From Motion Capture to Interactive Virtual Worlds

Towards Unconstrained Motion-Capture Algorithms for Real-time Performance-Driven Character Animation

A dissertation submitted towards the degree Doctor of Natural Sciences (Dr. rer. nat.) of the Faculty of Mathematics and Computer Science of Saarland University

Helge Rhodin

Saarbrücken 2016

UNIVERSITÄT DES SAARLANDES

Date of the colloquium:
15.12.2016

Dean:
Prof. Dr. Frank-Olaf Schreyer

Reporter:
Prof. Dr. Christian Theobalt
Prof. Dr. Hans-Peter Seidel
Prof. Dr. Christoph Bregler

Chairman of the Examination board:
Prof. Dr. Philipp Slusallek

Scientific Assistant:
Dr. Michael Zollhöfer

Abstract

This dissertation takes performance-driven character animation as a representative application and advances motion capture algorithms and animation methods to meet its high demands. Existing approaches have either coarse resolution and restricted capture volume, require expensive and complex multi-camera systems, or use intrusive suits and controllers.

For motion capture, set-up time is reduced using fewer cameras, accuracy is increased despite occlusions and general environments, initialization is automated, and free roaming is enabled by egocentric cameras. For animation, increased robustness enables the use of low-cost sensors input, custom control gesture definition is guided to support novice users, and animation expressiveness is increased. The important contributions are: 1) an analytic and differentiable visibility model for pose optimization under strong occlusions, 2) a volumetric contour model for automatic actor initialization in general scenes, 3) a method to annotate and augment image-pose databases automatically, 4) the utilization of unlabeled examples for character control, and 5) the generalization and disambiguation of cyclical gestures for faithful character animation. In summary, the whole process of human motion capture, processing, and application to animation is advanced. These advances on the state of the art have the potential to improve many interactive applications, within and outside virtual reality.

Kurzzusammenfassung

Diese Arbeit befasst sich mit Performance-driven Character Animation, insbesondere werden Motion Capture-Algorithmen entwickelt um den hohen Anforderungen dieser Beispielanwendung gerecht zu werden. Existierende Methoden haben entweder eine geringe Genauigkeit und einen eingeschränkten Aufnahmebereich oder benötigen teure Multi-Kamera-Systeme, oder benutzen störende Controller und spezielle Anzüge.

Für Motion Capture wird die Setup-Zeit verkürzt, die Genauigkeit für Verdeckungen und generelle Umgebungen erhöht, die Initialisierung automatisiert, und Bewegungseinschränkung verringert. Für Character Animation wird die Robustheit für ungenaue Sensoren erhöht, Hilfe für benutzerdefinierte Gestendefinition geboten, und die Ausdrucksstärke der Animation verbessert. Die wichtigsten Beiträge sind: 1) ein analytisches und differenzierbares Sichtbarkeitsmodell für Rekonstruktionen unter starken Verdeckungen, 2) ein volumetrisches Konturenmodell für automatische Körpermodellinitialisierung in genereller Umgebung, 3) eine Methode zur automatischen Annotation von Posen und Augmentation von Bildern in großen Datenbanken, 4) das Nutzen von Beispielbewegungen für Character Animation, und 5) die Generalisierung und Übertragung von zyklischen Gesten für genaue Charakteranimation. Es wird der gesamte Prozess erweitert, von Motion Capture bis hin zu Charakteranimation. Die Verbesserungen sind für viele interaktive Anwendungen geeignet, innerhalb und außerhalb von virtueller Realität.

Summary

Virtual and augmented reality applications call for non-intrusive human-computer interfaces that are cheap, easy to set up, and of high accuracy. Interfaces that are driven by human motion are promising; however, existing approaches have either coarse resolution and restricted capture volume, require expensive and complex multi-camera systems, or use intrusive suits and controllers. This forces commercial solutions to resort to physical controllers, such as the HTC Vive gear, that hamper motion and are limited to track hands instead of full-body motion.

This dissertation takes performance-driven character animation as a representative application and advances motion capture algorithms and animation methods to meet its high demands. For motion capture, the required number of (color) cameras is reduced, otherwise tedious initialization is eased, and free roaming is enabled by egocentric motion capture from body-worn cameras. For performance-driven character animation, methods are introduced that are designed to work with today's low-cost sensing technology, to broaden usability by guiding novice users during control definition, and to increase expressiveness of animation.

The most important technical contributions of this dissertation are: 1) an analytic and differentiable visibility model for local pose optimization under strong occlusions, 2) a volumetric contour model for automatic actor initialization in general scenes, 3) a method to annotate and augment image-pose databases automatically, 4) the utilization of unlabeled examples for character chontrol, and 5) the generalization and disambiguation of cyclical gestures for faithful character animation.

In summary, the whole process of human motion capture, processing, and application to animation is advanced to enable new levels of performance-driven interaction in virtual worlds. Beyond this specific goal, the attained improvements have the potential to enhance many interactive applications outside virtual reality.

Zusammenfassung

Anwendungen in virtueller und erweiterter Realität erfordern kostengünstige, nichtintrusive Schnittstellen zwischen Mensch und Computer, welche einfach handhabbar sind und eine hohe Genauigkeit haben. Schnittstellen, welche auf der Erfassung von menschlicher Körperbewegung basieren sind vielversprechend; bestehende Ansätze haben jedoch entweder eine grobe Auflösung und eingeschränkten Aufnahmebereich oder benötigen teure Multi-Kamera-Systeme, oder benutzen störende Controller und spezielle Anzüge. Dies zwingt kommerzielle Head-Mounted-Display-Systeme dazu physikalische Controller zu nutzen, welche jedoch die Bewegung behindern und nur die Handposition anstatt Vollkörperbewegung erfassen.

Diese Arbeit befasst sich mit Performance-driven Character Animation, insbesondere werden Motion Capture-Algorithmen entwickelt um den hohen Anforderungen dieser Beispielanwendung gerecht zu werden. Für Motion Capture wird die erforderliche Anzahl von (Farb-) Kameras reduziert, mühsame Initialisierung vereinfacht, und uneingeschränkte Bewegung durch tragbare Helmkameras ermöglicht. Für Performance-driven Charakter Animation werden Verfahren konzipiert, welche robust sind und mit kostengünstigen Sensoren funktionieren, benutzerdefinierte Gestendefinition für unerfahrene Nutzer assistieren, und die Ausdrucksstärke der Animation verbessern.

Die wichtigsten Beiträge dieser Arbeit sind: 1) ein analytisches und differenzierbares Sichtbarkeitsmodell für Rekonstruktionen unter starken Verdeckungen, 2) ein volumetrisches Konturenmodell für automatische Körpermodellinitialisierung in genereller Umgebung, 3) eine Methode zur automatischen Annotation von Posen und Augmentation von Bildern in großen Datenbanken, 4) das Nutzen von Beispielbewegungen für Character Control, und 5) die Generalisierung und Übertragung von zyklischen Gesten für genaue Charakteranimation.

Zusammenfassend wird der gesamte Prozess der menschlichen Bewegungserfassung, Verarbeitung, und Animation verbessert um neue Arten der Interaktion zwischen Mensch und Computer in virtuellen Welten zu ermöglichen. Jenseits diesem spezifischen Ziel haben die erzielten Entwicklungen das Potenzial, viele interaktive Anwendungen außerhalb der virtuellen Realität zu verbessern.

Acknowledgements

First of all, I thank Christian Theobalt for the contineous strong stream of support, the farsighted guidance complemented with the freedom and encouragement to try out my own paths. I further thank my thesis committee, Philipp Slusallek for his instant and sustained commitment since the start of my scientific endevours, Hans-Peter Seidel for familiar discussions at various levels and for forming and providing such a unique research environment, and Chris Bregler for his timely review and inspiering research.

A further thank you goes to all former and current members of the Computer Graphics department and GVV group, you form an increadiably rich and friendly atmosphere. In particular I thank Kiran Varanasi, James Tompkin, Kwang In Kim, Edilson de Aguiar, Dan Casas and Christian Richardt for guiding and contributing with their postdoc expearience, leveling out my weaknesses and amplifying my strengths, and for your friendship. I thank Srinath Sridhar, Nadia Robertini, Mohammad Shafiei, Dushyant Mehta and Eldar Insafutdinov for their productive collaboration, Oliver Klehm and Tobias Ritschel for their valuable expert feedback, and Antti Oulasvirta, Jemin Hwangbo, Fabrizio Pece and Rhaleb Zayer for discussing alternative paths. Moreover, I would like to thank Nils Hasler and Carsten Stoll for the particularly fruitful and open discussions, Michal Richter, Pablo Garrido and Nadia Robertini for creating a good mix of home and office, and Petr Kellnhofer, Bernhard Reinert, Hyeongwoo Kim, Martin Sunkel and Peter Grosche for the good admin-teamwork.

I thank all actors who performed in evaluation sequences and Cynthia Collins, Gabi Kussani, Gottfried Mentor and Wolfram Kampffmeyer for their creative influence.

I thank all my friends for their valuable friendship, Peter and Ralf also for sharing their expeariences as a researcher and improving this dissertation.

I thank my Family, Gabi, Andreas, Birte and Ulrike for giving me support during all stages of life.

Thank you for making this dissertation possible.

Contents

Part I

From Motion Capture to Interactive Virtual Worlds

Introduction 1

Digital motion and interaction

Motion is intrinsic to our everyday lives: we move through the world, we shape the world by operating tools, and we communicate with gestures, body language, and facial expressions. However, in the past few decades, technology has radically changed the way we interact with people and the world. Video calls open instant communication between people spread across continents, 3D displays and virtual reality glasses enable the exploration of digital worlds from the living room, and new forms of interaction and creative entertainment are enabled. For instance, virtual universes are explored with imaginary avatars, impossible to experience in classical theater and role-playing.

Current state

This digitization process strives for the fusion of real and virtual worlds, e.g, with augmented reality devices that render virtual content in our living room. While display technology advances rapidly, only a fraction of human expression that has evolved over millennia is represented in today's technology. Video streams only transmit simplified projections of the real world, hiding information of body language and subtle facial expressions. Human-computer interaction technologies are still centered around physical devices such as 2D touch screens, the keyboard, and the mouse, neglecting the dexterity and information content contained in the full 3D body motion. Existing virtual worlds reach gigantic extents, and show realistic appearance but do not reflect human appearance and motion adequately.

Current limitations

A prevailing limitation of existing approaches is the ability to reliably record and represent human users within these digital systems. This requires algorithms and devices to sense and reconstruct human appearance and motion, to extract and process the relevant information, and to display the result adequately for the human visual system—all of this must happen in real time to enable interaction. If we consider the human representation as an information flow, from capturing a real performance, through acquisition and processing, to display, then current systems drop information at each step of this pipeline to meet computational constraints of today's hardware as well as financial limits of the average consumer.

Goal

This dissertation examines the whole process—from acquisition over reconstruction and animation to display—and makes advances at each step with new algorithms and representations. We pick performance-driven character animation as a representative application and advance existing acquisition methods and reconstruction algorithms to best meet its demands. Furthermore, performance-driven character animation methods are improved in itself, including the handling of remaining constraints imposed by existing reconstruction methods.

Human shape and motion capture

In the first part of this dissertation, limitations of exiting performance acquisition and reconstruction approaches are identified and addressed, in particular those which have their application in virtual world interaction. Existing motion-capture algorithms can reconstruct skeleton motion from multiple video recordings; however, these often require manual initialization steps, a fixed camera placement restricts the capture volume, and most algorithms are limited to professional studio setups as they commonly require 6–12 calibrated cameras and depend on indoor studio conditions. To overcome these limitations, we propose new methods, that reduce the number and complexity of required sensors, and lower setup time and manual interaction. Furthermore, acquisition methodologies are changed to support reconstruction in enormously large recoding volumes and cluttered scenes with occluding objects, and previously required indoor studio conditions are relaxed, such as requiring controlled background. These advances enable low-cost motion capture for the average consumer in more general environments without degrading accuracy, enabling interactions with virtual worlds from within the living room or office.

Technical contribution I

To this end, new algorithms and representations for human motion capture are developed. A volumetric body model is introduced that provides analytically differentiable energy functions for photo consistency, and improves reconstruction accuracy for a low number of cameras. The volumetric model is further generalized to contour-based reconstruction, without requiring background segmentation, and is used for fully automatic human shape and appearance estimation in general environments. Moreover, new egocentric camera equipment is developed which, together with new egocentric motion-capture algorithms, enables motion estimation in cluttered scenes with many occluders and close interaction with objects and nearby persons, as well as general scenes with virtually infinite capture volume. This egocentric performance capture perspective enables new user-centric applications, but requires solutions to challenges that newly open up. Machine learning algorithms are commonly used for motion capture; however, they are not directly applicable as they depend on large annotated example databases, which do not exist for this new domain. We propose a way to create such a database with low effort through automatic annotation and augmentation of real recordings.

Performance-driven character animation

In the second part of this dissertation, the processing of user input from today's available consumer sensors, such as the Microsoft Kinect, for character animation is analyzed and extended to improve interaction in virtual worlds. Algorithms are developed to transfer human motion to non-human characters. The advances enable the embodiment of virtual characters that are neither restricted to human topology nor any specific shape and motion, such as, a horse, a caterpillar, alien, or robot. Many open challenges are overcome, especially the handling of topology differences between human and character and controlling skeleton-free characters. A major difficulty for such transfer is the causality of live input and output. For live input the future is unknown, and for interaction the delay must be minimal, leaving only a small time window for processing. We overcome these challenges and provide faithful performance-driven non-human character animation.

Technical contribution II

The transfer of user to character motion is posed as the problem of finding a mathematical function between input and output representations, which are inde-

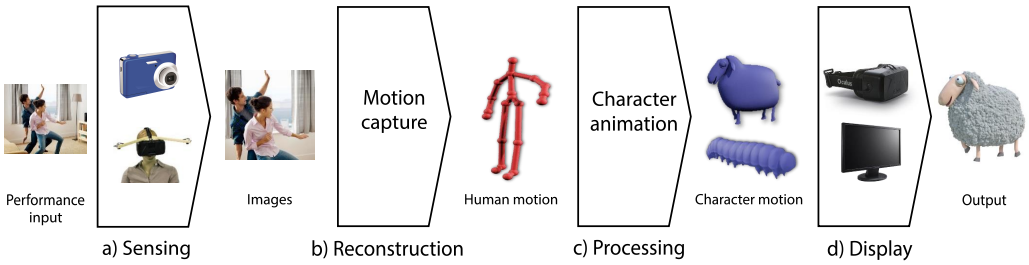


Figure 1.1: Process of human motion estimation for interactive worlds. a) Sensing of the human performance with cameras, b) motion and shape reconstruction, c) motion processing, and d) display. This dissertation focuses on algorithmic advances of steps a), b) and c).

pendent of input device, character type, and animation software. Different mapping functions known in the domain of machine learning are analyzed for this task, and a new mapping is designed that lifts the coarse motion input provided by today's acquisition methods to detailed and faithful character motion. A new facet is the utilization of unsupervised training data. The mapping is designed to allow the user to drive virtual characters with a dictionary of control motions, and to allow interactive control motion definition by performance, with support for novice users by automatic guidance towards suitable control gestures. Moreover, dynamics of the user's input motion are estimated and translated to character dynamics, which are specific and natural to the target character, for instance, for a horse speeding up locomotion should initiate a transition from trot to gallop with its characteristic motion style and dynamics. Furthermore, care is taken to disambiguate and separate simultaneously performed control motions.

Impact and perspective

Together, the contributions of this dissertation enable new levels of intuitive exploration and interaction in virtual worlds, and open the door for new gaming and entertainment forms. While we focus on the application to virtual reality, the developed advances have merit in diverse applications fields, such as, motion analysis of athletes, biomechanics, and robotics. For instance, we showcase a prototype that allows non-intrusive estimation of the full-body pose of a user wearing a head-mounted display (HMD) equipped with two tiny color cameras. These body-worn motion-capture sensing devices provide free roaming, and enable intuitive interaction in arbitrarily large worlds. Moreover, a prototype is demonstrated that allows control of a physical robot faithfully by intuitive gestures. It could be used by a handicapped person to control a service robot or electric wheelchair by simple hand gestures.

1.1. Overview

Outline

This dissertation addresses important open questions in the whole process from human motion, shape, and appearance acquisition to motion and pose processing. The individual processing steps are sketched in Figure 1.1. We consider four steps:

a) **Sensing**
The hardware design to record the human performance, e.g., a color camera.

b) **Reconstruction**
The reconstruction algorithm, e.g., to infer skeleton pose from an image.

c) **Processing**
Gesture separation, transfer, and animation, e.g., mapping human walk to horse gallop.

d) **Display**
Visualization of the outcome, e.g., 3D rendering of the character in an HMD.

The focus of this dissertation is on algorithmic advances of different components of the reconstruction and processing steps, and on new camera arrangements for sensing. Their exposition in the dissertation is ordered according to their position in the processing pipeline. Each component is extensively evaluated, particularly in the context of virtual worlds and character animation.

1.2. Structure

This dissertation is split into four parts. The main technical contributions are covered in Part II and Part III. Relations between chapters and related work are discussed at the beginning of each part.

Introduction and background

Part I gives an introduction and motivation for the dissertation topic, outlines the structure of exposition, and highlights the main contributions. Moreover, fundamental notations and representations that are used throughout the dissertation are introduced.

Motion and shape capture

Part II presents advances in human shape and motion estimation from as few as two video streams in general scenes. It covers three contributions: a novel volumetric scene model with differentiable visibility for pose estimation, automatic model initialization by spatio-temporal shape and pose estimation from volumetric contour cues, and estimation from an egocentric camera rig.

Processing for character animation

Part III presents two approaches for real-time performance-driven animation of non-human characters. The contributions are a frame-by-frame pose-based approach which introduces a mapping suitable for a large variety of (non-) human characters, and provides automatic guidance for control definition. As well as a motion-based approach that captures, disambiguates, transfers, and generalizes motion dynamics based on time-frequency analysis.

Conclusions

Part IV summarizes the core findings, restates the main results, and gives an outlook to future work.

1.3. Contributions

This section summarizes the main contributions of this dissertation to the state of the art in human motion estimation and character animation.

Volumetric model, smooth visibility

The main contributions of **Chapter 4** (published as Rhodin et al., 2015a) are:

- A 3D scene representation and image formation model that enables an analytic, continuous, and smooth visibility function that is differentiable everywhere in the scene.
- Similarity energies with rigorous visibility handling that are differentiable everywhere in the model parameters and efficient to evaluate.
- A human motion-capture algorithm that shows favorable and more robust convergence in cases where previous visibility approximations fail, such as disocclusions or multiple nearby occlusion boundaries.

Volumetric contours for initialization

The main contributions of **Chapter 5** (published as Rhodin et al., 2016b) are:

- A volumetric contour representation and 2D contour-based energy that measures contour alignment with image gradients on the raw RGB images. No explicit background segmentation is needed.
- A new data-driven body model that represents human surface variation, the space of skeleton dimensions, and the space of volumetric density distributions in a low-dimensional parametric space.
- A space-time optimization approach that fully automatically computes the shape and the 3D skeletal pose of the actor using both contour and ConvNet-based joint detection cues.

Egocentric tracking

The main contributions of **Chapter 6** (published as Rhodin et al., 2016a) are:

- A light-weight low-cost sensor rig of two head-mounted, downward-facing commodity video cameras with fisheye lenses.
- A new marker-less motion capture algorithm tailored to the strongly distorted egocentric fisheye views.
- A new semi-automatic approach for creating an extensive training dataset of real egocentric videos of general body poses for several people in different clothing. The preformed automation by marker-less performance capture and augmentation by intrinsic image decomposition generalized beyond egocentric databases.

Interactive pose mapping

The main contributions of **Chapter 8** (published as Rhodin et al., 2014) are:

- A real-time algorithm that can map between characters with different topology from sparse correspondences.
- A latent volume representation that efficiently exploits unlabeled data to allow robust performance-driven character animation.
- An automatic keyframe suggestion method to support the user during correspondence selection.

Wave gesture motion mapping

The main contributions of **Chapter 9** (published as Rhodin et al., 2015b) are:

- A live animation system, which couples wave gesture to parametric motion graphs and layers different input modalities.

- A technique to robustly and accurately estimate amplitude, frequency, and phase of simultaneous gestures in real time, generalized from a single user-defined reference motion.
- An interpolation method for motions with out-of-phase submotions that cannot be aligned by traditional time warping.

1.4. List of publications

This dissertation encompasses extended revisions of five scientific publications, peer reviewed and published at top-tier venues in graphics and vision. Two additional coauthored papers are only briefly discussed.

Three are in the field of human performance capture:

Pose estimation

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *ICCV*, 2015a

Shape estimation

Helge Rhodin, Nadia Robertini, Dan Casas, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. General Automatic Human Shape and Motion Capture Using Volumetric Contour Cues. In *ECCV*, 2016b

Egocentric capture

Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. Egocap: Egocentric marker-less motion capture with two fisheye cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 35(8), 2016a

Furthermore, two papers are in the field of performance-driven character animation:

Per-frame mapping

Helge Rhodin, James Tompkin, Kwang In Kim, Kiran Varanasi, Hans-Peter Seidel, and Christian Theobalt. Interactive motion mapping for real-time character control. *Computer Graphics Forum (Proceedings of Eurographics)*, 33(2), 2014

Motion mapping

Helge Rhodin, James Tompkin, Kwang In Kim, Edilson de Aguiar, Hanspeter Pfister, Hans-Peter Seidel, and Christian Theobalt. Generalizing wave gestures from sparse examples for real-time character control. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6), 2015b

The two additional papers are in the field of motion and performance capture:

Anisotropic Gaussian hand model

Srinath Sridhar, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, and Christian Theobalt. Real-time hand tracking using a sum of anisotropic Gaussians model. In *3DV*, 2014

General surface refinement

N. Robertini, D. Casas, H. Rhodin, H.-P. Seidel, and C. Theobalt. Model-based outdoor performance capture. In *3DV*, 2016

Technical background 2

Capture and animation

Marker-less motion capture and performance-driven character animation are closely related. The output of motion capture—the reconstructed (human) motion—is input to performance-driven character animation, and its (character) output is also a motion representation. Because of these connection and similarity, they build on common concepts, and many representations, data structures, and algorithms are shared.

Overview

In this chapter, we give a basic introductions to the representations commonly used throughout this dissertation to make the individual contributions of this dissertation accessible to inexperienced readers and we refer to the relevant literature for in-depth background information. Related work that is particular to individual parts of the pipeline is separately discussed within each chapter. The informed reader is invited to continue directly to Part II.

2.1. Skeleton representations

Anatomical abstraction

Following the physical anatomy of humans and vertebrate animals, virtual skeletons have been used to visualize and represent characters, see Figure 2.1 left. The skeleton consists of rigid bones connected by flexible joints, and follows a hierarchical tree structure. It is represented as a graph, where edges correspond to bones and nodes to joints. The advantage of skeletons is their sparse representation and high, physically-motivated, abstraction level. Storing only the skeleton joints and bones is a much lower-dimensional representation than modelling surface or volumetric detail. It allows for intuitive editing by artists and reduces the complexity for reconstruction tasks.

Skeleton representations

Representing the skeleton by the 3D position of each joint proved to be useful for efficient reconstruction [Shotton et al., 2011]; However, not constraining bone lengths and joint-angle limits allows for unrealistic configurations. Bone lengths can be forced to be constant by parameterizing the root node position, e.g. hip, explicitly and inferring the remaining joints by forward kinematics on joint angles and orientations. Rotational joints can be modeled as ball joints, represented by rotation matrices, unit quaternions [Sudderth et al., 2004], twists and the exponential map [Bregler and Malik, 1998], and Euler angles [Basu et al., 1996]. Some joints, such as the human knee, have only a single rotation axis and are well represented with a single angle that specifies the rotation around a fixed axis [Stoll et al., 2011] This further reduces the pose parameter space to be more realistic. Translational joints

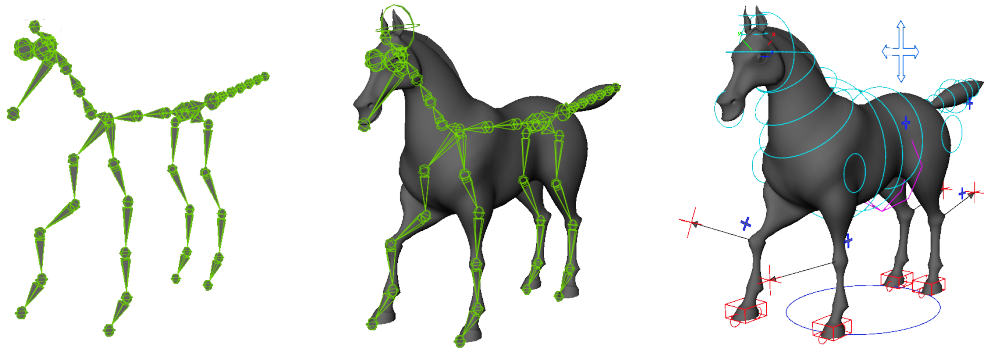


Figure 2.1: Characters are parametrized by lower dimensional representations to aid animation and reconstruction. Skeleton representations are anatomically motivated (left). Skinning drives surface deformation by an embedded skeleton (center). Rigs model complex characters with simple control handles, often using a combination of deformation techniques, such as inverse-kinematics handles and free-form deformations (right). *The rig is created by Hung Vodinh and Joel Anderson.*

are used to model the root position and to model flexible bone connections more accurately, such as the human shoulder.

Limited generalization

Characters and body parts which have no physical skeleton counterpart, such as facial deformations and deforming creatures, have been modeled with skeleton structures [James and Twigg, 2005]. However, skeleton structures fail in representing surfaces and volumes accurately.

To constrain the solution space for pose estimation in Chapters 4 to 6, we use a kinematic skeleton parametrized relatively by axis-angle joints. For performance-driven animation, we parametrize skeleton motion directly in terms of 3D joint positions, to attain the largest generality (Chapters 8 and 9).

2.2. Surface representations

Discrete approximation

Arbitrary surfaces are well approximated by mesh representations, [Botsch et al., 2007]. For instance, parametrized by 3D vertex positions and connecting edges, see Figure 2.2 left and center. The mesh graph defines faces, e.g. triangles, through sets of connected vertices, and the union of these flat faces forms a piecewise surface in 3D. Mesh representations are either reconstructed from real objects or manually shaped. Mesh creation is a tedious process; to model detailed geometry, realistic mesh characters require thousands of vertices and additional texture maps to define color appearance. In the following, we discuss lower-dimensional representations that have been proposed to ease editing and reconstruction.

2.2.1. Skinning

Skeleton-driven

Skinning is a common tool for character animation. It describes the relation between a surface mesh that is deformed according to the articulation of an underlying skeleton, see Figure 2.1 center. Linear blend skinning deforms each vertex of a *reference* mesh as a linear function of the skeleton bones. In a pre-process, the

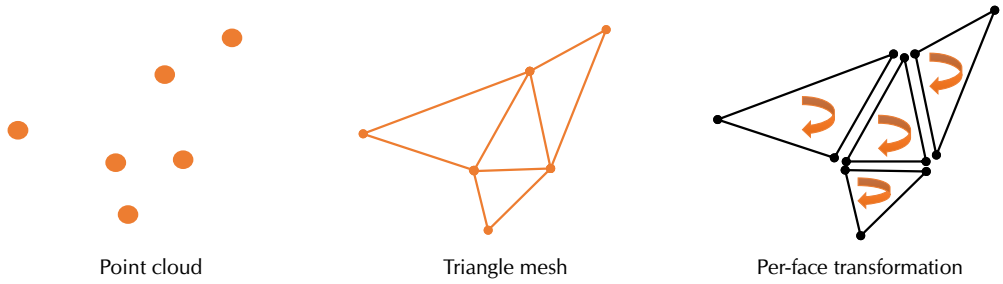


Figure 2.2: Solid objects can be represented by 3D point clouds (left), surface meshes (center), and local transformations (right). Point clouds are well suited to represent sparse estimates, such as skeleton joint positions. Meshes approximate arbitrary surfaces with piecewise flat faces that connect vertex points. Local transformations yield invariance, e.g., representing each face by a linear transformation gives translation invariance. Volumetric and continuous representations are shown in Figure 2.3.

relative position $\mathbf{v}_{i,b}$ of vertex $\mathbf{v}_i$ to bone b is computed for all vertex-bone pairs i, b in the reference mesh. Moreover, corresponding *skinning weights* $w_{i,b} \in [0,1]$ are predefined, the weight defines the influence of bone b on vertex i. Given a new skeleton deformation with affine bone transformations $\mathbf{A}_b$, for instance, bone orientation and position as a matrix operating in homogeneous coordinates, each vertex v_i of the reference mesh is deformed by the weighted sum

$$\hat{\mathbf{v}}_i = \sum_b w_{i,b} \mathbf{A}_b \mathbf{v}_{i,b}. \tag{2.1}$$

Weights are normalized to unity, $\sum_b w_{i,b} = 1$, i.e., the vertex position can be seen as an interpolation of the relative vertex-bone positions in the reference. This linear surface definition is highly efficient but suffers from artifacts, e.g. for large rotations, and many non-linear alternatives have been proposed [e.g. Kavan et al., 2007; Kim and Han, 2014]. In Chapter 5, we extend linear blend skinning to skinning by spherical proxy primitives.

2.2.2. Blend shapes and principal component analysis

Linear shape representation

Blend shapes define a semantic shape representation. The mesh is represented as a linear combination of blend shape vectors $\mathbf{u_j}$, which are added on top of a reference mesh $\mathbf{v}$ by

$$\hat{\mathbf{v}} = \mathbf{v} + \sum_j w_j \mathbf{u_j}, \tag{2.2}$$

where $\mathbf{v}$ and $\mathbf{u_j}$ are vectors that stack all 3D vertex positions of the mesh, and w_j is the activation weight of vector $\mathbf{u_j}$. For instance for facial animation, a blend shape $\mathbf{u_1}$ could correspond to the vertex displacement needed to lift the eyebrow.

While blend shapes are created by artists, relations exist to principal component analysis (PCA) on meshes. PCA extracts the principal components of a set of example meshes, these are linearly independent vectors that capture the variance of the examples. The mean shape over all examples corresponds to the blend shape reference mesh and the principal components are similar to blend shape vectors [Alexa and Müller, 2001; Allen et al., 2003]. We use a PCA basis for the volumetric

actor model introduced in Chapter 5 and for dimensionality reduction in Chapters 8 and 9.

2.2.3. Local surface representations

Local invariant representations

Instead of defining the mesh directly through absolute vertex positions, relative positions can be used to attain beneficial properties. The *mesh Laplacian* stores the relative positions of the vertices to their neighbours [Lipman et al., 2004; Sorkine et al., 2004], which gives invariance to the global translation of the mesh. Rotation invariance is gained by local representations that are independent of the global mesh orientation and translation [Lipman et al., 2005]. A very successful technique is to represent the mesh in terms of the affine transformation of each mesh face, see Figure 2.2 right. The transformations are called *deformation gradients* [Barr, 1984; Sumner, 2005; Lai et al., 2009], and can be seen as the linearization, i.e., Jacobian, of an arbitrary mesh deformation at each face. Such local representations are beneficial for surface smoothing and for interpolating or stitching different meshes, since they encode properties of the local shape. Handle-based deformation can be obtained by combining local representation and explicit vertex position constraints. It requires to solve for a consistent mesh that obeys both constraints, local shape and global handle position. As-rigid-as-possible (ARAP) deformations are constructed by decomposing the per-face affine transformation into rotation and shear components and iterative rotation compensation [Sorkine et al., 2004; Sorkine and Alexa, 2007; de Aguiar et al., 2008].

In the following, we explain two particular representations in more detail, as they are used as a baseline model for shape reconstruction in Chapter 5 and for character animation in Chapters 8 and 9.

Per-face transformations

Deformation gradient mesh representation The deformation gradient of a face f in mesh $\mathbf{v}$ is defined with respect to a reference mesh $\mathbf{v}^{\text{ref}}$. It is the affine transformation $\mathbf{A}_f$ of face f in $\mathbf{v}$ with respect to the same face in the reference mesh $\mathbf{v}^{\text{ref}}$. We use a representation that further decomposes deformation gradients $\mathbf{A}_f$ into rotation $\mathbf{R}_f$ and shear $\mathbf{S}_f$. We approximate the polar decomposition $\mathbf{A}_f = \mathbf{R}_f\mathbf{S}_f$ iteratively according to Higham [1986], which is computationally more efficient than using singular value decomposition. In practice, already three iterations give a reasonable separation. Rotations are processed in axis-angle form, the symmetric shear matrix is linearized to a vector of 6 elements. In addition to rotation and shear, we also store the absolute vertex positions $\mathbf{v}_i$ for each vertex i. The mesh is represented by a vector that concatenates the $3F$ rotation, $6F$ shear, and $3V$ point parameters, for a mesh with F faces and V vertices.

Overcomplete constraints

The combined space of vertex and face parametrization is overcomplete and neighboring affine transformations contradict if modified independently, e.g. in an editing tool, by a learned mapping function (Chapter 8), or after interpolating two meshes (Chapter 9). To mitigate this, a globally consistent surface is reconstructed by considering $\mathbf{A}_f$ and $\mathbf{v}_i$ as soft constraints with weight h on $\mathbf{v}$. We choose h by hand such that the contributions of face transformations are approximately one order of magnitude larger than the vertex position constraints. Thereby, the

face transformations control the shape reconstruction, while the vertex positions determine the global position of connected components.

Surface reconstruction

An efficient solution is possible using the previously introduced Laplacian coordinate representation. The differential coordinates δ depend linearly on the absolute vertex position, e.g.,

$$\delta^{\text{ref}} = \mathbf{L}\mathbf{v}^{\text{ref}}, \tag{2.3}$$

where the Laplacian $\mathbf{L}$ is constructed once from the reference pose vertex positions $\mathbf{v}^{\text{ref}}$ and the mesh faces, in our case using cotangent weights [Sorkine, 2006]. Given a surface representation $\mathbf{y}^{\text{shape}} = (\mathbf{A}, \mathbf{v})$, where $\mathbf{A} = (\mathbf{A}_1, \ldots, \mathbf{A}_F)$ and $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_V)$, we optimize for the mesh vertex positions $\mathbf{v}^*$ that simultaneously fit the reference pose updated by rotations $\mathbf{A}$ and goal vertex positions $\mathbf{v}$ in the least squares sense, similar to [Sorkine and Alexa, 2007]:

$$\mathbf{v}^* = \arg\min_{\hat{\mathbf{v}}} \|\mathbf{v} - \hat{\mathbf{v}}\|^2 + h\|\mathbf{A}(\delta^{\text{ref}}) - \mathbf{A}(L)\hat{\mathbf{v}}\|^2, \tag{2.4}$$

where $\mathbf{A}(\mathbf{v}^{\text{ref}})$ are the reference differential coordinates updated by per-face transformations $\mathbf{A}$, and $\mathbf{A}(L)$ updates the cotangent weights of L given A. Since Equation 2.4 is quadratic in $\hat{\mathbf{v}}$, and $\mathbf{L}$ is sparse, $\mathbf{v}^*$ can be computed efficiently by solving a sparse linear system of equations.

Explicit vertex constraints can be easily imposed by setting selected $\mathbf{v}_i$ to the goal location and increasing their weight. In a deformation tool, the goal locations can be used as control handles. This strategy is used to prevent ground penetration and foot skating in Chapter 9.

Pose and shape model

SCAPE model The shape completion and animation of people (SCAPE) model builds on the deformation gradient representation and constructs a human body model with separate shape and pose parameters by per-face transformations [Anguelov et al., 2005]. In comparison to the previously introduced deformation gradient representation, the affine matrix $\mathbf{A}$ is further decomposed into pose and shape factors by fitting to a database of laser scans to learn pose and shape variation and their dependency. The rotation component is used to model rigid transformations of bones, while additional linear components model shape variations and pose dependent deformation, such as muscle bulging. The result is a parametric body model that proved to have outstanding generalization properties. In Chapter 5 we introduce a linear model which has similar generalization properties for coarse shapes but improved computational performance.

2.2.4. Implicit surfaces

Smooth surfaces

Surfaces also have been represented through implicit functions, e.g. as the level set of a sum of basis functions [Plankers and Fua, 2003; Ilic and Fua, 2006; Kanai et al., 2006]. Instead of approximating a surface by piecewise linear faces, it is approximated through a finite set of basis functions. Using smooth basis functions leads to a smooth surface approximation. To visualize these implicit representations, an additional surface extractions step is necessary, e.g. by marching cubes [Lorensen and Cline, 1987]. We introduce a volumetric density formulation that additionally

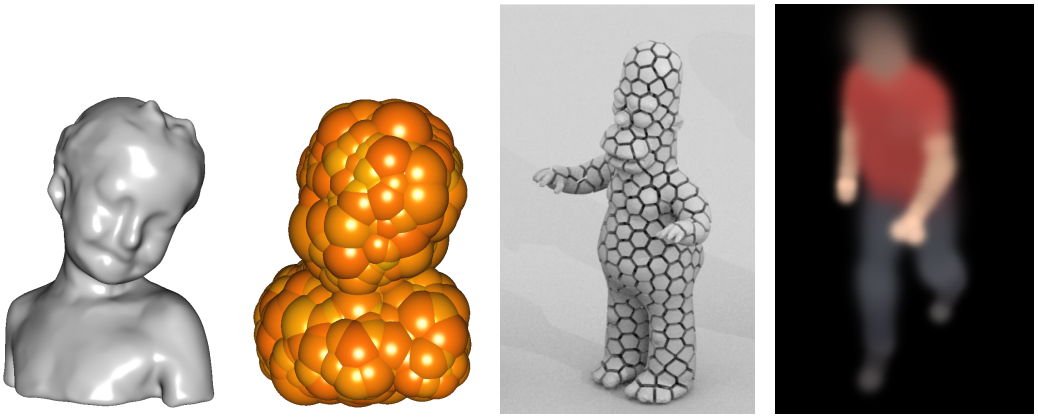


Figure 2.3: Implicit surfaces have been used to represent smooth surfaces (left). In this instance, constructed from a set of basis functions with spherical support [Wang et al., 2016b]. Volumetric meshes are also possible, they segment the volume into discrete cells and can be used to model volumetric deformations (center) [Kanai et al., 2006]. Smooth volumetric densities can be represented with a sum of Gaussians and have been utilized to approximate solid objects (right) [Rhodin et al., 2015a, Chapter 4].

models occlusions under projection as a smooth phenomenon, in Chapter 4 for colored surfaces and in Chapter 5 for model contours.

2.3. Volumetric representations

Volumetric discretization

A straightforward generalization of a surface mesh is a volumetric cell representation, see Figure 2.3. For instance, for a tetrahedral mesh representation groups of four triangles form a tetrahedron, and their concatenation discretizes the volume into cells. As for surface meshes, per vertex representations and relative representations are applicable [e.g. de Aguiar et al., 2008]. These methods model solid physical objects more accurately and have the advantage that deformation energies can represent volume preservation and anisotropic deformation behavior explicitly, to propagate deformations realistically in 3D. The tesselation of the volume is important. For instance, using Centroidal Voronoi Tesselation leads to a uniform discretization [Wang et al., 2016b], which yields improved tracking [Allain et al., 2015].

Smooth volumes

Smooth volumetric representations have also been utilized, e.g., an in-homogeneous density can be approximated by the sum over a set of Gaussian basis functions, which was used for human motion tracking [Stoll et al., 2011]. We build on these volumetric models for pose estimation in Chapter 4 and actor model initialization in Chapter 5.

2.4. Character rig representation

Rigging and deformation

In the animation domain, a *rig* refers to a low-dimensional character representation that allows for easy editing, see Figure 2.1 right. A combination of surface skinning and free-form deformation is commonly used. Such a rig manipulates near-rigid

character parts with a simple skeleton, and non-rigid deformations are driven by less constrained and more complex representations. Rigs have also been used for performance-driven character animation [Seol et al., 2013] and motion capture [e.g., Jain et al., 2010]. The advantage is that they are part of the traditional animation pipeline and are established in various animation tools. However, rig definitions and free-form deformation types vary between tools, which restricts their generality and leads to incompatibility. This was the reason for us to chose a more general mesh representation in Chapters 8 and 9.

Part II

Towards Unconstrained Motion-Capture Algorithms

Marker-less motion capture

3

Traditional application fields

Motion capture has traditionally been heavily used in digital movie production, with recent big-budget movies heavily drawing on increasingly detailed reconstructions of actors [Beeler et al., 2010]. Although not as prominent, motion capture for medical and sports supervision is also a traditional application field. It dates back to the analysis of human and animal gait by Étienne-Jules Marey using 'chronophotography', and is gaining attention in modern motion-capture approaches. For instance, the precision of a golf swing [Urtasun et al., 2005] or the strokes of an olympic swimmer [Bregler, 2012; Olstad et al., 2012] are analyzed for professional sports.

Traditional motion capture

In these traditional fields, the most common capture approaches are suit-based. Active point light sources or passively reflecting markers are attached to the actor, and their 3D motion is reconstructed using a static setup of surrounding inwards-facing cameras. Marker-less approaches succeed without suits, with recent methods reaching the accuracy of marker-based approaches. Additionally, they can reconstruct surface detail; however, existing high-quality solutions require expensive camera setups, expect controlled background and illumination, and are usually offline due to their computational complexity. The difficult setup of cameras and studio conditions, as well as their high cost, is a drawback and precludes their application in newly arising consumer applications, such as in virtual reality (VR), which require real-time, low-latency reconstruction.

