary extraction algorithm is proposed to automatically extract the body silhouette from the images. Then to avoid the possible overlapping for skin parts, we introduce a two-step modification, first rough matching just with feature points information and then fine matching with detected edge information. The body texture mapping is processed using two images. Moreover, we connect the individualized head to the individualized body to form a complete animatable human model. As a result, we are able to animate the cloned human models in virtual environments. This method shows better cloning of the whole body in terms of both reconstruction and animation. The robustness and practical usefulness of the face reconstruction method is proved on several public demonstrations such as ORBIT'98 in Basel (CH), CEBIT'99 in Hannover (DE), SMAU'99 in Milano (IT), and TELECOM'99 in Geneva (CH). In these events, hundreds of people (Asian/Caucasian/Africa, female/male, young/old) were cloned and animated in a virtual world in around 5 minutes. The whole body reconstruction also takes similar time.

The automatic reconstruction of 3D clothes from the same photo input as we use here is the ongoing research topic.

Acknowledgment

The authors would like to thank Laurent Moccozet, Prithweesh De, and Christian Babski for their help. We are grateful to people taken photos including eRENA partners. This project is supported by Swiss National Research Foundation and partially supported by Hong Kong University of Science and Technology.

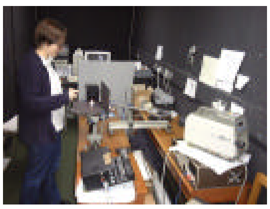
References

1. S. Addleman. Whole-body 3d scanner and scan data report. In *Three Dimensional Image Capture*, pages 2–5. 1997.
2. T. Akimoto, Y. Suenaga, and R. S. Wallace. Automatic creation of 3d facial models. In *IEEE Computer Graphics And Applications*, 1993.
3. C. Babski and D. Thalmann. A seamless shape for hanim compliant bodies. In *Proc. VRML 99,ACM Press*, pages 21–28, 1999.
4. C. Babski and D. Thalmann. Realtime animation and motion capture in web human director(whd). In *Proc. Web3D And VRML2000 Symposium*, 2000.
5. D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall InC, 1982.
6. P. J. Burt and E. H. Andelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.
7. H. Daanen, S. E. Talory, M. A. Brunsman, and J. H. Nurre. Absolute accuracy of the cyberware wb4 whole body scanner. In *Three Dimensional Image Capture*, pages 6–12. 1997.
8. T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. In *Proc. SIGGRAPH 1998*, pages 85–94, 1998.
9. P. Fua. Face models from uncalibrated video sequences. In *Captech98 (Modelling and Motion Capture Techniques for Virtual Environments)*, pages 215–228, 1998.
10. P. Fua. Human modeling from video sequence. *Geomatics Info Magazine*, 13(7):63–65, 1999.
11. J. Gu, T. Chang, S. Gopalsamy, and H. Shen. A 3d reconstruction system for human body modeling. In *Captech98 (Modelling and Motion Capture Techniques for Virtual Environments)*, pages 229–241, 1998.
12. A. Hilton, D. Beresford, T. Gentils, R. Smith, and W. Sun. Virtual people: Capturing human models to populate virtual worlds. In *Proc. Computer Animation 1999*, pages 174–185, 1999.
13. http://www.cyberware.com.
14. http://www.H-Anim.org.
15. H. S. Ip and L. Yin. Constructing a 3d individual head model from two orthogonal views. *Visual Computer*, 12:254–266, 1996.
16. I. A. Kakadiaris and D. Metaxas. Human body acquisition from multiple views. In *Proceedings of the Fifth ICCV 1995*, pages 618–623, 1995.
17. T. Kurihara and K. Arai. A transformation method for modeling and animation of the human face from photographs. In *Proceeding of Computer Animation*, pages 45–58, 1991.
18. K. F. Lai and R. T. Chin. Deformable contours: Modeling and extraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11:1084–1090, 1995.
19. W.-S. Lee, P. Kalra, and N. Magenat Thalmann. Model based face reconstruction for animation. In *Proc. Multimedia Modeling (MMM)*, pages 323–338, 1997.
20. W.-S. Lee and N. Magnenat-Thalmann. Head modeling from pictures and morphing in 3d with image metamorphosis based on triangulation. In *Captech98 (Modelling and Motion Capture Techniques for Virtual Environments)*, pages 254–267, 1998.
21. W.-S. Lee and N. Magnenat-Thalmann. Fast head modeling for animation. *Image and Vision Computing*, 18(4):355–364, 2000.
22. Y. Lee, D. Terzopoulos, and K. Waters. Realistic modeling for facial animation. In *Proc. SIGGRAPH 1996*, pages 55–62, 1996.
23. L. Moccozet and N. Magnenat-Thalmann. Dirichlet free-form deformations and their application to hand simulation. In *Proc. Computer Animation*, pages 93–102, 1997.
24. M. Proesmans and L. Van Gool. Reading between the lines - a method for extracting dynamic 3d with texture. In *Proceedings of VRST*, pages 95–102, 1997.