Interactive motion capture

VR glasses render a realistic virtual world and allow its exploration through natural (head) motion. The rendering in response to the head position grants an immersive visual experience; however, new input modalities are needed to transition from passive exploration to active interaction with the virtual world. The utilization of user motion as a natural interface for human-computer interaction is promising; the user could interact with the world naturally and avatars would directly mirror the users motion, creating an even deeper immersion. For this purpose, existing commercial solutions provide hand-held devices with maker-based tracking; however, there is a demand for non-intrusive marker-less motion capture solutions as hand-held devices have similar drawbacks as suits. Real-time non-intrusive reconstruction methods exist; however, they are limited to studio conditions or require tedious set up and calibration.

Goal of this part

In this part of the dissertation, we introduce advances to state-of-the-art marker-less human motion capture, which in particular address the requirements for interactive virtual reality and animation. We advance the first half of the introduced processing

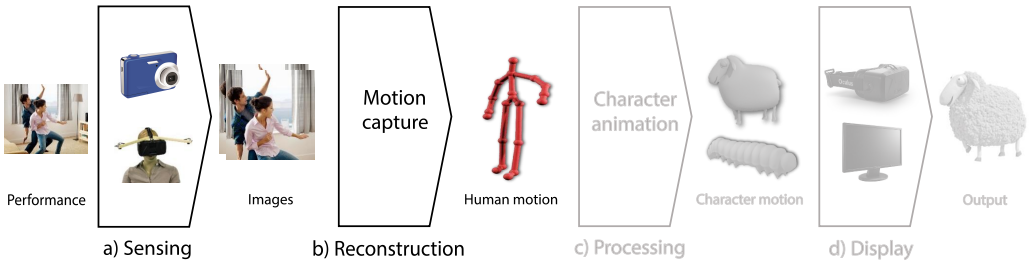


Figure 3.1: Human motion capture requires sensing hardware and reconstruction algorithms, which form, respectively, steps a), and b) of the processing pipeline introduced in Figure 1.1.

pipeline, namely the sensing and reconstruction step, see Figure 3.1 and Figure 3.1.

A top-down concept

To offer the best possible quality, we follow the top-down philosophy, and design global models that work without simplifying intermediate steps directly on the input image observations. To improve run time and robustness, bottom-up approaches are used for initialization and regularization, and care is taken to integrate these cues only as weak constraints to minimize bias and loss of information.

Evaluation

We demonstrate applicability to scenes that are less constrained than in previous methods, we test on diverse indoor and outdoor scenes with varying numbers of cameras. We evaluate the improvement of pose and shape estimation accuracy, setup complexity, and the automation of initialization. The improvement is quantified in comparison to ground-truth estimates and in comparison to state-of-the-art methods.

3.1. Overview

Three new motion-capture algorithms are presented. Each of them addresses a major shortcoming of existing state-of-the-art methods. Their contribution is described in three separate chapters.

Overview Chapter 4

In Chapter 4, we develop a motion-capture algorithm which reduces the required number of cameras, while maintaining reconstruction quality. A new analytic visibility and occlusion model is proposed, which gives rise to a top-down pose optimization method that includes occlusion and disocclusion effects into the objective function and optimization, and works directly on RGB input images.

Overview Chapter 5

In Chapter 5, we ease otherwise tedious initialization of the actor model by automatic shape estimation. A continuous volumetric contour model is proposed, that directly works on image gradients, avoiding the error-prone background segmentation used in previous bottom-up approaches. Bottom-up pose estimation is only used for initialization.

Overview Chapter 6

In Chapter 6, free roaming and reconstruction in general scenes is enabled with a new egocentric motion-capture concept. A new headgear, featuring head-mounted fisheye cameras, is developed. The user's pose is estimated from this egocentric perspective with a combination of the top-down generative model proposed in Chapters 4 and 5, and a bottom-up discriminative body-part detector. Core to the

approach is an automatic method for annotating and augmenting a large image database to train the discriminative body-part detector.

In the remainder of this chapter, we discuss the related work of Chapters 4 to 6. Chapter 4 introduces a common notation for these chapters.

3.2. Terminology

Motion capture and performance capture

We consider the problem of *marker-less human motion capture*, the estimation of 3D human skeleton pose from video input only, without requiring visual markers or specific appearance. It is closely connected to *human performance capture*, where the complete performances with surface-level detail is reconstructed.

Scene description

Dependent on the application scenario, we refer to the performing human as the *actor*, *subject*, or *user*. The actor forms the *foreground* of a scene and the rest of the scene, including static geometry and dynamic (non-tracked) elements, is the background.

Bottom-up and top-down

Algorithmically, we distinguish the major classes of *top-down* and *bottom-up* strategies. *Bottom-up* approaches incrementally process the input observations from low to high-level complexity abstractions. For instance, for performance capture, silhouettes of the actor are extracted from each input image, these are combined to visual hulls for each frame, and surface tracking merges per-frame estimates to a consistent mesh animation [Starck and Hilton, 2007]. *Top-down* approaches start from a high-level model and descend to mid- and low-level representations, e.g. starting with a 3D human body shape model, its 2D silhouette is formed by projection and is jointly refined with respect to the body shape and input image contours [Brox et al., 2005].

Generative and discriminative

In the probabilistic domain, the classification in *generative* and *discriminative models* is common. *Generative models* learn the joint probability distribution of input and output variables, and allow their joint sampling. In contrast, *discriminative models* learn the conditioned probability of the output given an input, but not their joint distribution. These terms are also used in non-probabilistic context; methods that classify or regress the output given an input directly are called *discriminative*, while those that generate output-input samples and follow the analysis-by-synthesis paradigm for output inference are termed *generative*. In the case of motion capture, *discriminative methods* directly regress body pose from the input images, whereas *generative methods* utilize an actor model, with the ability to synthesize candidate images that are sampled or optimized to match the input. Often, generative and discriminative models coincide with top-down and bottom-up approaches, respectively.

Pose and shape

We refer to *skeleton pose* as the skeleton parameters, i.e., joint angles and 3D root position, and to the position and orientation of the camera as *camera pose*. If the context is obvious, *pose* is used independently.

Model types

Parametric models represent a high-dimensional space through a fixed set of (low-dimensional) parameters. For instance, the high-dimensional space of human shape can be parametrized through a predefined linear basis. In contrast, *non-parametric*

models are adaptive to the model instance and can accommodate for the instance complexity. For example, a mesh representation can be subsampled to arbitrary resolution. Note that non-parametric models also have parameters, such as the vertex positions of a mesh or the values of a distribution histogram.

Projection Marker-less motion capture reconstructs 3D body pose from 2D projections. Most algorithms simulate the projection from 3D scene to 2D images, which is synonymously referred to as *rendering* or *image formation*.

Optimization types We use the term *local optimization* for gradient-based techniques, such as preconditioned gradient descent, as these approaches only consider the local surrounding around a guess and have local convergence. The term *global optimization* is used for methods that have non-local convergence. In general, it requires a search through the solution space, e.g. sparsely by simulated annealing [Gall et al., 2010].

Sensor arrangement We follow the sensor arrangement classification of Menache [2010]. *Outside-in* approaches place sensors around the capture volume, for instance cameras, with their views converging in the volume center. *Inside-out* approaches use body-worn sensors that look to the outside, i.e., utilize external references, such as IR markers or visual scene structure. *Inside-in* approaches succeed with body-worn sensors only, e.g. exoskeleton suits, without requiring external references and sensors.

3.3. Related work

> *This section is based on the work of Rhodin et al. [2015a, 2016a,b].*

Overview Human performance capture has been addressed from various angles and for diverse application fields. In the following, we discuss approaches separately within the following classes. We start by discussing common camera and sensor arrangements (Section 3.3.1), optical sensor types (Section 3.3.2), and suit-based motion capture (Section 3.3.3). The discussion is focused on marker-less motion capture, which is further classified into bottom-up, top-down, and hybrid algorithms (Section 3.3.4). Further, visibility models (Section 3.3.5) and actor model creation (Section 3.3.6) are discussed in more detail. For a general review, please refer to the in-depth surveys of Moeslund et al. [2006, 2011], Theobalt et al. [2010], and Holte et al. [2012].

3.3.1. Camera placements

Outside-in Traditional multi-view motion and performance capture uses an *outside-in* arrangement of cameras [Kanade et al., 1995; Gavrila and Davis, 1996], similar to the arrangement of human spectators around an action. Multiple views allow to explicitly reconstruct 3D information, but the performance is restricted to the capture volume, which is the intersection of the camera field of views. To the extreme, Joo et al. [2015] use a camera dome with 480 cameras for motion capture of closely interacting people, but domes do not scale to larger natural scenes. While static cameras ease reconstruction, movable cameras are possible by jointly optimizing

for camera and human pose [Elhayek et al., 2014], and by preceding camera pose estimation by structure-from-motion on the scene background [Hasler et al., 2009]. Movable cameras are naturally supported by methods that use a single stereo camera [Wu et al., 2013], or a monocular camera to reconstruct the human pose relative to the camera.

Inside-in

In the past, egocentric camera placements were used for tracking or model learning of certain parts of the body, for example for faces with a helmet-mounted camera or rig [Jones et al., 2011; Wang et al., 2016a], for fingers from a wrist-worn camera [Kim et al., 2012], or for eyes and eye gaze from cameras in a head-mounted rig [Sugano and Bulling, 2015]. Rogez et al. [2014] and Sridhar et al. [2015] track articulated hand motion from body- or chest-worn RGB-D cameras. Using a body-worn depth camera, Yonemoto et al. [2015] extrapolate arm and torso poses from arm-only RGB-D footage. These existing optical *inside-in* approaches succeed without external sensors or references, but do not capture full-body motion as our new formulation in Chapter 6 does.

Inside-out

Turning the standard outside-in capturing approach on its head, Shiratori et al. [2011] attach 16 cameras to body segments in an *inside-out* configuration, and estimate skeletal motion from the position and orientation of each camera as computed with structure-from-motion. Jiang and Grauman [2016] attempted full-body pose estimation from a chest-worn camera view by analyzing the scene, without observing the user directly and at very restricted accuracy.

Free roaming

Body-worn cameras have the advantage of a dynamic capture volume that moves with the user, but articulated full-body motion capture with a lightweight body-mounted setup was not yet attempted. In Chapter 6, we propose a lightweight headgear with two fisheye cameras facing down on the user and demonstrate. For the first time full-body capture from this egocentric *inside-in* camera perspective.

3.3.2. Optical motion-capture sensors

Restricted depth information

Recently, reconstruction from low-cost active depth cameras, which in addition to per-pixel color information provide an estimate of the object-camera distance, gained attention. 3D pose estimation is highly accurate and reliable when using multiple RGB-D cameras [Zhang et al., 2014], and even feasible from a single RGB-D camera in real time [e.g. Shotton et al., 2011; Baak et al., 2011; Wei et al., 2012], and also offline with surface detail [Helten et al., 2013; Cui et al., 2012; Bogo et al., 2015]. However, existing consumer devices have only a restricted capture volume, which severely restricts performances to on-spot motion and prohibits roaming. Moreover, their energy consumption is high, especially outdoors, due to the active sensing modality, which makes them unsuitable for mobile applications.

Following traditional marker-less motion capture, the work described in this dissertation uses passive color cameras, as they are generally applicable, are widely available, and have low energy consumption. Alternative non-optical sensor types are discussed in the following section.

3.3.3. Suit-based motion capture

Optical outside-in

Marker-based optical systems use a suit with passive retro-reflective spheres (e.g. Vicon) or active LEDs (e.g. PhaseSpace) and outside-in camera arrangement. Skeleton motion is reconstructed from observed marker positions in multiple cameras (usually ten or more) in an outside-in arrangement, producing highly accurate sparse motion data, even of soft tissue [Park and Hodgins, 2008; Loper et al., 2014], but the external cameras severely restrict the recording volume, in additional to the high cost of such systems. For character animation purposes, where motions are restricted, the use of motion sub-spaces can reduce requirements to six markers and two cameras [Chai and Hodgins, 2005], or a single foot pressure-sensor pad [Yin and Pai, 2003], which greatly improves usability. For hand tracking, a color glove and one camera is practical [Wang and Popović, 2009].

Inside-in

Inertial measurement units (IMUs) fitted to a suit (e.g. Xsens MVN) allow for inside-in pose estimation [Tautges et al., 2011]. Combinations with ultrasonic distance sensors [Vlasic et al., 2007], video input [Pons-Moll et al., 2010, 2011], and pressure plates [Ha et al., 2011], suppress the drift inherent to IMU measurements and reduce the number of required IMUs. Besides drift, the instrumentation with IMU sensors is the largest drawback, as it causes long set-up times and intrusion. Nevertheless, free roaming and high reliability in cluttered scenes have proven their merit in diverse applications. Exoskeleton suits (e.g. METAmotion Gypsy) provide inside-in estimation and avoid drift, but require even more cumbersome instrumentation.

Optical inside-out

The inside-out approach of Shiratori et al. [2011] requires full instrumentation of the human body as well as static backgrounds for structure-from-motion, but allows free roaming, i.e., it overcomes the strong capture volume limitation of systems using external cameras, which has been inspirational for our egocentric approach in Chapter 6.

3.3.4. Marker-less motion and performance capture

Top-down or bottom-up

Bottom-up and top-down motion and performance capture approaches lead to fundamentally different characteristics. We start to discuss the features of each approach separately, using representative algorithms which predominantly follow one of the strategies. Afterwards, hybrid approaches are discussed, which try to balance the advantages and disadvantages by combining multiple strategies.

Bottom-up approaches

Increasing level of complexity

Bottom-up motion and performance capture approaches incrementally extract representations from the input images with increasing levels of complexity. For multi-view studio setups, a common approach is to first segment foreground and background by color keying, e.g. by using green screen, or background subtraction. The actor silhouettes from multiple input views are then fused to carve out the actor's visual hull, formed by the intersection of all silhouettes back-projected into the scene [Laurentini, 1994; Matusik et al., 2000; Starck and Hilton, 2007]. An alternative bottom-up strategy is to extract stereo reconstructions for pairs of views, which are subsequently fused into a complete and consistent mesh [Bradley

et al., 2008]. Both of these mid-level representations are independently estimated per video frame, and an additional step is needed to obtain a consistent 3D mesh sequence with vertex correspondence [Budd et al., 2013; Cagniart et al., 2010]. The actor's skeleton can also be reconstructed by fitting to the inside of the per-frame mesh reconstructions, e.g. by medial axis extraction [Ménier et al., 2006].

Sparse features

Another bottom-up strategy is the extraction of 2D correspondences over time by tracking prominent local image features, such as with SIFT [Lowe, 2004], and dense optical flow. It is a core ingredient in the non-rigid structure-from-motion (SfM) domain [Bregler et al., 2000] and was also used to aid human performance capture [Tung et al., 2009].

Manual feature design

Incremental processing simplifies algorithms and allows efficient non-parametric reconstruction. However, hand-crafted features drop information in the abstraction process and errors occurring in early stages propagate to later stages.

Machine learning

Instead of designing individual steps by hand, discriminative machine learning techniques can be applied to learn intermediate representations and their extraction process automatically from a large corpus of annotated training examples. Convolutional neural networks (ConvNets) brought a major leap in performance, allowing for very accurate 2D pose prediction in images [Jain et al., 2014, 2015, 2014; Toshev and Szegedy, 2014]. Pure bottom-up prediction of 2D body pose is possible with deep networks that capture the complete pose context and have large receptive fields [Pishchulin et al., 2016]. While the manual design by layering multiple ConvNets allows for increased network depth [Pfister et al., 2015; Wei et al., 2016], the recent success of residual networks [Hernández-Vela et al., 2016; Newell et al., 2016] allows accurate pose prediction from a single deep network [Insafutdinov et al., 2016].

3D prediction

Deep ConvNets also enabled 3D pose estimation in bottom-up fashion. Different input image representations have been proposed, 3D HOG features in motion-compensated videos [Tekin et al., 2016b], giving 2D pose estimates as additional input [Park et al., 2016], and by using hand-crafted features [Ionescu et al., 2014]. For the output, relative joint positions [Li and Chan, 2014], auto encoders [Tekin et al., 2016a], and the integration of kinematic constraints into the neural network [Zhou et al., 2016] have been proposed. Besides neural networks, regression forests have also been used to derive mid-level *posebit descriptors*, answering questions such as "is the left leg in-front of the right leg", to query a 3D pose database with corresponding annotation [Pons-Moll et al., 2014]. Related is also the regression from 2D to 3D pose by Yasin et al. [2016].

Database limitations

Existing discriminative body-part detection techniques use simplified body models with few body parts to reduce the enormous cost of creating sufficiently large, annotated training datasets. Because existing pose databases have no motion information or limited motion variation, temporal information is often neglected and results exhibit jitter over time. Moreover, current databases are limited in pose, viewpoint, and appearance variation, such that the trained networks have limited generalization to new viewpoints, illumination conditions, and extreme poses. Some progress has been made by stitching multiple real images to increase variety [Rogez and Schmid, 2016] and by synthesizing and rendering characters

with varying textures and pose [Chen et al., 2016]. To overcome some of these limitations, we propose in Chapter 6 an automated way to annotate real training images and to increase appearance variety by augmentation.

Top-down approaches

A global model

Top-down motion-capture approaches use an actor and scene model that is matched against the image observation. Generative formulations commonly use analysis by synthesis; the relation between model parameters and observation is established with an image formation model, and the model parameters are optimized to minimize the difference between model and observation. The optimization can be performed locally, e.g. by gradient descent [Loper et al., 2014], in a larger neighbourhood, e.g. by particle filtering [Sidenbladh et al., 2000], or globally, e.g. by simulated annealing [Gall et al., 2010]. Local optimization is particularly suitable for tracking frameworks, where the previous frame initializes the optimization of the current frame.

Differentiable rendering

Loper et al. [2014] proposed *OpenDR*, an open-source renderer that models illumination and appearance of arbitrary mesh objects, and computes numerically approximated derivatives with respect to the model parameters for perspective projection. It can be used for inverse rendering, as the derivatives allow general gradient-based optimization of the model parameters. Finite differences are used for the spatial derivatives of pixel colors, as proposed for faces by Jones and Poggio [1996]. To attain smooth visibility at single occlusion boundaries, spatial smoothing by convolution of the model projection with a smooth kernel [Jones and Poggio, 1996; Yezzi and Soatto, 2003], and coarse-to-fine pyramid representations [Jones and Poggio, 1996; Blanz and Vetter, 1999; Loper et al., 2014] have been used. Some global dependencies in pose energy between distant scene elements are also handled by coarse-to-fine approaches.

Primitive-based models

Using simple primitives, like spheres [Ma and Wu, 2014] and (super-)quadrics [Krivic and Solina, 2003], reduces computational complexity. Instead of using a full-fledged render, simple primitives can be projected in closed form. Bregler and Malik [1998] model the actor shape with ellipsoid primitives and skeleton joints by twists, and this top-down model is then tracked with an optical flow formulation.

Smooth models

Gaussian functions have also been used for pose estimation [Wren et al., 1997; Stoll et al., 2011]. Stoll et al. use a collection of volumetric 3D Gaussians to represent the human body, as well as 2D Gaussians to model input images. The overlap between projected model and image Gaussians is maximized, using a coarse occlusion heuristic. This design allows for long-range effects between model and observation, and avoids expensive occlusion tests, but the occlusion heuristic leads to a problem formulation that is merely piecewise differentiable.

Smooth visibility

The handling of occlusions is an open challenge of existing top-down approaches in general. Occlusion models are discussed in detail in Section 3.3.5. In Chapter 4, a method is proposed that provides differentiable visibility based on ray tracing of non-homogeneous translucent volumes.

Hybrid approaches

Silhouette and mesh

Augmenting top-down pose estimation with bottom-up approaches led to increased tracking robustness and enabled reconstructions with a lower number of cameras. For multi-view input in controlled environments, actor silhouettes are the most common features. They have been used to guide top-down surface models [Starck and Hilton, 2007] and rigged template meshes [Vlasic et al., 2008; Wu et al., 2013]. A coupling and alternating of top-down and bottom-up silhouette extraction is particularly robust [Bray et al., 2006; Hasler et al., 2009; Wu et al., 2013], allowing for less controlled scenes. Tracked SIFT features can give additional sparse constraints [de Aguiar et al., 2008; Gall et al., 2009].

Generative and discriminative

Bottom up learning-based methods are robust, but accuracy and temporal stability is increase in hybrid approaches; local image evidence is extracted discriminatively in a bottom-up fashion, and coherent body pose is inferred top-down by generative human body models. Different skeleton models have been utilized: graphical models provide weak prior information and allow for exact inference of 2D pose from monocular images [Sigal et al., 2004; Lee and Nevatia, 2006; Felzenszwalb and Huttenlocher, 2005; Andriluka et al., 2009; Yang and Ramanan, 2013; Johnson and Everingham, 2011]. Moreover, 3D graphical models have been proposed for pose estimation from multi-view footage [Sigal et al., 2012; Amin et al., 2013; Burenius et al., 2013; Belagiannis et al., 2014]. Articulated skeleton models provide stronger kinematic constraints, but only local inference is feasiable [Elhayek et al., 2015]. To further constrain and regularize the solution space, skeleton models have been augmented with learned pose priors [Chai and Hodgins, 2005; Brox et al., 2006] and temporal priors [Sidenbladh et al., 2000; Urtasun et al., 2005, 2006b].

Monocular 3D pose

While the previously discussed hybrid models require multi-view input, monocular 3D pose prediction is also possible [Sminchisescu et al., 2006]. Kostrikov and Gall [2014] propose to regress from monocular images to a volumetric occupancy grid with regression forests and use a graphical top-down body model. Bogo et al. [2016] attain 3D pose estimation from monocular images by lifting ConvNet 2D pose predictions to 3D using a strong skeleton pose prior. Additionally, coarse shape estimates are predicted using a body shape prior. A similar combination of discriminative and generative pose estimation is used in Chapter 6 for pose estimation and in Chapter 5 for fully automatic pose and shape estimation.

3.3.5. Visibility and occlusion handling

Discrete visibility

Independent of the model representation, the image formation model needs to consider surface visibility, and needs to be differentiated for gradient-based optimization. The problem is that the (discrete) visibility function of a surface point is generally non-differentiable at occlusion boundaries of solid objects, and often hard to express in analytic form. Some approaches avoid explicit construction of the visibility function by heuristically fixing occlusion relations at the beginning of iterative numerical optimization, which can easily lead to convergence to erroneous local optima, or by re-computation of visibility before each iteration, which can become computationally prohibitive. Commonly the object's silhouette boundaries are handled as special cases, different from the shape interior [Matusik et al., 2000;

Starck and Hilton, 2007; Tung et al., 2009; Wu et al., 2013]. These approaches optimize the model configuration (e.g. pose and/or shape) such that the projected model boundaries align with multi-view input silhouette boundaries and possible additional features away from the silhouette [e.g. Deutscher and Reid, 2005; Rosenhahn et al., 2005; Urtasun et al., 2006a; Sigal et al., 2010; Ballan et al., 2012].

Optimizing visibility

The integration of binary visibility into the objective function is more complex. Analytic visibility gradients can be obtained for implicit shapes [Gargallo et al., 2007] and mesh surfaces [Delaunoy and Prados, 2011], by resorting to distributional derivatives [Yezzi and Soatto, 2003; Gargallo et al., 2007; Delaunoy and Prados, 2011], and by geometric considerations on the replacement of a surface to another with respect to motions of the occlusion boundary [Jalobeanu et al., 2004; de La Gorce et al., 2008; Loper et al., 2014]. For multi-view reconstruction of convex objects, visibility can be inferred efficiently from surface orientation [Lempitsky et al., 2006]. While these approaches yield similarity functions that are mathematically differentiable almost everywhere, non-differentiability is resolved only locally, which still leads to abrupt changes of object visibility, as illustrated in Figure 4.1. Efficient gradient-based numerical optimization of the similarity does not fare well under such abrupt and localized changes, leading to frequent convergence to erroneous local optima. In Chapter 4, we introduce a scene and visibility model that handles the important case of double occlusion boundaries well, which occur at the point of complete occlusion of an object, see Figure 4.1.

Gaussian visibility

The visibility approach that we introduce in Chapter 4 is inspired by the method of Stoll et al. [2011], which models the actor and image with a set of Gaussians. Stoll et al.'s method computes the model-to-image similarity analytically through the integral of pairs of Gaussians. However, there is no differentiable occlusion handling. For object tracking, Milan et al. [2011] use a similar Gaussian overlap model and add a continuous depth ordering heuristic through a sigmoid function on the depth difference of Gaussians. It was later generalized to per-ray depth ordering by approximations with sigmoid functions [Chandraker and Dhiman, 2016].

Stochastic visibility

Most similar to our visibility model is the method of Hansard [2015], which reasons about visibility in the domain of point cloud reconstruction with a statistical interpretation. Nevertheless, the same reasoning on the attenuation in an absorbing medium is followed and the same simplification based on Gaussian basis functions is used. Both methods were developed independently and reviewed concurrently.

3.3.6. Actor model initialization

Manual initialization

Many multi-view marker-less motion-capture approaches consider model initialization and tracking as separate problems [Holte et al., 2012]. Even in recent methods working outdoors, shape and skeleton dimensions of the tracked model are either initialized manually prior to tracking [Elhayek et al., 2015], or estimated from manually segmented initialization poses [Stoll et al., 2011; Ahmed et al., 2005]. In controlled studios, static shape [Hilton et al., 1999; Bălan and Black, 2008] or dimensions and pose of simple parametric human models can be optimized by matching against chroma-keyed multi-view image silhouettes [Kakadiaris and Metaxas, 1998],

and stereo cues [Plänkers and Fua, 2001]. Many multi-view performance capture methods [Theobalt et al., 2010] deform a static full-body shape template obtained with a full-body scanner [de Aguiar et al., 2008; Gall et al., 2009; Wu et al., 2013, 2012], or through fitting against the visual hull [Vlasic et al., 2008; Starck and Hilton, 2003; Ballan and Cortelazzo, 2008; Allain et al., 2015] to match scene motion. Again, all these require controlled in-studio footage, an offline scan, or both.

Semi-automatic initialization

Shape estimation of persons in skintight clothing in single images is feasible using shading and edge cues and a parametric body model [Guan et al., 2009], or monocular pose and shape estimation from video [Sminchisescu and Telea, 2002; Sminchisescu and Triggs, 2003; Guo et al., 2012; Jain et al., 2010], but require substantial manual intervention (joint labeling, feature/pose correction, background subtraction etc.). For multi-view in-studio setups with three to four views, where background subtraction works, Bălan et al. [2007] estimate shape and pose of the SCAPE parametric body model. Optimization is independent for each frame and requires initialization by a coarse cylindrical shape model. Implicit surface representations yield beneficial properties for pose and body dimension estimation [Plänkers and Fua, 2001; Plankers and Fua, 2003] and surface reconstruction [Ilic and Fua, 2006], but do not avoid the dependency on explicit silhouette or stereo input. Bogo et al. [2016] show fully automatic pose estimation from a single image and rough shape estimates by correlating 2D body-part locations to actor shape, but could not demonstrate reconstruction of extreme body shapes. In contrast to all previously mentioned methods, the approach we introduce in Chapter 5 requires no manual interaction, succeeds even with only two camera views, works for slim and well-build actors, and on scenes recorded outdoors without any background segmentation.

Outdoor segmentation

In outdoor settings with moving backgrounds and uncontrolled illumination, foreground-background segmentation is hard, but progress has been made by multi-view segmentation [Campbell et al., 2007; Wang et al., 2014; Djelouah et al., 2015], joint segmentation and reconstruction [Szeliski and Golland, 1998; Guillemaut and Hilton, 2011; Bray et al., 2006; Mustafa et al., 2015], and also aided by propagation of a manual initialization [Brox et al., 2005; Rosenhahn et al., 2006; Hasler et al., 2009; Wu et al., 2013]. However, the obtained segmentations are still noisy, enabling only rather coarse 3D reconstructions [Mustafa et al., 2015]. Many methods do not work with only two cameras, and require manual initialization.

Edge cues

Edge cues have been widely used in human shape and motion estimation [Moeslund et al., 2006; Holte et al., 2012; Deutscher et al., 2000; Sidenbladh and Black, 2003; Sigal et al., 2012; Kehl et al., 2005]. In Chapter 5, we provide a new formulation for their use and make edges in general scenes the primary cue. In contrast, existing shape estimation techniques use edges as supplemental information, for example to find self-occluding edges in silhouette-based methods or to correct rough silhouette borders [Guan et al., 2009]. Our formulation in Chapter 5 is inspired by the work of Nagel et al., where model contours are directly matched to image edges for rigid object pose [Kollnig and Nagel, 1995] and human pose tracking [Wachter and Nagel, 1997]. Contour edges on tracked meshes are found by a visibility test, and are convolved with a Gaussian kernel. This approach forms piecewise-smooth and differentiable model contours which are optimized to maximize overlap with image

gradients. We advance this idea in several ways: our model is volumetric, analytic visibility is incorporated in the model and optimization, and occlusion changes are differentiable. Furthermore, the human is represented as a deformable object, allowing for shape estimation. Most importantly, contour direction is handled separately from contour magnitude, to compensate unknown contrast variations in the input image.

NRSfM

Our work has links to non-rigid structure-from-motion that finds sparse 3D point trajectories (e.g. on the body) from single-view images of a non-rigidly moving scene [Park et al., 2015]. Articulation constraints [Fayad et al., 2011] can help to find the sparse scene structure, but the goal is different from our automatic estimation of a fully dense, rigged 3D character and stable skeleton motion.

Parametric body models

Actor model initialization methods build on the success of parametric body models for surface representation, [e.g., Anguelov et al., 2005; Jain et al., 2010; Loper et al., 2014, 2015; Pishchulin et al., 2015]. Most common are linear subspaces of vertex positions by PCA [Allen et al., 2003], which are simple and efficient. Improved generalization to strong deformations and shape variations are obtained by per-triangle rotation and shear representations, such as SCAPE [Anguelov et al., 2005], at the cost of increased complexity. The method described in Chapter 5 extends linear parametric surface models to represent the space of volumetric shape models, along with a rigged surface and skeleton.

A versatile scene model with differentiable visibility

4

This chapter is based on Rhodin et al. [2015a].

Generative pose estimation

Vision algorithms that employ a top-down generative approach estimate the configuration $\mathbf{p}$ of a 3D shape by optimizing a function measuring the similarity of the projected 3D model with one or more input camera views of a scene. In rigid object tracking, for example, $\mathbf{p}$ models the global pose and orientation of an object, whereas in generative marker-less human motion capture, $\mathbf{p}$ instead models the skeleton pose, in addition to the rigid pose, and optionally surface geometry and appearance.

Ideal objective function

The ideal objective function for optimizing similarity has several desirable properties that are often difficult to satisfy: it should have analytic form, analytic derivative, exhibit few local minima, be efficient to evaluate, and numerically well-behaved, i.e., smooth. Many approaches already fail to satisfy the first condition and use similarity functions that cannot be expressed or differentiated analytically. This necessitates the use of methods that do not require explicit gradients, such as expensive particle-based optimization, or numerical gradient approximations that may cause instability and inaccuracy.

Visibility and occlusion

A major difficulty in achieving the above properties is the handling of occlusions when projecting from 3D to 2D. Only those parts of a 3D model visible from a camera view should contribute to the similarity. In general, this can be handled by using a visibility function $\mathcal{V}(\mathbf{p})$ in the similarity measure that describes the visibility of a surface point in pose $\mathbf{p}$ when viewed from a certain direction. For many shape representations, this function is unfortunately not only hard to formulate explicitly, but it is also binary for solid objects, and hence non-differentiable at points along occlusion boundaries. This renders the similarity function non-differentiable.

Smooth and differentiable visibility

In this chapter, we introduce a 3D scene representation and image formation model that holistically addresses visibility within a generative similarity energy. It is the first model that satisfies all of the following properties:

1. It enables an analytic, continuous and smooth visibility function that is differentiable everywhere in the scene.
2. It enables similarity energies with rigorous visibility handling that are differentiable everywhere in the model and camera parameters.

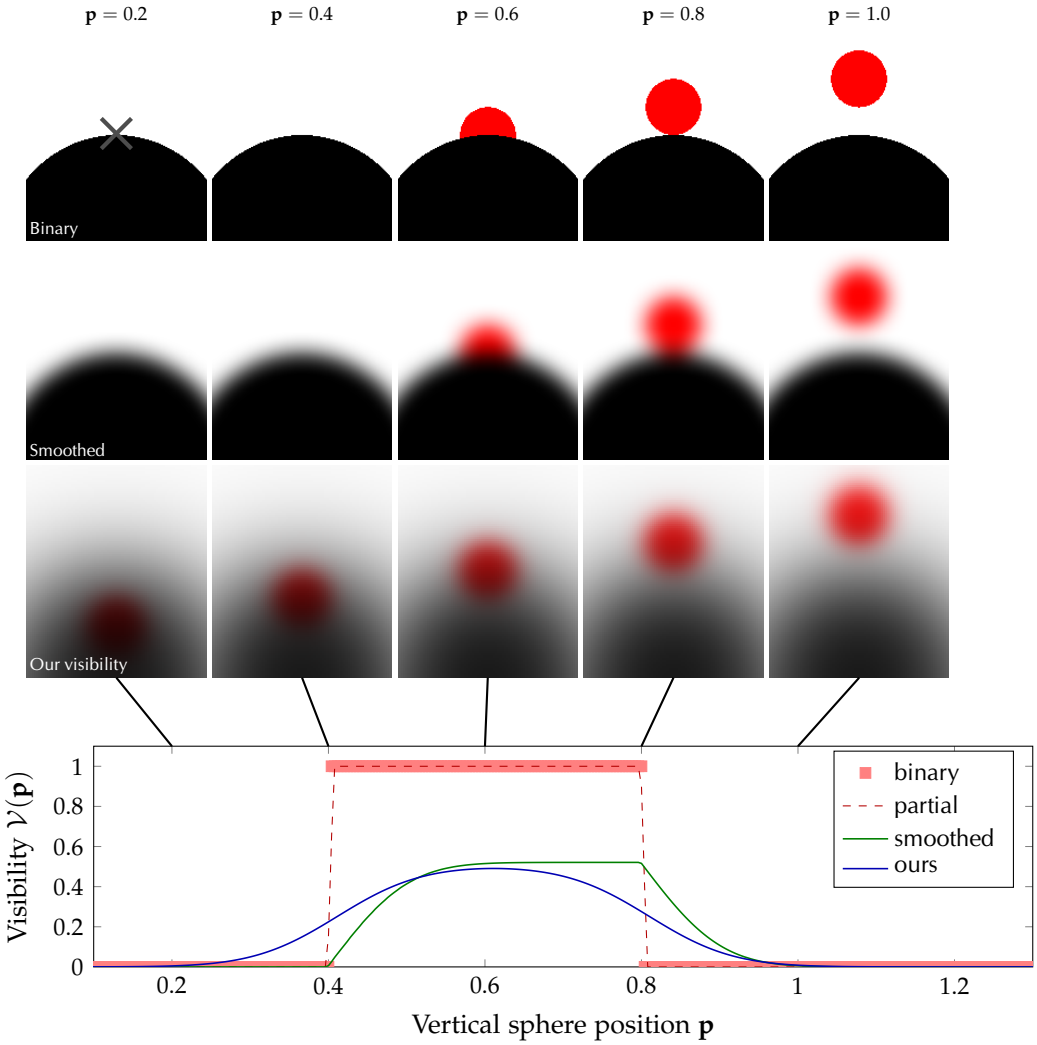


Figure 4.1: Visibility comparison on a vertically moving sphere. Top to bottom: solid scene with binary visibility, spatial image smoothing, and our visibility model for positions $\mathbf{p} \in \{0.2, 0.4, 0.6, 0.8, 1\}$. Bottom: plot of the red sphere's visibility at the central pixel (marked by the gray cross in the first image) versus sphere position $\mathbf{p}$ for the different visibilities. Only our method is smooth at the double occlusion boundaries at $\mathbf{p} = 0.4$ and $\mathbf{p} = 0.8$.

3. It enables similarity energies that can be optimized efficiently with gradient-based techniques, and which exhibit favorable and more robust convergence in cases where previous visibility approximations fail, such as disocclusions or multiple nearby occlusion boundaries.

Our method approximates opaque objects by a translucent medium with a smooth density distribution defined via a collection of Gaussian functions. This turns occlusion and visibility into smooth phenomena, a property which has not been attained by existing models, see Figure 4.1. Based on this representation, we derive a new rigorous image formation model that is inspired by the principles of light transport in translucent media common in volumetric rendering [Cerezo et al., 2005], and which ensures the advantageous properties above.

Evaluated for pose and shape estimation

Although visibility of solid objects is non-differentiable by nature, we demonstrate experimentally in Section 4.5 that introducing approximations on the scene level is advantageous compared to state-of-the-art methods that employ binary visibility and use spatial image smoothing. We demonstrate the advantages of our approach in several scenarios. For marker-less human motion capture with a low number of cameras, we show improvements on state-of-the-art methods that lack rigorous visibility modeling [Stoll et al., 2011; Elhayek et al., 2015], For multi-object pose optimization, we demonstrate increased robustness compared to methods using local visibility approximations [Loper et al., 2014]. For body shape and appearance estimation, we show automatic fitting to silhouettes.

4.1. Notation and overview

Input and output

Input to our algorithm are RGB image sequences $\mathbf{I}_{c,t}$, recorded with calibrated cameras $c = 1, \ldots, C$ across frames $t = 1, \ldots, T$. The output of our approach is a per-frame pose vector $\mathbf{p}_t$, for human motion capture it is a vector of 3D root position and skeleton joint angles. Furthermore, a volumetric scene model $\gamma(\mathbf{p}_t)$ is used, which needs to be created manually or automatically (see Section 4.3). For motion capture, the scene model is an actor model that defines appearance, shape, and skeleton structure, and it's articulation is parametrized by joint angles $\mathbf{p}_t$ (see Section 4.5.3). Pose optimization is initialized in a tracking framework with the estimate from the previous frame, the frame index t is dropped in the notation where the context admits.

Volumetri scene model

We propose a scene model that approximates solid objects by a smooth density representation, resulting in a visibility function that is well-behaved and differentiable everywhere. We start with a general introduction of the scene model in Section 4.2.1, a physically-based intuition of the resulting visibility function in terms of a translucent medium is given in Section 4.2.2, the corresponding image formation model is presented in Section 4.2.3, and the optimization is explained in Section 4.4. Results of our scene model applied to rigid pose tracking, shape estimation, and marker-less motion capture from sparse cameras are shown in Section 4.5.

4.2. Volumetric body model

4.2.1. Smooth scene approximation

Smoothnes by diffusion

Hard object boundaries cause discontinuities of visibility at occlusion boundaries. To obtain a smooth visibility function, we propose a smooth scene representation. We diffuse objects to a smooth translucent medium – with high density at the inside of the original object and a smooth falloff to the outside. In our model, the density defines the extinction coefficient which models how opaque a point in space is, and thus how much it occludes [Cerezo et al., 2005]. To obtain an analytic form and for performance reasons, we use a parametric representation for the density $D(\mathbf{x})$ at

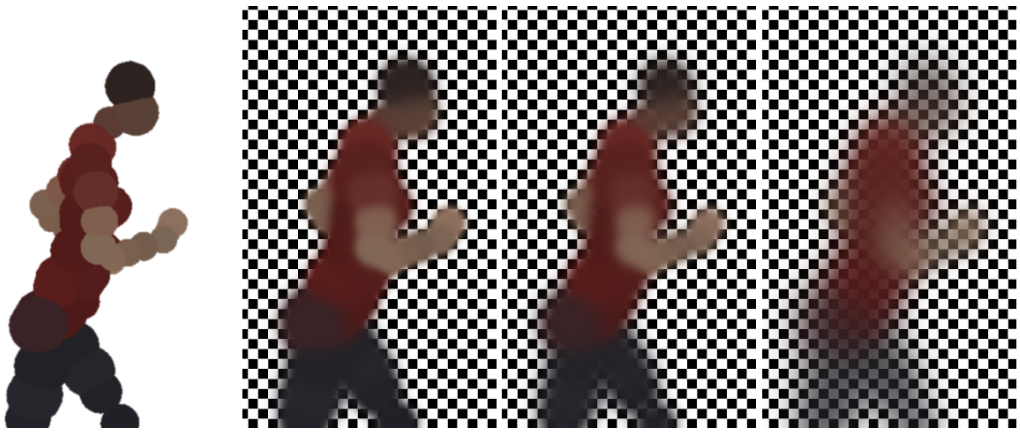


Figure 4.2: From left to right: A solid sphere actor model, our representation by a translucent medium with Gaussian density visualized on a checkerboard background for increasing smoothness levels (m=$\{0.0001, 0.01, 0.5\}$). Note the proper occlusion, e.g. of the left arm and the torso.

position $\mathbf{x}$ as the sum of scaled isotropic Gaussians $\mathcal{G} = \{G_q\}_q$, defined as

$$D(\mathbf{x}) = \sum_{G_q \in \mathcal{G}} G_q(\mathbf{x}), \text{ where each Gaussian}$$
$$G_q(\mathbf{x}) = c_q \cdot \exp\left(-\frac{\left\|\mathbf{x} - \boldsymbol{\mu}_q\right\|^2}{2\sigma_q^2}\right) \tag{4.1}$$

has a magnitude c_q, center $\boldsymbol{\mu}_q$, and standard deviation σ_q. Appearance is modeled by annotating each Gaussian with an albedo attribute $\mathbf{a}_q$. Figure 4.2 shows an example of the colored density representation for a human actor, consisting of Gaussians of varying size and albedo.

Gaussian parametrization

Our model leads to a low-dimensional scene representation parametrized by $\boldsymbol{\gamma}(\mathbf{p}) = \{c_q, \boldsymbol{\mu}_q(\mathbf{p}), \sigma_q, \mathbf{a}_q\}_q$. For readability, we use $G_q(\mathbf{x})$ for Gaussians and omit the dependence on $\boldsymbol{\gamma}$ and $\mathbf{p}$.

Adaptive smoothness

The degree of opaqueness and smoothness is adjustable by tuning the magnitudes c_q and standard deviations σ_q of the Gaussians. We discuss the conversion of a general scene to our Gaussian density representation in Section 4.3. While other smooth basis functions are conceivable, Gaussians lead to simple analytic expressions for the visibility that work well in practice. While our Gaussian representation is similar to the method of Stoll et al. [2011], its semantics of a translucent medium is fundamentally different and our image formation model with rigorous visibility is phrased in entirely new ways, as explained in the following sections.

4.2.2. Light transport and visibility

Physical light transport

Our computation of the visibility $\mathcal{V}$ of a 3D point from a given camera position is inspired by the physical laws of light transport in translucent media, and based on simulation techniques from computer graphics [Cerezo et al., 2005]. As the translucent medium is only used as a tool to model continuous visibility, we assume a medium with uniform absorption of all colors without scattering. According to

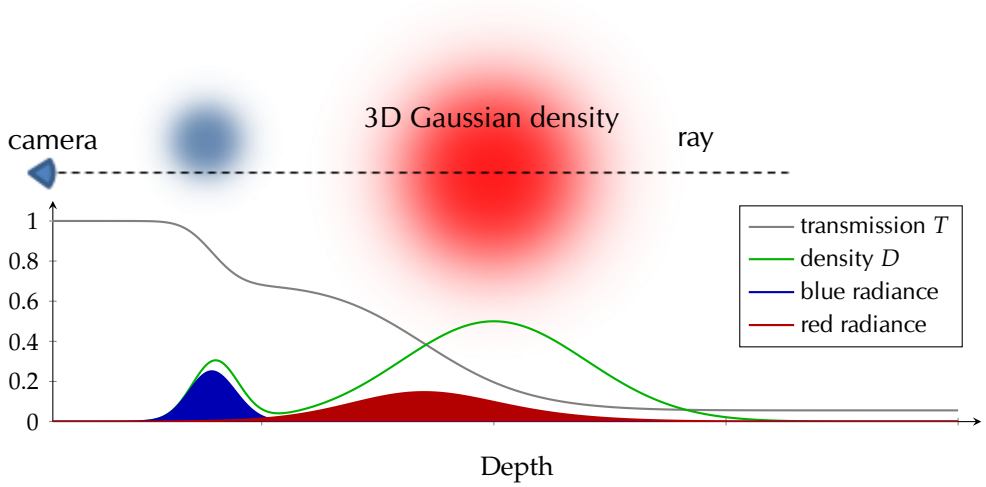


Figure 4.3: Top: Raytracing of a Gaussian density. Bottom: Light transport along the ray. The density along a ray is a sum of 1D Gaussians (green), and transmittance (gray) falls off from one for increasing optical depth. The radiance is the fraction of reflected light that reaches the camera (red and blue areas). We use it to compute the visibility of a particular Gaussian.

the Beer-Lambert law of light attenuation, the transmittance (the percentage of light transmitted between two points in space) decays exponentially with the optical thickness of a medium, i.e., the accumulated density, as visualized in Figure 4.3. Specifically, the transmittance T of a 3D point at distance s along a ray from a camera position $\mathbf{o}$ in direction $\mathbf{n}$ is

$$T(\mathbf{o},\mathbf{n},s,\gamma) = \exp\left(-\int_0^s D(\mathbf{o}+t\mathbf{n})\,\mathrm{d}t\right). \tag{4.2}$$

1D ray Gaussian

Note that for a specific camera, $\mathbf{n}(u,v)$ is uniquely defined for each pixel location (u,v); from now on, we use the short notation $\mathbf{n}$ that is implicitly dependent on the pixel position. With our Gaussian density representation, the density at any point on a line through a sum of 3D Gaussians is in turn the sum of 1D Gaussians. Specifically, inserting the line equation $\mathbf{x} = \mathbf{o}+s\mathbf{n}$ into the 3D Gaussian G_q (4.1) results in a scaled 1D Gaussian, $\bar{G}_q$ of form

$$\bar{G}_q = \bar{c}\exp\left(-\frac{(x-\bar{\mu})^2}{2\bar{\sigma}^2}\right) \tag{4.3}$$

with 1D Gaussian parameters

$$\bar{\mu} = (\boldsymbol{\mu}-\mathbf{o})^\top\mathbf{n},\ \ \bar{\sigma} = \sigma,\ \text{ and }\ \bar{c} = c\cdot\exp\left(-\frac{(\boldsymbol{\mu}-\mathbf{o})^\top(\boldsymbol{\mu}-\mathbf{o})-\bar{\mu}^2}{2\bar{\sigma}^2}\right). \tag{4.4}$$

Transmittance

Using the Gaussian form of the density, we can rewrite the transmittance function (4.2) in analytic form in terms of the error function, $\mathrm{erf}(s) = \frac{2}{\sqrt{\pi}}\int_0^s \exp(-t^2)\,\mathrm{d}t$, as

$$T(\mathbf{o},\mathbf{n},s,\gamma) = \exp\left(-\int_0^s \sum_q G_q(\mathbf{o}+t\mathbf{n})\,\mathrm{d}t\right) \tag{4.5}$$

$$= \exp\left(\sum_q \frac{\bar{\sigma}_q\bar{c}_q}{\sqrt{\frac{2}{\pi}}}\left(\mathrm{erf}\left(\frac{-\bar{\mu}_q}{\sqrt{2}\bar{\sigma}_q}\right)-\mathrm{erf}\left(\frac{s-\bar{\mu}_q}{\sqrt{2}\bar{\sigma}_q}\right)\right)\right). \tag{4.6}$$

Similar formulations are used for cloud rendering [Zhou et al., 2007; Jakob et al., 2011]. The transmittance of a medium is symmetric, it also measures the *fractional visibility* of a point $\mathbf{x} = \mathbf{o} + s\mathbf{n}$ from position $\mathbf{o}$, which we denote by $\mathcal{V}(\mathbf{x}, \gamma) := T(\mathbf{o}, \mathbf{n}, s, \gamma)$.

Fractional visibility

4.2.3. Image formation and Gaussian visibility

Image formation

For image formation, we assume that all scene elements emit an equal amount of light L_e. To produce a discrete image from the proposed Gaussian density model, we shoot a ray through each pixel of a virtual pinhole camera. The pixel color is the fraction of source radiance that is emitted along the ray and reaches the camera (color and radiance are related by the camera transfer function; we assume a linear camera response and use pixel color and radiance interchangeably). For the defined medium with pure absorption, the received radiance is the product of transmittance T, density D, albedo $\mathbf{a}$ and ambient radiance L_e, integrated along the ray $\mathbf{x} = \mathbf{o} + s\mathbf{n}$,

$$L(\mathbf{o}, \mathbf{n}) = \int_0^\infty T(\mathbf{o}, \mathbf{n}, s, \gamma) D(\mathbf{x}(s)) \mathbf{a}(\mathbf{x}(s)) L_e \, \mathrm{d}s. \tag{4.7}$$

Ambient illumination

This is a special form of the integrated *radiative transfer equation* [Cerezo et al., 2005; Chandrasekhar, 1960], and it models the fact that each point in space emits light proportional to its density $D(\mathbf{x})$ and illumination L_e. For our Gaussian density with parameters γ and fixed $L_e = 1$, we obtain

$$L(\mathbf{o}, \mathbf{n}, \gamma) = \int_0^\infty T(\mathbf{o}, \mathbf{n}, s, \gamma) \sum_q G_q(\mathbf{o} + s\mathbf{n}) \mathbf{a}_q \, \mathrm{d}s. \tag{4.8}$$

To obtain an analytic form, we approximate the infinite integral by sampling a compact interval $S_q = \{\bar{\mu}_q + k\lambda_q \mid k \in K \subset \mathbb{Z}\}$ around the mean of each G_q:

$$\hat{L}(\mathbf{o}, \mathbf{n}, \gamma) = \sum_q \mathbf{a}_q \sum_{s \in S_q} \lambda_q T(\mathbf{o}, \mathbf{n}, s, \gamma) G_q(\mathbf{o} + s\mathbf{n}), \tag{4.9}$$

where $\lambda_q \sim \bar{\sigma}_q$ is the sampling step length, which is adaptive to the Gaussian's size.

Integration by sampling

Gaussians have infinite support ($G_q(\mathbf{x}) > 0$ everywhere), but each Gaussian's contribution vanishes exponentially with the distance from its mean, so local sampling is a good approximation. In practice, we found that five samples with $K = \{-4, -3, \ldots, 0\}$ and $\lambda = \bar{\sigma}$ suffice. Importance sampling could further enhance accuracy.

Gaussian visibility

A final insight is that the inner sum in the radiance equation (4.9), the sum of the product of source radiance and transmission, measures the contribution of each Gaussian to the pixel color, and therefore computes the *Gaussian visibility*

$$\mathcal{V}_q(\mathbf{o}, \mathbf{n}, \gamma) := \sum_{s \in S_q} \lambda_q T(\mathbf{o}, \mathbf{n}, s, \gamma) G_q(\mathbf{o} + s\mathbf{n}), \tag{4.10}$$

of G_q from camera $\mathbf{o}$ in direction $\mathbf{n}$. The Gaussian visibility from pixel (u, v), $\mathcal{V}_q((u, v), \gamma)$, is equivalent to $\mathcal{V}_q(\mathbf{o}, \mathbf{n}(u, v), \gamma)$. The radiance $\hat{L}$ and Gaussian visibility $\mathcal{V}_q$ depend on the set of all Gaussians in the scene. However, our model enables

us to represent most scenes with a moderate number of Gaussians (see Section 4.3), such that the analytic forms of $\hat{L}$ and $\mathcal{V}_q$ can be evaluated efficiently. For increased performance, we also exclude Gaussians with magnitude $\bar{c}_q < 10^{-5}$ for the given ray direction, which does not impair tracking quality (see Section 4.5.3).

4.3. Model creation

Sphere packing

In principle, arbitrary shapes can be approximated using a sufficiently large number of small, localized Gaussians. We convert an existing mesh model to our representation by first filling the object's volume with spheres. In our experiments, like the actor in Figure 4.2, we place spheres manually, but automatic sphere packing could be used instead [Wang et al., 2006].

Smoothness level

We then replace the spheres by Gaussians of 'equal perceived extent'. A translucent object forms its boundary at the point of strongest transmittance change. To find suitable parameters c_q and σ_q that approximate a sphere of radius r, we place a Gaussian G_q at its center and analyze the visibility $\mathcal{V}_q(\mathbf{o}, \mathbf{n}, \{c_q, \sigma_q\})$ viewed from an orthographic camera (i.e., $\mathbf{n}$ is fixed in view direction and $\mathbf{o}$ is the pixel location). We solve for magnitude c_q and standard deviation σ_q such that the transparency at the Gaussian's center, $1 - \mathcal{V}_q$, is equal to a constant m, and the inflection point of $\mathcal{V}_q$ lies at distance r from the center. Here, m is a free parameter determining the level of smoothness and translucency, see Figure 4.2. This is a useful tool to tune robustness versus specificity, as we demonstrate in Section 4.5.3. This procedure aligns the perceived Gaussian size with the reference sphere outline while maintaining a consistent opacity across Gaussians of different size. An example is Figure 4.1, where the inflection point is aligned with the binary occlusion boundary. The optimization is necessary, as the inflection point of visibility deviates from the density's inflection point, and parameters c_q and σ_q jointly influence its location.

Rigid and articulated motion

For generative tracking, the configuration of the tracked model $\mathbf{p}$ needs to be mapped to our scene representation using a function $\gamma(\mathbf{p})$. In rigid object tracking, $\gamma(\mathbf{p})$ is a single rigid transform that determines the position $\boldsymbol{\mu}_q$ of all Gaussians G_q; the sizes σ_q and densities c_q are then fixed. For skeletal motion capture, each Gaussian is rigidly attached to a bone in the skeleton, and $\mathbf{p}$ represents global pose and joint angles. Details on skeleton representations are explained in Section 2.1. Other mappings for non-rigidly deforming shapes can also be used.

4.4. Pose optimization

Objective function

Reconstruction methods based on our representation will compute $\mathbf{p}$ from a set of image observations $\{\mathbf{I}_c\}_c$ captured from different cameras c, by minimizing an objective function of the general form

$$\mathrm{F}(\mathbf{p}, \{\mathbf{I}_c\}_c) = \sum_c \mathrm{D}(\gamma(\mathbf{p}), \mathbf{I}_c) + \mathrm{P}(\mathbf{p}), \tag{4.11}$$

where $\mathrm{D}(\gamma, \mathbf{I})$ is a data term, e.g. photo-consistency, and $\mathrm{P}(\mathbf{p})$ is a prior on configurations, e.g. a general regularization terms. With our image formation model

(Section 4.2.3), we can formulate a photo-consistency-based model-to-image overlap in a fully visibility-aware, yet analytic and analytically differentiable way as:

$$\mathrm{D}_{\mathrm{pc}}(\boldsymbol{\gamma}, \mathbf{I}) := \sum_{(u,v)\in I} \left\| \hat{L}(\mathbf{o}, \mathbf{n}(u,v), \boldsymbol{\gamma}) - \mathbf{I}(u,v) \right\|_2^2, \tag{4.12}$$

where $\mathbf{I}(u,v)$ is the image color at pixel (u,v). In Section 4.5.1, we use the photo-consistency energy $\mathrm{F}_{\mathrm{pc}}(\mathbf{p}, \{\mathbf{I}_c\}_c) = \sum_c \mathrm{D}_{\mathrm{pc}}(\boldsymbol{\gamma}(\mathbf{p}), \mathbf{I}_c)$ without prior for rigid object tracking and body shape and appearance estimation.

Motion capture

We also demonstrate our approach for marker-less human motion capture. The generative method by Stoll et al. [2011] use a Gaussian representation for the skeletal body model, and transforms the input image into a collection of Gaussians using color clustering. Their data term sums the color-weighted overlap of all image and projected model Gaussians using a scaled orthographic projection and without rigorous visibility handling, see their paper for details.

Motion capture objective function

For visibility-aware motion capture, we define a new pose energy F_{mc} with a perspective camera model and a new visibility-aware data term that accumulates the color dissimilarity $d(\mathbf{I}(u,v), a_q)$ over all pixels (u,v) in image $\mathbf{I}$ and Gaussians G_q, weighted by the Gaussian visibility $\mathcal{V}_q$:

$$\mathrm{D}_{\mathrm{mc}}(\boldsymbol{\gamma}, \mathbf{I}) = \sum_{(u,v)} \sum_{q} d(\mathbf{I}(u,v), a_q) \mathcal{V}_q(\mathbf{o}, \mathbf{n}(u,v), \boldsymbol{\gamma}). \tag{4.13}$$

To analyze the influence of our new visibility function in isolation, we adopt the remaining model components from the baseline method of Elhayek et al. [2014]. To compensate for illumination changes, colors are represented in HSV space and the value channel is scaled by 0.2. To ensure temporal smoothness and anatomical joint limits, accelerations and joint limit violations are quadratically penalized in the prior term $\mathrm{P}(\mathbf{p})$. Motion capture with the new visibility-aware energy leads to significantly improved results, as we demonstrate in Section 4.5.3.

Gradient descent

For all our experiments on rigid and articulated tracking, we utilize a conditioned conjugate gradient descent solver to minimize the objective function. The analytic derivatives of the objective functions F_{pc} and F_{mc} with respect to all parameters are listed in the supplemental document published alongside Rhodin et al. [2015a].

4.5. Results

Overview

We first validate the advantageous properties of our model in general (Section 4.5.1), and then show how our scene representation and image formation model lead to improvements over the state of the art in rigid object tracking (Section 4.5.2), shape estimation, and marker-less human motion capture (Section 4.5.3).

4.5.1. General validation

Synthetic example

We validate the smoothness and global support of our visibility handling using a scene with simple occlusions: a red sphere, initially hidden by an occluder, moves up vertically and becomes visible (Figure 4.1). In our model, the visibility $\mathcal{V}$ of the

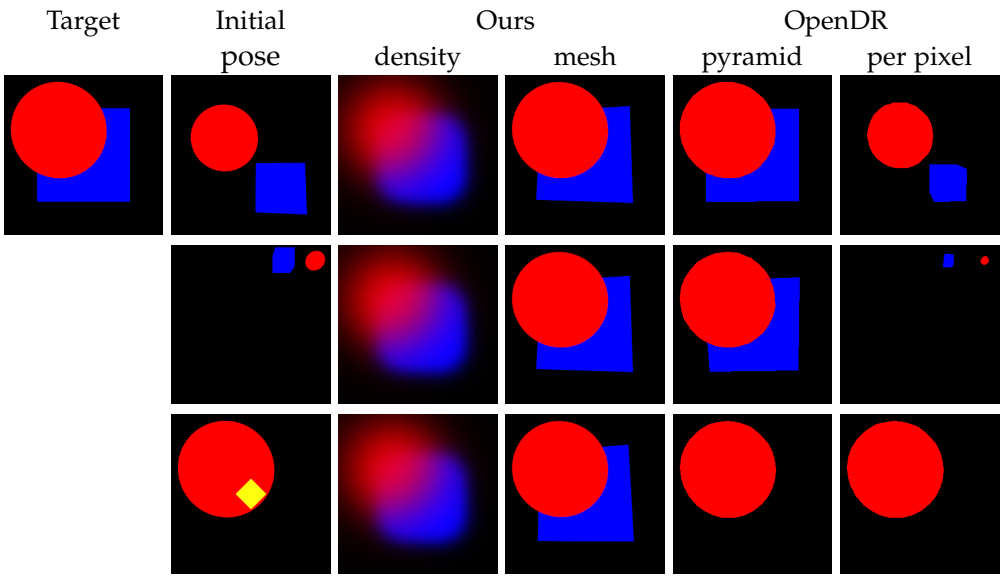


Figure 4.4: 3D reconstruction of a red sphere (position) and blue cube (position and orientation) using photo-consistency energy F_{pc} from one image for three different initializations. Top to bottom: initialization with overlap to final pose, distant initialization, occluded initialization (the occluded cube is shown in yellow). OpenDR does not find the right solution for initializations without overlap or when far from the solution, and fails under full occlusions. Our method finds the correct pose in all three cases.

red sphere (a single Gaussian) is smooth, and hence differentiable with respect to position γ (blue line). This is in contrast to surface representations which are only piecewise differentiable: binary visibility functions have discontinuities (red line), and visibility with partial pixel coverage is continuous but non-differentiable at occlusion boundaries (dashed red line). Methods that smooth pixel intensities spatially as a post-process obtain smoothness at single occlusion boundaries. However, when an object occludes or disoccludes behind another occlusion boundary, like the red sphere becoming visible behind the black sphere, and thus two or more occlusion boundaries are in spatial vicinity, visibility is non-differentiable and localized (green line). Improper handling of this case is a major limitation in practical applications, for instance, in motion capture where an arm may disocclude from behind the body. Our approach handles these cases by considering near-visible objects, it 'peeks' behind occlusion boundaries.

4.5.2. Object tracking

Comparison to binary visibility

We show the advantages of our representation for gradient-based multi-object pose optimization from a single view under photo-consistency and compare against OpenDR [Loper et al., 2014]. The nine parameters in $\mathbf{p}$ for the synthetic test scene are the 3D position of a red sphere, and position and orientation of the blue cube. Both objects shall reach the pose shown in the *Target* image of Figure 4.4. We compare the optimization of F_{pc} with $m = 0.1$ using our model, OpenDR with per-pixel photo-consistency, and OpenDR with a Gaussian pyramid of 6 levels. The optimizer is initialized with 100 random and three manual cases (rows in

Table 4.1: A table describing each scene and the relevant parameters, such as number, type and resolution of cameras, pose parameter, run time per gradient iteration, and reconstruction error (average Euclidean 3D distance over all joints and frames to the ground truth).

Sequence	*Soccer* (two actors)		*Soccer* (one actor)		*Marker*	*Walker*	Shape estimate	Rigid objects
Published by	Elhayek et al. [2015]					Stoll et al. [2011]		Our
Number of cameras	3				2	4	11	1
Number of frames	300				500	522	1	
Frame rate	23.8				25	25	n/a	
Camera type	mobile phone *(HTC One X)*				PhaseSpace Vision Camera			simulated
Raw image resolution	1280×720				1296×972			256×256
Environment	outdoor, uncontrolled background and lighting				studio, uncontrolled background			synthetic
Tracked subjects	2		1		1			2
Number of joints	118		59		61	66	0	
Number of parameters	84		42		44	43	800	9
Number of Gaussians	182		72		72	77	200	28
Input image resolution	320×180	640×360	320×360	640×360	324×243			128×128
Input pixels per frame	≈12,000	≈202,000	≈7,000	≈122,000	≈10,000	≈25,000	≈80,000	≈16,000
Ground truth	Manual annotation, 3D triangulation				Marker	12 cam	n/a	constructed
Average error [cm]	4.81	4.69	5.88	4.70	3.79	2.55	n/a	
Timing [iterations/s]	3.33	0.23	10.0	0.86	8.1	5.01	0.68	2.14

Figure 4.4). Without smoothing, OpenDR fails in all cases as object and observation boundary do not overlap sufficiently (last column). With smoothing, OpenDR captures 70% of all random initializations, when one object is fully occluded it fails (fifth column, last row). Our solution captures the correct pose in 88% of all random initializations, even under full occlusion if the occluded object is in the vicinity of the occlusion boundary (forth column, last row). Only in few cases an erroneous local minimum is reached. Averaged over all successful optimizations, the 3D Euclidean positional error of the sphere is 7×10^{-3} times its diameter (ours) vs. 4×10^{-5} (OpenDR) and for the cube 1.7×10^{-2} (ours) vs. 1.3×10^{-2} (OpenDR). In essence, the density approximation (one Gaussian for the sphere, 27 Gaussians for the cube) increases robustness and is essential for certain scene configurations. The inaccuracy due to the approximation of sharp edges is of small scale, $\approx 10^{-2}$ compared to OpenDR, model and observation align very well when visualized as meshes (fourth column).

4.5.3. Marker-less human motion capture

Three sequences, vs. two approaches

We now show the benefits of our approach for marker-less human motion capture on three multi-view video sequences with single and multiple actors. Table 4.1 describes each scene and the relevant parameters, such as number, type and resolution of cameras, how many actors, pose parameters, and run time. Our approach optimizes F_{mc} in the skeletal joint parameters (see Section 4.4). We compare against the purely generative approach by Stoll et al. [2011], and the recent combination of their generative method with a ConvNet-based joint detection [Elhayek et al., 2015], which was previously the only approach capable of marker-less skeletal motion capture in outdoor scenes with only 2–3 cameras.

Tracking accuracy

Indoor

We first quantitatively compare against both methods using the average Euclidean reconstruction error against ground-truth 3D joint positions (from a concurrently run marker-based system) using two cameras of the indoor sequence *Marker* [Elhayek et al., 2015], see Figure 4.5. All three algorithms use the same skeleton with 44 pose parameters, 72 Gaussians and data terms using HSV color space [Elhayek

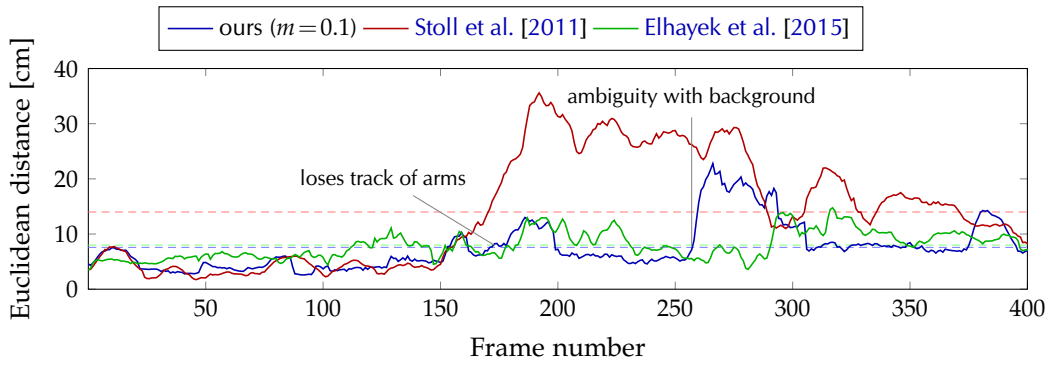
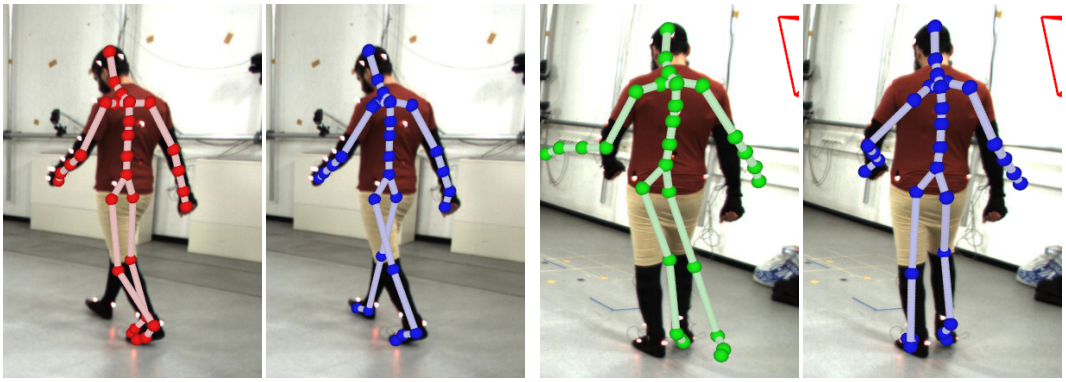


Figure 4.5: Reconstruction accuracy against marker-based ground truth. Stoll et al. loses track of the arms after frame 150, and Elhayek et al. lacks accuracy during the first half of the sequence. Our method has a 7.4 cm average joint position error – 45% better than Stoll et al. with 14 cm and best overall (dashed lines).



Stoll et al. [2011] | ours ($m = 0.1$) | Elhayek et al. [2015] | ours ($m = 0.1$)

Figure 4.6: Pose estimates for the *Marker* sequence, using two views for reconstruction. Our method properly handles occlusion of the legs in frame 38 (left), and has much higher accuracy for frame 131 (right), here viewed from a third camera not used for tracking.

et al., 2015] to be comparable. The implicit model needed for our approach is created as described in Section 4.3. Our method improves over the baseline [Stoll et al., 2011] by a 45% lower average error (7.4 cm versus 14 cm), as the baseline cannot track large parts of the sequence at all from only two views. [1] Compared to Elhayek et al., who use a discriminative component together with generative tracking, our method with visibility-aware, purely generative tracking is more precise for the first 250 frames and comparable for the last frames, where all three methods show errors due to ambiguities with the black background. We thus achieve similarly robust marker-less captured with only two cameras as their more complex method. These quantitative improvements also manifest as clear qualitative pose improvements, as shown in Figure 4.6. The proper visibility handling overcomes failures of previous techniques when arms and legs occlude.

[1] The reported numbers of all listed approaches are corrected by a factor of two compared to the respective original publications, correcting a mistake in the camera calibration of the test sequence introduced in [Elhayek et al., 2015].

Table 4.2: Reconstruction accuracy for the *Soccer* sequence against manually annotated ground truth and comparison to Stoll et al. [2011] and Elhayek et al. [2015]. Our approach and the combined generative-discriminative approach from Elhayek et al. [2015] keep good average tracking with low 3D reprojection error for all joint positions.

	Avg. error [cm]		
	ours	Stoll et al. [2011]	Elhayek et al. [2015]
all joints	7.18	10.72	4.53
limbs	4.81	9.39	4.80

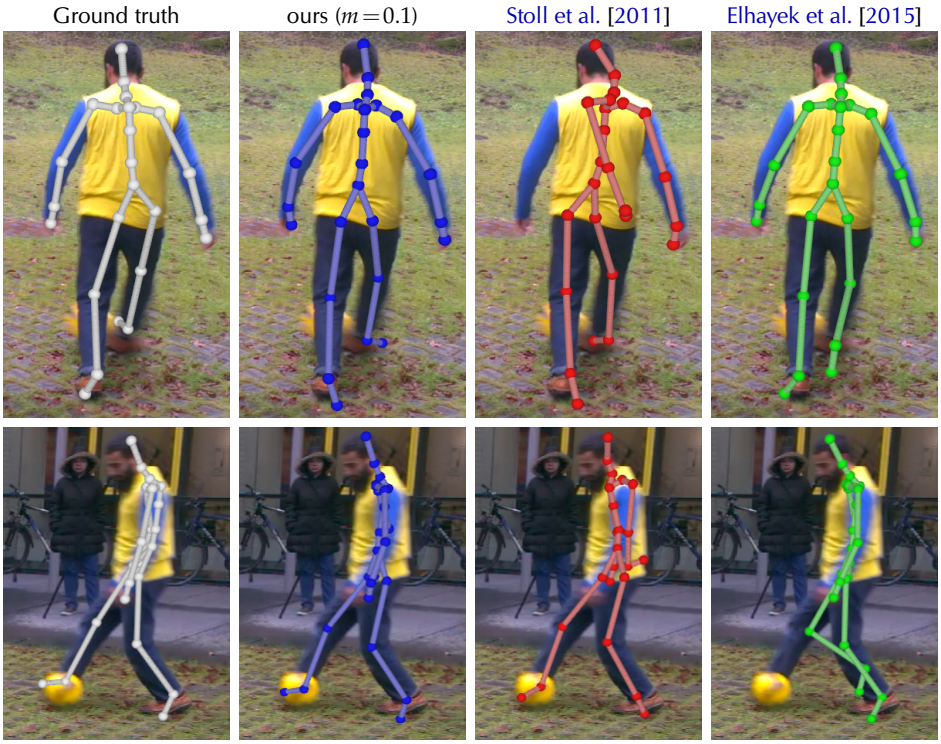


Figure 4.7: Visual comparison for the *Soccer* sequence against manually annotated ground truth, Stoll et al. [2011] and Elhayek et al. [2015]. The selected frame illustrates a case of ambiguity across two views, where the generative approach [Stoll et al., 2011] loses track of the person's arm and the combined generative-discriminative approach confuses the right foot.

Outdoor We repeat the same comparison on the outdoor sequence *Soccer* with two actors, strong occlusions and fast actions, from only three views, see Figure 4.8. Again, we obtain significantly better accuracy than Stoll et al. in terms of 3D joint position, in particular for the limb joints (see Table 4.2), as their results quickly show severe failures with so few cameras (see Figure 4.7). To analyze the impact of occlusions, we run our method once for a single actor, and in a second run we jointly track both actors (in total 84 parameters and 182 Gaussians). Simultaneous optimization not only handles self-occlusions but also the mutual occlusion of both actors. This improves by 10.6% and demonstrates the strength of precise and differentiable occlusion handling (see also Figure 4.8).

ours (2 actors) ours (1 actor)

Figure 4.8: Reconstruction for the *Soccer* sequence with comparison to tracking and modeling all versus a single actor. As shown in the figure, tracking of both subjects at the same time is advantageous as occluding regions are effectively handled by our approach.

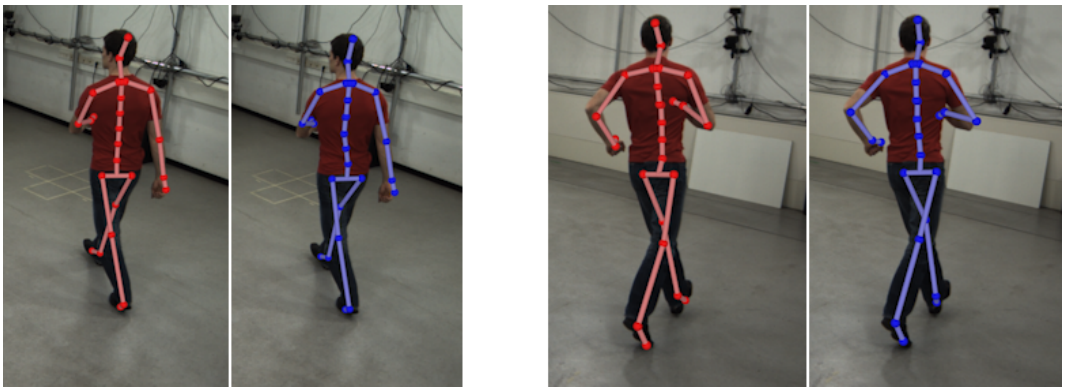


Figure 4.9: Evaluation of our method on the *Walker* sequence [Stoll et al., 2011]. With only four cameras, our method (blue) is able to accurately track the whole sequence containing walking (left) and jogging (right) motions. Here compared to 12 camera tracking with the method of Stoll et al. (red).

Tracking quality

Qualitative comparison

We further evaluate the proposed visibility model on Stoll et al.'s *Walker* sequence [Stoll et al., 2011], for which no ground truth is available. Instead, a reimplementation of the method of Stoll et al. using 12 cameras is used as reference. The same skeleton with 77 pose parameters and 43 Gaussians and image resolution 324×243 is used for both methods. The approach of Stoll et al. applied on the full available camera acquisition setup, consisting of 12 cameras, produces qualitatively comparable results to our approach applied on 4 cameras only, see Figure 4.9. The average Euclidean 3D joint position distance between both results is only 2.55 cm. At the point of very fast arm-jogging motions with strong occlusions, small errors are visible, however, our method recovers quickly.

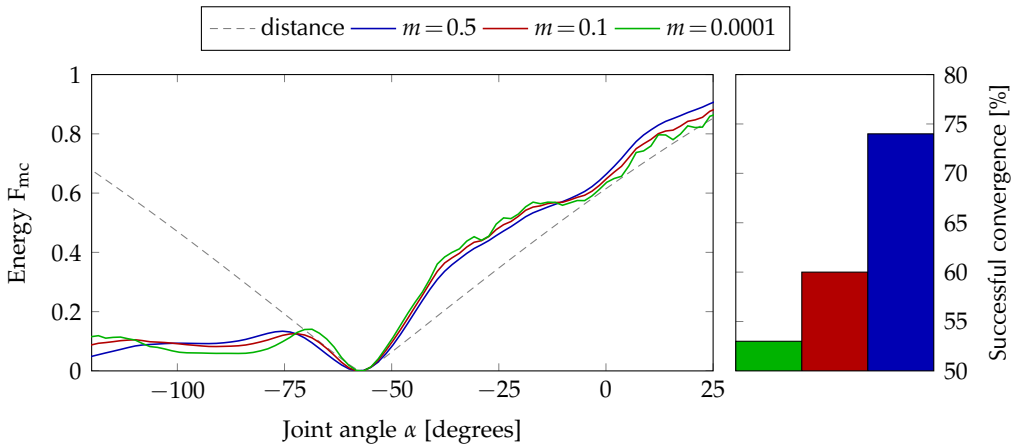


Figure 4.10: A 1D slice through the energy landscape (for the shoulder joint angle) for different smoothness values m. Higher values lead to a smoother energy with fewer local minima, and larger peaks further from the occlusion boundary (at $\alpha = -75$). The global minimum of all configurations aligns well with the Euclidean distance to the ground truth (at $\alpha = -58$). Right: Convergences from 100 initializations within the shown interval.

Radius of convergence

Smoothness analysis

Our improved scene model with rigorous visibility handling leads to more *well-behaved* similarity energies with a large *radius of convergence*, i.e., they converge for points further away from the global minimum, and a *smooth energy landscape* with few local minima (already observed in OpenDR comparison). We now validate these properties for the motion capture energy F_{mc}, see Figure 4.10 left. For frame 83 of the *Marker* sequence, we initialized the shoulder joint with $\alpha \in [-127°, 43°]$ and analyzed different choices of m. As expected, the energy F_{mc} is smoother and contains fewer local minima for smaller values of m. We measure the convergence radius by optimizing from 100 initializations with α equally spaced over the shown interval and count successful convergences. While all configurations succeed for initializations $\alpha \in [-100°, -20°]$ close to the minimum $\alpha = -58°$, only smoother versions ($m \geq 0.1$) converge for distant initializations, see Figure 4.10 right. The case $m = 0.0001$ models very sharp object boundaries, and hence gives results similar to methods with binary visibility.

Visibility gradient and smoothness level

Fixed visibility

In our final experiment, we show that our new differentiable and well-behaved visibility function is essential for the success of our approach in marker-less human motion capture with very few cameras. For the *Marker* sequence, we fix the visibility for each Gaussian and each camera prior to each iteration of the gradient-based optimizer, i.e., changes in occlusion are ignored during optimization. This setup quickly loses track of the limbs and fails completely after 110 frames, see Figure 4.11. Moreover, to analyze the behavior of our method for different degrees of smoothness in our scene model, we compare multiple fixed smoothness levels. The best trade-off between smoothness and specificity is attained for $m = 0.1$. Which we use for all our experiments, unless otherwise specified.

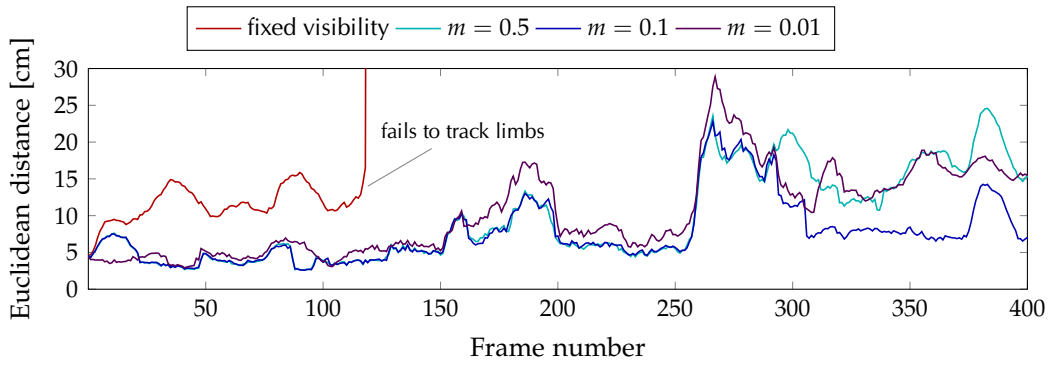


Figure 4.11: Reconstruction accuracy of joints for different smoothness levels m, and for pre-computed fixed visibility per Gaussian.

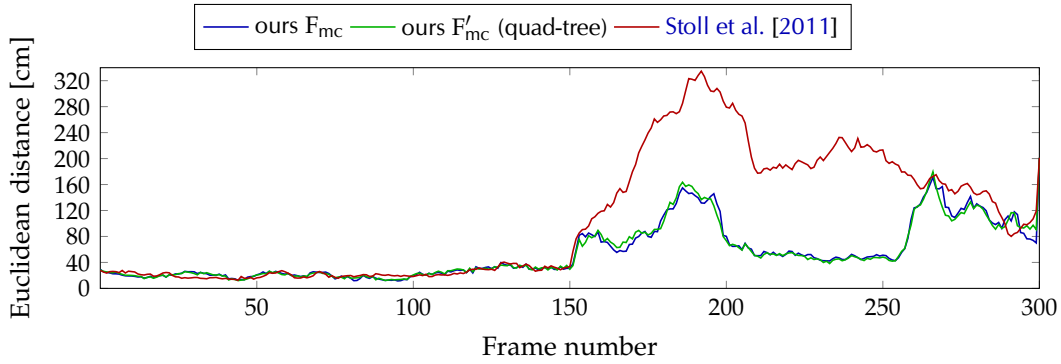


Figure 4.12: Comparison of F_{mc} and F'_{mc}, based on the Euclidean 3D joint position error of all limb joints with respect to marker-based ground truth. The impact of using a hierarchical representation is much smaller than the improvement on Stoll et al.

Input image resolution and thresholding

Image discretization

The method of Stoll et al. operates on a hierarchical image representation, that clusters regions of similar color in a quad-tree, whereas our method operates on pixels. To verify that the gained improvements are primarily due to our introduced visibility model, and not due to different image resolutions, we run our algorithm on the studio sequence with the same quad-tree representation that models squared areas of similar color by a single pixel of corresponding size. To make our energy model applicable to representations with varying pixel size, we construct the energy F'_{mc}, which is equivalent to F_{mc}, but weights each pixel by its area. The influence of the hierarchical representation on the reconstruction quality is much smaller than the improvement on Stoll et al., as shown in Figure 4.12.

Approximation error

Moreover, we validate that the error due to excluding model blobs with negligible contribution (see Section 4.2.3) is vanishingly small; the average error across the first 100 frames of the *Marker* sequence is increased by only 0.0076 cm (0.1% of the total error).