Archaeological Visualisation
The need for realism

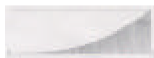
- Computer Graphics allow virtual environments to be “constructed” on a computer in a straightforward manner
- Computer reconstructions can be easily misleading
- Realism is *essential* if we are to provide an insight into how these sites may have appeared

Realistic flame



oil

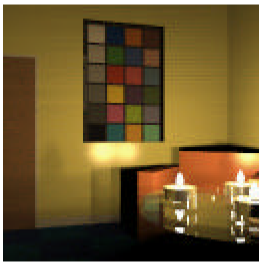
oil+water



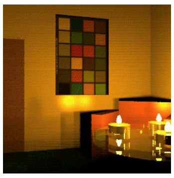
oil+salt

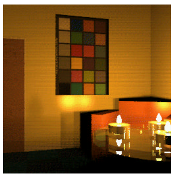
Spectral readings of different fuel types

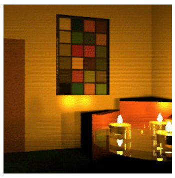
Different fuel types


Modern lighting

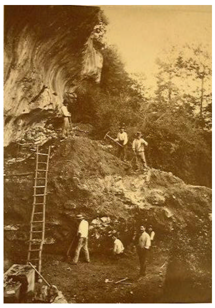

Tallow candle


Olive oil

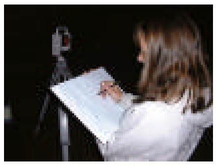

With salt


With water

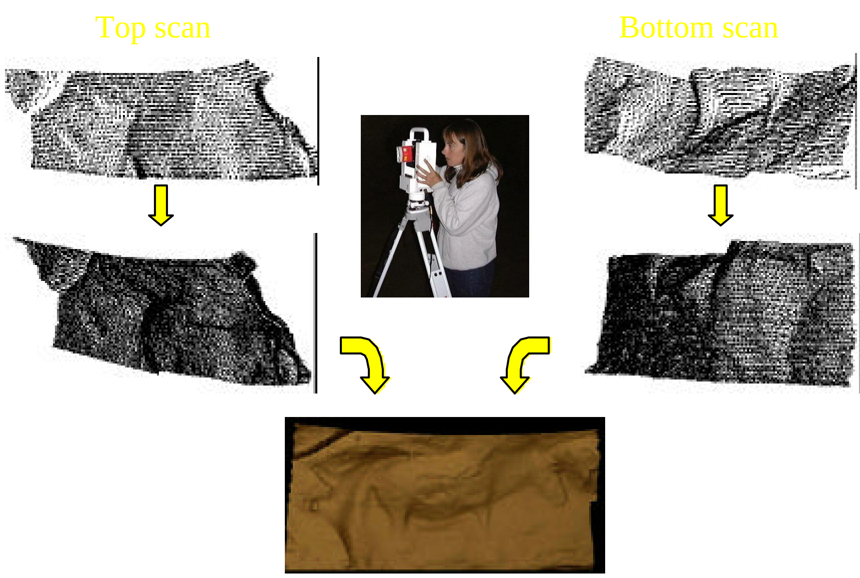
Cap Blanc



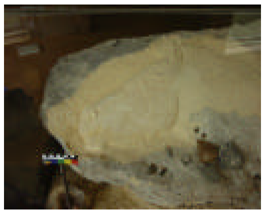
Capturing the data at Cap Blanc

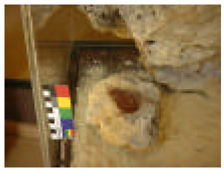


From depth map to 3D model



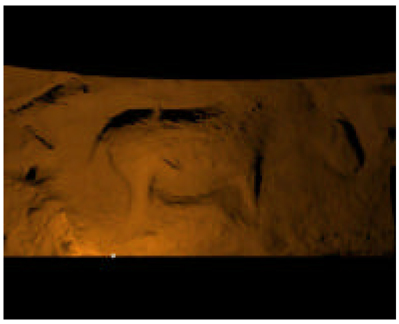
Reconstructing Cap Blanc





Realistic Lighting

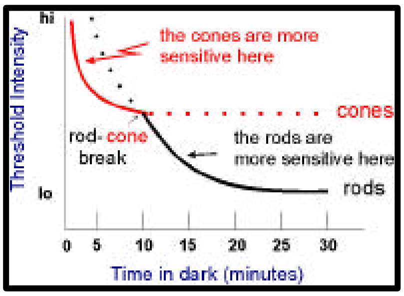
Modern lighting

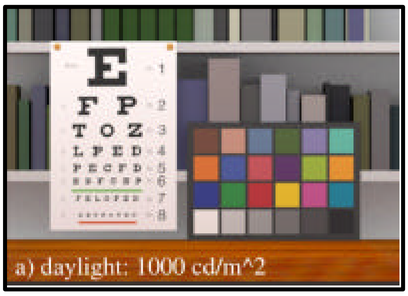


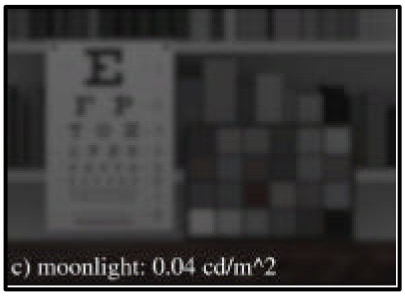
Animal tallow candle

Visual Adaptation

Ferwerda et al, 1996







Heritage sites offer significant challenges to computer graphics

- Media participation
  - accurate flame and smoke
- The need for realism
  - laser scanning, psychophysics
- Complexity of environments
  - parallel processing, visual perception
- Multi-disciplinary nature
  - archaeology, psychology, engineering, art history
- Multi-sensory
  - acoustic rendering

Autonomous Virtual Humans and Crowds for Cultural Heritage Virtual Environments

Daniel Thalmann

Why autonomous Virtual Humans and Crowds in Cultural Heritage Virtual Environments?

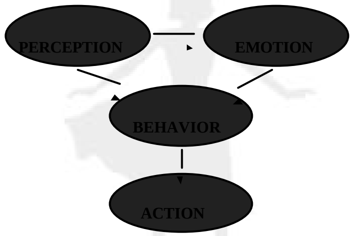
- **Because old cities were inhabited**
- **Because nobody wants to animate the details of animation of the inhabitants**
- **Problems to solve: individual behaviors, collective behaviors, interaction with objects**

Autonomous Virtual Humans

- **Autonomous:**

- **Have own goals, rules**
- **react to environment**
- **make decisions based on perception systems, memory and reasoning.**
- **communicates with other virtual humans and real humans**

Our starting structure



behavioral loop (on time)

initialize animation environment
while (not terminated) {
  update scene
  for each actor
    realize perception of environment
    select actions based on sensorial
    input, actual state, specific behavior
  for each actor
    execute selected actions }

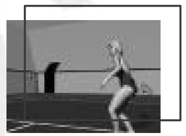
Action

Actions may be at several degrees of complexity.

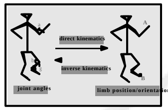
- **actor may simply evolve in environment or may interact with environment or even communicate with other actors.**

  e.g.: navigation, ball games.

But, action is based on motion



Motion Control



Kinematics

Motion capture

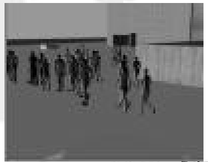
Dynamics

Walking

Grasping

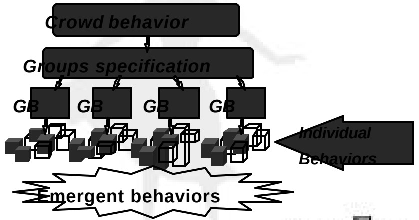
Human Crowd Simulation

- Goal: Simulate in a realistic way the human crowd motion in specific environments.



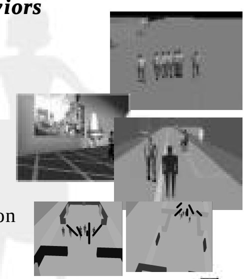
Human Crowd Simulation

- Crowd behavior based on group behaviors



Group Behaviors

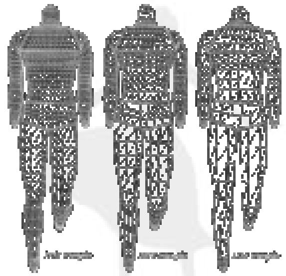
- Seek goal
- Flocking motion
- Collision avoidance
- Formation to a goal
- Following leaders
- Dispersion / Agregation



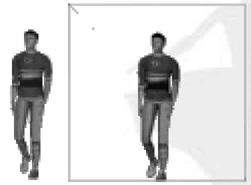
Acceleration Techniques for rendering

- **Culling (visibility, occlusion)**
- **Levels of detail (geometry, animation)**
- **Image-based rendering (HW layers, impostors)**

LOD: Human model



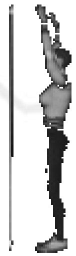
Impostors
What are they?



- Textured plane(s) that rotate(s) to face the camera
- Texture is preferably dynamically generated
- Can be recursive (e.g. hands)

Impostors
Texture Generation

- Takes place in a hidden buffer
- Cashes in on hardware acceleration
- Uses an orthographic camera whose viewing frustum is set once for all



The Multiplane Impostor

- Virtual human divided into 17 parts: waist, torso, neck, head, upper arms, lower arms, upper legs, lower legs, hands and feet.
- Each body part associated with a quadrilateral, which is placed in the scene so that it faces the viewer.
- Textures for each body part are generated dynamically in an off-screen buffer and copied to texture memory.

Example

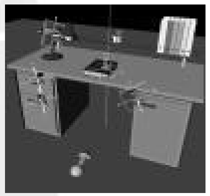
Geometric Object Model

\+

Modelling of its Interaction Features

=

Smart Object



Smart Objects

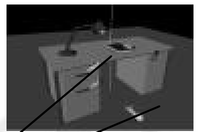
```
STATE VARIABLES
open1 false
open2 false
...
END
```

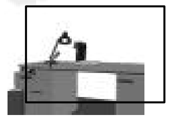
State Variables +

```
BEHAVIOUR open_drawer_1
  UserGoto ref_position
  CheckVar open1 false
  UserDoGest gest1 LeftHand
  SetVar open1 true
  DoCmd open1
END
BEHAVIOUR open_drawer_2
...
```

\+ Geometric Parameters

\+ Scripted Plans (Behaviours)





Simulation in the VE

Soraia Raupp Musse
University do Vale dos rio dos Sinos
Brazil

Daniel Thalmann
EPFL, Switzerland

1 CROWD MODEL

1.1 Crowd Structure

We defined a crowd as a set of groups composed of virtual agents. Our model distributes the crowd behaviors to the groups (GB) and then to the individuals. Further details about this distribution are presented in section 1.3.