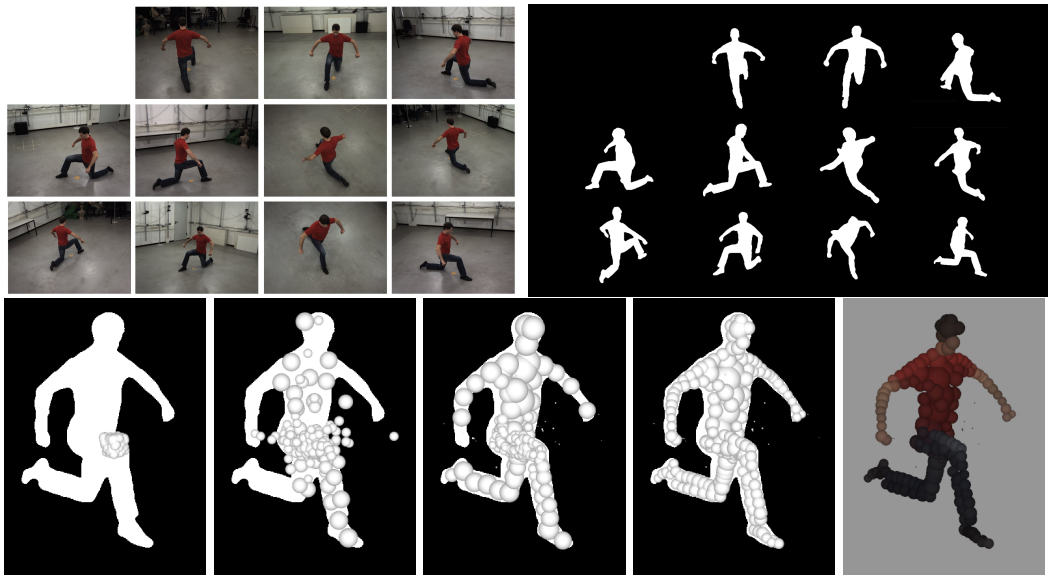


Figure 4.13: Shape and appearance estimation using the photo-consistency F_{pc}. Top: Input RGB images and extracted silhouettes. Bottom: reconstruction process from an unused camera view. From left to right: initialization, after 100, 300, 10000 iterations, and color back-projection. Each Gaussian is represented by a sphere of radius equal to its standard deviation.

4.5.4. Computational complexity and efficiency

Quadratic complexity

Our implementation of functions F_{mc} and F_{pc} and their gradients has complexity $O(N_I N_q^2 N_K + N_{\mathbf{p}} N_q)$ for N_I input pixels (summed over all views), and a scene of N_q Gaussians, N_K radiance samples and $N_{\mathbf{p}}$ parameters. The quadratic complexity in terms of the number of Gaussians originates from the handling of multiple occlusion boundaries (occlusion test for each pair of Gaussians). Our energy can nevertheless be efficiently evaluated, as the Gaussian density allows us to model even complex objects, such as a human, by few primitives. For higher accuracy, coarse-to-fine approaches could be applied. For the *Marker* sequence the performance is 8.1 gradient iterations per second for F_{mc}, 10K input pixels (pixels far from the model do not contribute and are excluded), 72 Gaussians, and 44 pose parameters. The experiments are executed for 200 iterations on a quad-core CPU with 3.6 GHz. As the visibility evaluation of each pixel is independent, further speedups could be obtained by stochastic optimization and parallel execution on GPUs.

4.5.5. Shape optimization

Shape from silhouette

The goal of this experiment is to evaluate the applicability of our image formation model to geometry and appearance estimation from multiple RGB images. Input to the method are 11 RGB images from calibrated cameras and corresponding silhouettes obtained by background subtraction (see Figure 4.13). We initialize the model with 200 small white Gaussians positioned randomly around the center of the capture volume. Subsequently, the position $\boldsymbol{\mu}_q$ and size σ_q of each Gaussian G_q is optimized for photo-consistency F_{pc} between silhouette images and model density.

Optimization stability

Figure 4.13 shows the reconstructed shape after 100, 300 and 10000 gradient iterations from a 12th camera which is not used for optimization. This verifies that the geometry of an object can accurately be estimated in the same manner as the object pose. Note that a few Gaussians are pushed outside of the silhouette (presumably due to too large gradient steps) and vanish in size. These could be removed in a post-process.

Color projection

The color of each Gaussian G_q is inferred by the weighted average over the pixel colors of all pixels (u, v) in all RGB input images, weighted by the corresponding Gaussian visibility $\mathcal{V}_q((u, v), \gamma)$. This shape estimation of a human actor is an extension of the actor model creation step proposed by Stoll et al. [2011], who optimize a parametric actor model that consists of Gaussians constrained to move with a skeleton. We show that our method is versatile enough to approximate the actor's shape without positional constraints between blobs.

4.6. Discussion and limitations

Scalability to thin shapes

A prevailing limitation is the increasing number of isotropic Gaussians needed to represent fine geometric detail. Spherical shapes are well approximated, but for thin and elongated structures the Gaussian size must be small and many isotropic Gaussians are required to fill the volume. For instance, to model a thin plate the Gaussian size is limited to the plate's vertical thickness and many Gaussians are required to fill the horizontal dimensions. For hand and object tracking, we developed an anisotropic Gaussian representation, showing increased reconstruction accuracy using fewer Gaussians [Sridhar et al., 2014]. It appears promising to extend this formulation to the algorithms developed in this chapter. Anisotropic Gaussians would introduce additional model parameters and complicate equations, but could pay off for applications requiring fine detail.

Smoothness advantages

OpenDR and other surface models may more accurately represent shape and texture, and also integrate light sources into the scene model. This allows for higher alignment precision for some shapes, but it comes at the cost of a smaller convergence radius, failure under full occlusion, and lower computational efficiency than with our model.

4.7. Summary

Smooth scene approximation

The introduced scene model and corresponding image formation model approximates a scene by a translucent medium defined by Gaussian basis functions. This intentionally smoothes out 3D shape and appearance. While this may introduce some uncertainty of shape models, it enables a visibility function and an image formation model that are differentiable everywhere, and efficient to evaluate. Analytic pose optimization energies were already used for motion capture [Stoll et al., 2011; Elhayek et al., 2015], but visibility was only approximated.

Effect of 3D model smoothing

Our new approach advances the state of the art by enabling analytic, smooth, and differentiable pose energies with analytic and differentiable visibility. It also leads to larger convergence radii of these similarity energies. Previous surface-based approaches used smoothing in the 2D image domain to improve convergence

[Loper et al., 2014]. We demonstrated that the proposed 3D scene smoothing has the effect of smoothing the entire high-dimensional pose parameter space and improves pose estimation accuracy. This not only enables us to perform purely generative motion capture at the same accuracy but with far fewer cameras than Stoll et al. [2011], but also to achieve comparable accuracy with only 2–3 cameras as the more complex method by Elhayek et al. [2015], which combines generative and discriminative approaches.

Semi-automatic initialization

A prevailing limitation of the presented approach and most existing generative motion-capture approaches is the dependence on an actor model, which is either created manually or needs correction in semi-automatic methods [Stoll et al., 2011]. The actor model accuracy is crucial for tracking accuracy, but manual correction precludes applications with non-technical users. This limitation is addressed in the succeeding chapter with a method for fully-automatic actor model and pose estimation.

General automatic human shape and motion capture using volumetric contour cues

5

This chapter is based on Rhodin et al. [2016b].

Model initialization

Marker-less full-body motion capture techniques promise to be an important enabling technique in computer animation and visual effects production, in sports and biomechanics research, and the growing fields of virtual and augmented reality. Recent methods, such as the one proposed in the previous chapter, succeed in general outdoor scenes with as few as two cameras [Holte et al., 2012; Elhayek et al., 2015; Rhodin et al., 2015a, Chapter 4]. However, before motion capture commences, the 3D body model for tracking needs to be personalized to the captured human. This includes personalization of the bone lengths, but often also of biomechanical shape and surface, including appearance. This essential initialization is, unfortunately, neglected by many methods and solved with an entirely different approach, or with specific and complex manual or semi-automatic initialization steps. For instance, some methods for motion capture in studios with controlled backgrounds rely on static full-body scans [de Aguiar et al., 2008; Gall et al., 2009; Zollhöfer et al., 2014], or personalization on manually segmented initialization poses [Stoll et al., 2011]. Recent outdoor motion-capture methods use entirely manual model initialization [Elhayek et al., 2015]. When using depth cameras, automatic model initialization was shown [Shotton et al., 2013; Bogo et al., 2015; Tong et al., 2012; Helten et al., 2013; Newcombe et al., 2015], but RGB-D cameras are less accessible and not usable outdoors. Simultaneous pose and shape estimation from in-studio multi-view footage with background subtraction was also shown [Kakadiaris and Metaxas, 1998; Ahmed et al., 2005; Bălan et al., 2007], but not on footage of less constrained setups such as outdoor scenes filmed with very few cameras.

Full automation

We therefore propose a fully-automatic space-time approach for simultaneous model initialization and motion capture. Our approach is specifically designed to solve this problem automatically for multi-view video footage recorded in general environments (moving background, no background subtraction) and filmed with as few as two cameras. Motions can be arbitrary and unchoreographed. It takes a further step towards making marker-less motion capture practical in the aforementioned application areas, and enables motion capture from third-party

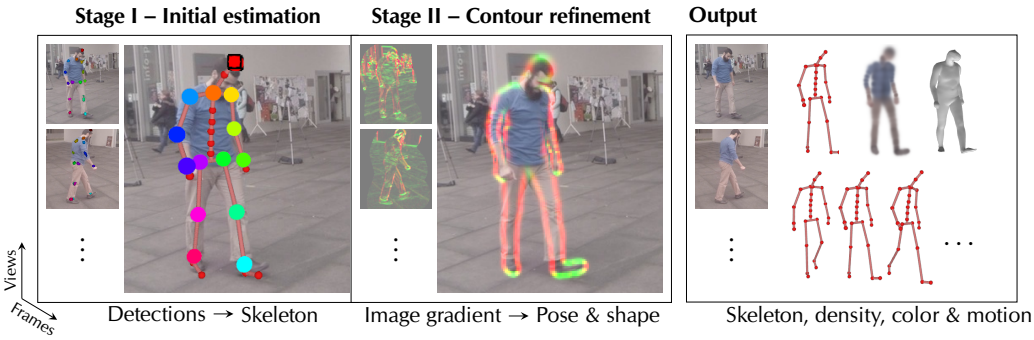


Figure 5.1: Method overview. Pose is estimated from detections in Stage I, actor shape and pose is refined through contour alignment in Stage II by space-time optimization. Outputs are the actor skeleton, attached density, mesh and motion.

video footage, where dedicated initialization pose images or the shape model altogether are unavailable. Our approach builds on the following contributions.

Volumetric contours First, we extend the volumetric scene model presented in Chapter 4, that represents the volumetric body shape by Gaussian density functions attached to a kinematic skeleton. We define a novel 2D contour-based energy that measures contour alignment with image gradients on the raw RGB images using a new volume raycasting image formation model. We define contour direction and magnitude for each image position, which form a ridge at the model outline, see Figure 5.1. No explicit background segmentation is needed for this top-down approach. Importantly, our energy features analytic derivatives, including fully-differentiable visibility everywhere.

Volumetric parametric body model The second contribution is a new data-driven body model that represents human surface variation, the space of skeleton dimensions, and the space of volumetric density distributions optimally for reconstruction using a low-dimensional parameter space.

Space-time optimization Finally, we propose a space-time optimization approach that fully automatically computes both the shape and the 3D skeletal pose of the actor using both contour and bottom-up ConvNet-based joint detection cues. The final outputs are 1) a rigged character, as commonly used in animation, comprising a personalized skeleton and attached surface, along with the (optionally colored) volumetric human shape representation, and 2) the joint angles for each video frame. We tested our method on eleven sequences, indoors and outdoors, showing reconstructions with fewer cameras and less manual effort compared to the state of the art.

5.1. Notation and overview

Input and output As in the previous chapter, input to our algorithm are RGB image sequences $\mathbf{I}_{c,t}$, recorded with calibrated cameras $c = 1, \ldots, C$ and synchronized to within a frame. In contrast to the previous approach, the actor model needs no manual initialization. The output is the complete configuration of a virtual actor model $\mathcal{K}(\mathbf{p}_t, \mathbf{b}, \gamma)$ for each frame $t = 1, \ldots, T$, comprising the per-frame joint angles $\mathbf{p}_t$, the personalized

bone lengths $\mathbf{b}$, as well as the personalized volumetric Gaussian representation γ, including color, of the actor.

Two stages

In the following, we first explain the basis of our new image formation model, the Gaussian density scene representation, and our new parametric human shape model building on it (Section 5.2). Subsequently, we detail our space-time optimization approach (Section 5.3) in two stages: (I) using ConvNet-based joint detection constraints (Section 5.3.1); and (II) using a new ray-casting-based volumetric image formation model and a new contour-based alignment energy (Section 5.3.2). The approach is evaluated in detail in terms of pose and shape estimation accuracy in Section 5.4, showing fully automatic shape estimation in less constrained environments while maintaining the accuracy of existing methods.

5.2. Volumetric statistical body shape model

Volumetric density

To model the human in 3D for reconstruction, we build on the volumetric model introduced in the previous chapter and model the volumetric extent of the actor using a set of 91 isotropic Gaussian density functions distributed in 3D space. As before, each Gaussian G_q is parametrized by its standard deviation σ_q, mean location $\boldsymbol{\mu}_q$ in 3D, and density c_q, which define the Gaussian shape parameters $\gamma = \{\boldsymbol{\mu}_q, \sigma_q, c_q\}_q$. The combined density field of the Gaussians, $\sum_q c_q G_q$, smoothly describes the volumetric occupancy of the human in 3D space, see Figure 5.1. Each Gaussian is rigidly attached to one of the bones of an articulated skeleton with bone lengths $\mathbf{b}$ and 16 joints, whose pose is parameterized with 43 twist pose parameters, i.e., the Gaussian position $\boldsymbol{\mu}_q$ is relative to the attached bone. This representation allows us to formulate a new alignment energy tailored to pose fitting in general scenes, featuring analytic derivatives and fully-differentiable visibility (Section 5.3).

Goal

In the previous chapter the 3D density models was created by a semi-automatic placement of Gaussians in 3D. Since the shape of humans varies drastically, a different distribution of Gaussians and skeleton dimensions is needed for each individual to ensure optimal tracking. In this chapter, we propose a method to automatically find such a skeleton and optimal attached Gaussian distribution, along with a good body surface. Rather than optimizing in the combined high-dimensional space of skeleton dimensions, the number of Gaussians and all their parameters, we build a new specialized, low-dimensional parametric body model.

Linear model

Traditional statistical human body models represent variations in body surface only across individuals, as well as pose-dependent surface deformations using linear [Allen et al., 2003; Loper et al., 2015] or non-linear [Anguelov et al., 2005; Hasler et al., 2010] subspaces of the mesh vertex positions. For our task, we build an enriched statistical body model that parameterizes, in addition to the body surface, the optimal volumetric Gaussian density distribution γ for tracking, and the space of skeleton dimensions $\mathbf{b}$, through linear functions $\gamma(\mathbf{s})$, $\mathbf{b}(\mathbf{s})$ of a low-dimensional shape vector $\mathbf{s}$. To build our model, we use an existing database of 228 registered scanned meshes of human bodies in neutral pose [Hasler and Stoll, 2009]. We take one of the scans as *reference* mesh, and place the articulated skeleton inside. The 91 Gaussians are attached to bones. Following the strategy proposed in Section 4.3, their position is set to uniformly fill the mesh volume, and their standard deviation

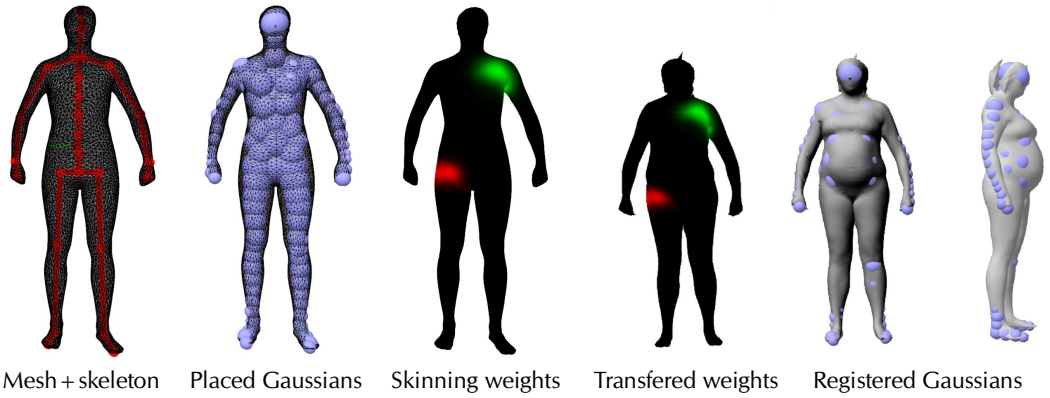


Figure 5.2: Registration process of the body shape model. Skeleton and Gaussians are once manually placed into the *reference* mesh, vertex correspondence transfers Gaussian- and joint-neighborhood weights (green and red respectively), to register reference bones and Gaussians to all instance meshes.

and density is set such that a density gradient forms at the mesh surface, see Figure 5.2 (left). This manual step has to be done only once to obtain Gaussian parameters γ_{ref} for the database reference, and can also be automated by silhouette alignment [Rhodin et al., 2015a, Chapter 4].

Inverse skinning

The best positions $\{\boldsymbol{\mu}_q\}_q$ and scales $\{\sigma_q\}_q$ of Gaussians γ_i for each remaining database *instance* mesh i are automatically derived by weighted Procrustes alignment. Each Gaussian G_q in the reference has a set of neighboring surface mesh vertices. The set is inferred by weighting vertices proportional to the density of G_q at their position in the reference mesh, see Figure 5.2 (right). For each Gaussian G_q, vertices are optimally translated, rotated and scaled to align to the corresponding instance mesh vertices. These similarity transforms are applied on γ_{ref} to obtain γ_i, where scaling multiplies σ_q and translation shifts $\boldsymbol{\mu}_q$.

Bone size

To infer the adapted skeleton dimensions $\mathbf{b}_i$ for each instance mesh, we follow a similar strategy: we place Gaussians of standard deviation 10 cm at each joint in the reference mesh, which are then scaled and repositioned to fit the target mesh using the same Procrustes strategy as before. This yields properly scaled bone lengths for each target mesh.

PCA model

Having estimates of volume γ_i and bone lengths $\mathbf{b}_i$ for each database entry i, we now learn a joint body model. We build a PCA model on the data matrix $[(\gamma_1;\mathbf{b}_1),(\gamma_2;\mathbf{b}_2),\ldots]$, where each column vector $(\gamma_i;\mathbf{b}_i)$ is the stack of estimates for entry i. The mean is the average human shape $(\bar{\gamma};\bar{\mathbf{b}})$, and the PCA basis vectors span the principal shape variations of the database. The PCA coefficients are the elements of our shape model $\mathbf{s}$, and hence define the volume $\gamma(\mathbf{s})$ and bone lengths $\mathbf{b}(\mathbf{s})$. Due to the joined model, bone length and Gaussian parameters are correlated, and optimizing $\mathbf{s}$ for bone length during pose estimation (stage I) thus moves and scales the attached Gaussians accordingly. To reduce dimensionality, we use only the first 50 coefficients in our experiments.

To infer the actor body surface, we introduce a volumetric skinning approach. The reference surface mesh is deformed in a free-form manner along with the Gaussian

set under new pose and shape parameters. Similar to linear blend skinning [Lewis et al., 2000], each surface vertex is deformed with the set of 3D transforms of nearby Gaussians, weighted by the density weights used earlier for Procrustes alignment. This coupling of body surfaces to volumetric model is as computationally efficient as using a linear PCA space on mesh vertices [Pishchulin et al., 2015], while yielding comparable shape generalization and extrapolation qualities to methods using more expensive non-linear reconstruction [Anguelov et al., 2005], see Section 5.4.5. Isosurface reconstruction using Marching Cubes would also be more costly [Plankers and Fua, 2003].

5.3. Pose and shape estimation

Space-time objective

We formulate the estimation of the time-independent 50 shape parameters $\mathbf{s}$ and the time-dependent $43T$ pose parameters $\mathbf{P} = \{\mathbf{p}_1, \ldots, \mathbf{p}_T\}$ as a combined space-time optimization problem over all frames $\mathbf{I}_{c,t}$ and camera viewpoints c of the input sequence of length T:

$$E(\mathbf{P}, \mathbf{s}) = E_{\text{shape}}(\mathbf{s}) + \sum_t \Big(E_{\text{smooth}}(\mathbf{P}, \mathbf{s}, t) + E_{\text{pose}}(\mathbf{p}_t) + \sum_c E_{\text{data}}(c, \mathbf{p}_t, \mathbf{s}) \Big). \tag{5.1}$$

Our energy uses quadratic prior terms to regularize the solution: E_{shape} penalizes shape parameters that have larger absolute value than any of the database instances, E_{smooth} penalizes joint-angle accelerations to favor smooth motions, and E_{pose} penalizes violation of manually specified anatomical joint-angle limits. The data term E_{data} measures the alignment of the projected model to all video frames. To make the optimization of Equation 5.1 succeed in general scenes with few cameras, we solve in two subsequent stages. In Stage I (Section 5.3.1), we optimize for a coarse skeleton estimate and pose set without the volumetric distribution, but using 2D joint detections as primary constraints. In Stage II (Section 5.3.2), we refine this initial estimate and optimize for all shape and pose parameters using our new contour-based alignment energy. Consequently, the data terms used in the respective stages differ:

$$E_{\text{data}}(c, \mathbf{p}_t, \mathbf{s}) = \begin{cases} E_{\text{detection}}(c, \mathbf{p}_t, \mathbf{s}) & \text{for Stage I (Section 5.3.1)} \\ E_{\text{contour}}(c, \mathbf{p}_t, \mathbf{s}) & \text{for Stage II (Section 5.3.2).} \end{cases} \tag{5.2}$$

Gradient descent

The analytic form of all terms as well as the smoothness in all model parameters enables efficient optimization by gradient descent. In our experiments we apply the conditioned gradient descent method of Stoll et al. [2011].

5.3.1. Stage I – Initial estimation

Buttom-up

Stage I roughly estimates the 3D skeleton dimensions and pose, it is not the main contribution of our work, and well-founded solutions already exist [Sigal et al., 2012; Amin et al., 2013; Belagiannis et al., 2014]. We employ the discriminative ConvNet-based body-part detector by Tompson et al. [2014] to estimate the approximate 2D skeletal joint positions, which are subsequently lifted to a consistent 3D skeleton. The detector is independently applied to each input frame $\mathbf{I}_{c,t}$, and outputs heat maps of joint location probability $\mathcal{D}_{c,t,j}$ for each joint j in frame t

seen from camera c. The heat map $\mathcal{D}_{c,t,j}$ classifies for each pixel whether joint j is apparent, the classification score can be interpreted as a per-pixel likelihood. Importantly, the detector discriminates the joints on the left and right side of the body (see Figure 5.1).

Top-down

There is no direct, one-to-one correspondence between detection and skeleton joint, as heat maps localize joint positions only roughly and are in general multi-modal due to detection ambiguities and the presence of multiple people. To nevertheless infer the poses $\mathbf{P}$ and an initial guess for the body shape of the subject, we optimize Equation 5.1 with data term $E_{\text{detection}}$, which measures the overlap of the heat maps $\mathcal{D}$ with the projected skeleton joints in terms of Gaussian overlap; Each joint in the model skeleton has an attached colored 3D joint-Gaussian G_j. The detection heat maps and joint-Gaussians (colored blobs) are both shown in the overview Figure 5.1. Each Gaussian is projected into each camera view using the projection model of the previous chapter and Rhodin et al. [2015a]. It defines the visibility $\mathcal{V}_j(u,v)$ of Gaussian G_j as seen from pixel (u,v) in the heat map. The energy term $E_{\text{detection}}$ thus measures the overlap of each Gaussian with the corresponding heat map as

$$E_{\text{detection}}(c,t,\mathbf{p}_t,\mathbf{s}) = -\sum_{(u,v)}\sum_{j}\mathcal{V}_j(u,v,\mathbf{p}_t,\mathbf{s})\cdot\mathcal{D}_{c,t,j}. \tag{5.3}$$

Hierarchical optimization

Because $E_{\text{detection}}$ is non-convex, we employ a hierarchical optimization approach. The optimization is initialized with the average human shape $(\bar{\boldsymbol{\gamma}},\bar{\mathbf{b}})$ in T-pose, at the center of the capture volume. We assume a single person in the capture volume; people in the background are implicitly ignored, as they are typically not visible from all cameras and are dominated by the foreground actor. In level I, the coarse skeleton position and orientation is determined. For this, we set joint-Gaussians to have large support (standard deviation of $\sigma = 1\,\text{m}$), and only optimize the rigid skeleton pose based on the torso joints for the first frame of the sequence. Level II refines the global pose across the whole sequence (in our experiments around 100 frames) with medium-sized Gaussians ($\sigma = 0.4\,\text{m}$). Level III adjusts bone length by optimizing the shape $\mathbf{s}$. Level IV adds elbow and knee joints with $\sigma = 0.1\,\text{m}$. Level V adds the remaining wrist and ankle joints. We observed that enabling self-occlusion for leg-Gaussians and ignoring occlusion for torso and arm joints gives best results overall.

Shape initialization

Please note that bone lengths $\mathbf{b}(\mathbf{s})$ and volume $\gamma(\mathbf{s})$ are determined through $\mathbf{s}$, hence, Stage I yields a rough estimate of γ. In Stage II, we use more informative image constraints than pure joint locations to better estimate volumetric extent.

5.3.2. Stage II – Contour-based refinement

No silhouettes

The pose $\mathbf{P}$ and shape $\mathbf{s}$ found in the previous stage are now refined by using a new density-based contour model in the alignment energy. This model explains the spatial image gradients formed at the edge of the projected model, between actor and background, and thus bypasses the need for silhouette extraction, which is difficult for general scenes. To this end, we extend the ray-casting image formation model of Chapter 4, as summarized in the following paragraph, and subsequently explain how to use it in the contour data term E_{contour}.

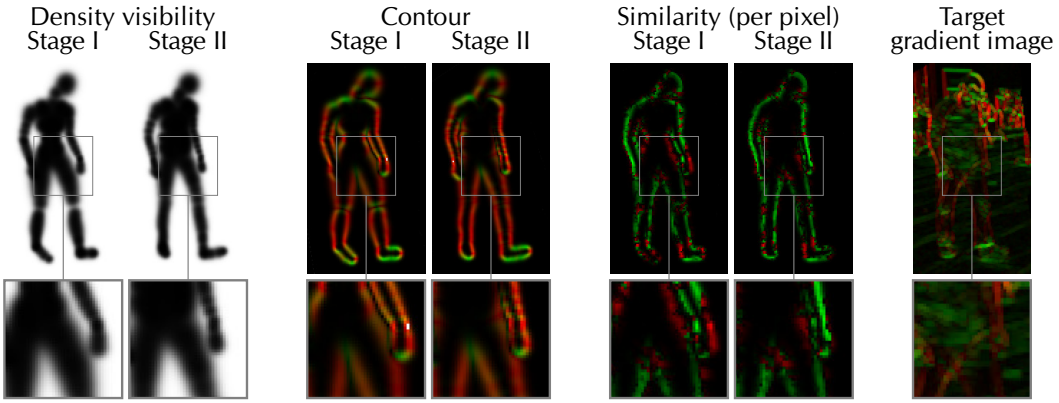


Figure 5.3: Contour refinement to image gradients through per-pixel similarity. Contour color indicates direction, green and red energy indicate agreement and disagreement between model and image gradients, respectively. Close-ups highlight the shape optimization: left arm and right leg pose are corrected in Stage II.

Ray-casting image formation model

Model visibility

Each image pixel spawns a ray that starts at the camera center $\mathbf{o}$ and points in direction $\mathbf{n}$. In the previous chapter, the visibility of a particular model Gaussian G_q along the ray $(\mathbf{o}, \mathbf{n})$ was defined as

$$\mathcal{V}_q(\mathbf{o},\mathbf{n}) = \int_0^\infty \exp\left(-\int_0^s \sum_i G_i(\mathbf{o}+t\mathbf{n})\,\mathrm{d}t\right) G_q(\mathbf{o}+s\mathbf{n})\,\mathrm{d}s. \tag{5.4}$$

This equation models light transport in a heterogeneous translucent medium [Cerezo et al., 2005], i.e., $\mathcal{V}_q$ is the fraction of light along the ray that is absorbed by Gaussian G_q. Chapter 4 and Rhodin et al. [2015a] describe an analytic approximation to Equation 5.4 by sampling the outer integral.

Background visibility

Different to their work, we apply this ray casting model to infer the visibility of the background, $\mathcal{B}(\mathbf{o},\mathbf{n}) = 1 - \sum_q \mathcal{V}_q(\mathbf{o},\mathbf{n})$. Assuming that the background is infinitely distant, $\mathcal{B}$ is the fraction of light not absorbed by the Gaussian model:

$$\mathcal{B}(\mathbf{o},\mathbf{n}) = \exp\left(-\int_0^\infty \sum_q G_q(\mathbf{o}+t\mathbf{n})\,\mathrm{d}t\right) = \exp\left(-\sqrt{2\pi}\sum_q \bar{\sigma}_q \bar{c}_q\right). \tag{5.5}$$

This analytic form is obtained without sampling, but rather it stems from the Gaussian parametrization: the density along ray $(\mathbf{o}, \mathbf{n})$ though 3D Gaussian G_q is a 1D Gaussian, with standard deviation $\bar{\sigma}_q = \sigma_q$ and density maximum

$$\bar{c}_q = c_q \cdot \exp\left(-0.5\frac{(\boldsymbol{\mu}_q-\mathbf{o})^\top(\boldsymbol{\mu}_q-\mathbf{o}) - ((\boldsymbol{\mu}_q-\mathbf{o})^\top\mathbf{n})^2}{2\sigma_q^2}\right), \tag{5.6}$$

and the integral over the Gaussian density evaluates to a constant (when the negligible density behind the camera is ignored). A model visibility example is shown in Figure 5.3 left.

Contour gradients

To extract the contour of our model, we compute the gradient of the background visibility $\nabla\mathcal{B} = (\frac{\partial\mathcal{B}}{\partial u}, \frac{\partial\mathcal{B}}{\partial v})^\top$ with respect to pixel location (u, v):

$$\nabla\mathcal{B} = \mathcal{B}\sqrt{2\pi}\sum_q \frac{\bar{c}_q}{\bar{\sigma}_q}(\boldsymbol{\mu}_q - \mathbf{o})^\top \mathbf{n}(\boldsymbol{\mu}_q - \mathbf{o})^\top \nabla\mathbf{n}. \tag{5.7}$$

$\nabla\mathcal{B}$ forms a 2D vector field, where the gradient direction points outwards from the model, and the magnitude forms a ridge at the model contour, see Figure 5.3 center. In (calibrated pinhole) camera coordinates, the ray direction thus depends on the 2D pixel location (u, v) by $\mathbf{n} = (u, v, 1)^\top / \|(u, v, 1)\|_2$ and $\nabla\mathbf{n} = (\frac{\partial\mathbf{n}}{\partial u}, \frac{\partial\mathbf{n}}{\partial v})^\top$.

Efficiency

In contrast to the visibility model presented in Chapter 4, our model is specific to background visibility, but more accurate and efficient to evaluate. It does not require sampling along the ray to obtain a smooth analytic form, has linear complexity in the number of model Gaussians instead of their quadratic complexity, and improves execution time by an order of magnitude.

Contour energy

Contour objective

To refine the initial pose and shape estimates from Stage I (Section 5.3.1), we optimize Equation 5.1 with a new contour data term E_{contour}, to estimate the per-pixel similarity of model and image gradients:

$$E_{\text{contour}}(c, \mathbf{p}_t, \mathbf{s}) = \sum_{(u,v)} E_{\text{sim}}(c, \mathbf{p}_t, \mathbf{s}, u, v) + E_{\text{flat}}(c, \mathbf{p}_t, \mathbf{s}, u, v). \tag{5.8}$$

Separation of contour magnitude and orientation

In the following, we omit the arguments $(c, \mathbf{p}_t, \mathbf{s}, u, v)$ for better readability. E_{sim} measures the similarity between the gradient magnitude $\|\nabla\mathbf{I}\|_2$ in the input image and the contour magnitude $\|\nabla\mathcal{B}\|_2$ of our model, and penalizes orientation misalignment (contours can be in opposite directions in model and image):

$$E_{\text{sim}} = -\|\nabla\mathcal{B}\|_2 \|\nabla\mathbf{I}\|_2 \cos\big(2\angle(\nabla\mathcal{B}, \nabla\mathbf{I})\big). \tag{5.9}$$

The term E_{flat} models contours forming in flat regions with gradient magnitude smaller than $\delta_{\text{low}} = 0.1$:

$$E_{\text{flat}} = \|\nabla\mathcal{B}\|_2 \max(0, \delta_{\text{low}} - \|\nabla\mathbf{I}\|_2). \tag{5.10}$$

We compute spatial image gradients $\nabla\mathbf{I} = (\frac{\partial\mathbf{I}}{\partial u}, \frac{\partial\mathbf{I}}{\partial v})^\top$ using the Sobel operator [Szeliski, 2010], smoothed with a Gaussian ($\sigma = 1.1$ px), summed over the RGB channels and clamped to a maximum of $\delta_{\text{high}} = 0.2$.

Appearance estimation

Color projection

Our method is versatile: given the shape and pose estimates from Stage II, we can also estimate a color for each Gaussian. This is needed by earlier tracking methods that use similar volume models, but color appearance-based alignment energies [Stoll et al., 2011; Rhodin et al., 2015a, Chapter 4] – we compare against them in our experiments. Direct back-projection of the image color onto the model suffers from occasional reconstruction errors in Stages I and II. Instead, we compute the

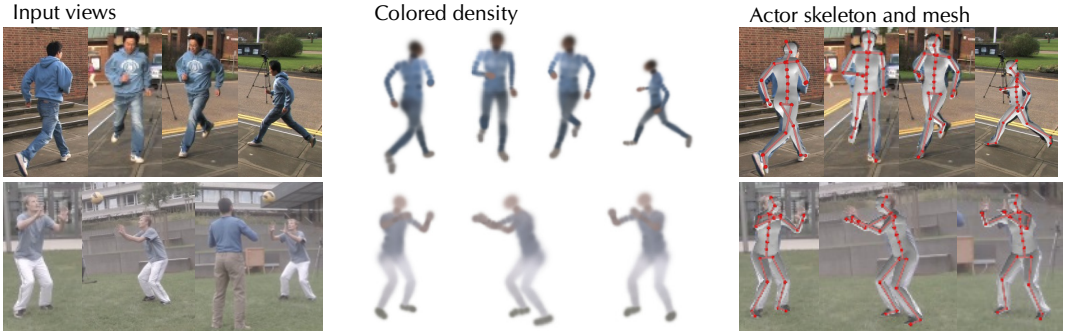


Figure 5.4: Reconstruction of challenging outdoor sequences with complex motions from only 3–4 views, showing accurate shape and pose reconstruction.

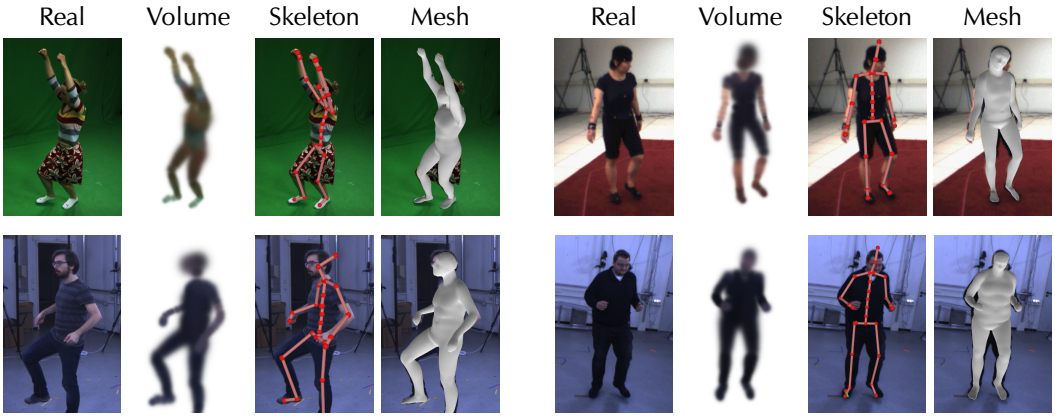


Figure 5.5: In-studio reconstruction of several subjects. Our estimates are accurate across diverse body shapes and robust to highly articulated poses.

Outlier rejection

weighted mean color $\bar{a}_{q,c}$ over all pixels separately for each Gaussian G_q and view c, where the contribution of each pixel is weighted by the Gaussian's visibility $\mathcal{V}_q$ (Equation 5.4). Colors $\bar{a}_{q,c}$ are taken as candidates from which outliers are removed by iteratively computing the mean color and removing the largest outlier (in Euclidean distance). In our experiments, removing 50% of the candidates leads to consistently clean color estimates, as shown in Figures 5.4 and 5.5.

5.4. Evaluation

11 sequences indoor and outdoor

We evaluate our method on 11 sequences of publicly available datasets with large variety, both indoors and outdoors, and show comparisons to state-of-the-art methods, details on the dataset are summarized in Table 5.1. The quality of pose and shape reconstruction is best assessed in the supplemental video published alongside [Rhodin et al., 2016b], where we also apply and compare our reconstructions to tracking with the volumetric Gaussian representations of Chapter 4 [Rhodin et al., 2015a] and Stoll et al. [2011].

5.4.1. Robustness in general scenes

Outdoor

We validate the robustness of our method on three outdoor sequences. On the `Walk`

Table 5.1: List of sequences used in this chapter and their characteristics. 'Input resolution' refers to the resolution used by our algorithm, not the original video resolution. Runtime is measured in minutes.

Sequence	Source	# Cameras	# Frames	Motion	Environment	Ground truth	Input resolution	Stage I runtime	Stage II runtime
`Walk`	Elhayek et al. [2015]	6	100	walking	outdoor, moving background, ambiguous color	—	320×180	123	24
`Cathedral`	Kim and Hilton [2014]	4	20	runing, falling	outdoor	—	240×140	32	5
`Subject3`	Our	3	100	volleyball	outdoor, cluttered background	shape laser scan	180×90	65	15
`Subject2`	Our	6	100	gymnastics	studio, few colors	shape laser scan	200×160	62	15
`Subject1`	Our	6	100	gymnastics	studio, few colors	shape laser scan	200×160	68	18
`HumanEva-Walk`	Sigal et al. [2010]	3	586	walking	studio, low-quality image	markers, manual silhouettes	160×120	176	42
`HumanEva-Box`	Sigal et al. [2010]	3	382	boxing	studio, low-quality image	markers, manual silhouettes	160×120	113	65
`Marker`	Elhayek et al. [2015]	2	100	walking	studio	markers, joint positions	128×128	22	8
`Skirt`	Gall et al. [2009]	6	100	dancing	studio, green screen	—	128×128	97	28
`Monocular`	Guan et al. [2009]	1	3×1	posing	studio, segmented	multi-view reconstruction	125×125	—	0.3
`Studio`	Stoll et al. [2011]	10	4 (300 tracking)	gymnastics, walking	studio	—	162×121	14	1.5

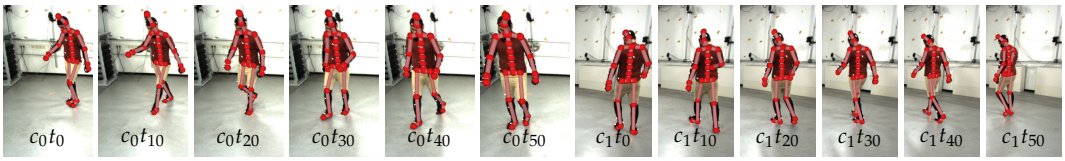


Figure 5.6: We obtained accurate results even using only two views on the `Marker` sequence.

dataset [Elhayek et al., 2015], people move in the background, and background and foreground color are very similar. Our method is nevertheless able to accurately estimate shape and pose across 100 frames from 6 views, see Figure 5.1. We also qualitatively compare against the recent model-free method of Mustafa et al. [2015]. On the `Cathedral` sequence of Kim and Hilton [2014], they achieve rough surface reconstruction using 8 cameras without the explicit need for silhouettes; in contrast, 4 views and 20 frames are sufficient for us to reconstruct shape and pose of a quick outdoor run, see Figure 5.4 (top). Furthermore, we demonstrate reconstruction of complex motions on `Subject3` during a two-person volleyball play from only 3 views and 100 frames, see Figure 5.4 (bottom). The second player was segmented out during Stage I, but Stage II was executed automatically. Fully-automatic model and pose estimation are even possible from only two views as we demonstrate on the `Marker` sequence [Elhayek et al., 2015], see Figure 5.6.

5.4.2. Shape estimation accuracy

Shape accuracy metrics

To assess the accuracy of the estimated actor models, we tested our method on a variety of subjects performing general motions such as walking, kicking and gymnastics. Evaluation of estimated shape is performed in two ways: 1) the estimated body shape is compared against ground-truth measurements, and 2) the 3D mesh derived from Stage II is projected from the captured camera viewpoints

(a) `Subject1` **(b)** `Subject2` **(c)** `Subject3`

Figure 5.7: Visual comparison of estimated body shapes at the different stages. In each subfigure (from left to right): mean PCA $(\bar{\gamma}, \bar{\mathbf{b}})$, Stage I, Stage II and ground-truth shape, respectively.

to compute the overlap with a manually segmented foreground. We introduce two datasets `Subject1` and `Subject2`, in addition to `Subject3`, with pronounced body proportions and ground-truth laser scans for quantitative evaluation. Please note that shape estimates are constant across the sequence and can be evaluated at sparse frames, while pose varies and is separately evaluated in the following section.