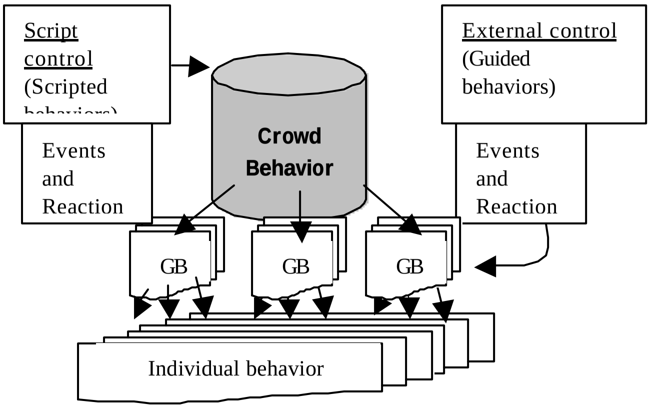


Figure 1: **Hierarchical structure of the model.**

As shown in Figure 1, there are two ways of setting the parameters of our model: scripted and external control. Scripted control defines *scripted behaviors* of the crowd whereas external control specifies *guided behaviors*.

As mentioned before, our crowd model is represented through a hierarchical architecture where the minor entity to be treated consists of groups. 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 ViCrowd, defined by the user or can emerge from the sociological rules.

Concerning the crowd control features, ViCrowd aims at providing autonomous, guided and programmed crowds (Table 2). 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 [18].

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 (see Section 5). 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 (Figure 1). We call the groups of virtual agents who apply the behavioral rules <***autonomous groups***>. Considering the levels of autonomy presented in this work, Figure 2 shows the priority criteria.

High ↕ *Low*

External Control	Reaction
	Group Behavior
Scripted Control	Reaction
	Group Behavior
Innate Behavior	

Figure 2: **The behavior priority.**

These priority criteria aim at solving the problems that occur when different types of control are applied at the same time to same characters implying the same nature of tasks. For instance, when the external controller sends an order to go to the restaurant, it can not turn off the collision avoidance, or change the way the trajectories are computed (innate behavior). On the other hand, external controller could explicitly change the group behavior "collision avoidance" to be applied to turn ON or OFF interactively with more priority than the innate behavior.

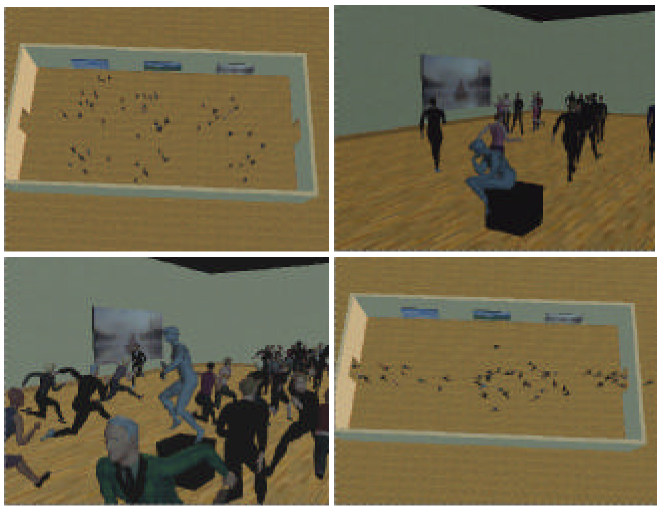


Figure 3: **Scenes of simulation of evacuation due to a panic situation. Up-left and up right: before the panic situation, the crowd walks. Down-left and down right: crowd reacts because an event generated when the statue becomes alive.**

Another example of multiple controls is: if a group's intention is to visit a museum (scripted group behavior), but a panic situation occurs (event), this group can then perform the programmed reaction associated with the event. This reaction can either be externally specified (during the simulation) or pre-programmed in the script, e.g. exit the environment. Figure 3 shows some images of a panic situation simulation, where 100 agents react by exiting the museum because a statue has become alive. Further details about the reactive behaviors are given in Section 5.

1.2 Crowd Information

We deal with three categories of information in order to characterize the crowds: knowledge, beliefs and intentions. Knowledge represents the information of the virtual environment, for example: <the real position of a chair>. Beliefs describe the internal status of groups and individuals, for instance: <group 0 is happy>. Finally, intention represents the goals of the crowd and groups of agents, e.g. <group 1 goes to the bank>. Table 3 describes the information existent in each one of three levels of entity in our model: (crowd, groups and individuals).

CROWD		GROUPS			INDIVIDUALS	
Knowledge	*Beliefs*	*Knowledge*	*Beliefs*	*Intentions*	*Beliefs*	*Intentions*
Obstacles to avoid	Crowd parameters	Group memory	Group parameters	Goals and actions to be applied	Relationship with other groups	Follow group or change of groups
Interest points of the scene		Group Perception			Status of domination	Be leader
Actions points						

Table 3: **Categories of information distributed among the entities of crowd and dynamically changed during the simulation**

The next sections present further details about crowd information.

1.2.1 Knowledge

The crowd knowledge represents the information about the virtual environment. Examples of crowd knowledge are locations of the interest points of the scene and information about the action to be applied in some locations. Group knowledge concerns the memory of groups related to the past experiences as well as perception related to agents and groups.

1.2.1.1 Crowd Obstacles

The obstacles to be avoided by the crowd are defined in two ways. The first one relies on the declaration of all objects of the scene; the second one concerns the declaration of the areas where the crowd can walk. The information can also be mixed, declaring some regions where the crowd can walk with some obstacles to be avoided.

1.2.1.2 Crowd Motion and Action

In addition to avoiding obstacles, it is possible to define crowd motion and action. Crowd motion is described using goals that can be: interest points (IP – locations where the crowd must pass through) and action points (AP - locations where the crowd can if necessary go and on arrival must perform an action). These points thus define the crowd paths [18]. Basically, the path followed by the crowd is specified using a set of IPs and APs that are associated with the groups of agents. Figures 4 and 5 show IPs and APs respectively.

As the agents from the same group share the same list of AP/IP (group's goals), each time one group arrives in a goal, we computed one different Bézier curve for each individual between the current goal and next one. Then, these curves are stored for each individual only until the end of their application (during the simulation).

The paths for the different agents from the same group can be similar but are never the same because they cannot occupy the same sub-region, as showed in Figure 4.

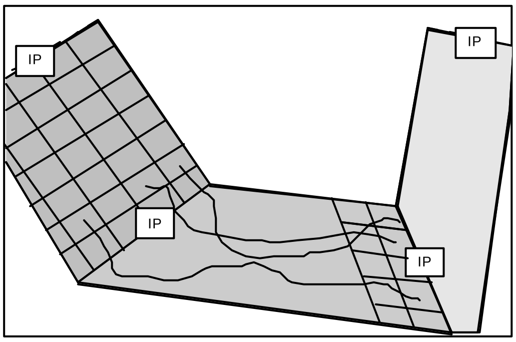


Figure 4: **Family of Bézier curves to define the group paths.**

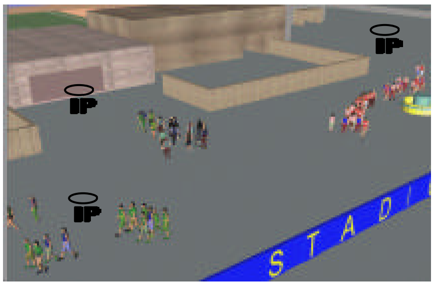


Figure 5: **Some IPs used to drive the crowd motion.**

Figure 6: **An AP where the action is applause.**

The APs parameters are checked to know if the action to be applied can be used by more than one agent at the same time or not. For instance, a counter (which means in our context, a flat surface in a bank or shop where individuals can be served) is considered as an individual AP whereas a piece of art in a Museum is considered as a shared AP. This classification is useful in order to coherently distribute the agents in AP regions.

1.2.1.3 Group Knowledge

The memory of groups is processed only by the leader of the group. In fact the memory is a structure where the leader's perceived information can be stored and processed afterwards depending on the specified behavioral rules. The size of the memory (capacity of storage) can be pre-defined for each group of crowd.

Group perception concerns the information about the location of groups/agents as well as some associated parameters: knowledge, beliefs and intentions. In this way, a group can perceive the internal status of another group, for instance. As with memory, the perception is associated to just the leader of group.

1.2.2 Beliefs

The crowd beliefs represent the list of behaviors to be applied by the groups as well as the emotion. The crowd beliefs are used to generate the group beliefs. These can be shared specifications or be redefined. The individual beliefs concern internal variables used by the sociological model in order to specify crowd effects: For instance, <relationship with others groups> describes a value for relationships with all the groups of the crowd which can influence the agents' intention to change groups or not (more details about agents' ability to change groups in **Goal Changing** behavior). The parameter <status of domination> represents the individual intention to be a leader or not. These two parameters are only used if the simulation has to apply sociological effects.

1.2.2.1 Crowd and Group Behaviors

High-level behaviors are specified in order to characterize crowds. These behaviors can be programmed in the script language or directly informed using guided control (Fig. 1). The list above presents the eight group behaviors actually existent in ViCrowd.

1. Flocking: Group ability to walk together in a structured group movement where agents from the same group walk at the same speed towards the same goals [18]. This behavior is responsible for flocking formation presented in some group motions in the real world, e.g. flock of birds. In our case, we defined four rules to model the flocking formation.

1. the agents from the same group share the same list of goals;
2. they walk at similar speeds;
3. they follow the paths generated as showed in section 3.2.1;
4. one agent can wait for another on arrival at a goal when another agent from the same group is missing.

Consequently, the agents from the same group walk together. We considered it as an important characteristic of our model, because in real life people walk in groups. To decide whether one agent must wait or not for another (rule 4), it is necessary to evaluate if all the agents from the same group arrived on a specific goal. If not, the agents who have arrived must wait.

2. Following: Group ability to follow a group or an individual motion. In this case we have defined the assumption of group goals which can be permanent or temporary. Let be *Group A*, a group which follows *Group B*. If the following motion is permanent, *Group A* adopts the goals information of *Group B* until the end of simulation. If this behavior is temporary, *Group A* shares the list of goals of *Group B* at some periods of the simulation but in a randomly defined manner.

3. Goal Changing: Agents can have the intention to change groups, consequently assuming the goals of its new group. It can occur only when the sociological effects are applied [17]. Basically, individuals have a more complex structure of parameters including: i) a value for the relationship with all groups (value between 0 and 1) and ii) a value for its domination status, which describes how much the considered agent is able to dominate the others (leadership ability). If the relationship with other groups is better than the current group, individuals can change groups. Also, if the individual presents a high value for leadership ability, he/she can become the new leader of the group, which can change the group behavior too.

4. Attraction: Groups of agents are attracted around an attraction point. Using a graphical interface, the user draws bi-dimensional regions or selects specific positions where the crowd should be positioned at a specific time. In association with this command, a <look_at> behavior can be added in order to determine the required orientation for each agent. Figure 7 shows the Open Inventor interface through which the regions are specified where the crowd should be located, as well as the attraction point defined by the <look_at> command.