Body measures

The shape accuracy is evaluated by measurements of chest, waist and hip circumference. `Subject1` and `Subject2` are captured indoor and are processed using 6 cameras and 40 frames uniformly sampled over 200 frames. `Subject3` is an outdoor sequence and only 3 camera views are used, see Figure 5.4. All subjects are reconstructed with high quality in shape, skeleton dimensions, and color, despite inaccurately estimated poses in Stage I for some frames. We only observed little variation dependent on the performed motions, i.e., a simple walking motion is sufficient, but bone length estimation degrades if joints are not sufficiently articulated during performance. All estimates are presented quantitatively in Table 5.2 and qualitatively in Figure 5.7. In addition, we compare against Guan et al. [2009] on their single-camera and single-frame datasets `Pose1`, `Pose2` and `Pose3`. Stage I requires multi-view input and was not used; instead, we manually initialized the pose roughly, as shown in Figure 5.8, and body height is normalized to 185 cm [Guan et al., 2009]. Our reconstructions are within the same error range, demonstrating that Stage II is well suited even for monocular shape and pose refinement. Our reconstruction is accurate overall, with a mean error of only 2.3 ± 1.9 cm, measured across all sequences with known ground truth.

Silhouette overlap

On top of these sparse measurements (chest, waist, and hips), we also evaluate silhouette overlap for sequences `Walk` and `Box` of subject 1 of the publicly available `HumanEva-I` dataset [Sigal et al., 2010], using only 3 cameras. We compute how

Table 5.2: Quantitative evaluation of estimated shapes in different stages and comparison to Guan et al. [2009]. We use three body measures (chest, waist, and hips, as shown on the right) to evaluate predicted body shapes against the ground truth (GT) captured using a laser scan.

	Chest size [cm]				**Waist size** [cm]				**Hip size** [cm]				**Height** [cm]			
	Guan et al. [2009]	Stage I	Stage II	GT	Guan et al. [2009]	Stage I	Stage II	GT	Guan et al. [2009]	Stage I	Stage II	GT	Guan et al. [2009]	Stage I	Stage II	GT
`Pose1`	92.7	–	92.8	92.6	79.6	–	82.5	80.2	–	–	98.0	–	–	–	183.2	185.0
`Pose2`	87.4	–	91.3	91.6	78.5	–	82.1	79.4	–	–	98.9	–	–	–	181.9	185.0
`Pose3`	91.9	–	93.5	91.4	76.9	–	83.2	80.3	–	–	101.3	–	–	–	182.9	185.0
`Subject1`	–	92.7	132.6	131.3	–	76.7	127.1	132.7	–	108.2	135.4	136.1	–	187.5	194.2	195.0
`Subject2`	–	92.3	99.3	100.1	–	77.3	90.6	96.5	–	92.5	102.3	99.7	–	168.5	162.5	162.0

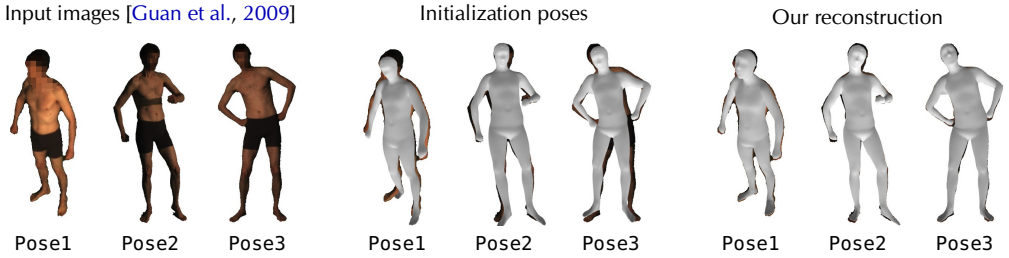


Figure 5.8: Monocular reconstruction experiment. Our reconstruction (right) shows high-quality contour alignment, and improved pose and shape estimates.

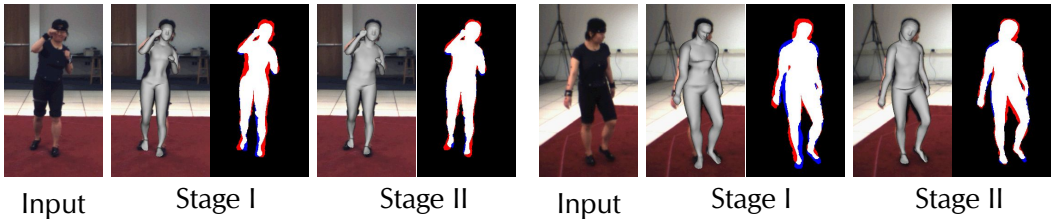


Figure 5.9: Overlap of the estimated shape in Stages I and II for an input frame of `Box` (left) and `Walk` (right) sequences [Sigal et al., 2010]. Note how the white area (correct estimated shape) significantly increases between Stage I and II, while blue (overestimation) and red (underestimation) areas decrease.

much the predicted body shape overlaps the actual foreground (*precision*) and how much of the foreground is overlapped by the model (*recall*). Despite the low number of cameras, low-quality images, and without requiring background subtraction, our reconstructions are accurate with 95% precision and 85% recall, and improve slightly on the results of Bălan et al. [2007]. Results are presented in Figure 5.9 and Table 5.3. Note that Stage II significantly improves shape estimation.

5.4.3. Pose estimation accuracy

Pose accuracy

Pose estimation accuracy is quantitatively evaluated on the public `HumanEva-I` dataset, where ground-truth data is available, see Table 5.4. We tested the method on the designated validation sequences `Walk` and `Box` of subject `S1`. Reconstruction quality is measured as the average Euclidean distance of estimated and ground-truth joint locations, frames with ground-truth inaccuracies are excluded by the provided scripts.

Table 5.3: Quantitative evaluation of Figure 5.9. See Section 5.4 for definitions of *Precision* and *Recall*.

	Precision	**Recall**
`Walk` Stage I	87.43%	87.25%
`Walk` Stage II	95.18%	86.89%
`Box` Stage I	93.26%	81.11%
`Box` Stage II	95.42%	85.28%

Table 5.4: Pose estimation accuracy measured in mm on the `HumanEva-I` dataset. The standard deviation is reported in parentheses.

Seq.	Trained on	Our	Amin et al. [2013]	Sigal et al. [2012]	Belagiannis et al. [2014]	Elhayek et al. [2015]
S1,	general	74.9 (21.9)	—	66	68.3	66.5
`Walk`	HumanEva	54.6 (24.2)	54.5	—	—	—
S2,	general	59.7 (15.0)	—	—	62.7	60.0
`Box`	HumanEva	35.1 (19.0)	47.7	—	—	—

On par with manual initialization

Our pose estimation results are on par with state-of-the-art methods with 6–7 cm average accuracy [Sigal et al., 2012; Amin et al., 2013; Belagiannis et al., 2014; Elhayek et al., 2015]. In particular, we obtain comparable results to Elhayek et al. [2015], which however requires a separately initialized actor model. Please note that Amin et al. [2013] specifically trained their model on manually annotated sequences of the same subject in the same room. For best tracking performance, the ideal joint placement and bone lengths of the virtual skeleton may deviate from the real human anatomy, and may generally vary for different tracking approaches. To compensate differences in the skeleton structure, we also report results where the offset between ground truth and estimated joint locations is estimated in the first frame and compensated in the remaining frames, reducing the reconstruction error to 3–5 cm. Datasets without ground-truth data cannot be quantitatively evaluated; however, our shape overlap evaluation results suggest that pose estimation is generally accurate. In summary, pose estimation is reliable, with only occasional failures in Stage I, although the main focus of our work is on the combination with shape estimation.

5.4.4. Automatic vs. manual actor model

Automatic vs. manual model dimensions

Our reconstructed actor model provides the Gaussian parameters and underlying skeleton dimensions. We tested our model's applicability to the volumetric Gaussian representations proposed by Stoll et al. [2011] and Rhodin et al. [2015a] from Chapter 4 on the `Marker` and `Walking` sequences, with 3 and 10 cameras, respectively. We found that our automatically generated model matches their manually initialized and hand-crafted body dimensions, see Figure 5.10.

Suitable for tracking

We additionally tested the model quality by tracking the same sequence once with the automatically estimated body model and once with the original, manually created models. The overall tracking performance of our actor model was equivalent to method in Chapter 4, and improved on Stoll et al.'s model.

5.4.5. Body shape space generalization

Alternative shape models

We qualitatively assess the generalization capability of our body shape model by comparing against representing meshes directly as a vector of vertex positions [Allen et al., 2003], and using per-triangle rotation and shear with respect to a rest shape, similar to SCAPE [Anguelov et al., 2005]. For each database mesh instance, we build a combined feature vector by stacking $(\gamma_i, \mathbf{b}_i)$, vertex positions $\mathbf{v_i}$, and per-triangle shear $\mathbf{a_i}$ into a single vector, as explained in Section 2.2.3. We perform PCA on the combined features, which generates principal vectors that jointly express the

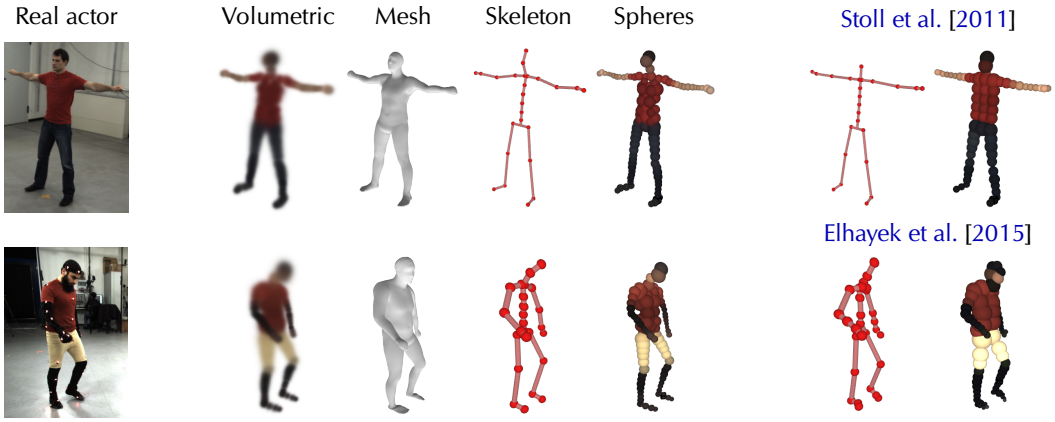


Figure 5.10: Comparison of the estimated actor models for the `Walk` sequence (top) and `Marker` sequence (bottom) to the manually created skeletons of Stoll et al. [2011] and Elhayek et al. [2015]. For comparison between methods, we represent Gaussians as spheres with radius equal to one standard deviation.

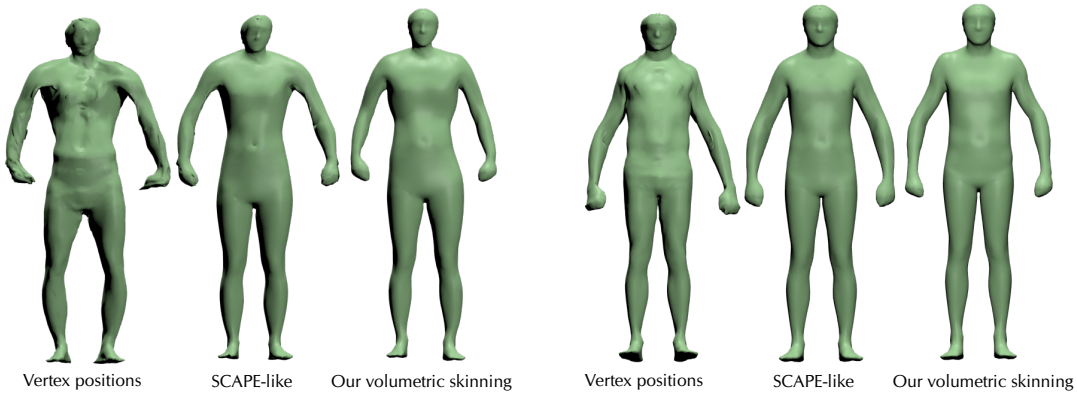


Figure 5.11: Comparison of PCA body shape spaces: vertex positions, SCAPE-like per-triangle transformations, and our volumetric skinning. Our volumetric skinning is computationally as efficient as vertex encodings, and yields comparable shape generalization to the SCAPE-like method.

variation in all three representations. To test generalization capability, we explore different PCA coefficients and analyze the mesh predicted by each representation.

Efficient and robust

Our volumetric skinning is computationally more efficient than per-triangle encodings, like the SCAPE model [Anguelov et al., 2005], while still yielding comparable shape generalization, see Figure 5.11. Per-triangle deformations only encode the mesh vertex positions implicitly and surface mesh reconstruction requires solving a linear system. Direct encoding in terms of vertex positions is as efficient as our volumetric skinning—in both cases, vertex positions depend linearly on the PCA coefficients. However, it exhibits stronger artefacts, see Figure 5.11. We believe this is due to the coupling of each vertex to neighbouring Gaussians in our model, which introduces an implicit spatial smoothing regularization. Each vertex is influenced by multiple Gaussians, and each Gaussian was registered based on all neighbouring vertices, which compensates for inaccuracies in the mesh registration.

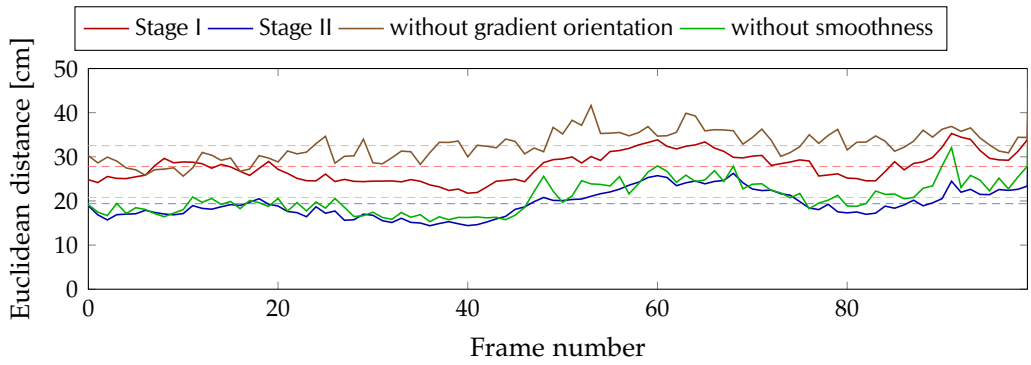


Figure 5.12: Model component influence evaluation. All model components are important: stage II consistently improves on stage I; smoothness term E_{smooth} removes temporal jitter; and without contour direction in E_{contour} the reconstruction error doubled.

5.4.6. Model components

Importance of terms

We quantitatively assess the contribution of each of our model components by comparing estimated poses to the marker-based ground truth of the `Marker` sequence of Elhayek et al. [2015].[2] We execute all algorithm variants on 3 camera views and 100 frames, the mean Euclidean joint error is plotted in Figure 5.12. Stage II consistently improves the initial estimates of Stage I. Without the smoothness term E_{smooth}, temporal jitter emerged. Disregarding contour direction and image gradient direction in E_{contour} results in doubling the reconstruction error, which indicates that the integration of contour direction is crucial for the success of the proposed algorithm. In our experiments E_{smooth} is weighted by 0.01 and E_{flat} by 0.05.

2 vs. 3 cameras

The influence of using only two or three cameras is analyzed on the same sequence, see Figure 5.13. Automatic reconstruction with three cameras is as accurate as tracking with the handcrafted model and method presented in Chapter 4 [Rhodin et al., 2015a]. Pose reconstruction from only two cameras is still accurate for large parts of the sequence, and sometimes more accurate than tracking with two cameras and a manual actor model. Dramatic errors occur only occasionally in the second half of the sequence. Shape estimation nevertheless succeeds due to the robustness provided by the underlaying parametric model. Please note that our skeleton model has a slightly different structure than the ground-truth skeleton. To compensate, we computed joint position offsets in the first frames and propagates these across the whole sequence. However, some differences remain, which likely explains some of the error.

5.4.7. Runtime

Efficient space-time optimization

In our experiments, runtime scaled linearly with the number of cameras and frames. Contour-based shape optimization is efficient: it only takes 3 seconds per view,

[2] The reported numbers of all listed approaches are corrected by a factor of two compared to the respective original publications, correcting a mistake in the camera calibration of the test sequence introduced in [Elhayek et al., 2015].

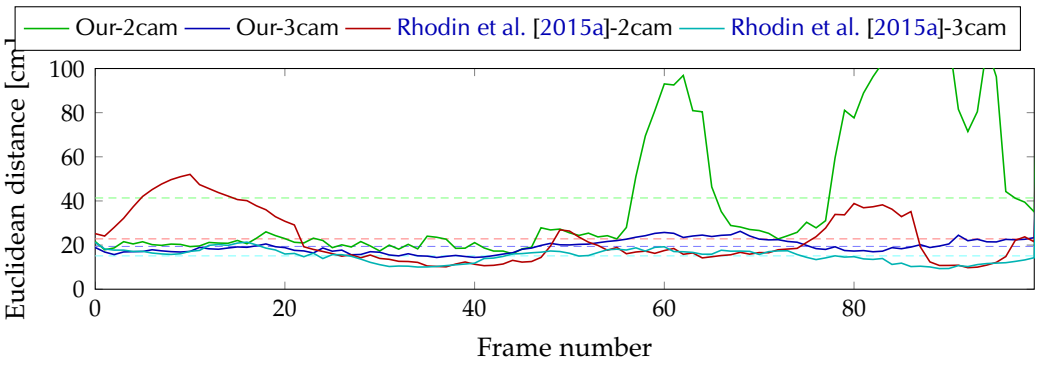


Figure 5.13: Influence of the number of camera views. Using three cameras, our reconstruction is as accurate as tracking with the manual model. It is still accurate for two cameras for large parts of the sequence.

totaling 15 minutes for 50 frames and 6 views on a standard desktop machine. Skeleton pose estimation is not the main focus of this work and the code is not optimized; it takes 10 seconds per frame and view, totaling 50 minutes.

5.5. Discussion and limitations

Clothing

Even though the body model was learned from tight clothing scans, our approach handles general apparel well, correctly reconstructing the overall shape and body dimensions. We demonstrate that even if not all assumptions are fulfilled, our method produces acceptable results, such as for the dance performance `Skirt` of Gall et al. [2009] in Figure 5.5 (top left) that features a skirt. However, our method was not designed to accurately reconstruct fine wrinkles, facial details, hand articulation, or highly non-rigid clothing.

Surface refinement

Methods for high-detail surface refinement exist for controlled environments, but challenges remain in general scenes. We developed surface refinement in outdoor scenes with uncontrolled background [Robertini et al., 2016]. It works without background subtraction, but requires a textured actor template for initialization. The template could be provided by the method proposed in this chapter, only automatic and detailed texturing of the model remains open.

Monocular?

We demonstrate fully automatic reconstructions from as few as two cameras and semi-automatic shape estimation using a single image. Fully automatic pose and shape estimates from a single image remains difficult, but becomes conceivable with the recent advance of 3D pose estimation from monocular images [Tekin et al., 2016b] and coarse shape regression [Bogo et al., 2016] which could be used to improve our stage I and to initialize stage II.

5.6. Summary

Automatic shape through a volumetric contour model

We proposed a fully automatic approach for estimating the shape and pose of a rigged actor model from general multi-view video input with just a few cameras. The method is robust to moving background, general motions can be reconstructed, and succeeds with uncontrolled illumination. It is the first approach that reasons

about contours within sum-of-Gaussians representations and which transfers their beneficial properties, such as analytic form and smoothness, and differentiable visibility, to the domain of edge- and silhouette-based shape estimation. This results in an analytic volumetric contour alignment energy that efficiently and fully automatically optimizes the pose and shape parameters. Based on a new statistical body model, our approach reconstructs a personalized kinematic skeleton, a volumetric Gaussian density representation with appearance modeling, a surface mesh, and the time-varying poses of an actor.

Indoor and outdoor

We demonstrated shape estimation and motion capture results on challenging datasets, indoors and outdoors, captured with very few cameras. This is an important step towards making motion capture more practical.

Calibration is required

A remaining limitation of multi-view pose and shape estimation methods is the burden of installing and calibrating cameras before recording and offline reconstruction. Moreover fixed camera placements do not scale to recordings in vast scenes, it would require hundreds of cameras for sufficient coverage. To overcome these limitations, the next chapter proposes an egocentric pre-calibrated camera rig and a dedicated algorithm that enables full-body motion capture in vast as well as very confined recording volumes. Using the automatic actor model initialization of this chapter, it offers motion-capture technology for consumers—easy to use hard- and software that operates in general casual environments.

Egocentric marker-less motion capture with two fisheye cameras

6

This chapter is based on Rhodin et al. [2016a].

Outside-in

Traditional optical skeletal motion-capture methods – both marker-based and marker-less – use several cameras typically placed around a scene in an *outside-in* arrangement, with camera views approximately converging in the center of a confined recording volume. This greatly constrains the spatial extent of motions that can be recorded; simply enlarging the recording volume by using more cameras, for instance to capture an athlete, is not scalable. Outside-in arrangements also constrain the type of scene that can be recorded, even if it fits into a confined space. If a recording location is too small, cameras can often not be placed sufficiently far away. In other cases, a scene may be cluttered with objects or furniture, or other dynamic scene elements, such as people in close interaction, may obstruct a motion-captured person in the scene or create unwanted dynamics in the background. In such cases, even state-of-the-art outside-in marker-less optical methods that succeed with just a few cameras and are designed for less controlled and outdoor scenes, such as the ones introduced in the preceding chapters, quickly fail. Scenes with dense social interaction were previously captured with outside-in camera arrays of a few hundred sensors [Joo et al., 2015], a very complex and hardly scalable setup.

Inside-in suits

Inside-out

These strong constraints on recording volume and scene density prevent the use of optical motion capture in the majority of real-world scenes. This problem can partly be bypassed with motion-capture methods that use body-worn sensors, such as the Xsens MVN inertial measurement unit suit. However, the special suit and cabling are obstructive and require tedious calibration. Shiratori et al. [2011] propose to wear 16 cameras placed on body parts facing *inside-out*, and capture the skeletal motion through structure-from-motion relative to the environment. This clever solution requires instrumentation, calibration, and a static background, but enables free roaming. This design was inspirational for our egocentric approach.

EgoCap: optical inside-in

We propose egocentric motion capture (EgoCap), an approach that estimates full-body pose from a pair of optical cameras carried by lightweight headgear (see Figure 6.1). The body-worn cameras are rigidly attached to the user and simultaneously record the user's motion, thus forming an *inside-in* camera arrangement, which overcomes many limitations of previous methods. It reduces the setup effort, enables free roaming, and minimizes body instrumentation. EgoCap decouples the estimation of local body pose with respect to the headgear cameras and

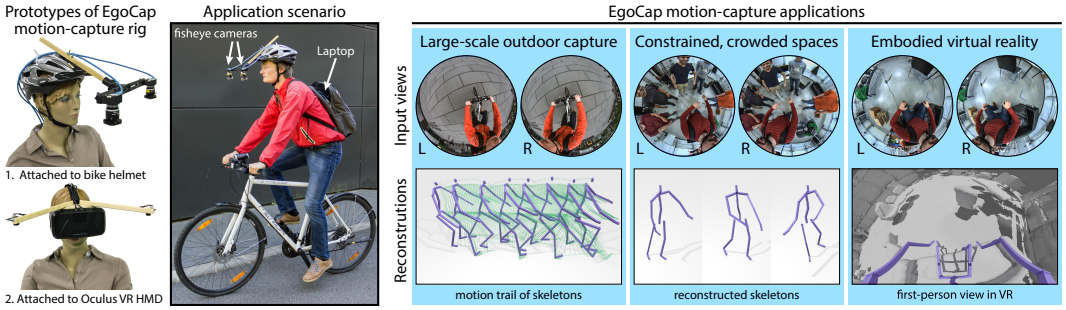


Figure 6.1: We propose a marker-less optical motion-capture approach that only uses two head-mounted fisheye cameras (see rigs on the left). Our approach enables three new application scenarios: 1) capturing human motions in outdoor environments of virtually unlimited size, 2) capturing motions in space-constrained environments, e.g. during social interactions, and 3) rendering the reconstruction of one's real body in virtual reality for embodied immersion.

global headgear position, which we infer by structure-from-motion on the scene background.

Fisheye gear

Our first contribution is a new egocentric inside-in sensor rig with only two head-mounted, downward-facing commodity video cameras with fisheye lenses (see Figure 6.1). The rig can be attached to a helmet or a head-mounted VR display, and, hence, requires less instrumentation and calibration than other body-worn systems. The stereo fisheye optics keep the whole body in view in all poses, despite the cameras' proximity to the body. We prefer conventional video cameras over IR-based RGB-D cameras, which were for example used for egocentric hand tracking [Sridhar et al., 2015], since the latter fail outdoors. Video cameras work indoors and outdoors, also have lower energy consumption, and are easily fitted with the required fisheye optics.

Generative and discriminative

Our second contribution is a new marker-less motion-capture algorithm tailored to the strongly distorted egocentric fisheye views. It combines a generative model-based skeletal pose estimation approach (Chapter 4 and Section 6.2) with evidence from a trained ConvNet-based body-part detector (Section 6.2.2). The approach features an analytically differentiable objective energy that can be minimized efficiently, is designed to work with unsegmented frames and general backgrounds, succeeds even on poses exhibiting notable self-occlusions (e.g. when walking), as the part detector predicts occluded parts, and enables recovery from tracking errors after severe occlusions.

Automatic database creation

Our third contribution is a new approach for automatically creating body-part detection training datasets. We record test subjects in front of green screen with an existing outside-in marker-less motion-capture system to get ground-truth skeletal poses, which are reprojected into the simultaneously recorded head-mounted fisheye views to get 2D body-part annotations. We augment the training images by replacing the green screen with random background images, and vary the appearance in terms of color and shading by intrinsic recoloring [Meka et al., 2016]. With this technique, we annotate a total of 100,000 egocentric images of eight people in different clothing (Section 6.2.2), with 75,000 images from six people used for training. We publish the dataset for research purposes [EgoCap, 2016].

Prototypes for crowded and large-scale scenes

We designed and extensively tested two system prototypes featuring (1) cameras fitted to a bike helmet, and (2) small cameras attached to an Oculus Rift headset. We show reliable egocentric motion capture, both offline and in real time. The egocentric tracking meets the accuracy of outside-in approaches using 2–3 cameras; additional advances are necessary to match the accuracy of many-camera systems. Nevertheless, we succeed in scenes that are challenging for outside-in approaches, such as close interaction with many people, as well outdoor and indoor scenes in cluttered environments with frequent occlusions, for example when working in a kitchen or at a desk. We also show successful capturing in large volumes, for example of the skeletal motion of a cyclist. The lightweight Oculus Rift gear is designed for egocentric motion capture for virtual reality, where the user can move in the real world to roam and interact in a virtual environment seen through a head-mounted display, while perceiving increased immersion thanks to the rendering of the motion-captured body, which is not obtained with current HMD head pose tracking.

Application to first-person vision

The proposed full-body tracking could also be used as an intermediate representation for recognition in the field of first-person vision, where body worn cameras are used for activity recognition [e.g. Fathi et al., 2011; Kitani et al., 2011; Ohnishi et al., 2016; Ma et al., 2016], for learning engagement and saliency patterns of users when interacting with the real world [e.g. Park et al., 2012; Su and Grauman, 2016], and for understanding the utility of surrounding objects [Rhinehart and Kitani, 2016].

6.1. Egocentric camera design

Lightweight pre-calibrated rig

We designed a mobile egocentric camera setup to enable human motion capture within a virtually unlimited recording volume. We attach two fisheye cameras rigidly to a helmet or VR headset, such that their field of view captures the user's full body, see Figure 6.2. The wide field of view allows us to observe interactions in front and beside the user, irrespective of their global motion and head orientation, and without requiring additional sensors or suits. The stereo setup ensures that most actions are observed by at least one camera, despite substantial self-occlusions of arms, torso and legs in such an egocentric setup. A baseline of 30–40 cm proved to be best in our experiments. The impact of the headgear on the user's motion is limited as it is lightweight: our prototype camera rig for VR headsets (see Figure 6.1, bottom left) only adds about 65 grams of weight.

6.2. Egocentric inside-in motion capture

Local and global pose

Our egocentric setup separates human motion capture into two subproblems: 1) local skeleton pose estimation with respect to the camera rig, and 2) global rig pose estimation relative to the environment. Global pose is estimated with existing structure-from-motion techniques, see Section 6.4.3. We formulate skeletal pose estimation as an analysis-by-synthesis-style optimization problem in the pose parameters $\mathbf{p}^t$, that maximizes the alignment of a projected 3D human body model with the human in the left $\mathbf{I}^t_{\text{left}}$ and the right $\mathbf{I}^t_{\text{right}}$ stereo fisheye views, at each video time step t. We use a hybrid alignment energy combining evidence from

a generative image-formation model, as well as from a discriminative detection approach.

Top-down and bottom-up

For the generative component we adapt the volumetric ray-tracing model introduced in Chapters 4 and 5. The existing model facilitates generative pose estimation with only two cameras, and we adapt it in Section 6.2.1 to the strongly distorted fisheye views. Our energy also employs constraints from one-shot joint-location predictions in the form of $E_{\text{detection}}$. These predictions are found with a new ConvNet-based 2D joint detector for head-mounted fisheye views, which is learned from a large corpus of annotated training data, and which generalizes to different users and cluttered scenes (Section 6.2.2). The combined energy that we optimize takes the following form:

$$E(\mathbf{p}^t) = E_{\text{color}}(\mathbf{p}^t) + E_{\text{detection}}(\mathbf{p}^t) + E_{\text{pose}}(\mathbf{p}^t) + E_{\text{smooth}}(\mathbf{p}^t). \tag{6.1}$$

Here, $E_{\text{pose}}(\mathbf{p}^t)$ is a regularizer that penalizes violations of anatomical joint-angle limits as well as poses deviating strongly from the rest pose ($\mathbf{p} = \mathbf{0}$):

$$\begin{aligned} E_{\text{pose}}(\mathbf{p}^t) = \lambda_{\text{limit}} \cdot \Big(&\max(0, \mathbf{p}^t - \mathbf{l}_{\text{upper}})^2 + \max(0, \mathbf{l}_{\text{lower}} - \mathbf{p}^t)^2 \Big) \\ &+ \lambda_{\text{pose}} \cdot \text{huber}(\mathbf{p}^t), \end{aligned} \tag{6.2}$$

where $\mathbf{l}_{\text{lower}}$ and $\mathbf{l}_{\text{upper}}$ are lower and upper joint-angle limits, and $\text{huber}(x) = \sqrt{1+x^2} - 1$ is the Pseudo-Huber loss function. $E_{\text{smooth}}(\mathbf{p}^t)$ is a temporal smoothness term:

$$E_{\text{smooth}}(\mathbf{p}^t) = \lambda_{\text{smooth}} \cdot \text{huber}(\mathbf{p}^{t-1} + \zeta(\mathbf{p}^{t-1} - \mathbf{p}^{t-2}) - \mathbf{p}^t), \tag{6.3}$$

where $\zeta = 0.25$ is a damping factor. The total energy in Equation 6.1 is optimized for every frame, as described in Section 6.2.3. In the following, we describe the generative and discriminative terms in more detail, while omitting the temporal dependency t in the notation for better readability. We use weights $\lambda_{\text{pose}} = 10^{-4}$, $\lambda_{\text{limit}} = 0.1$ and $\lambda_{\text{smooth}} = 0.1$.

6.2.1. Egocentric ray-casting model

Top-down

We adapt the volumetric ray-tracing model introduced in Chapters 4 and 5, because it has several key advantages over previous generative models for image-based pose estimation. It enables analytic derivatives of the pose energy, including a smooth analytically differentiable visibility model everywhere in pose space. This makes it perform well with only a few camera views. Furthermore, it handles non-stationary backgrounds and occlusions well, a requirement for the egocentric tracking setting. However, it applies only to static cameras, does not support the distortion of fisheye lenses, and it does not run in real time.

Root at head

In our egocentric camera rig, the cameras move rigidly with the user's head. In contrast to commonly used skeleton configurations, where the hip is taken as the root joint, our skeleton hierarchy is rooted at the head. Like a puppet, the lower body parts are then relative to the head motion, see Figure 6.2. This formulation factors out the user's global motion, which can be estimated independently, see

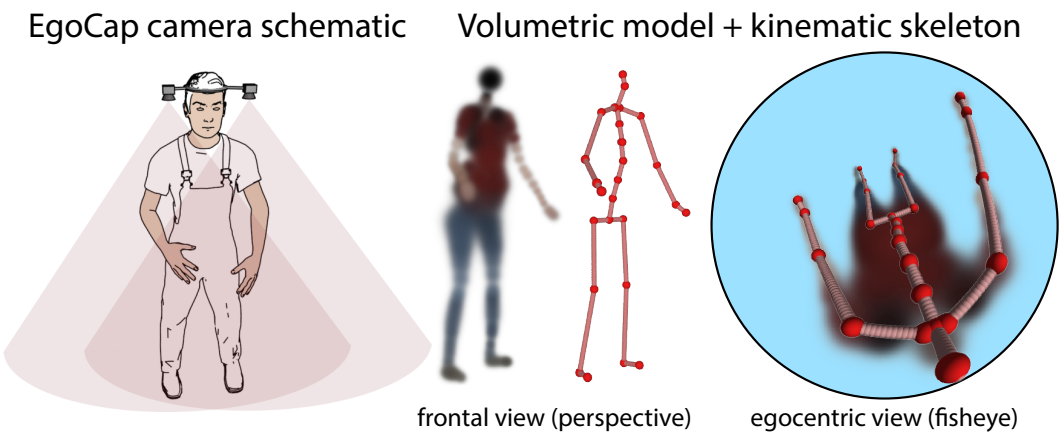


Figure 6.2: Schematic of EgoCap, our egocentric motion-capture rig (left), visualization of the corresponding volumetric body model and kinematic skeleton (center), and the egocentric view of both in our head-mounted fisheye cameras (right).

Section 6.4.3, and reduces the dimensionality of the pose estimation by 6 degrees of freedom. By attaching the cameras to the skeleton root, the movable cameras are reduced to a static camera formulation, as assumed in the previous chapters.

Fisheye camera model

Simply undistorting the fisheye images before optimization is impractical as resolution at the image center reduces and pinhole cameras cannot capture fields of view approaching 180 degrees – their image planes would need to be infinitely large. To apply the ray-casting formulation described in Chapter 4 to our egocentric motion-capture rig, with its 180° field of view, we replace the original pinhole camera model with the omnidirectional camera model of Scaramuzza et al. [2006]. The ray direction $\mathbf{n}(u,v)$ of a pixel (u,v) is then given by $\mathbf{n}(u,v) = [u,v,f(\rho)]^\top$, where f is a polynomial of the distance ρ of (u,v) to the estimated image center. We apply the energy term $\mathrm{D_{mc}}$ defined in Equation 4.13 (Chapter 4, Section 4.4) to both cameras of our egocentric camera rig using

$$E_{\text{color}}(\mathbf{p}) = \mathrm{D_{mc}}(\mathbf{p}, \mathbf{I}_{\text{left}}) + \mathrm{D_{mc}}(\mathbf{p}, \mathbf{I}_{\text{right}}). \tag{6.4}$$

These extensions also generalize the contour model of Rhodin et al. [2016b] and Chapter 5 to enable egocentric body model initialization.

Color similarity metric

Color dissimilarity For measuring the dissimilarity $d(\mathbf{m},\mathbf{i})$ of model color $\mathbf{m}$ and image pixel color $\mathbf{i}$ in Equation 4.13, we use the HSV color space (with all dimensions normalized to unit range) and combine three dissimilarity components:

1. For saturated colors, the color dissimilarity d_s is computed using the squared (minimum angular) hue distance. Using the hue channel alone is less sensitive to illumination changes.
2. For dark colors, the color dissimilarity d_d is computed as twice the squared value difference, i.e., $d_d(\mathbf{m},\mathbf{i}) = 2(m_v - i_v)^2$. Hue and saturation are ignored as they are unreliable for dark colors.
3. For gray colors, the distance d_g is computed as the sum of absolute value and saturation difference, i.e., $d_g(\mathbf{m},\mathbf{i}) = |m_v - i_v| + |m_s - i_s|$. Hue is unreliable and

thus ignored.

We weight these three dissimilarity components by $w_s = \sqrt{m_s}/Z$, $w_d = \max(0, 0.5 - m_v)/Z$ and $w_g = \max(0, 0.5 - m_s)/Z$ respectively, where Z normalizes the sum of these weights to unity. The total dissimilarity is computed by $d(\mathbf{m}, \mathbf{i}) = \phi(w_s d_s + w_d d_d + w_g w_g)$ where $\phi(x) = 1 - (1-x)^4(8x+2)$ is a smooth step function. We employ a two-sided energy, i.e., E_{color} can be negative: For dissimilar colors, $d \approx 1$ and approaches -1 for similar colors.

6.2.2. Egocentric body-part detection

Bottom-up detections

We combine the generative model-based alignment from the previous section with evidence from the discriminative joint-location detector of Insafutdinov et al. [2016], trained on annotated egocentric fisheye images. The discriminative component dramatically improves the quality and stability of reconstructed poses, provides efficient recovery from tracking failures, and enables plausible tracking even under notable self-occlusions. To apply Insafutdinov et al.'s body-part detector, which has shown state-of-the-art results on human pose estimation from outside-in RGB images, to the top-down perspective and fisheye distortion of our novel egocentric camera setup, the largest burden is to gather and annotate a training dataset that is sufficiently large and varied, containing tens of thousands of images. As our camera rig is novel, there are no existing public datasets, and we therefore designed a method to automatically annotate real fisheye images by outside-in motion capture and to augment appearance with the help of intrinsic image decomposition.

Dataset creation

Annotation by motion capture

We propose a novel approach for semi-automatically creating large, realistic training datasets for body-part detection that comprise tens of thousands of camera images annotated with the joint locations of a kinematic skeleton and other body parts such as the hands and feet. To avoid the tedious and error-prone manual annotation of locations in thousands of images, as in previous work, we use a state-of-the-art marker-less motion-capture system (Captury Studio [Captury]) to estimate the skeleton motion in 3D from eight stationary cameras placed around the scene. We then project the skeleton joints into the fisheye images of our head-mounted camera rig. The projection requires tracking the rigid motion of our head-mounted camera rig relative to the stationary cameras of the motion-capture system, for which we use a large checkerboard rigidly attached to our camera rig (Figure 6.3). We detect the checkerboard in all stationary cameras in which it is visible, and triangulate the 3D positions of its corners to estimate the pose and orientation of the camera rig. Using Scaramuzza et al.'s camera distortion model, we then project the 3D joint locations into the fisheye images recorded by our camera rig.

Augmentation by intrinsic image decomposition

Dataset augmentation We record video sequences of eight subjects performing various motions in a green-screen studio. For the training set, we replace the background of each video frame, using chroma keying, with a random, floor-related image from Flickr, as our fisheye cameras mostly see the ground below the tracked subject. Please note that training with real backgrounds could give the CNN additional context, but is prone to overfitting to a (necessarily) small set of

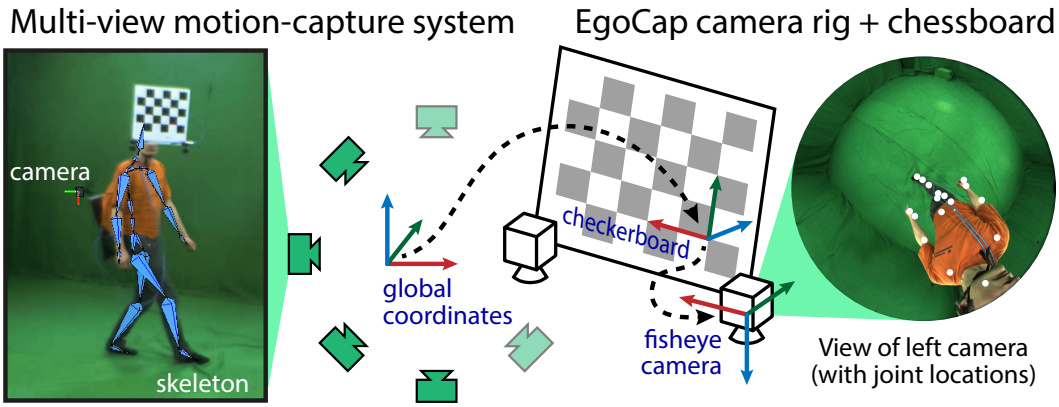


Figure 6.3: For database annotation, the skeleton estimated from the multi-view motion-capture system (left), is converted from global coordinates (center) into each fisheye camera's coordinate system (right) via the checkerboard.