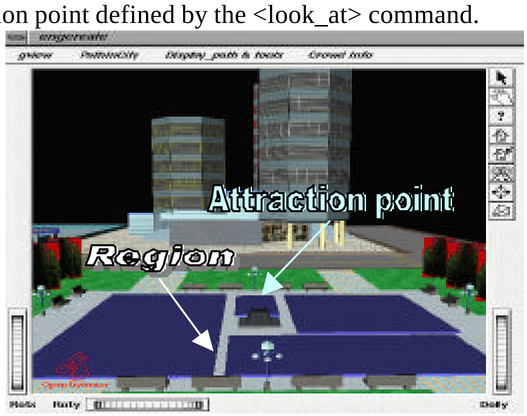


Fig. 7: **Blue boxes representing the regions where the crowd should be located in and the attraction point is the place where the crowd must look at.**

Fig. 8: **The crowd positioned inside the regions looking at the attraction point.**

To distribute the crowd over bi-dimensional regions, we apply a simple spatial distribution method to define the sub-regions where the agents are placed (Fig. 8). In addition, this distribution considers the required crowd density to be simulated (high or low density) depending on the number of agents. Afterwards, the agents are able to walk to their goal avoiding collision with the others.

5. Repulsion: Group ability to be repulsed from a specific location or region. The opposite of attraction behavior, the repulsion behavior relies on the generation of crowd goals outside the area to be expulsed. Consequently, the agents stay outside the area to be avoided and take their next intentions (goals) representing avoid the repulsed area. At the end of repulsion behavior, the agents are again free to walk in all defined areas. Fig. 9 shows two images of a repulsion simulation.

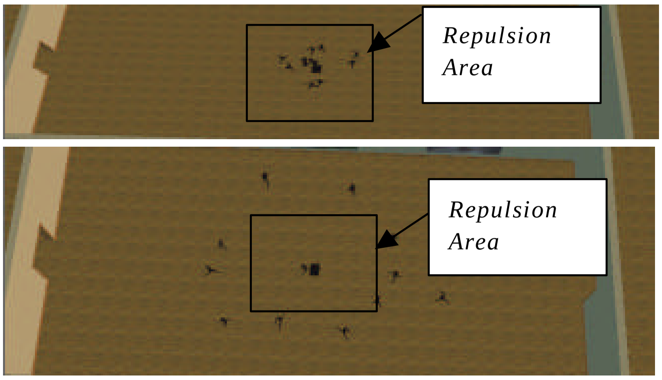


Fig. 9: **Agents are repulsed from a specific location**

6. Split: This behavior concerns the subdivision of a group to generate one or more groups (Figs. 10 and 11). This behavior concerns the randomly generation of intentions to create new groups. The number of agents to be transferred to the new group is random as well as the list of agents. The opposite idea of this behavior (addition of one or more groups) can be programmed using **following behavior.**

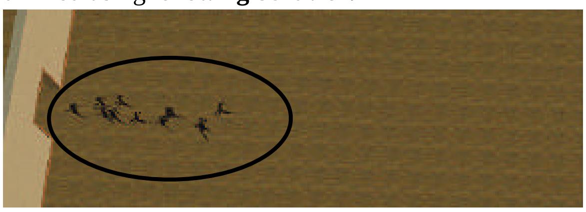


Fig. 10: **A group formed by 12 agents has the same intention and walk in the same direction.**

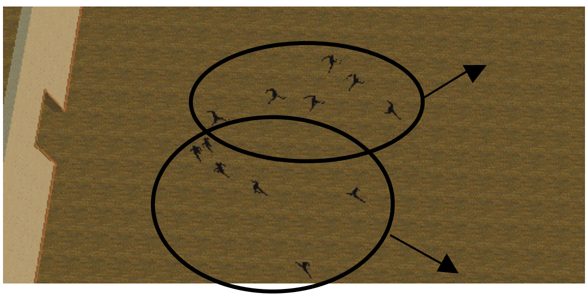


Fig. 11: **The group of Fig. 10 split into two different groups with different intentions.**

7. Space Adaptability: Group ability to occupy all the walking space. The computed trajectory between IPs and/or APs is represented by a Bézier curve. The information considered computing the curve is the region where the IP/AP's are located and divided in sub-regions using a simple spatial distribution method as a function of the required density of people. Adaptability behavior considers the full region to distribute the trajectories. If the agents do not adapt themselves to occupy all the space (without adaptability behavior), the region distributed is smaller and localized close to the goals' location. For example in Figure 12, there are two simulations; the first one represents a group with adaptability behavior and the second one, without it.

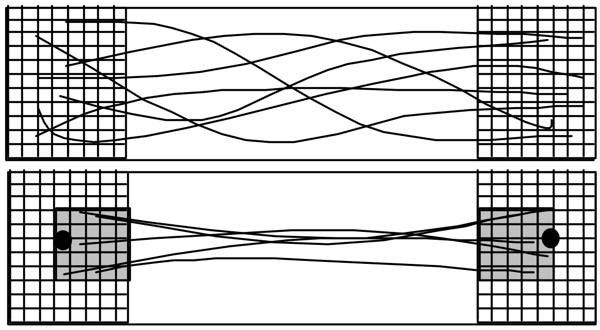


Fig. 12: **(up) All the space information is used to distribute the Bézier curves. 11) (down) A smaller part around the goals' location (black circles) is used in order to determine the trajectories.**