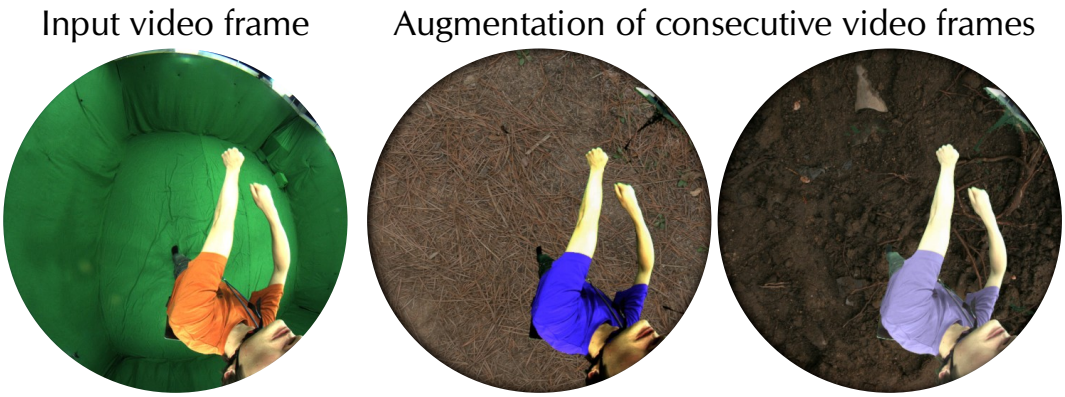


Figure 6.4: Illustration of our dataset augmentation using randomized backgrounds, intrinsic recoloring and gamma jittering. Note the varied shirt colors as well as brightness of the trousers and skin, which help prevent overtraining of the ConvNet-based joint detector.

recorded real backgrounds. In addition, we augment the appearance of subjects by varying the colors of clothing, while preserving shading effects, using intrinsic recoloring [Meka et al., 2016]. This is, to our knowledge, the first application of intrinsic recoloring for augmenting datasets. We also apply a random gamma curve ($\gamma \in [0.5, 2]$) to simulate changing lighting conditions. We furthermore exploit the shared plane of symmetry of our camera rig and the human body to train a single detector on a dataset twice the size by mirroring the images and joint-location annotations of the right-hand camera to match those of the left-hand camera during training, and vice versa during motion capture. Thanks to the augmentation, both background and clothing colors are different for every frame (see Figure 6.4), which prevents overfitting to the limited variety of the captured appearances. This results in a training set of six subjects and ~75,000 annotated fisheye images. Two additional subjects are captured and prepared for validation purposes.

Detector learning

Our starting point for learning an egocentric body-part detector for fisheye images is *ConvNet training*

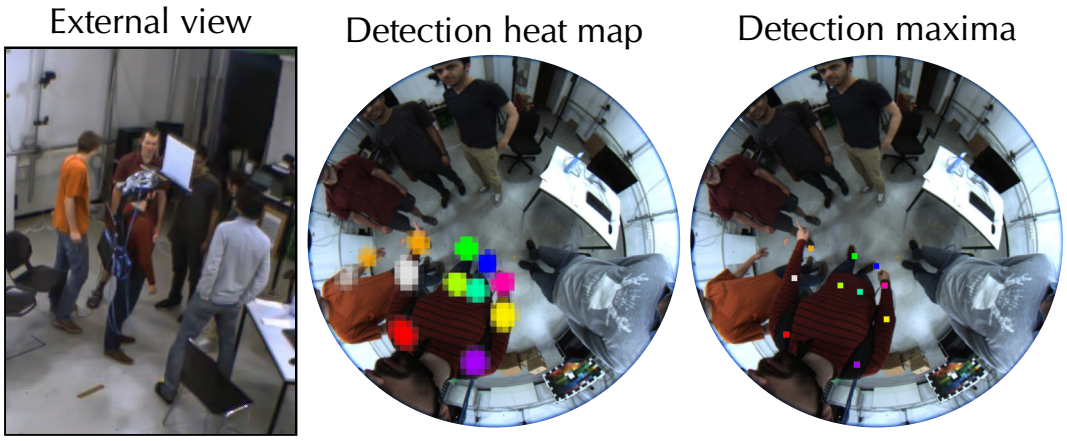


Figure 6.5: Color-coded joint-location detections on the `Crowded` sequence. For crowded scenes (left), detections can be multi-modal (center). However, the maximum (right) lies on the user. We exclude knee, hand, and ankle locations for clearer visualization.

the 101-layer residual network [Hernández-Vela et al., 2016] trained by Insafutdinov et al. [2016] on the MPII Human Pose dataset [Andriluka et al., 2014], which contains ~19,000 internet images that were manually annotated in a crowd-sourced effort, and the Leeds Sports Extended dataset [Johnson and Everingham, 2011] of 10,000 images. We remove the original prediction layers and replace them with ones that output 18 body-part heat maps[3]. The input video frames are scaled to a resolution of 640×512 pixels, the predicted heat maps are of 8× coarser resolution. We then fine-tune the ConvNet on our fisheye dataset for 220,000 iterations with a learning rate of 0.002, and drop it to 0.0002 for 20,000 additional iterations. The number of training iterations is chosen based on performance on the validation set. We randomly scale images during training by up to ±15% to be more robust to variations in user size. Figure 6.5 (center) visualizes the computed heat maps for selected body parts. We demonstrate generalization capability to a large variety of backgrounds, changing illumination, and clothing colors in Section 6.3.3.

Body-part detection energy

Bottom-up to top-down

Inspired by Elhayek et al. [2015], who exploit detections in outside-in motion capture, we integrate the learned detections, in the form of heat maps as shown in Figure 6.5, into the objective energy (Equation 6.1) as a soft constraint. For each detection label, the location with maximum confidence, $(\hat{u}, \hat{v})$, is selected and an associated 3D Gaussian is attached to the corresponding skeleton body part. This association can be thought of as giving a distinct color to each body-part label. The Gaussian is used to compute the spatial agreement of the detection and body-part location in the same way as in the color similarity E_{color}, only the color distance $d(\cdot,\cdot)$ in Equation 4.13 is replaced with the predicted detection confidence at $(\hat{u}, \hat{v})$. For instance, a light green Gaussian is placed at the right knee and is associated with the light green knee detection heat map at $(\hat{u}, \hat{v})$, then their agreement is

[3] We jointly learn heat maps for the head and neck, plus the left and right shoulders, elbows, wrists, hands, hips, knees, ankles and feet.

maximal when the Gaussian's center projects on $(\hat{u}, \hat{v})$. By this definition, $E_{\text{detection}}$ forms the sum over the detection agreements of all body parts and in both cameras, it behaves similar to the detection similarity measure introduced in Section 5.3.1. We weight its influence by $\lambda_{\text{detection}} = 1/3$.

6.2.3. Real-time optimization

Quadratic complexity

The volumetric ray-casting method presented in Chapter 4 models occlusion as a smooth phenomenon by integrating the visibility computations within the objective function instead of applying a depth test once before optimization. While this is beneficial for optimizing disocclusions, it introduces dense pairwise dependencies between all Gaussians: the visibility $\mathcal{V}_q$ (Equation 4.10) of a single Gaussian can be evaluated in linear time in terms of the number of Gaussians, N_q, but E_{color}—and its gradient with respect to all Gaussians—has quadratic complexity in N_q, see Section 4.5.4.

Parallel implementation

To nevertheless reach real-time performance, we introduce a new parallel stochastic optimization approach. The ray-casting formulation enables a natural parallelization of $E_{\text{detection}}$ and E_{color} terms and their gradient computation across pixels (u, v) and Gaussians G_q. We also introduce a traversal step, which determines the Gaussians that are close to each ray, and excludes distant Gaussians with negligible contribution to the energy. These optimizations lead to significant run-time improvements, particularly when executed on a GPU, but only enable interactive frame rates.

Stochastic optimization

We achieve further reductions in run times by introducing a statistical optimization approach that is tailored to the ray-casting framework. The input image pixels are statistically sampled for each gradient iteration step, as proposed by Blanz and Vetter [1999]. In addition, we sample the volumetric body model by excluding Gaussians from the gradient computation at random, individually for each pixel, which improves the optimization time to 10 fps and more.

6.3. Evaluation

6.3.1. Hardware prototypes

Prototype details

We show the two EgoCap prototypes used in this work in Figure 6.1 (left). *EgoRig1* consists of two fisheye cameras attached to a standard bike helmet. It is robust and well-suited for capturing outdoor activities and sports. *EgoRig2* builds on a lightweight wooden rig that holds two consumer cameras and is glued to an Oculus VR headset. It weighs only 65 grams and adds minimal discomfort on the user. Both prototypes are equipped with 180° fisheye lenses and record with a resolution of 1280×1024 pixels at 30 Hz. Note that the checkerboard attached to *EgoRig1* in several images is not used for tracking (only used in training and validation dataset recordings).

Fisheye FOV

Body-Part visibility For egocentric tracking of unconstrained motions, the full 180° field of view is essential for egocentric tracking. We evaluate the visibility of selected body parts from our egocentric rig with different (virtual) field-of-view angles in

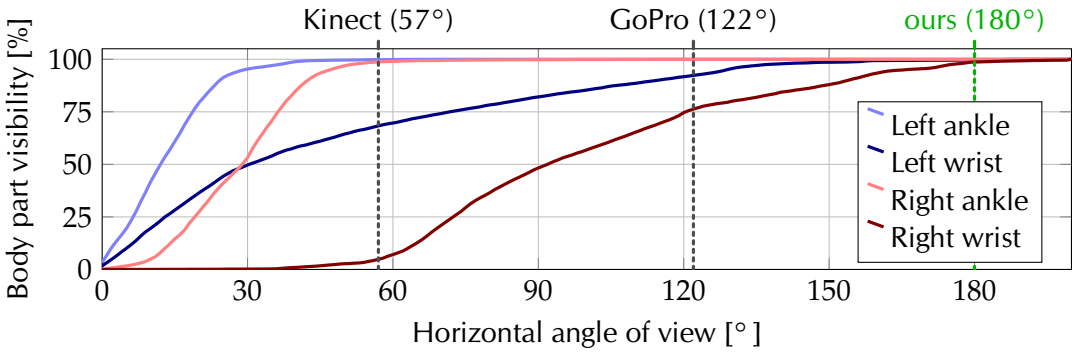


Figure 6.6: Visibility of selected body parts for different camera angles of view, for the left-hand camera in our rig over a 5-minute recording. Seeing the right wrist 95 percent of the time requires an angle of view in excess of 160°, which is only practical with fisheye lenses.

Figure 6.6. Only at 180 degrees are almost all body parts captured, otherwise even small motions of the head can cause the hand to leave the recording volume. The limited field of view of existing active depth sensors of 60–80 degrees restricts their applicability to egocentric motion capture in addition to their higher energy consumption and interference with other light sources.

6.3.2. Runtime

Resolution vs. runtime

For most tracking results, we use a resolution of 128×128 pixels and 200 gradient-descent iterations. Our CPU implementation runs at ten seconds per frame on a Xeon E5-1620 3.6 GHz, which is similar to run times reported in Chapter 4. Straightforward parallelization on the GPU reduces run times to two seconds per frame. The body-part detector runs on a separate machine, and processes 6 images per second on an Nvidia Titan GPU and a Xeon E5-2643 3.30 GHz.

Real-time

For some experiments (see Section 6.4.3), we use a resolution of 120×100 pixels and enable stochastic optimization. Then, purely color-based optimization reaches 10 to 15 fps for 50 gradient iterations (2–3 ms per iteration), i.e., close to real-time performance. Our body-part detector is not optimized for speed and cannot yet run at this frame rate, but its implementation could be optimized for real-time processing, so a real-time end-to-end approach would be feasible without algorithmic changes.

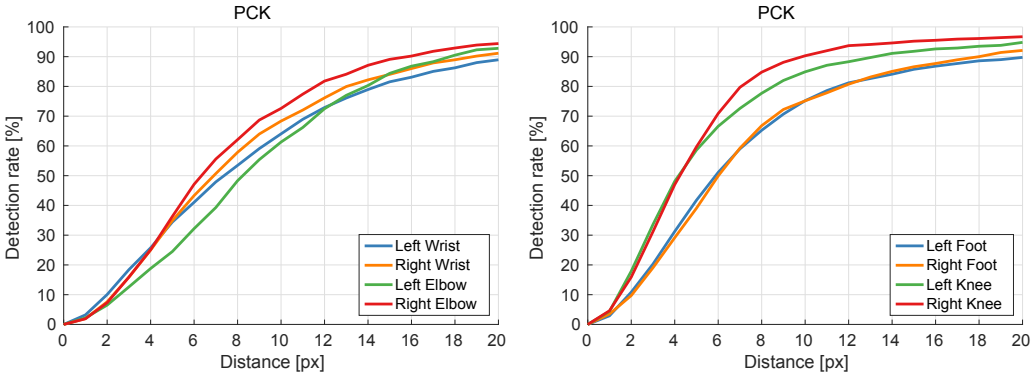
6.3.3. Body-part detections

2D validation set

We first evaluate the learned body-part detectors, irrespective of generative components, using the percentage of correct keypoints (PCK) metric [Sapp and Taskar, 2013; Jain et al., 2014]. We evaluate on a validation set, `Validation2D`, of 1000 images from a 30,000-frame sequence of two subjects that are not part of the training set and wear dissimilar clothing. `Validation2D` is augmented with random backgrounds using the same procedure as for the training set, such that the difficulty of the detection task matches the real-world sequences. We further validated that overfitting to augmentation is minimal, by testing on green-screen background, with equivalent results.

Table 6.1: Part detection accuracy in terms of the percentage of correct keypoints (PCK) on the validation dataset `Validation2D` of 1000 images, evaluated at 20 pixel threshold for three ConvNets trained with different data augmentation strategies (Section 6.2.2). AUC is area under curve evaluated for all thresholds up to 20 pixels.

Training dataset setting	Head	Sho.	Elb.	Wri.	Hip	Knee	Ank.	PCK	AUC
green-screen background	75.5	46.8	18.8	13.6	17.4	7.2	4.5	22.4	10.0
+ background augmentation	84.7	87.5	90.9	89.1	97.7	94.2	86.4	89.5	56.9
+ intrinsic recoloring	**86.2**	**96.1**	**93.6**	**90.1**	**99.1**	**95.8**	**90.9**	**92.5**	**59.4**



(a) Arm joints (b) Leg joints

Figure 6.7: Pose estimation results in terms of percentage of correct keypoints (PCK) for different distance thresholds on `Validation2D`.

Augmentation improves accuracy

Dataset augmentations Table 6.1 presents the evaluation of proposed data augmentation strategies. Background augmentation during training brings a clear improvement. It provides a variety of challenging negative samples for the training of the detector, which is of high importance. Secondly, the performance is further boosted by employing intrinsic video for cloth recoloring, which additionally increases the diversity of training samples. The improvement of about two percent is consistent across all body parts.

High 2D accuracy

Detection accuracy Figure 6.7 contains the plots of PCK at different distance thresholds for arms and legs evaluated on sequence `Validation2D`. We achieve high accuracy, with slightly lower detection reliability of terminal limbs (wrists, feet). This can either be due to more articulation or, in case of the feet, due to higher occlusion by knees and their small appearance due to the strong fisheye distortion.

The 2D detection accuracy of feet and wrists is comparable, even though feet are further away, and similar pixel error hence translates to larger 3D errors, as evaluated in the next section.

Dense sampling helps

We additionally evaluated the training set size. We found that subject variation is important: using only three out of six subjects, the PCK performance dropped by 2.5 percent points. Moreover, using a random subset of 10% of the original database size reduces the PCK by 2 points, i.e., using more than three frames per second is beneficial. Using a 50% subset did not degrade performance, showing that consecutive frames are not crucial for our per-frame model, but could be beneficial for future research, such as for temporal models.

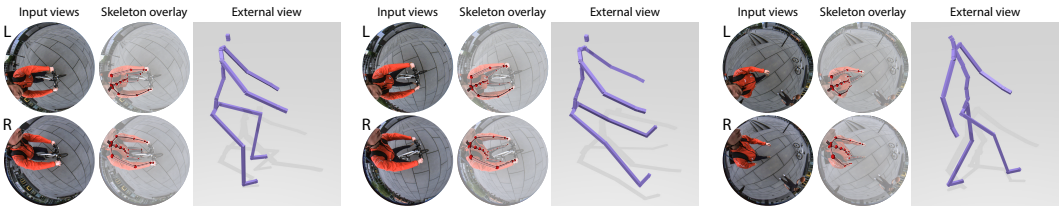


Figure 6.8: EgoCap enables outdoor motion capture with virtually unconstrained extent. Full-body pose is accurately estimated for fast `Biking` (left and center) and for unconstrained `Walk` (right). The model is tailored to handle the present occlusions and strong image distortion.

6.3.4. 3D body pose accuracy

3D accuracy

Our main objective is to infer 3D human pose from the egocentric views, despite occlusions and strong fisheye image distortions. We quantitatively evaluate the 3D body pose accuracy of our approach on two sequences, `ValidationWalk` and `ValidationGest`. Ground-truth data is obtained with the Captury Studio, a state-of-the-art marker-less commercial multi-view solution with eight video cameras and 1–2 cm accuracy. The two systems are used simultaneously and their relative transformation is estimated with a reference checkerboard, see Figure 6.3. We experimented with raw green-screen and with randomly replaced background. Error values are estimated as the average Euclidean 3D distance over 17 joints, including all joints with detection labels, except the head. Reconstructions on green and replaced backgrounds are both 7 ± 1 cm for a challenging 250-frame walking sequence with occlusions, and 7 ± 1 cm on a long sequence of 750 frames of gesturing and interaction.

Lower vs. upper body

During gesturing, where arms are close to the camera, upper body (shoulder, elbow, wrist, finger) joint accuracy is higher than for the lower body (hip, knee, ankle, and toe) with 6 cm and 8 cm average error, respectively. During walking, upper and lower body error is similar with 7 cm. Please note that slight differences in skeleton topology between ground truth and EgoCap exist, which might bias the errors.

Comparable to 3 cam outside-in

Despite the difficult viewing angle and image distortion of our egocentric setup, the overall 3D reconstruction error is comparable to state-of-the-art results of outside-in approaches [Elhayek et al., 2015; Amin et al., 2013; Sigal et al., 2010; Belagiannis et al., 2014; Rhodin et al., 2015a, Chapter 4], which reach 5–7 cm accuracy from two or more cameras, but only in small and open recording volumes, and for static cameras. In contrast, our algorithm scales to very narrow and cluttered scenes (see Figure 6.9) as well as to wide unconstrained performances (see Figure 6.8). No existing algorithm is directly applicable to these conditions and the strong distortions of the fisheye cameras, precluding a direct comparison. Closest to our approach is the fundamentally offline inside-out method of Shiratori et al. [2011], who use 16 body-worn cameras facing outwards, reporting a mean joint position error of 2 cm on a slowly performed indoor walking sequence. Visually, their outdoor results show similar quality to our reconstructions, although we require fewer cameras, and can handle crowded scenes. It depends on the application whether head gear or body-worn cameras less impair the user's performance.

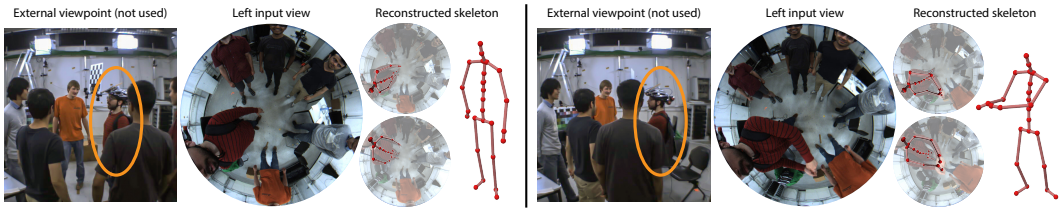


Figure 6.9: Capturing social interaction in crowded scenes is of importance, but occlusions pose difficulties for existing outside-in approaches (left). The egocentric view enables 3D pose estimation, as demonstrated on the `Crowded` sequence. The visible checkerboard is not used.

6.3.5. Model components

Importance of energy terms

Our objective energy consists of detection, color, smoothness, and pose prior terms. Disabling the smoothness term increases the reconstruction error on the validation sequences by 3 cm. Without the color term, accuracy is reduced by 0.5 cm. We demonstrated in the supplemental video of [Rhodin et al., 2016a] that the influence of the color term is more significant in the outdoor sequences for motions that are very dissimilar to the training set. Disabling the detection term removes the ability to recover from tracking failures, which are usually unavoidable for fully automatic motion capture of long sequences with challenging motions. High-frequency noise is filtered with a Gaussian low-pass filter of window size 5.

6.4. Applications

Extensive evaluation

We further evaluate our approach in three application scenarios with seven sequences of lengths of up to 1500 frames using *EgoRig1*, in addition to the three quantitative evaluation sequences. The captured users wear clothes not present in the training set. The qualitative results are best observed in the supplemental video of Rhodin et al. [2016a].

6.4.1. Unconstrained and large-scale motion capture

Vast outdoor space

We captured a `Basketball` sequence outdoors, which shows quick motions, large steps on a steep staircase, and close interaction of arms, legs and the basketball. We also recorded an outdoor `Walk` sequence with frequent arm-leg self-occlusions (Figure 6.8, right). With EgoCap, a user can even motion capture themselves while riding a bike in a larger volume of space (`Bike` sequence, Figure 6.8, left and center). The pedaling motion of the legs is nicely captured, despite frequent self-occlusions; the steering motion of the arms and the torso is also reconstructed. Even for very fast absolute motions, like this one on a bike, our egocentric rig with cameras attached to the body leads to little motion blur, which challenges outside-in optical systems. All this would have been difficult with alternative motion-capture approaches.

Generalization

Note that our outdoor sequences also show the resilience of our method to different appearance and lighting conditions, as well as the generalization of our detector to a large range of scenes.

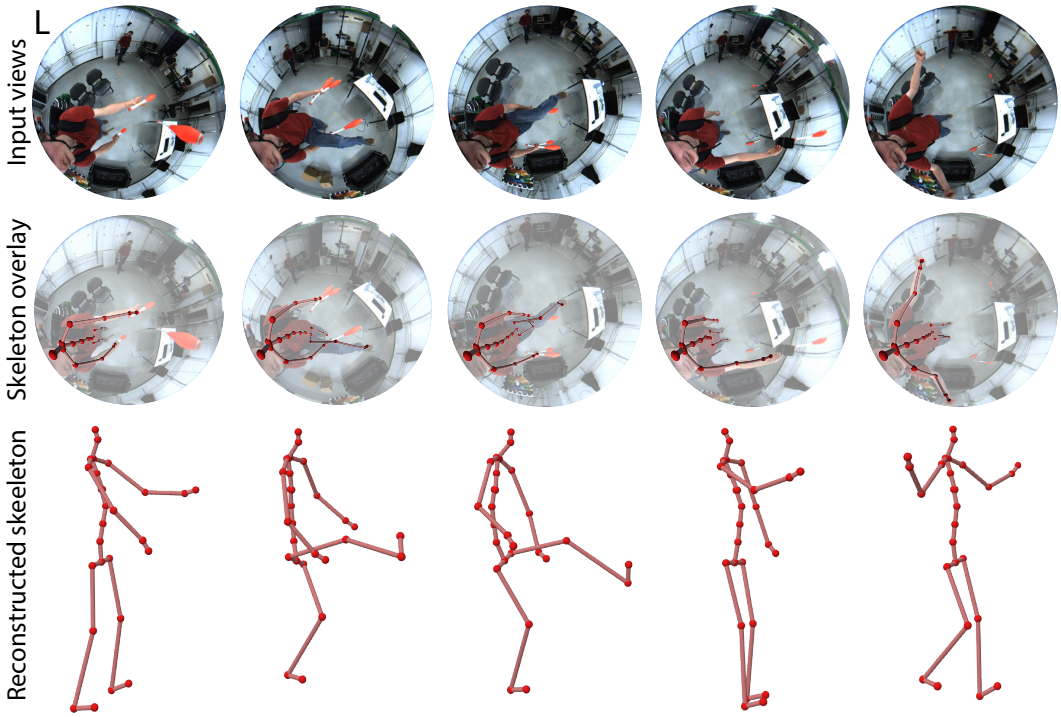


Figure 6.10: Reconstruction results on the `Juggler` sequence, showing one input view and the estimated skeleton. Despite frequent self-occlusions, our approach robustly recovers the skeleton motion.

6.4.2. Constrained and crowded spaces

Crowd occlusions

We also tested EgoCap with *EgoRig1* for motion capture on the `Crowded` sequence, where many spectators are interacting and occluding the tracked user from the outside (Figure 6.9). In such a setting, as well as in settings with many obstacles and narrow sections, outside-in motion capture, even with a dense camera system, would be difficult. In contrast, EgoCap captures the skeletal motion of the user in the center with only two head-mounted cameras.

Object interaction

The egocentric camera placement is also well-suited for capturing human-object interactions, such as the juggling performance `Juggler` (Figure 6.10). Fast throwing motions as well as occlusions are handled well. The central camera placement ensures that objects that are manipulated by the user are always in view.

6.4.3. Tracking for immersive VR

Real-time demo

We also performed an experiment to show how EgoCap could be used in immersive virtual reality (VR) applications. To this end, we use *EgoRig2* attached to an Oculus VR headset and track the motion of a user wearing it. We build a real-time demo application running at up to 15 fps, showing that real-time performance is feasible with additional improvements on currently unoptimized code. In this `Live` test, we only use color-based tracking of the upper body, without detections, as the detector code is not yet optimized for speed. The `Live` sequence shows that body motions are tracked well, and that with such an even more lightweight capture rig, geared for HMD-based VR, egocentric motion capture is feasible. Current HMD-based

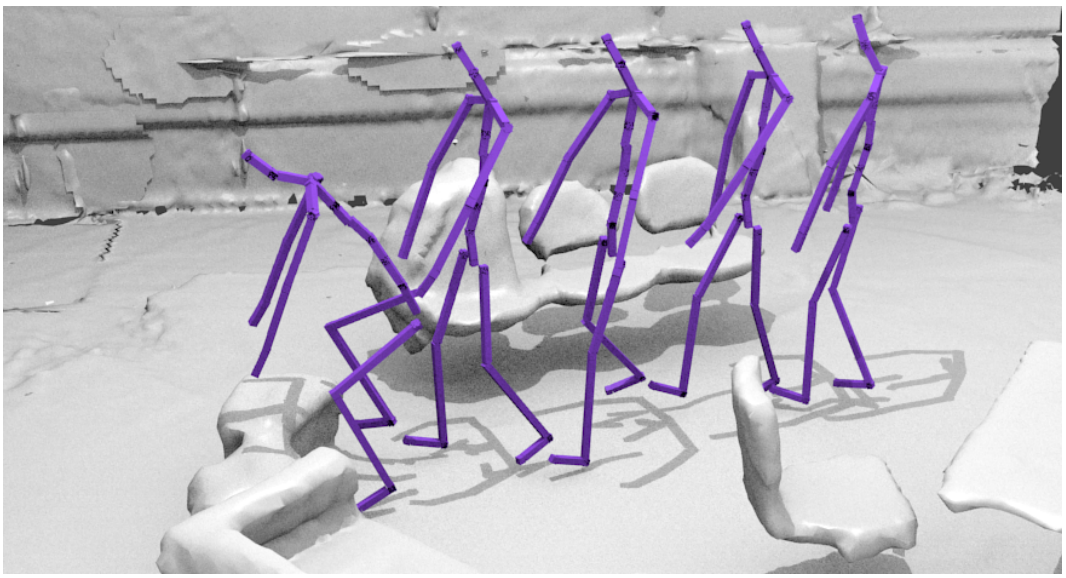


Figure 6.11: Complete motion-capture example VR, in which our egocentric pose tracking is combined with global pose tracking using structure-from-motion, shown as a motion sequence in a 3D reconstruction of the scene. In a VR scenario, this would allow free roaming and interaction with virtual objects.

systems only track the pose of the display; our approach adds motion capture of the wearer's full body, which enables a much higher level of immersion.

Complete motion capture

Global pose estimation For free roaming, the global rig pose can be tracked independently of external devices using structure-from-motion in the fisheye views. We demonstrate combined local and global pose estimation on the `Biking`, `Walk`, and `VR` sequence, using the structure-from-motion implementation of Moulon et al. [2013] provided in the OpenMVG library, see Figure 6.11 and the accompanying video. Such complete motion capture paves the way for immersive roaming in a fully virtual 3D environment.

6.5. Discussion and limitations

Full-body optical inside-in

We developed the first stereo egocentric motion-capture approach for indoor and outdoor scenes, that also works well for very crowded scenes. The combination of generative and detection-based pose estimation make it fare well even under poses with notable self-occlusions. Similar to other outside-in optical methods, tracking under occlusions by objects in the environment, e.g a table, may lead to tracking failures. However, the detections enable our tracker to quickly recover from such occlusion failures. Interestingly, the egocentric fisheye camera setup provides stronger perspective cues for motion towards and away from the camera than with normal optics. The perspective effect of the same motion increases with proximity to the camera. For instance, bending an arm is a subtle motion when observed from an external camera, but when observed in proximity, the same absolute motion causes large relative motion, manifesting in large displacements and scaling of the object in motion.

Rig construction

The algorithm in this chapter focuses on an entirely new way of capturing the full

egocentric skeletal body pose, that is decoupled from global pose and rotation relative to the environment. Global pose can be inferred separately by structure-from-motion from the fisheye cameras or is provided by HMD tracking in VR applications. Fisheye cameras keep the whole body in view, but cause distortions reducing the image resolution of distant body parts such as the legs. Therefore, tracking accuracy of the upper body is slightly higher than that of the lower body. While overall tracking accuracy of our research prototype is still lower than with commercial outside-in methods, it shows a new path towards less constrained capture in the future. Currently, we have no real-time end-to-end prototype. We are confident that this would be feasible without algorithm redesign, yet felt that real-time performance is not essential to demonstrate the algorithm and its general feasibility. Our current prototype systems may still be a bit bulky, but much stronger miniaturization becomes feasible in mass production; the design of *EgoRig2* shows this possibility. Some camera extension is required for lower-body tracking and might pose a problem with respect to social acceptance for some applications; However, we did not encounter practical issues during our recordings and VR tests, as users naturally keep the area in front of their head clear to not impair their vision.

Remaining limitations

Moreover, handling changing illumination is still an open problem for motion capture in general and is not the focus of our work. For dynamic illumination, the color model would need to be extended. However, the CNN performs one-shot estimation and does not suffer from illumination changes. The training data also contains shadowing from the studio illumination, although extreme directional light might still cause inaccuracies. Additionally, loose clothing, such as a skirt, is not part of the training dataset and hence likely to reduce pose accuracy.

6.6. Summary

EgoCap

We presented EgoCap, the first approach for marker-less egocentric full-body motion capture with a head-mounted fisheye stereo rig. It is based on a pose optimization approach that jointly employs two components. The first is a new generative pose estimation approach based on a ray-casting image formation model enabling an analytically differentiable alignment energy and visibility model. The second component is a new ConvNet-based body-part detector for fisheye cameras that was trained on the first automatically annotated real-image training dataset of egocentric fisheye body poses. EgoCap's lightweight on-body capture strategy bears many advantages over other motion-capture methods. It enables motion capture of dense and crowded scenes, and reconstruction of large-scale activities that would not fit into the constrained recording volumes of outside-in motion-capture methods. It requires far less instrumentation than suit-based or exoskeleton-based approaches. EgoCap is particularly suited for HMD-based VR applications; two cameras attached to an HMD enable full-body pose reconstruction of your own virtual body to pave the way for immersive VR experiences and interactions.

Immersive interaction

Direct retargetting of tracked performance to a virtual avatar allows the user to explore virtual worlds from first person perspective. Such direct retargetting is particularly interesting for virtual reality applications, where typical interfaces

may be impractical due to impaired real-world vision and hamper roaming, as mouse, keyboard, or gamepad physically bound the user to the desk. Moreover, if the avatar mimics the user motion an additional immersive sensation is created, because the VR display, such as VR glasses, can provide the user with 'overlay' of the virtual body on the position of the real body, which leads to a stronger feeling of being in the virtual world. Similar immersion can be obtained in augmented reality applications where the transition of virtual and real world is seamless.

Use for performance-driven animation

The next part of this dissertation takes motion capture as input and addresses open challenges for performance-driven character animation, especially for mappings that go beyond human avatars. It demonstrates how the methods developed in this part open the door for VR interaction, new computer game applications, and for intuitive content creation by performance-driven animation.

Part III

Real-time Performance-Driven Character Animation

Performance-driven character animation

7

Current state of interaction

New interactions are enabled by recent advances in consumer level motion-capture sensors, such as the Microsoft Kinect, and algorithmic advances on marker-less motion capture from RGB video, as presented in the previous part of this dissertation. However, this new technology has more potential than existing applications show. Recent 3D movies can take us on a journey into fantasies of our mind, to science fiction scenes, to historic places, and to alternative worlds, but such large and complex virtual worlds are exclusively created by professional teams relying on many artists and expensive motion capture equipment. Furthermore, existing worlds are mostly static, movies are passively consumed, and most games still offer only basic control interfaces. It lacks intuitive interfaces that enable non-professionals to interact and animate complex virtual worlds. Intriguing is the use of performance-driven interfaces. However, existing solutions struggle to utilize today's low-cost commodity sensing and reconstruction technology, because it only provides medium accuracy and detail.

Goal of this part

This part of the dissertation addresses the second half of the introduced processing pipeline, namely performance-driven character animation, see Figure 7.1. Taking the captured user performance as input, the goal is to enable intuitive and precise gestural interaction for virtual worlds. We solve existing limitations to go beyond classical mouse-keyboard and predefined gesture input.

Focus on non-human characters

We develop algorithms and representations that allow the user to not only retarget their motion in real time onto human avatars, but to go one step further and enable embodiment of arbitrary virtual creatures, for instance, a horse, dog, caterpillar, and humanoid aliens, and even to control real physical robots. While retargetting onto humanoid characters has been extensively investigated, the faithful and real-time transfer of user performance onto arbitrary characters remains an open challenge, in terms of reliability, character shape variety, and motion diversity.

Evaluation

The methods are evaluated in diverse experiments, covering hand, full-body and facial input motions that are mapped to humanoids, quadrupeds, and non-skeletal creatures. The advance on existing approaches is substantiated with quantitative experiments and an extensive user-study.

7.1. Overview

We overcome four major challenges which remain unsolved in existing state-of-the-art methods. Their description is split in the following two chapters.

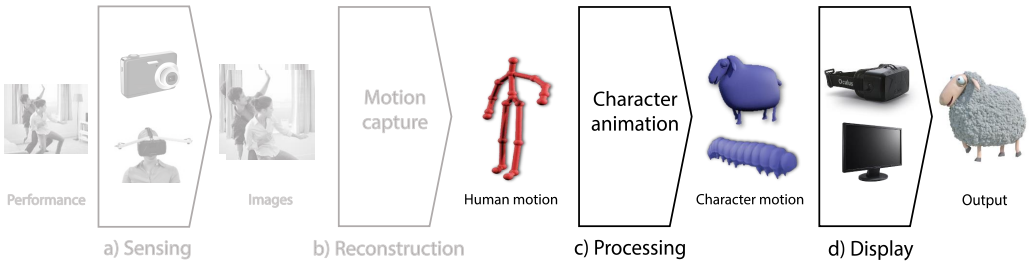


Figure 7.1: Performance-driven character animation provides a tool for interactive animation by transferring human motion to character motion. It involves step c) of the processing pipeline introduced in Figure 1.1.

Overview Chapter 8

First, mappings to non-human characters exist; however, they are either limited to skeletal representations, not suitable to characters like a caterpillar which does not have a skeleton naturally, or are specific to a particular animation system. In Chapter 8, we develop a general mesh-based representation and mathematical mapping that overcomes these limitations by exploiting unlabeled example animations.

Second, existing systems allow the definition of control motions, a particular gesture which should drive a corresponding character motion, but require explicit manual correspondence definition between character and user limbs, which is tedious and difficult for novice users [Seol et al., 2013]. We propose an approach that only requires a single performance of the user gesture and generalizes from this example, and which guides the user in the control motion selection. The algorithms to attain these two advances are explained in Chapter 8.

Overview Chapter 9

Third, existing methods do not generalize well to different motion styles. In Chapter 9, we focus on estimating and transferring dynamic motion properties which are crucial for faithful character control, such as steering the transition from trot to gallop of a horse character by speeding up the control motion. This approach of abstracting motion properties further allows the mapping between different motion styles and forms, such as mapping a stilted finger walk to a complex horse gait and at the same time to smoothly transition from a slow trot to a fast gallop. Fourth, existing sensors exhibit recording noise and provide a limited capture volume, and the user's ability to exactly reproduce control motions is limited, leading to inaccurate and stilted input motions. We develop a filtering technique which preserves important motion properties and reduces the influence of noise and control inaccuracy, in particular for periodic motions and for simultaneously performed motions. These aspects of dynamic motion control are covered in Chapter 9.

The remainder of this chapter introduces terminology and related work, which is necessary to explain the contributions of Chapter 8 and Chapter 9.

7.2. Terminology

Mathematical mapping description

We consider the problem of *performance-driven* character animation, the process of transferring a *source motion*—the user's input sensed with the input device—to a *target motion* of the target character. We focus on real-time methods, and refer to them as *real-time character control* methods. We call the transfer process from source

to target a *mapping* and treat the mapping as a mathematical function with source domain equal to the user motion representation, e.g. a sequence of skeleton poses, and target domain equal to the character representation, e.g. a mesh animation. We further separate between *reference* (source/target) motions and *control* (source) motions. Reference motions are used as examples during training, to construct the source to target mapping, whereas control motions drive the target character animation by live application of the constructed mapping. For the target character terms *motion* and *animation* are used interchangeable.

Character classes

When discriminating different character *shapes* we reason about different body proportions, such as, different limb lengths or volume. Structurally different characters are referred to have different *topology*, that means having a different number of limbs for multi-legged creatures and completely different structure such as a worm and a biped.

Mapping classes

We distinguish *topology-preserving mappings*, which map between source and target character of the same topology, and *topology-independent mappings* that can cope with topology and shape differences. Furthermore, we separate instantaneous *pose mappings*, which operate frame by frame, and *motion mappings*, which map dynamic motion properties that are estimated over a time window.

Interaction type

The processing time of approaches is classified as *offline* when in the order of minutes, as *interactive* when in the order of seconds, and *real-time* when results approach 30 Hz and more. We develop real-time approaches.

7.3. Related work

> *This section is based on the work of Rhodin et al. [2014, 2015b].*

Overview

We introduce and discuss performance-driven character animation approaches. We start to discuss topology preserving mappings in Section 7.3.1. Most of these approaches map human input to humanoid characters, and we cover offline, interactive and real-time solutions. The strengths of existing topology-independent mappings for controlling non-human characters and their remaining limitations are discussed in Section 7.3.2. We consider pose as well as motion mappings. To compare to the state of the art, the major features of the most closely related work are summarized and compared to the methods of Chapter 8 and Chapter 9 in Table 7.1.

The cost of animation

Many of the discussed methods require some form of character animation as input. Creating and animating virtual characters is a time consuming task and requires artistic and technical skill. Also recording and reconstructing real performances is expensive, as professional equipment is required for high quality. Animation by simulation is discussed in Section 7.3.3; these techniques can automate animations and are an alternative to performance-driven character animation.

Background

Finally, Section 7.3.4 covers underlying methods and representations, which form the basis of the algorithms presented in this part of the dissertation.

Feature	Gleicher [1998]	Shiratori and Hodgins [2008]	Rhodin et al. [2014], Chapter 8	Seol et al. [2013]	Ishigaki et al. [2009]	Oore et al. [2002]	Rhodin et al. [2015b], Chapter 9
Easy to use amplitude, frequency and phase	○	●	○	○	○	○	●
Control of motion dynamics	○	●	○	○	●	○	●
Extensible motion graph / intentions	○	◑	○	◑	●	○	●
Control of fast and slow motions	●	●	○	○	●	●	●
Superposition control	●	○	●	●	●	●	●
Direct control of skeleton DOF	●	○	○	●	●	●	○
Interactive user-defined control motions	○	○	●	●	○	○	●
No body part or DOF assignment required	◑	○	●	○	◑	○	●
Independent of rig	○	●	●	○	○	○	●
Non-biped topology	○	◑	●	●	○	●	●
Robust to user size and shape	●	●	◑	○	●	●	●
No database or predefined controller required	●	○	○	○	○	●	○
Diverse and multiple tracker capable	○	○	●	◑	○	●	●
Robust to low quality input device	○	●	●	●	○	○	●
High-dimensional tracking input	●	○	●	●	●	○	●
Real time	●	●	●	●	●	●	●
Low control delay	●	○	●	◑	●	●	◑
Non-bipedal complex motion transitions	○	○	○	○	○	●	●
Foot-sliding prevention (biped & quadruped)	◑	◑	○	○	◑	○	●
Physical realism	○	●	○	○	●	○	○
Live animation (game, performance)	○	●	●	●	●	○	●
Prototyping / blocking animation	●	●	◑	◑	◑	●	●
Professional 'final' animation	●	○	○	○	◑	●	○

Table 7.1: ● / ◑ / ○ : full / partial / no support; Feature table comparison of state-of-the-art real-time character control methods to Rhodin et al. [2014], and Rhodin et al. [2015b], which are explained in Chapter 8 and Chapter 9, respectively. For our task of versatile character control with application to games, the proposed methods are often a better fit than existing methods, though the system does require a pre-existing animation database.

7.3.1. Topology-preserving mappings

Human motion retargetting

Gleicher [1998] showed that user motion can be retargeted onto new bipedal characters with different body shape and proportions, but the same (bipedal) topology, through a set of position objectives and motion frequency constraints. The goal of topology preserving mappings is to obtain a target character motion which is as similar in position and dynamics as the original input motion, but adheres to the target dimension, i.e., supports shorter legs leading to reduced step-length when walking. Monzani et al. [2000] introduced an intermediate skeleton to handle different skeleton structures with the same topology. Shin et al. [2001] added importance sampling through the proximity of end effectors to the environment, to balance between conflicting motion and positional constraints. Reusing motion behavior across characters is not straightforward, as motion retargetting produces awkward artifacts even when the characters are similar in geometry.

Real-time retargetting

Ishigaki et al. [2009] applied retargetting to real-time games: user motion is blended in real time with example animations, depending on user intention, and environmental and physical constraints. This approach addresses the problem that input motions are often stilted, because the admissible user motion is limited by the motion-capture volume and is bound to the position of the display. For instance, an on-spot swimming motion is transferred to proper virtual character swimming, including global translation. This and the previously mentioned methods rely on expensive optical motion capture, only available in professional motion-capture studios.

Real-time consumer level

For low-cost systems, Chai and Hodgins [2005] tracked sparse 2D marker positions in stereo video, and achieved 3D pose reconstruction by constraining motion to a local linear model. Lee et al. [2002] transition a motion graph by performance according to silhouette features from video. Barnes et al. [2008] track hand-drawn paper cut-out characters in video input and transfer their rigid motion to 2D characters. Microsoft Kinect brought real-time control of virtual characters to casual users by variants of classical retargetting: The KinÊtre algorithm of Chen et al. [2012] and its extension by Jiang and Zhang [2015] map joint positions captured with Kinect onto a target character by means of a mesh deformation framework. Vögele et al. [2012] map the skeleton motions of two humans, captured by Kinect, in real time to a quadripedal target skeleton, for instance by mapping one skeleton to the front half of a horse, and one to the back. Both of these approaches provide intuitive character control for casual users, however, their mapping is limited to target characters with parts or sub-skeletons resembling a human skeleton, i.e., it is topology preserving; further, the motion mapping is an exact mapping between sources and target. Weise et al. [2011] extract facial expressions from the depth map provided by Kinect and transfer these to virtual faces by corresponding blend-shape face representations. Input and output are represented by the same set of face blendshape expression components (see Section 2.2). Held et al. [2012] transfer object motion into virtual worlds, but their approach is limited to rigid motions without articulation and deformation. While these approaches have many applications requiring intuitive interaction with characters, they do not allow to target characters with arbitrary shape, do not support transfer between fundamentally different motion forms, and

control of dynamic motion properties is limited.

7.3.2. Topology-independent mappings

Topology-independent mappings are the focus of this chapter and are covered in more depth. We discuss virtual puppetry, generalized retargetting, data-driven pose mappings, and data-driven motion mappings.

Classical puppetry principles

Virtual puppetry The transfer of user performance onto virtual characters via motion tracking control has been used successfully in animation for many years. Sturman [1998] reviews early work in performance-driven character animation, describing pedals, gloves, joysticks, and body suits to empower multiple users to orchestrate control of an animated character, similar to classical puppetry [Oore et al., 2002]. Layering time-sequential performances allows for rich animations [Dontcheva et al., 2003; Shiratori et al., 2013; Tyler and Neff, 2012], even from mouse and keyboard input [Neff et al., 2007]. Acting out animations proves useful for animation timing [Terra and Metoyer, 2004], but control becomes difficult for complex characters and motions. Fender et al. [2015] create an interactive VR-integrated animation system for layering multiple cyclic motions, where individual cycles are manipulated through tracked glove motion. Particularly intuitive are tangible and modular input devices, such as the one proposed by Jacobson et al. [2014] and Glauser et al. [2016], that can be modified to match the character topology and provide tangible feedback.

Retargetting to non-humans

Generalized retargetting Hecker et al. [2008] showed that some generalization to characters of different topology and shape is possible by duplicating and mirroring. New control rigs and control parameters are estimated by a particle-based inverse kinematics solver which makes the character move appropriately. Poirier and Paquette [2009] adapt an existing reference skeleton to a new character mesh by topology graph matching. Bharaj et al. [2011] automatically estimate a skeleton for multi-component characters, and describe a way of mapping source motion to the inferred target rig which succeeds as long as each target subchain can be assigned to an equivalent subchain in the source rig. Jin et al. [2015] integrate retargetting and shape editing into a single system that supports interleaved animation and editing and is easily accessible to novice users. However, these generalized retargetting approaches still rely on skeletons, are hard to map to soft-bodied creatures, and do not apply to characters with entirely different topology and motion style. Feng et al. [2008] extract a set of control points from a single mesh animation and use kernel canonical correlation analysis between control points and mesh to reproduce fine-scale surface details from the articulated movements of an underlying skeleton. While these approaches generalize to characters on non-human form, the underlying skeleton representations limit their generalization to arbitrary shapes, for instance, the deforming body of a caterpillar is unsuitable for a skeleton rig.

Data-driven

Data-driven pose mappings The similarity requirement between user and character can be relaxed by constructing indirect mappings. The mapping from user motion to character motion can automatically be inferred in a training step by learning

their dependency from examples. Training input motions are usually performed by the user. Training character animations are pre-authored by artists.

Frame to frame

Most common are *pose mappings*, which map the tracked user pose, frame-by-frame, to a corresponding character pose. Pose mappings have been constructed through linear transfer of interpolations weights [Bregler et al., 2002], non-linear interpolation through a hierarchical mesh shape space [Baran et al., 2009], Gaussian process latent variable model (GPLVM) [Lawrence, 2004; Yamane et al., 2010], and linear maps to rigged characters [Dontcheva et al., 2003]. Real-time control has been attained through linear mappings to rigged [Seol et al., 2013] and mesh characters [Celikcan et al., 2014]. Dontcheva et al. [2003] obviate the common manual selection of corresponding degrees of freedom by CCA on a dense temporally aligned training sequence. Baran et al. [2009] rely on a set of example temporal correspondences that encode the semantics of the mapping. It nicely drives one mesh animation by another, but is not applicable to the output of today's real-time motion estimation algorithms as it does not support point based and skeleton input. Zhou et al. [2010] extend the framework of Baran et al. to multi-component objects. One key difference of our approaches is that we utilize unlabeled samples in the training motion examples, not just for user defined correspondences.

Yamane et al. (offline)

The approach of Yamane et al. [2010] is offline, it maps the exact motions of a human to a target character, e.g. a hopping motion to make a Luxo lamp hop, or a toddle to control the walk of a penguin. With 30-50 correspondences hand-distributed at important poses, they build a mapping between source and target in a shared latent space obtained by GPLVM. As this mapping is noisy, they regularize with a post-process inverse kinematics solver, as well as a further post-process to ensure foot planting.

Seol et al. (real time)

To improve real-time control of arbitrary characters, Seol et al. [2013] classify user input poses in two: *simple* motions, where character features map to human features directly and a linear map is applied, and *complex* motions, where features do not correspond (e.g. many pairs of legs) and so a nearest-neighbor lookup (NN) finds the closest animation frame from a pre-defined coupling. Both outputs are then blended. This setup generalizes via the linear map from the authored character animations as new live motions are performed. In Chapter 9 we overcome the following remaining limitations. The NN map does not generalize, which leads to stilted animation when given new live motion variations, with no extrapolation to provide control over motion style variations. Moreover, detailed manual selection of features and a character rig are required.

Current limitations

As demonstrated in Chapter 8, pose mapping techniques allow real-time control of non-human characters, as example-driven mappings decouple input and output shape and motion style. However, the output animation quality and detail is limited by the ambiguity of pose mappings. The goal of Chapter 9 is to overcome these ambiguities by generalizing properties of motions from sparse examples.

Motion style and dynamics

Data-driven motion mappings To overcome pose ambiguities, the temporal evolution of an animation can be utilized. Motion contains information on the performed action, the mood of the character, and its intentions, information which is hidden

in instantaneous shots, i.e., a single pose. For instance, slow waving usually resembles a farewell gesture, while hectic waving is used to catch attention. The same motion performed at different speed can encode different intentions. Extracting and mapping such motion content is difficult and was only partially addressed in related work.

Action control

Action control methods trigger (non-human) character actions by detecting user intention. Seol et al. [2013] classify user input by accumulating per-frame classifications. More advanced are classification methods using dynamic time warping [Raptis et al., 2011] and dynamical movement primitives [Ijspeert et al., 2013]. Such high-level action control enables *sympathetic interfaces*, e.g. animation through a sensor equipped plush doll [Johnson et al., 1999], and automatic maintenance of the emotional character state to stay *in character* [Tomlinson et al., 2002]. Please note that these approaches only classify in broad categories, it is not possible to map continuous properties of motion, which would provide a much finer level of control.

Shiratori et al. (real time)

An inspiring step in this direction is the method of Shiratori and Hodgins [2008], where amplitude, phase, and frequency of low-dimensional accelerometer sensors are mapped to a physically-simulated character. The input motion is classified into a set of discrete states and each input state maps to a predefined character state. For instance, acceleration frequency is classified into slow and fast states and is respectively mapped to a slow or fast character walk, utilizing that a slow and a fast walk input creates low and high frequency accelerations, respectively. The simulation takes care of transitions and directional changes. The method of Lockwood and Singh [2012] applies a similar principle and classifies *finger walking* motions from touchpad input by contact features, e.g. frequency, into gross motion classes. We generalize these ideas in Chapter 9, to handle high-dimensional input motions, continuous output (e.g. speed regression vs. classification into slow or fast), user-defined control motions with arbitrary periodic trajectories, and simultaneously-performed motions.

7.3.3. Character control by simulation

Animation by simulation

Direct manipulation of selected degrees of freedom (DOF) of characters is tedious for the artist and can be eased by adding physical simulation. In contrast to the previously introduced performance mapping approaches, control by simulation creates physically-motivated animations from scratch. Only the start and boundary constraints for simulation systems need to be set. The difficulty lies in finding the right balance between user specifications and automatic simulation. Coros et al. [2012] build controllers for purposeful motion of soft deformable characters via elastic simulation. They couple mouse input via rest state adaptation in an elastic physical simulation, and Laszlo et al. [2000] couple mouse and keyboard input with physical simulation of bipedal motion through proportional derivative controllers. The precise control of complex simulated characters is difficult, as the outcome of the simulation from user input is hard to imagine by the user. To ease simulation control Laszlo et al. [2005] proposed to provide guidance by simulating and displaying the outcome of possible future user inputs interactively, giving

the user immediate feedback on the temporal influence of his input. Celikcan et al. [2014] combine data-driven pose mappings with physical constraints by solving a spatio-temporal system of equations, leading to physically plausible performance-driven animation.

7.3.4. Character and motion representations

As important as how to map between human input and character output motion is to decide on their representation. The representations most related to the ones chosen in this dissertation are introduced in the following.

Animation representation

Dimensionality reduction Data-driven animation schemes parameterize a character motion space by example motions. Animation researchers have experimented with dimensionality reduction techniques for data reduction and for designing new motion controllers that are intuitive to use [Alexa and Müller, 2001; Kry et al., 2009], and both linear mappings and non-linear approaches have been investigated [Shin and Lee, 2006]. In linear frameworks, the movement space can be considered to be spanned by a set of basis shapes [Torresani et al., 2001], or by a set of basis trajectories for each point [Akhter et al., 2010]. In this new basis space, the character can be controlled by modifying the influence of basis shapes or trajectories. Akhter et al. [2012] propose a bilinear spatio-temporal basis that describes oscillations around a set of example shapes. Some animation types cannot be well represented in a linear framework, and so these techniques might lead to a small latent control space with few meaningful dimensions. Non-linear dimensionality reduction techniques, such as Gaussian process latent variable models (GPLVM), kernel methods, or multi-dimensional scaling (MDS), have also been applied to animation parametrization [Levine et al., 2012]. For example, Cashman and Hormann [2012] project arbitrary motions onto a 2D control plane, obtained through MDS, to create new animations as paths within that latent space. In contrast to these approaches, where each dimension in the latent space affects the surface deformation globally, direct local parameterizations in the source space are also feasible. James and Twigg [2005] represent general mesh deformations with a set of proxy bones and skinning weights. Kavan et al. [2010] and Jacobson et al. [2012] extend this framework for fast and automatic computation.

Motion parametrization

Dynamics can be integrated into the model by considering previous frames of motion. de Aguiar et al. [2009] use projection into a linear latent space in combination with a second-order linear dynamic system to simulate cloth motion, whereas Wang et al. [2008] use a non-linear system built through Gaussian process dynamic models.

Transition graphs

Motion graphs Realistic character motion is heavily constrained. Transitions between two motions can only happen at limited time frames and not between all motion classes. For instance, it is impossible to walk while lying on the ground, the character has to stand up first. And transitions from run to walk are only possible during ground contact, not in mid air. Such constraints are modelled in motion graphs of animation databases: Nodes in the graph represent decision points, and edges represent feasible transitions between motions. Rose et al. [1998] and Heck

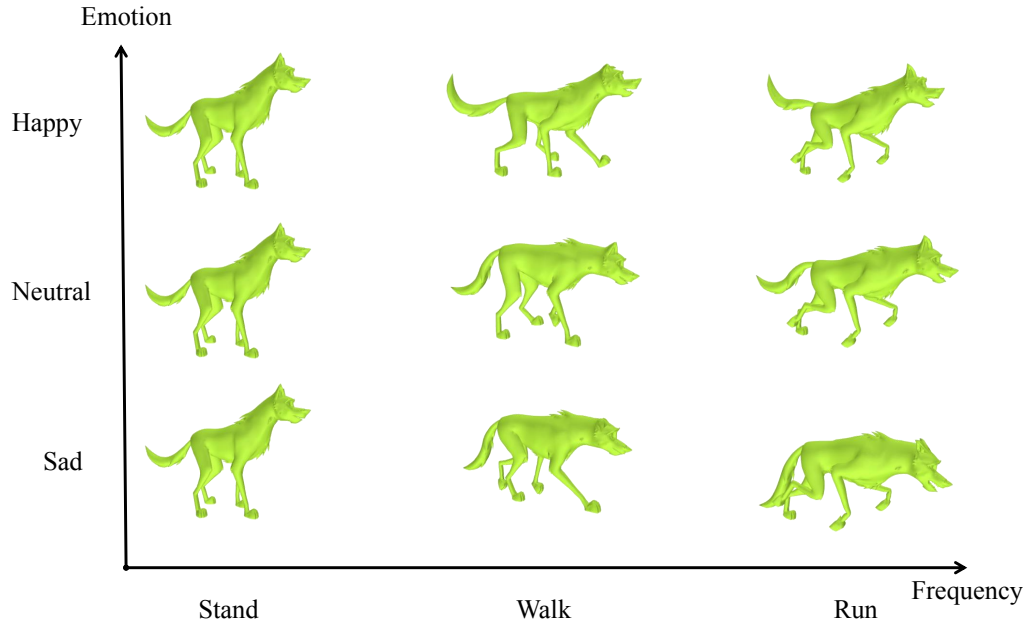


Figure 7.2: Database character animations are structured into parameterized motion classes, such as this locomotion class for a dog. In Figure 9.3, the phase-frequency slice through the parametrized motion class is visualized, here we give the speed-emotion dimension.

and Gleicher [2007] extend this structure where a node represents a motion class (e.g. walking) which is parametrized by motion properties such as speed or style.

In Chapter 9, we use a simple motion graph of a dog character with shake, sit down, and move classes. Figure 7.2 shows an emotion-frequency slice of the parametrized move class.

Such parametrized spaces are traditionally created by manual positioning of examples [Igarashi et al., 2005], which may require artistic skill or knowledge of the animation pipeline to produce good results. Automatic extensions include parametrized transitions [Shin and Oh, 2006], dense graphs where possible transitions are stored for each database frame [Lee et al., 2010], statistical motion models [Min and Chai, 2012], and interpolation and transition for mesh characters [Casas et al., 2012]. Graph construction has also been automated [Kovar et al., 2002; Heck and Gleicher, 2007]. The interpolation quality and realism can be improved by using pre-recorded motions as transitions [Tanco and Hilton, 2000; Arikan and Forsyth, 2002]. Animations are commonly synthesized according to high-level task constraints such as motion goals, and foot-plant and end-effector positions [Wiley and Hahn, 1997; Lee et al., 2010; Levine et al., 2012; Lockwood and Singh, 2012], though this part of the dissertation focuses on real-time gestural motion control.

Space-time representations

Fourier representations While motions can be represented as a time series of poses, they do not capture important properties, such as periodicity, and are not well suited for motion analysis. Fourier representations separate motion properties in frequency bands and have been widely used in character animation. Frequency band decompositions have been used to create animation variations by multi-level

sampling [Pullen and Bregler, 2000], to blend animations, and to alter motion style [Unuma et al., 1995]. The bilinear spatio-temporal basis of Akhter et al. [2012] also expresses frequency bands. In Chapter 9, we use a particular form of Fourier decomposition of high-dimensional live input motions to transfer amplitude, frequency and phase properties from input motion to target character.

Interactive pose mapping for real-time character control

8

This chapter is based on Rhodin et al. [2014].

Tasks and open challenges

The motion-capture algorithms explained in the first half of this dissertation provide great opportunities as input for interactive animation. With traditional skeleton-based animation pipelines, human joint data can be mapped to a virtual character so long as it has a similar skeleton to a human, thus providing character control. However, often structurally different non-humanoid characters need to be controlled. For instance, deformable characters like caterpillars are poorly represented by skeleton rigging. Spiders can be parameterized by skeletons with different numbers of limbs, but even if a skeletal mapping can be defined, direct human skeletal control would be difficult due to the entirely different character motion style. In these cases, mapping and retargetting of human motions would require tim- consuming manual work by animators. In general, retargetting has practical limitations, and the question of how to map arbitrary motions remains.

Existing approaches

Existing character control algorithms cannot map arbitrary source and target motions, and new approaches need to fundamentally generalize. One approach is to deform meshes by using joints as deformation constraints, e.g. to embody and move a chair [Chen et al., 2012]. Another approach controls a quadrupedal target skeleton by mapping the motion to two humans, similar to a pantomime horse [Vögele et al., 2012]. However, these approaches do not scale to target character shape and motions that are very different from human.

Need

Solutions to the generalized motion retargetting problem need to map a source input space to the target output space of a virtual character, even if that character is structurally dissimilar to a human and even if its motion is not well represented by a skeleton. It should be possible to map an undulating arm motion to a crawling caterpillar motion, or to map a jockey saddle bounce motion to a horse gallop motion, all without explicitly specifying spatial correspondences.

Our approach

We move towards this goal with a solution for interactive mapping and real-time character control. Our approach is general and abstracts from the classical skeleton-rig-based motion parameterization which is not suitable for mapping structurally different motion spaces. We take as input a reference source sequence of sparse 3D points or meshes, such as full-body, hand, or face motion from any motion-capture system. This approach bypasses time consuming and potentially error prone skeleton reconstruction, see Figure 8.1. We also take as input a target

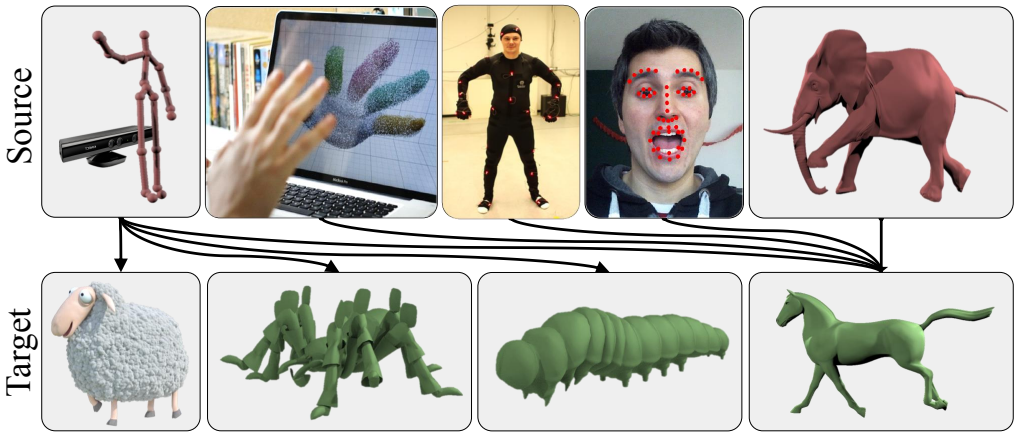


Figure 8.1: Different motions are easily mapped between very different morphologies for fast and versatile real-time character control. We require only 4-8 interactively defined correspondences and support 3D point sequences and meshes as input.

character as a sequence of meshes, and no additional control rig is required. Thus, any animation type can be puppeted by our system, be it keyframe animation, physics-based simulation, or rig-based animation.

Latent volume and mapping

To achieve this, we represent both motions in dedicated feature spaces and learn a mapping between these intermediate representations, with bounds stemming from a *latent volume* which constrains the predicted poses. Latent volumes are bound from training motions which reduce the risk of causing artifacts if new live source motions for puppetry are unlike the reference source motions, and so they remove the need for the puppeteer to restrict their poses only to those correspondences defined. Furthermore, this makes our approach robust, and mappings learned for one source can be successfully applied to other sources, such as in training/repetition scenarios or across actors. The learned mapping is pose-based, a motion-based approach is described in Chapter 9.

Interactive definition and real-time control

Our approach is fast, flexible, and user-friendly as our algorithm only requires the specification of 4-8 temporal correspondences between source and target sequences. We neither require spatial correspondences, e.g. limb mapping, nor correspondences to every pose in the target character motion. This enables an efficient *animate-correspond-synthesize* workflow: The user interactively corresponds their body poses to the target character, the mapping is learned in a few seconds, and then new animation is synthesized in real-time. As this cycle is very fast, users can iterate to find which source mappings are most intuitive and make on-the-fly control adjustments. This naturally leads to applications in simple motion and party games; however, our approach also gives new tools to animation artists. In conversations with animators, they found our system useful to 'give life' to existing animations through creative puppetry, for example, to produce variation when animating crowds or when adding timing nuances to a walk cycle through finger tracking. In our experiments, we show the performance and reliability of our method with a variety of real-time control examples with different source and target motions, such as mapping full-body motion of a human to a caterpillar and mapping face motions onto a sheep.

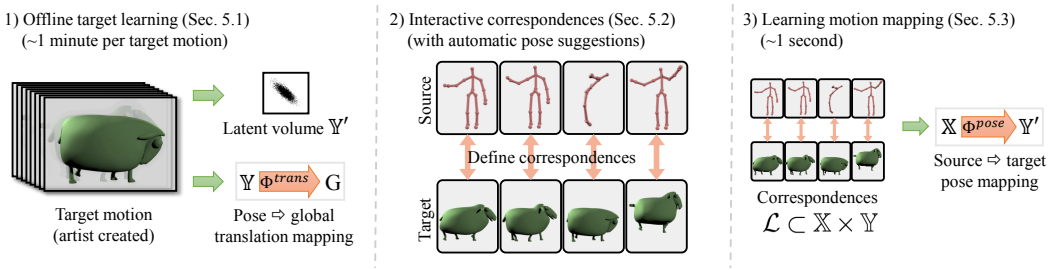


Figure 8.2: Learning a motion mapping: 1) Offline, an artist creates an unlabeled target motion. We learn latent volume $\mathbb{Y}'$ and the dependency Φ^{trans} between changes in pose and global translation G. 2) Sparse key pose correspondences are automatically suggested and then interactively refined. 3) The map Φ^{pose} between source space $\mathbb{X}$ and target space $\mathbb{Y}$ is learned.

We summarize our contributions: *Contribution and impact*

- A real-time algorithm that can map between characters with different topology from sparse correspondences.
- A latent volume representation that efficiently exploits unlabeled data to allow robust character control.
- An automatic keyframe suggestion method to support the user during correspondence selection.

These create a robust and easy to set up system for interactive motion mapping of characters with arbitrarily different shapes and motions, which opens the door for more creative and intuitive real-time character animation.

8.1. Overview and notation

We wish to control a target character by performance. First, we learn a high-dimensional oriented bounding box, or *latent volume*, which represents the space of controllable target poses (Figure 8.2, left, and Section 8.2.1). This offline training computation takes about a minute. We wish to puppet the target motions using arbitrary source motions, and so the next step is for the user to interactively define a small number of pose correspondences (4 to 8) between example or reference source motions and the desired target motions (Figure 8.2, middle, and Section 8.3). The reference source motions need only be performed once, and help is provided with automatic suggestions of appropriate poses. From these correspondences, we learn a mapping between reference source and target motions (Figure 8.2, right, and Section 8.4), and this takes less than one second. This completes the learning stage. *Structure*

In the synthesis phase, live user control motions are inputted to the learned motion transfer mapping to synthesize new motion sequences of the target character in real time (Figure 8.6, and Section 8.5). Control motions are similar repetitions of the reference source motions used to define correspondence, and can vary in intensity, e.g., faster or slower, or arms raised less high, and can be performed simultaneously to cause motion combination or superposition effects. In most cases, the input reference and live source motions will both come from a tracking system *Control capabilities*

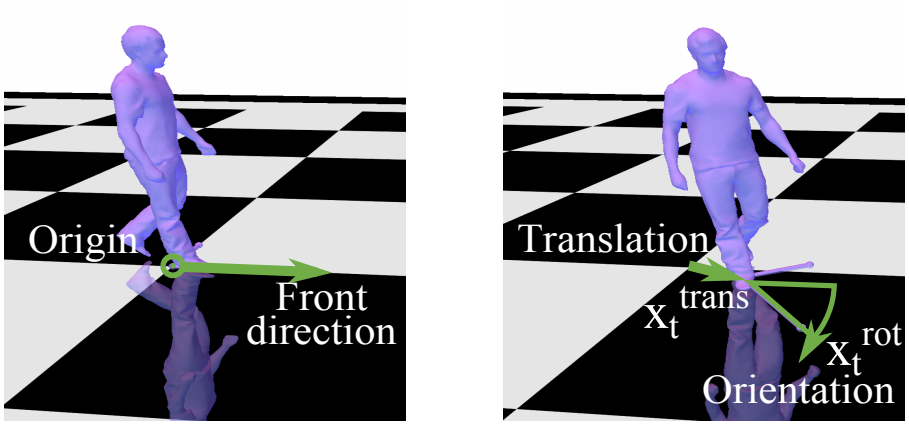


Figure 8.3: Global translation $\mathbf{x}_t^{\text{trans}}$ and rotation $\mathbf{x}_t^{\text{rot}}$ of a general pose (*right*) extracted in relation to the rest pose *(left)*.

such as Kinect, LeapMotion, or a face tracker, which creates a system for interactive motion mapping and real-time character control.

Input and output notation

In the diagrams of this chapter, the input reference source motion used in training and the live control motion used in synthesis are shown in red, the input target motion is in green, and the output synthesized motion is in blue. Our source motion $[\mathbf{x}_1, \ldots, \mathbf{x}_M]$ is a 3D point sequence with M frames. Target character motions $[\mathbf{y}_1, \ldots, \mathbf{y}_N]$ are mesh sequences. Our representation must separate global motion from character pose, such as, to isolate a walk cycle from world movement, so that we can independently map these to different controls. We parametrize a motion into character pose feature vectors $[\mathbf{x}_1^{\text{pose}}, \ldots, \mathbf{x}_M^{\text{pose}}]$, global translations $[\mathbf{x}_1^{\text{trans}}, \ldots, \mathbf{x}_M^{\text{trans}}]$, and global rotations $[\mathbf{x}_1^{\text{rot}}, \ldots, \mathbf{x}_M^{\text{rot}}]$.

8.1.1. Global motion

Global pose

For each frame in the source $\mathbf{x}_t$ (or target $\mathbf{y}_t$), we estimate the global position and orientation of the character on the ground plane as an offset to $\mathbf{x}_1$ by the least squares fit of $\mathbf{x}_t$ to $\mathbf{x}_1$ using orthogonal Procrustes analysis [Sorkine, 2009], see Figure 8.3. Global motion is represented by 3 degrees of freedom: a translation vector $\mathbf{x}_t^{\text{trans}} \in \mathbb{R}^2$ and a yaw rotation angle $\mathbf{x}_t^{\text{rot}} \in \mathbb{R}$.

8.1.2. Source point-based representation

Temporal information

We represent character pose as the concatenated feature vector of 3D point positions and their velocities after compensating for global motion and mean centering. The velocities help disambiguate similar poses in simple source motions (e.g. raising vs. lowering limbs). For a pose $\mathbf{x}_t$ at time t, velocity vector $\dot{\mathbf{x}}_t = (\mathbf{x}_t - \mathbf{x}_{t-1})/\Delta t$, where Δt is the time between two frames. The source pose vector is $\mathbf{x}_t^{\text{pose}} = [\mathbf{x}_t^\top, w\dot{\mathbf{x}}_t^\top]^\top$, which is 6V in length for a character of V vertices. Factor w balances the contribution of static and dynamic information in the regression and is set to 0.1. For noisy vertex position input, we apply a small one-sided Gaussian filter to smooth temporally, with a 3-frame standard deviation (≈ 0.1 sec.).

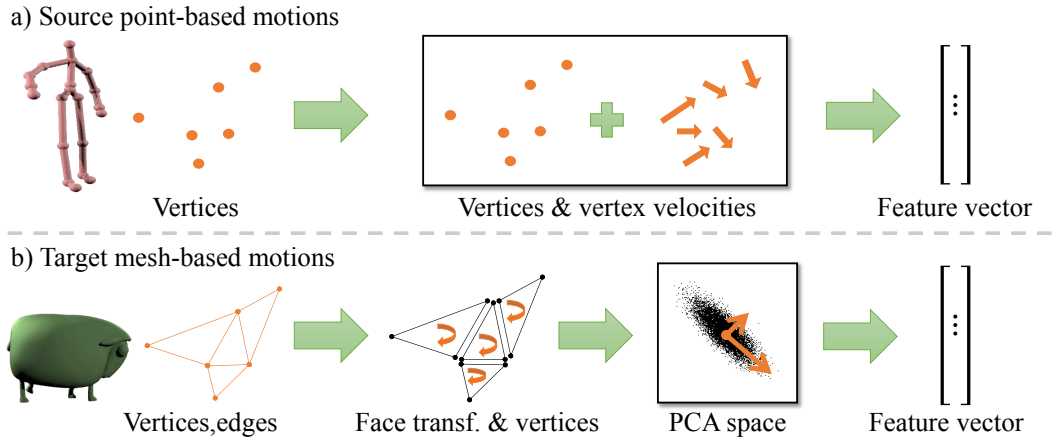


Figure 8.4: a) Source point-based characters are represented as vertex positions and velocities. b) Target mesh characters are decomposed into vertex positions and face transformations to model rotations explicitly, and are projected into low-dimensional PCA space to reduce complexity.

8.1.3. Target mesh representation

Per-face representation

The simple point-based representation leads to strong distortions if used for target mesh characters. We remove distortions by exploiting information contained in the connectivity of the mesh. We extend the deformation gradient representation, which models surface deformation by combining per face transformations [Sumner and Popović, 2004], with explicit vertex features (Figure 8.4). We discuss the strength of this representation against alternatives in Section 8.6.1.

Rotation and shear encoding

For each mesh face f we extract the affine transformation $\mathbf{A}_f$ in relation to the rest pose and decompose it into rotation and shear by polar decomposition. Rotations are compactly stored in axis-angle form and the six degrees of freedom of the symmetric shear matrix are linearized to a vector. To model the absolute position of potentially disconnected mesh components, which is not included in the original deformation gradient representation, we additionally store all vertex positions $\mathbf{v}_i$. Therefore, each pose vector $\mathbf{y}_t^{\text{pose}}$ is a concatenation of $3F$ rotation, $6F$ shear, and $3V$ point parameters for a character with F faces and V vertices. Details of this representation are given in Section 2.2.3.

8.2. Learning a motion mapping

Mapping domains

We aim to learn a motion transfer function $\Phi : \mathbb{X} \to \mathbb{Y}$ between the space of source poses $\mathbb{X}$ and the space of target poses $\mathbb{Y}$ based on a sparse set of *labeled* pose correspondences $\mathcal{L} \subset \mathbb{X} \times \mathbb{Y}$ and additional *unlabeled* character example motions $\mathcal{U}_\mathbf{y} \subset \mathbb{Y}$.

Semi-supervised regression

This is an instance of the regression problem. In this context, previous works have successfully applied combinations of non-linear feature extraction (or data representation, e.g. GPLVM) and regression (e.g. kernel-based Gaussian process regression (GPR)) to offline character animation [Yamane et al., 2010; Vögele et al., 2012; Levine et al., 2012]. However, our application poses new challenges that

are not well addressed in existing systems: 1) only a limited number of labeled correspondences are given, and 2) character animation synthesis should be real-time. Fortunately, unlike typical regression, we have the set of unlabeled examples $\mathcal{U}_{\mathbf{y}} \subset \mathbb{Y}$ which is larger than the labeled training set of correspondences. We exploit this additional information on the potential variations to assist the regression process and estimate a latent volume $\mathbb{Y}'$ that effectively encodes and limits the predicted pose.

Mapping type

As such, our goals are three-fold (Figure 8.2): First, to learn a mapping Φ^{trans} between pose and global translation for the target sequence $[\mathbf{y}_1, \ldots, \mathbf{y}_N]$, and to learn the range of admissible target motion as a latent volume $\mathbb{Y}'$ (Section 8.2.1); Second, to generate pose correspondences $\mathcal{L}$ between reference source motions and target motions, in which we adopt Bayesian regression to suggest appropriate candidates (Section 8.3); and, third, to learn a motion transfer function Φ^{pose} between source and target poses from these correspondences (Section 8.4).

8.2.1. Offline target learning

Local and global motion

The offline learning includes two steps (Figure 8.2, left): learning a mapping Φ^{trans} which links pose to global translation, and the computation of the latent volume $\mathbb{Y}'$.

Simplification by PCA

Dimensionality reduction We reduce the dimensionality of mesh representations to $D = 50$ by principal component analysis (PCA) on $\mathcal{U}_{\mathbf{y}}$. D was chosen experimentally such that the dimensionality is drastically reduced while still preserving more than 99% of the original variance, and this improves the memory footprint and performance of the mapping for our real-time scenario. Henceforth, we refer to the resulting feature vector as $\mathbf{y}_t^{\text{pose}} \in \mathbb{R}^D$. The reconstruction from this low-dimensional feature vector is explained when we come to synthesize a new motion in Section 8.5.

Admissible poses

Learning a latent volume We represent pose as a vector $\mathbf{y}_t^{\text{pose}}$, an element of $\mathbb{R}^D$. However, characters are physical objects whose pose variations are constrained by their bodies, and the plausible range of variation in $\mathbf{y}_t^{\text{pose}}$ is significantly smaller than $\mathbb{R}^D$. For instance, human pose is limited by joints, while caterpillar and crab poses are constrained by their skins and carapaces.

Latent volume

Traditionally, these constraints have been explicitly constructed through virtual skeletons with fine-tuned degrees of freedom and joint limits. In our case, we expect arbitrary source and target animations and, accordingly, we have no a priori knowledge about the admissible range of motions. Our approach is to exploit the unlabeled data $\mathcal{U}_{\mathbf{y}}$ for inferring implicitly the admissible variations through a high-dimensional oriented bounding box, or latent volume. By assuming that the distribution of poses is approximated by a Gaussian distribution, we find the principal axes of variation by decorrelating $\mathcal{U}_{\mathbf{y}}$ with PCA and estimating the range of motion to be the minimum interval I_c encompassing all data points in $\mathcal{U}_{\mathbf{y}}$ across PCA components c. The effectiveness of this operation is empirically demonstrated in Section 8.6.

On-spot input to global motion

Global translation map Global character motion $(\mathbf{y}_t^{\text{trans}}, \mathbf{y}_t^{\text{rot}})$ often depends on the change of pose $\dot{\mathbf{y}}_t^{\text{pose}}$, e.g., a pose in a run motion should cause more global translation than a pose in a walk motion. We learn a relationship between pose and translation with a linear map Φ^{trans} from the target pose velocities, which uses the heuristic that a quicker pose change leads to quicker locomotion:

$$\Phi^{\text{trans}} = \arg\min_{\mathbf{W}} \sum_{=1,\ldots,\mathtt{N}} \|\mathbf{W}\,|\dot{\mathbf{y}}_t^{\text{pose}}| - \dot{\mathbf{y}}_t^{\text{trans}}\|^2 + \|\sigma_{\text{trans}} W\|^2, \tag{8.1}$$

where velocities are estimated by finite differencing, $|\mathbf{y}|$ is the coefficient-wise absolute value of vector $\mathbf{y}$, and σ_{trans} is the regularization parameter ($= 0.001$). We learn Φ^{trans} from $\mathcal{U}_{\mathbf{y}}$ and apply it during synthesis to recover global translation, and separately apply rotations from the live control motion (Section 8.5).

8.3. Guided interactive control definition

Semantic alignment

Given the latent volume of the target sequence, our next goal along the way to defining the pose motion transfer function Φ^{pose} is to label corresponding pose frame pairs $\mathcal{L} = \left[(\mathbf{x}_{l_1}^{\text{pose}}, \mathbf{y}_{l_1}^{\text{pose}}), \ldots, (\mathbf{x}_{l_n}^{\text{pose}}, \mathbf{y}_{l_n}^{\text{pose}})\right]$ between the reference source and target sequences, where each label element $[l_1, \ldots, l_n]$ contains indices for $\mathbf{x}^{\text{pose}}$ and $\mathbf{y}^{\text{pose}}$ in their respective frame ranges $[1, M]$ and $[1, N]$. From existing work, we might think that time-warping is a suitable method to align reference source and target character sequences by synchronising spatial correspondences; however, this assumes a priori temporal or spatial correlations. For instance, Vögele et al. [2012] use time-warping to align the legs of two humans to the front and rear limbs of a horse, but this 'limbed creature' spatial correlation assumption fails for arbitrary motions and characters.

Dissimilarity measure

In arbitrary cases, selecting good correspondences between reference source motions and target motions requires some skill as it is not always intuitive which source motions will lead to good control. To ease this process, we exploit a Bayesian regression model (detailed in Section 8.4) to provide assistance in two different use cases. As correspondences are defined and as $\mathcal{L}$ increases in size, this model provides us with the most probable corresponding pose $\mathbf{y}_*^{\text{pose}}$ for the current source pose $\mathbf{x}^{\text{pose}}$, where the probability density $P(\mathbf{y}^{\text{pose}}|\mathbf{x}^{\text{pose}})$ is inferred from the present set of correspondences $\mathcal{L}$ and where the variance corresponds to the uncertainty of the prediction. Therefore, we choose the predicted variance as a metric $q(\mathbf{x}^{\text{pose}})$ to suggest correspondences which will make mappings with good control.

Interactive control definition

Performance-based Often it is natural to perform the desired reference source motions, and so in this mode the user defines correspondences to target poses by performing the reference source motions and timing button presses. For instance, when capturing reference source motions with Kinect, our system uses a hand-held remote-controlled trigger which the user activates when their poses align to the desired target poses. To help the user, q is shown as a bar which increases in size and changes color from red to green when the performed pose is underrepresented by the present correspondence selection, i.e., when the performed pose explains much of the remaining variance (Figure 8.5, bottom).

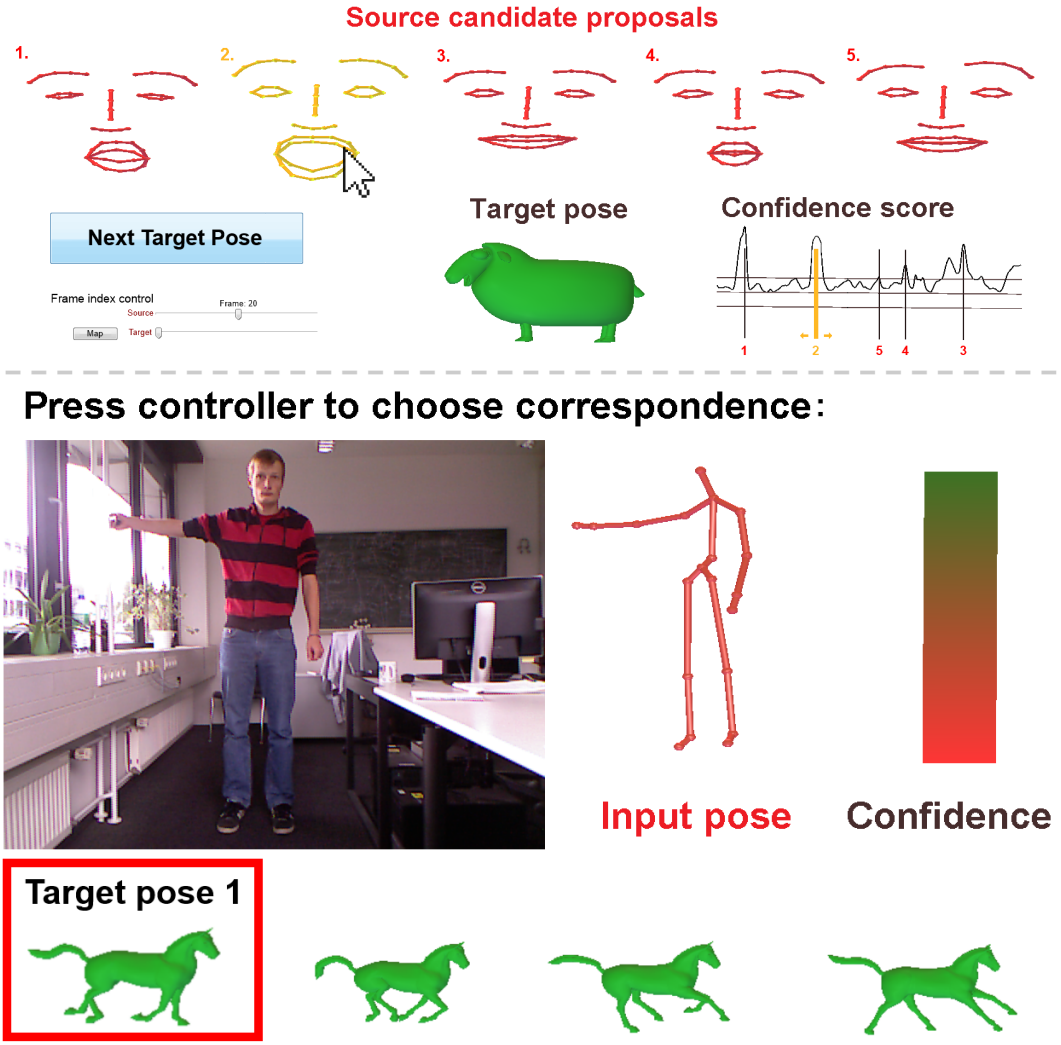


Figure 8.5: *Top:* Automatic correspondence suggestion for mapping source face tracking data to a sheep character. The confidence plot score allows users to fine-tune the suggestions. *Bottom:* Performance-based correspondence definition, where the confidence of the current pose being a good controller is visualized to the user as a colored bar.