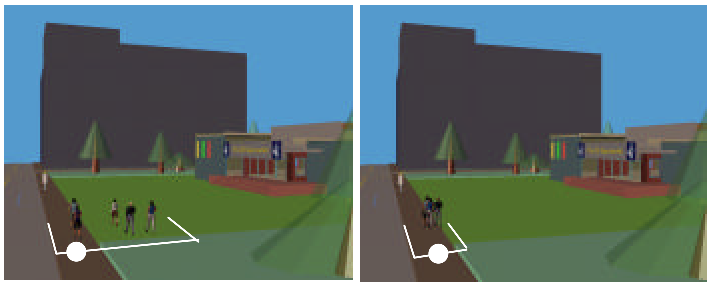


Fig. 13: **(left) Agents walking in all associated region. (right) agents walking in sub-region close to the goal (white circles)**

8. Safe-Wandering: We have used a procedural method in order to evaluate and avoid collision contacts with agents and objects. Our approach is based on the directions changes. The agents can predict the collision event (knowing the position of next virtual humans or obstacles) through simple geometric computing (intersection of two lines). It can therefore avoid the collision by changing its directions through its angular velocity changes.

After a specific period of time, the virtual human returns to its last angular velocity (which was stored in the data structures), and returns to its previous direction.

Fig. 14 shows an image of the collision detection method where agents avoiding collision with other agents and obstacles

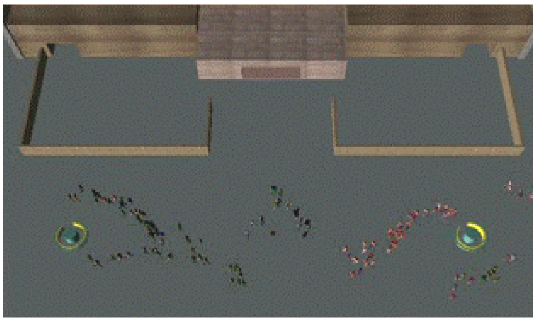


Fig. 14: **Groups avoiding collision with obstacles.**

1.2.2.2 Emotional Status

The emotion property represents the subjective climate to be simulated, e.g., "sad", "calm", "regular", "happy", or "explosive" (here "explosive" means an emotional status happier than happy). These are pre-defined emotional parameters, which can be changed by the user in order to work with other list of names. Depending on this overall definition, the groups are created according to the following parameters: *way of walking, walking speed and range of basic actions*. The correspondence between the names recently cited (sad, calm, etc) and the parameters (e.g., way of walking) is made through a normalized value [0;1].

Using the ViCrowd script language (see Section 4), one can specify how the emotions should be distributed among groups, as shown in Listing 1. When events and reactions are specified, the emotional status can also be used to define a condition (section 5) to trigger an event.

```
CROWD_EMOTION
 PARTY            PERC 80 EXPLOSIVE
```

Listing 1: **Script language to specify crowd emotion.**

Listing 1 shows a crowd of which 80 percent of the people have the emotional status <explosive>. The other 20 percent will be generated in a random way. Yet, the emotional status for specific groups can also be redefined. For example, it is possible to simulate a <happy> crowd with one or several <sad> groups. In the same way, the emotion can be changed as a function of triggered events and reactions. Figures 15 and 16 show some postures and ways of walking [4][5] taken on by the crowd according to their emotion.

Fig. 15: **Different postures as a consequence of various emotions.**

Fig. 16: **Different ways of walking as a result of various emotions.**

1.2.2.3 Individual Beliefs

As mentioned before, we consider the individual agents more simple than the groups of agents. While the groups contain goals, emotions, beliefs, knowledge and intentions, the individuals are just able to walk whereas avoid collision with obstacles and other agents. However, when the sociological effects are applied, it's possible for the individuals to have a more complex structure of parameters including: i) *goal-changing behavior*, group behavior (see Section 1.2.2.1); ii) *a value for the relationship with all groups* (value between 0 and 1) and iii) *a value for its domination status* describing how much the considered agent is able to dominate the others (leadership ability).

1.2.3 Intentions

The crowd does not have intentions (see Table 3), unless the group intentions are not specified. In this case, the crowd knowledge is used to generate crowd intentions, which afterwards are used to generate group intentions in a random way. For example, the crowd knowledge can define some interest points (IPs) and action points (APs). If the group intention is not defined, ViCrowd computes for each group a list of IPs and APs to be followed. The individual intention determines if the agent will follow its group's specification or change groups, for example exiting $Group_i$ and joining $Group_j$. The other individual intention is dependent of the <domination value> (individual belief) specified for each agent. If this value (between 0 and 1) is the highest of its group, this agent can became the new leader.

1.2.4 Inter-Relationship between the Various Categories of Information

The three types of information distributed in the multi-layered architecture can present some inter-dependence. The *knowledge* represents the information coming from the virtual environment and this can be used together with *beliefs* in order to apply different *intentions*. For example if a group has the *intention* to <go to the bank> and the agents from this group have the *belief* to <follow the group>, they are able to go to the bank if the crowd *knowledge* contains the information about <where is the bank>. Figure 17 shows the general graph of internal dependence between the various levels of information. The knowledge cannot be changed except if it is made through an external control during the simulation. The beliefs and intentions are dependent on one another, and also dependent on knowledge.

REFERENCES

[1] Arkin, R. A., "Integrating behavioral, perceptual and world knowledge in reactive navigation". In P. Maes (ed.), Designing Autonomous Agents, pp. 105-122, Cambridge, MA: MIT Press, 1990.

[2] Arbib, M. A. & Lee, H. B., "Anural visuomotor coordination for detour behaviour: from retina to motor schemas". In J.-A. Mayer, H. L. Roitblat and S. W. Wilson (ed.), From Animals to Animats II, Cambridge, MA: MIT Press, 1993.

[3] Blumberg, B.; Galyean, T. "Multi-Level Direction of Autonomous Creatures for Real-Time Virtual Environments". SIGGRAPH - Computer Graphics Proceedings, pp 47-54. Los Angeles, 1995.

[4] Boulic, R; Capin, T.; Huang, Z.; Kalra, P.; Lintermann, B.; Magnenat-Thalmann, N.; Moccozet, L.; Molet, T.; Pandzic, I.; Saar, K.; Schmitt, A.; Shen, J. and Thalmann, D. "The HUMANOID Environment for interactive Animation of Multiple Deformable Human Characters". Proceedings of EUROGRAPHICS'95, p.337-348 (Maastricht, The Netherlands, August 28 september, 1995).

[5] Boulic, R.; Huang, Z.; Thalmann. D. "Goal Oriented Design and Correction of Articulated Figure Motion with the TRACK System". Journal of Computer and Graphics, v.18, n.4, pp. 443-452, Pergamon Press, October 1994.

[6] Bouvier, E.; Cohen E.; and Najman. L. "From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation". Journal of Electronic Imaging 6(1), 94-107 (January 1997).

[7] Brogan, D. and Hodgins, J. "Group Behaviours for Systems with Significant Dynamics". Autonomous Robots, 4, 137-153. 1997.

[8] Brogan, D.C., Metoyer, R.A. and Hodgins, J.K. "Dynamically simulated characters in virtual environments". In IEEE COmputer Graphics and Applications. Vol.18, No5, pp 58-69. Sept. 1998.

[9] DIVE - *http://www.sics.se/dive/*

[10] Drogou, L.A and Ferber, J. "Multi-agent simulation as a tool for studying emergent processes in societies". Gilbert, N and Doran, J. editors, In Proceedings of Simulating Societies: the computer simulation of social phenomena, North-Holland, 1994.

[11] eRENA - *http://www.nada.kth.se/erena/index.html*

[12] Farenc, N., Musse, S.R, Schweiss, E., Kallmann, M., Aune, O., Boulic, R and Thalmann, D. "A Paradigm for Controlling Virtual Humans in Urban Environment Simulations". Special Issue on Intelligent Virtual Environments, 1999.

[13] Giroux, S. "Open Reflective Agents". M.; Wooldridge, J.P. Muller and M.; Tambe, editors, Intelligent Agents vol. II, Agent Theories, Architectures, and Languages, pp. 315-330. Springer-verlag, lnai (1037) edition, 1996.

[14] Kallmann, M. and Thalmann, D "Modeling Objects for Interaction Tasks". Proc. Eurographics Workshop on Animation and Simulation, 1998

[15] Mataric, M. J. "Learning to Behave Socially, in D. Cliff, P. Husbands, J.-A. Meyer & S.Wilson, eds, From Animals to Animats: International Conference on Simulation of Adaptive Behavior, pp.453-462. 1994.

[16] Meyer, J.A.; Guillot, A. "From SAB90 to SAB94: Four Years of Animat Research". In: Proceedings of Third International Conference on Simulation of Adaptive Behavior. Brighton, England, 1994.

[17] Musse, S.R. and Thalmann, D. "A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis". Proc. Workshop of Computer Animation and Simulation of Eurographics'97, Sept, 1997. Budapest, Hungary.

[18] Musse, S.R., Babski, C., Capin, T. and Thalmann, D. "Crowd Modelling in Collaborative Virtual Environments". ACM VRST /98, Taiwan

[19] Noser, H., Thalmann, D. "The Animation of Autonomous Actors Based on Production Rules". Proc. Computer Animation '96, 1996, Geneva, Switzerland.

[20] Parent, R. Notes on Computer Animation: "Computer Animation: Algorithms and Techniques". http://www.cis.ohiostate.edu/~parent/book/outline.html

[21] Perlin, K.; Goldberg, A. "Improv: A System for Scripting Interactive Actors I Virtual Worlds". SIGGRAPH – Computer Graphics Proceedings. Pp. 205-216. New Orleans, 1996.

[22] Reynolds, C. "Flocks, Herds and Schools: A Distributed Behavioral Model". Proc. SIGGRAPH '87, Computer Graphics, v.21, n.4, July, 1987.

[23] Reynolds, C. "Steering Behaviors for Autonomous Characters". Game Developers Conference, 1999.

[24] Schweiss, E., Musse, S.R.; Garat, F. "An Architecture to Guide Crowds based on rule-based systems". Autonomous Agents '99, Seattle, Washington, USA, 1999.

[25] Tu, X. and Terzopoulos, D. "Artificial Fishes: Physics, Locomotion, Perception, Behavior". Proc. SIGGRAPH '94, Computer Graphics, July 1994.

[26] Treiber, M., Hennecke, A. and Helbing, D. "Microscopic Simulation of Congested Traffic". In: Traffic and Granular Flow'99: Social Traffic, and Granular Dynamics (Springer, Berlim) 1999.

[27] AntZ. http://www.antz.com [28] Bugs Life. http://bugslife.com