Refinement

Automatic correspondence suggestion In other cases, the reference source motion is captured offline, or the user may wish to refine performance-based correspondences. For each target correspondence $\mathbf{y}_l^{\text{pose}}$, we suggest the 5 largest local maxima of $q\left([\mathbf{x}_1^{\text{pose}}, \ldots, \mathbf{x}_M^{\text{pose}}]\right)$ from the reference source as candidates. Users can accept one of the suggestions, or are able to make adjustment with a pose time slider. After acceptance, the pose correspondence is added to $\mathcal{L}$, metric q is instantly recomputed for the updated $\mathcal{L}$, and new suggestions are proposed for the next correspondence (Figure 8.5, top).

8.4. Learning a user-to-character pose mapping

Linear Bayesian regression

Given our correspondences, the next step is to learn Φ^{pose}, the pose transfer function. We construct a linear map $\Phi^{\text{pose}}(\mathbf{x}^{\text{pose}}) = \mathbf{M}\,\mathbf{x}^{\text{pose}}$ that fits to a given

set of labeled correspondences $\mathcal{L}$. Matrix $\mathbf{M}$ is of size $D \times$ *input pose dimension*. Adopting the standard Bayesian linear regression framework, we apply an identical isotropic Gaussian prior to each row of $\mathbf{M}$; $\text{row}(\mathbf{M}) \sim \mathcal{N}(\mathbf{0}, I)$ with $\mathcal{N}(\mathbf{m}, C)$ being a Gaussian distribution with mean vector $\mathbf{m}$ and covariance matrix C, and a Gaussian noise model:

$$\mathbf{y}^{\text{pose}} = f(\mathbf{x}^{\text{pose}}) + \epsilon I, \tag{8.2}$$

where f is to be estimated and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. We set the noise value σ^2 to 0.05 for all our results; manual tuning may further improve results. We concatenate $\mathbf{x}^{\text{pose}}$ with 1 to model a constant offset between source and target.

Posterior distribution

Marginalizing the prior of $\mathbf{M}$ and the likelihood (8.2) over $\mathbf{M}$, we obtain the predictive Gaussian distribution on the output target pose $\mathbf{y}^{\text{pose}}$ given test input $\mathbf{x}_*^{\text{pose}}$ [Rasmussen and Williams, 2006]:

$$p(\mathbf{y}^{\text{pose}} | \mathbf{x}_*^{\text{pose}}, \mathcal{L}) = N(\sigma^{-2}\mathbf{Y}\mathbf{X}^\top \mathbf{A}^{-\top}\mathbf{x}_*^{\text{pose}}, \mathbf{x}_*^{\text{pose}\top}\mathbf{A}^{-1}\mathbf{x}_*^{\text{pose}} I), \tag{8.3}$$

where $\mathbf{A} = \sigma^{-2}\mathbf{X}\mathbf{X}^\top + I$ and matrices $\mathbf{X} = [\mathbf{x}_{l_1}^{\text{pose}}, \ldots, \mathbf{x}_{l_n}^{\text{pose}}]$, $\mathbf{Y} = [\mathbf{y}_{l_1}^{\text{pose}}, \ldots, \mathbf{y}_{l_n}^{\text{pose}}]$ are formed from $\mathcal{L}$. The map Φ^{pose} is then defined as the mode $\mathbf{y}_*^{\text{pose}} = \sigma^{-2}\mathbf{Y}\mathbf{X}^\top \mathbf{A}^{-\top}\mathbf{x}_*^{\text{pose}}$ of the predictive distribution for test input $\mathbf{x}_*^{\text{pose}}$. Computing Φ^{pose} is equivalent to ridge regression. The Bayesian framework provides the variance $\mathbf{x}_*^{\text{pose}\top}\mathbf{A}^{-1}\mathbf{x}_*^{\text{pose}}$ of the predictive distribution which corresponds to the uncertainty on the made prediction. This information is exploited as metric $q(\mathbf{x}_*^{\text{pose}})$ for suggesting label candidates as shown in Section 8.3. We apply Φ^{pose} to live control motions in the next section.

8.5. Live character animation

Pose mapping

With pose mapping Φ^{pose}, we can now sequentially map endless live control motions $[\ldots, \chi_{t-1}, \chi_t, \ldots]$ to create new target motions $[\ldots, \gamma_{t-1}, \gamma_t, \ldots]$. Input χ and output γ are novel, and are not contained in the training data $[\mathbf{x}_1, \ldots, \mathbf{x}_M]$, $[\mathbf{y}_1, \ldots, \mathbf{y}_N]$. As summarized in Figure 8.6, synthesizing each output frame γ_t is performed in two steps: 1) Inferring the pose γ_t^{pose} by Φ^{pose} on χ_t^{pose}, constrained by the latent volume; and 2) Applying the global translation map and rotation map $\Phi^{\text{trans}}, \Phi^{\text{rot}}$ on the newly synthesized pose γ_t^{pose} to obtain global motion velocities $\dot{\gamma}_t^{\text{trans}}, \dot{\gamma}_t^{\text{rot}}$. Finally, these are integrated to yield $\gamma_t^{\text{trans}}, \gamma_t^{\text{rot}}$.

Bounded result

The latent volume constrains a new pose estimate γ_t^{pose} by projecting onto the PCA space and clipping each PCA component c to the respective interval I_c. The effect is that of a high-dimensional bounding box in $\mathbb{R}^D$ whose sides are aligned with the principal axes of pose variation. This ensures that the outputs are strictly contained in the volume spanned by $\mathcal{U}_\mathbf{y}$, and unwanted deformations of the target character are prevented (see Section 8.6.1). Except for the latent volume bounding, all operations are linear and only depend on the most recent frames. This allows real-time control.

Orientation control

In principle, a rotation map Φ^{rot} could be learned in a similar way to Φ^{trans}; however, in practice this requires target sequences to contain sufficient examples of character rotation, and this is often not the case. Instead, the target rotation is controlled

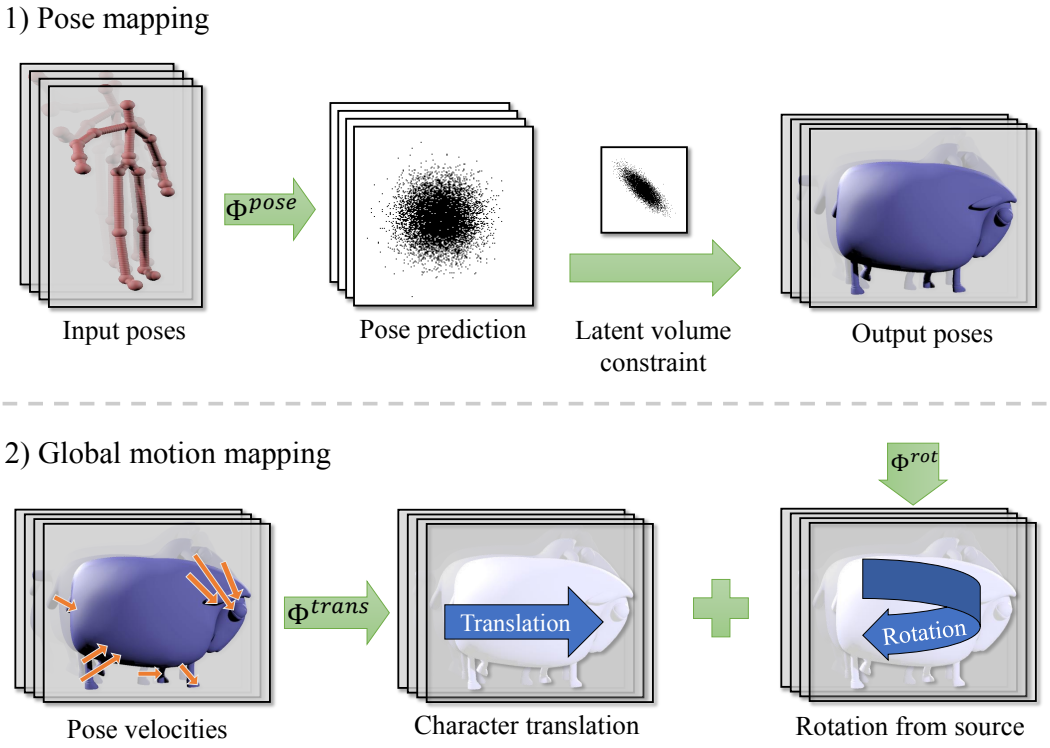


Figure 8.6: Synthesis: 1) Given a new input pose, the distribution of most likely target poses is inferred through Φ^{pose} (Section 8.4) and constrained by the latent volume to obtain the final target character pose. 2) The global translation is inferred through Φ^{trans} from pose velocities (Section 8.2.1) and source rotation is mapped to output rotation by Φ^{rot} (Section 8.5).

directly from source rotation χ_t^{rot} through $\Phi^{\text{rot}} : \chi^{\text{rot}} \rightarrow r\chi^{\text{rot}}$, with r manually adjusted to the agility of the character. Practically, this leads to intuitive control where rotating the body in front of Kinect also rotates the character.

8.5.1. Mesh reconstruction

Mesh reconstruction

While the motion mapping is simple, we still need to return from our dedicated feature space. This is more complicated for mesh sequences as we decompose them into vertex positions $\mathbf{v}_i$ and per-face rotations and shears $\mathbf{A}_f$, and project these into PCA space (Section 8.2.1). Thus, given a new pose γ_t^{pose} in feature space, first, the PCA coefficients are back-projected to obtain face transformations $\mathbf{A}_f$ and positional features $\mathbf{v}_i$. After the mapping procedure, the transformations of $\mathbf{A}_f$ and $\mathbf{v}_i$ might be inconsistent and might yield a disconnected surface. We reconstruct a coherent surface by solving a Poisson system similar to the approach of [Sumner and Popović, 2004], Section 2.2.3 gives a more detailed explanation.

8.6. Evaluation

Sequence details

Our dataset consists of 8 different motions. The *CMUHuman* mocap data is from the CMU Motion of Body Database. The *KinectHuman* sequences are captured with a

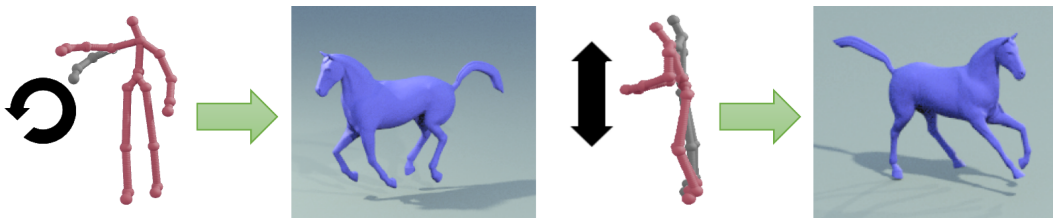


Figure 8.7: *KinectHuman-Handwave* arm motions *(left)* and *KinectHuman-Jockey (right)* knee bend motions synthesize *Horse* gallops.

Kinect sensor Shotton et al. [2011]. The *LeapMotion* sequence from the Leap motion controller [LeapMotion]. These three sequences are all 3D point-based, representing the 3D joint positions of the full body and hand. *Face* contains 66 2D feature points tracked automatically from a video sequence. *Elephant*, *Horse*, *Caterpillar*, and *Sheep* are mesh sequences created by artists containing different motions. The data acquisition processes for *KinectHuman* and *LeapMotion* are real-time. Table 8.1 details these sequences and the number of correspondences between them.

Hyper parameters

The (hyper-)parameters of our algorithm are: temporal motion derivative weight $w = 0.1$ (Section 8.1.2), latent volume dimensionality $D = 50$ (Section 8.2.1), pose and translation regression regularization parameters $\sigma = 0.05, \sigma_{\text{trans}} = 0.001$ (Section 8.4), and per-character rotation angle r (Section 8.5). Except for r, which is a data-dependent fixed ratio, these parameters were tuned once and fixed throughout the entire set of experiments. Experiments were performed at 30-40fps on an Xeon 3.6Ghz Quad-Core for meshes of 3-16k triangles.

Full-body motion examples

Figure 8.7 (left) shows how *KinectHuman-Handwave* is transfered to *Horse*. Although our algorithm was not explicitly directed to focus on the right hand in synthesizing the target, it successfully inferred the desired effect from only 4 correspondences. The hand waving speed controls the target motion speed. Any combination of source and target is possible—we replace *KinectHuman-Handwave* with *KinectHuman-Jockey* in Figure 8.7 (right), and now knee bounces control the target.

Motion capture input

Figure 8.8 (left) shows an example where a *CMUHuman* walk is transferred to a *Caterpillar* that has different topology in static shape and demonstrates completely distinct dynamics in motion. With only 4 labels, our algorithm successfully recovered the underlying correspondence in class. Furthermore, this mapping can be directly applied to other source motions, and we puppet the caterpillar with stylized *slow*, *medium*, and *fast* walks.

Controlling multiple motion classes

Figure 8.8 (right) shows a more challenging scenario with walking, leaning, raising of the thorax, and jumping. This wider range of motions was covered by 8 interactively defined example correspondences. In Figure 8.11 and, we show that the linearity of our mapping allows for realistic superposition of motions such as turn and crawl, and head-lift while bending.

2D input

In Figure 8.9, we show the transfer of facial motion between a human (*Face*) and a sheep (*Sheep-Face*) using only 6 example correspondences, which were defined using the automatic correspondence suggestion.

Animation

Figure 8.10 shows the last example in which a *KinectHuman* controls a *Sheep*. Local

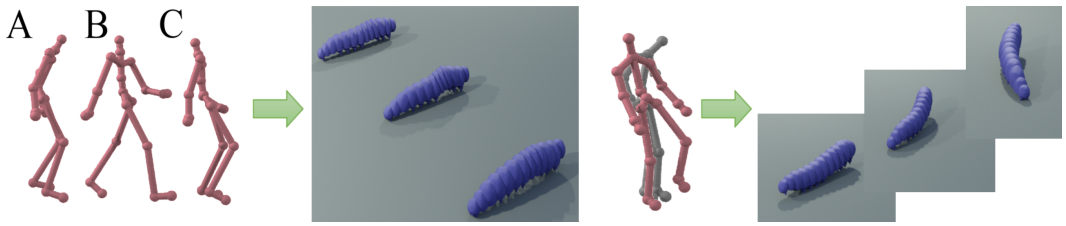


Figure 8.8: Examples of motion transfer to a caterpillar. *Left:* Mapping learned from *CMU Human-Walk* style C and applied on different styles A,B,C varying in speed and step size, which displays the capability of style transfer. *Right: KinectHuman-Various* and the synthesized *Caterpillar*.

Figure 8.9: Examples of facial motion transfer learned from 6 correspondences. The first and second columns: *Face* and the second and fourth columns: the synthesized *Sheep*.

arm and global body motions of the source sequence are assigned to walking and jumping of the sheep based on 6 correspondences. Our algorithm generalizes these different classes of motions well. In practical applications, a fine-detailed geometry (such as fleece) can be added to generate a visually pleasing character.

8.6.1. Comparison to alternative approaches

Representation comparison

The advantage of the combined shape space is shown in Figure 8.13 with a map from an *Elephant* to *Horse*. Raw point features lead to distortions when synthesizing rotational motions and the deformation gradient representation is agnostic to vertical translation. The latent volume and use of unlabeled data is compared to pure PCA in Figure 8.12. The gain from the remaining method components, such as the velocity features and regularization, are especially noticeable in motion; the supplemental video published alongside [Rhodin et al., 2014] shows these gains with a comparison of our model to weighted nearest neighbor lookup and to the method of Yamane et al. [2010]. The quantitative improvement of different model components are analyzed in the following section.

8.6.2. Quantitative analysis

Ground truth definition

We wish to compare our animation results quantitatively, e.g. as the mean Euclidean distance to a ground-truth animation. Since there is no obvious ground truth in motion mapping, we employ the different strategy of dividing user-provided correspondences into a learning set and a disjoint validation set as ground truth. For quantitative evaluation, we measured the errors of the outputs of our algorithm for the *KinectHuman-Horse* sequence (Figure 8.12).

Table 8.1: Target character frame length, walk cycles frame length, and the number of source-target pose correspondences for each sequence. † denotes Kinect and Elephant.

Source char.	Kinect	† Kinect		CMU Leap,	Face Mocap	
Target char.	Caterp.	Horse	Sheep	Caterp.	Sheep	Lamp
Target frames	700	12	57	100	100	35
Target walk cycle length	20	12	18	150	n/a	n/a
# correspondences	8	4	6	4	6	8

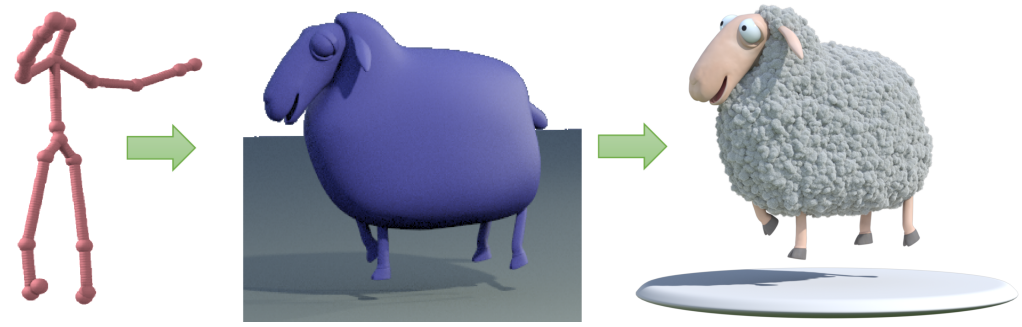


Figure 8.10: With appropriate rendering, our system can create expressive characters. We integrate our motion transfer method into a professional animation-rendering pipeline to show transfer from sparse input to fine geometry.

SNR metric

For a given set of output meshes of our algorithm $\mathbf{Z} = [Z_1, \ldots, Z_{T'}]$ and the corresponding ground truths $\mathbf{G} = [G_1, \ldots, G_{T'}]$, the results are summarized based on the signal-to-noise ratio (SNR):

$$\mathrm{SNR}(\mathbf{Z}|\mathbf{G}) = 10 \log_{10} \frac{\mathbf{V}}{\mathbf{E}}, \tag{8.4}$$

where the error $\mathbf{E}$ is defined as

$$\mathbf{E} = \frac{1}{T'} \frac{1}{N_v} \sum_{i=1}^{T'} \sum_{j=1}^{N_v} \|Z_i^j - G_i^j\|^2 \tag{8.5}$$

with N_v being the number of vertices in each mesh. The variance $\mathbf{V}$ is calculated in the same way as $\mathbf{E}$ except that each ground truth is replaced by the mean mesh of $\mathbf{G}$. Figures 8.14 and 8.15 show the results. For comparison, we also show the results of variations of our algorithm constructed by replacing each system component with an alternative, to show the importance of each step. In Figure 8.15, 'latent volume' denotes our original algorithm while 'PCA only' implies removing the projection onto the estimated support in PCA space (i.e., only axis rotation plus reduction of dimensionality to 50).

Cross validation

The results were generated based on 50 correspondences from which a varying number T' of examples (correspondences) were used for training (x-axis in the figure) while the remaining $50 - T'$ examples were used for evaluation. For each T', we generated 10 different combinations of training and evaluation sets and the mean SNRs were calculated. The results show steady increase of SNR as T'

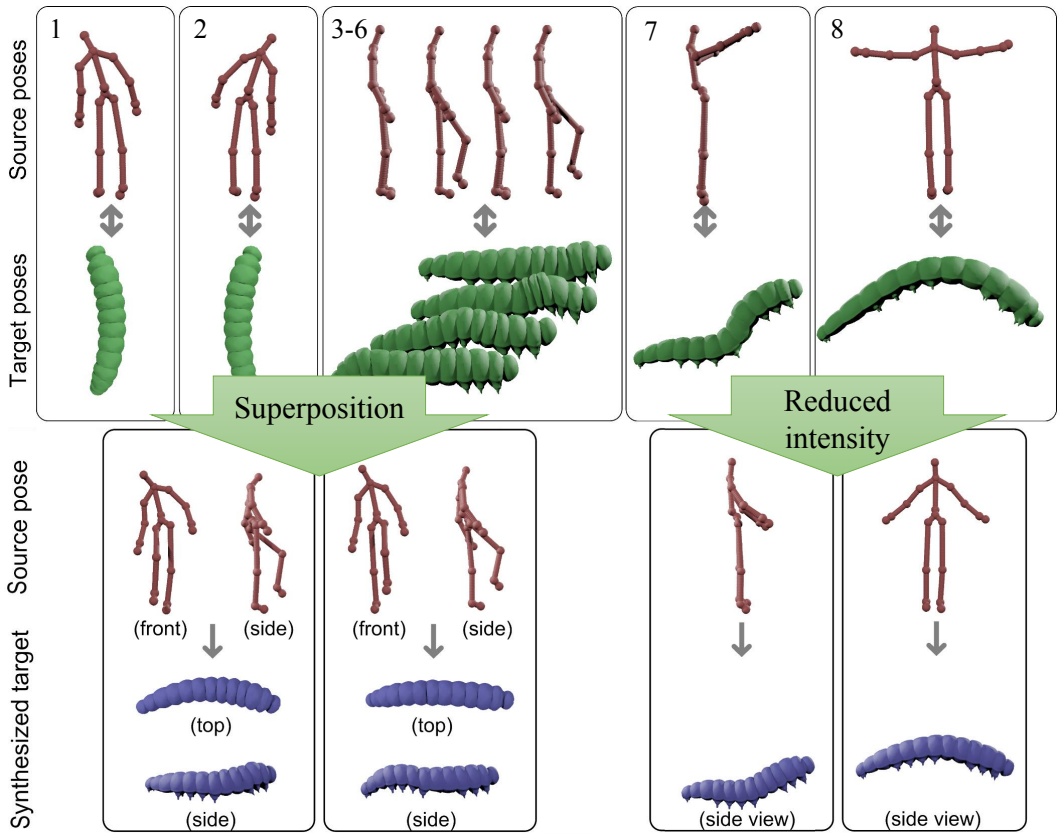


Figure 8.11: *Top:* Correspondences labeled between joint positions and caterpillar mesh to control bending (1-2), crawl (3-6), head lift (7) and jump (8). *Bottom:* New pose synthesis of the caterpillar character showing intermediate and newly inferred superposition of, e.g., bend and crawl.

increases as expected. The use of latent volumes (specifically, the use of unlabeled priors) is justified by the improvement in performance over regular PCA.

Approximation impact

Interestingly, the PCA dimensionality reduction degrades the performance when the resulting dimensionality is low (20 and below). In our preliminary experiments, we observed that the performance improves as the dimensionality increases, and eventually, it approaches to the level of not using dimensionality reduction at all. This suggests that the feasible set of motions does not lie on a low-dimensional linear sub-space of the data space. While, in theory, this problem could be addressed by adopting non-linear representations, the high computational complexity of existing algorithms is unsuitable for real-time systems.

Latent volume impact

As demonstrated in results, our latent volume can be regarded as a practical trade-off between complexity and flexibility. Figure 8.14 shows the results of additional variations of our algorithm with different choices of regression functions: here, 'Latent volume' (without velocities) denotes the setup where the first-order motion information is removed from the input features while 'Polynomial' denotes a non-linear regression implemented by performing linear regression with monomial features $[x^{1/2}, x, x^2]$ for each feature x. For these variations, the regularization parameters were manually tuned in favour of SNR. The superior performance of our algorithm 'with velocities' indicates the advantage of exploiting the temporal

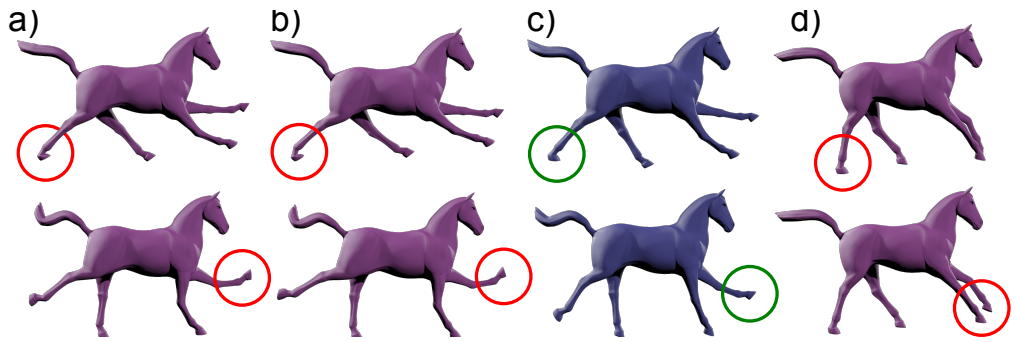


Figure 8.12: Comparison of our method to related mapping techniques. The horse pose is synthesized from Kinect input through a) a linear map without unlabeled examples, b) in a PCA latent space of dimension 10, c) by our method, and d) our method without unlabeled examples. Artifacts are marked in red, successfully regularization in green.

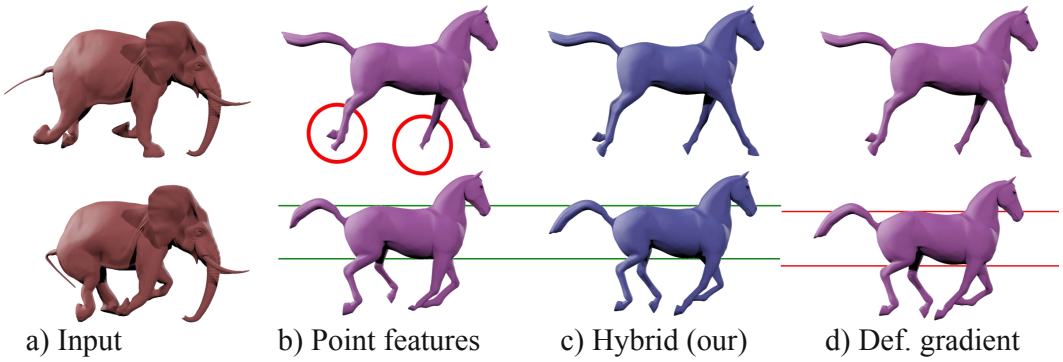


Figure 8.13: Result of deformation transfer using a) point features that lead to shrinkage artifacts, b) our artifact free representation, and c) the deformation gradient representation which does not capture vertical translations.

data for motion synthesis. Polynomial regression overfits to the data even with regularization, that can be attributed to the lack of labeled correspondences. We expect that, in general, when the number of examples is very large, a non-linear regressor (e.g. the polynomial regressor) should perform better than a linear regressor.

8.7. Discussion

Interactive and real-time

Our final algorithm is a design that reflects the requirements imposed by the application scenario: interactive correspondence declaration, real-time mapping, and versatility to very different source and target sequences. Here, we briefly discuss how these choices were made.

Speed vs. complexity

Linear regression The choice of a linear regressor over non-linear algorithms is not driven by the complexity but by the wish to succeed with sparse labels. With only 4-8 examples, the training and learning time of Gaussian process regression is comparable to linear regression but requires careful tuning of additional parameters. However, in our setting, the quality of outputs obtained from the linear regression was approximately equivalent to the output from non-linear ones.

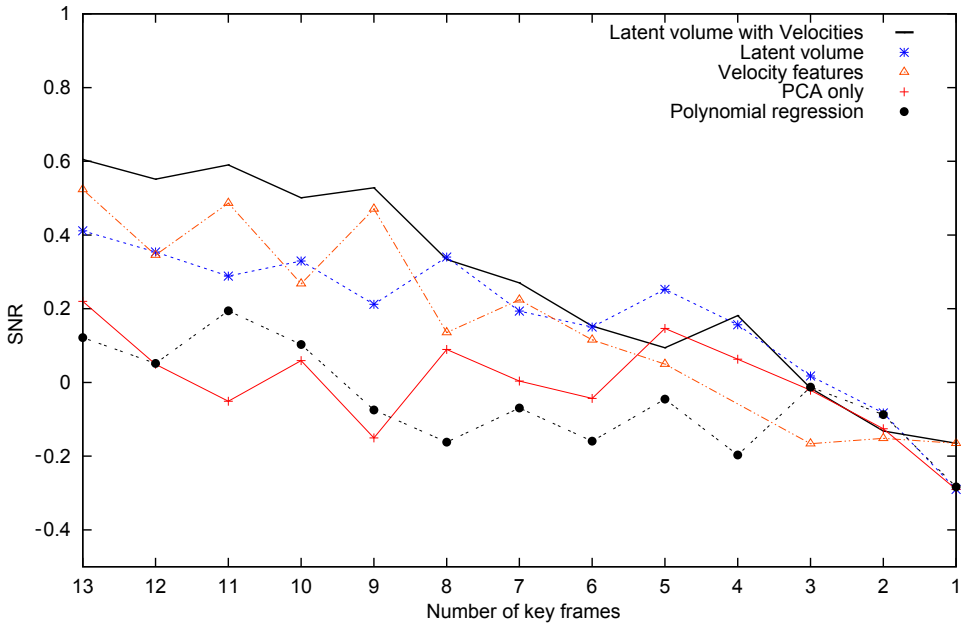


Figure 8.14: Motion transfer performance as a function of the number of training examples (key frames) with different choices of feature and regression function combinations, quantified as the SNR. Our full model (Latent volume with velocity features) consistently outperforms the baseline.

Ill-posed problem

Even for the linear regressor (with limited complexity), the problem is severely ill-posed as the number of correspondences is much smaller than the dimensionality of input and output data points. Exploiting a priori knowledge is essential for regularizing the regression process. Fortunately, unlike typical regression, we exploit unlabeled examples $\mathcal{U}_\mathbf{y} \subset \mathbb{Y}$ to estimate a latent volume $\mathbb{Y}'$ that effectively encodes and limits the plausible motion range.

Latent volume One of our main contributions is that we demonstrate that a simple orthogonal transform followed by independent bounding of variables is effective for regularizing the regression process (see Section 8.2.1). Recently, automatic approaches have been proposed for implicitly inferring the constraints from the data using principal component analysis (PCA) or GPLVM.

Alternative latent spaces

In preliminary experiments, classical use of PCA for estimating constraints—PCA followed by significant dimensionality reduction—was insufficient for our purpose. Furthermore, flexible GPLVMs or sophisticated non-linear density estimators are too inefficient for interactive applications due to their high computational complexity in learning and prediction. Learning can take several hours even for low dimensional skeleton characters [Levine et al., 2012], and prediction has yet to be achieved in real time [Yamane et al., 2010].

Speed vs. complexity

Our latent volume representation is the result of the trade-off between performance and quality. In general, when properly regularized, sophisticated non-linear algorithms should lead to as good or better representations of plausible motion spaces; unfortunately, these algorithms cannot be used in real-time applications. Direct bounding by the sparse correspondences without considering the unlabeled data for decorrelation and support estimation is insufficient as it improperly restricts

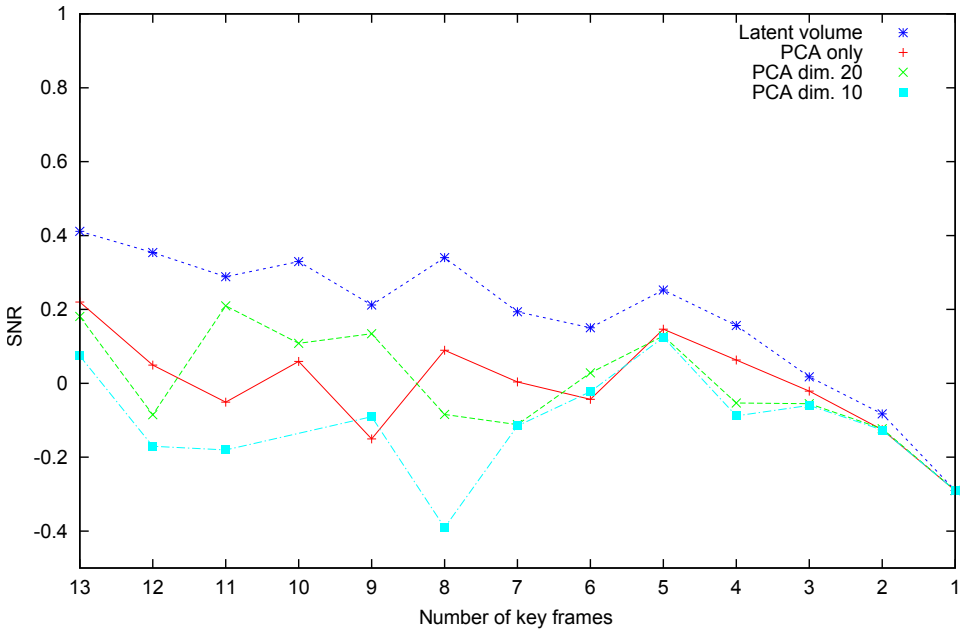


Figure 8.15: The effect of latent volume representation for different numbers of training examples (key frames), quantified as the SNR. The proposed latent volume is more effective than simple dimensionality reduction.

the admissible range and thereby leads to jerky animations.

Latent volume, a prior

From a probabilistic point of view, deriving the latent volume from the available unlabeled data corresponds to finding the support of probability distribution $P_{\mathbf{y}^{\text{pose}}}$ which is applied as a prior to infer the admissible variations. In general, finding the support of a probability distribution is a non-trivial problem. However, by assuming that $P_{\mathbf{y}^{\text{pose}}}$ is approximated by a Gaussian distribution, we find an efficient estimate of its support by first decorrelating the data and then finding the support of the resulting empirical distribution within each data dimension. That is, the latent volume defines the support of $P_{\mathbf{y}^{\text{pose}}}$ and is specified as the minimum interval I_c encompassing all decorrelated data points for each dimension c. In the context of the regression problem, the latent volume may be interpreted as an approximate prior distribution $P_{\mathbf{y}^{\text{pose}}}$ on the target character pose.

8.7.1. Pose mapping limitations

Collisions and scene interaction

During synthesis, we do not address collision detection, interaction of multiple components, or interaction with the environment. To achieve this, a post-processing similar to Yamane et al. [2010] would have to be extended to work in real time on mesh characters.

Scalability

Our algorithm is only applied to single source/target sequences. For complex puppetry applications, it might be helpful to allow multiple humans to control (different parts of) the same target. Conversely, a single human could control the movement of a swarm of small characters as a whole.

Expressiveness

Currently, our method yields too coarse a control to steer characters with broader

dynamics. For example, only the basic facial expressions are transferred to the sheep in Figure 8.9. The utilized linear regression and latent volume representation needs to be extended to more flexible non-linear alternatives while retaining the computational efficiency for real-time applications.

8.8. Summary

Interactive definition real-time control

We have presented a data driven method for real-time performance-driven character animation that offers fast and flexible interactive motion mapping. Our contribution is a mapping from source to target characters that ensures robustness by limiting the range of admissible poses through the use of a dedicated latent volume. Unlike other approaches, we interactively define only 4-8 correspondences between sequences, and so we must robustly learn this mapping by using unlabeled frames. This allows puppetry with many different motion sources such as body-, face-, and hand-tracking systems. Combined with the automatic correspondence suggestion, this leads to an animation system for quickly defining mappings between very different shapes and motions, and then intuitively controlling characters in real time.

From pose to motion

In the next chapter, we build on this work, improving on limitations caused by the chosen instantaneous pose mapping. We look for alternative character representations which encode not only the feasible character poses (latent volume), but entire motion sequences with varying and plausible dynamics, while preserving the general representation of arbitrary character shapes in mesh form. Furthermore, we seek a mapping which is able to transfer dynamics of the input onto non-human characters, such as control speed and intensity, as well as additional high-level attributes such as the users facial expressions, e.g. happy or sad face. This is in particular challenging for our setting of low-cost input sensors with limited accuracy and real-time application, and is of high importance for performance-driven VR applications, where typical interfaces may be impractical due to impaired real-world vision.

Generalizing wave gestures from sparse examples for real-time character control

9

This chapter is based on Rhodin et al. [2015b]

Retargetting and gesture recognition

Gestural and motion-driven character animation has been approached from two primary directions: Retargeting motion controllers [Gleicher, 1998] map human and character bodies 'one-to-one' at the bone level. While expressive, character motion is consequently restricted to the sensing volume and to the ability of the user to perform complex motions, with mappings hard to generalize to non-human characters. At the other end of the spectrum, gestural control-action pairs [Johnson et al., 1999; Raptis et al., 2011] trigger character motions from a database, but there is no control of motion style variation through extrapolation.

Current limitations

The techniques introduced in Chapter 8 [Rhodin et al., 2014] and by Seol et al. [2013] lie in-between, by mapping poses to character animations. This forfeits retargetting, but expands control flexibility and expressivity over gesture-action pairs, and is a good fit for games. However, there are problems: pure pose mappings are under-constrained for complex character motion, and so velocity and acceleration features add constraints. These are sensitive to noise, and may result in jerky or stilted animations. Furthermore, a user moving his legs faster will induce a faster playing walk animation, when the desire is for a progression of dynamism from walking to running. While in principle this could be accomplished with multiple maps, in practice this is difficult and better extrapolation of control tempo is needed.

Our Fourier domain approach

Inspired by Fourier domain representations [Unuma et al., 1995; Shiratori and Hodgins, 2008], we robustly estimate instantaneous wave properties—amplitude, frequency, and phase (AFP)—of high-dimensional user pose motions. Our versatile method allows interactive part- or full-body control motion definition via diverse capture sensors. Key to our approach is the separation of simultaneously-performed control motions into independent low-dimensional waves. We learn this mapping interactively from a single example per control motion, such that wave properties can be estimated by windowed Fourier analysis and extrapolated to control motion variations. With a mapping learned, instantaneous real-time estimates are still challenging as future user input is unknown. We overcome this by exploiting the temporal dependence of phase and frequency.

Cyclic motions

Cyclic animations are common, and our wave gestures are often well suited to

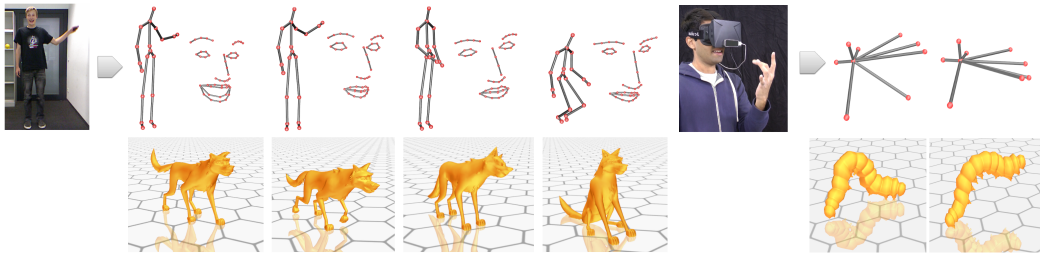


Figure 9.1: *Top:* Skeletal, facial, and hand motions are tracked by off-the-shelf sensors. From sparse user-defined examples, our method reliably separates and extrapolates simultaneously-performed gestures to control characters in games or VR. *Bottom left:* Dog happy walk, neutral run, shaking, and sitting. *Bottom right:* Caterpillar crawl variations, controlled by varying amplitude, phase, and frequency of input gestures.

cyclic control tasks, e.g. for walking locomotion, frequency maps to number of steps per second, amplitude to step size, and phase to walk progression. This allows control over variations in style which would be complex for existing pose mapping approaches, such as a slow wide steps or fast narrow steps. Non-cyclic linear motions are also needed, e.g., sitting down or raising an arm. Seol et al. [2013] classify input motions and blend different pose mappings to improve cyclic motion control. We extend this idea by allowing superpositions of both linear pose mappings and cyclic motion mappings, all within a motion graph. Furthermore, our approach extends to additional input modes with shared parameter spaces, for instance, a face tracker measuring users mood to control character emotion.

Quadruped space-time interpolation

We generate new animations by interpolating wave properties within parametric animation database spaces. Our general interpolation scheme operates on meshes. It is based on the representation introduced in Section 8.1.3 and is independent of a character rig or skeleton, supports arbitrary character shapes and topologies, and enables foot sliding prevention. For locomotion, often quadrupeds switch the phase of individual limbs between walks to runs. We propose to reduce classical artifacts by aligning limbs individually by time warping, and interpolating limb timing differences separately from limb positions. This minimizes the size of the character animation database needed for smooth dynamic state changes.

Our improvements

We show that wave gestures are more robust to noise and operate over a larger temporal domain than commonly-used velocity and acceleration features [Seol et al., 2013; Rhodin et al., 2014, Chapter 8], and that they successfully decorrelate ambiguous control motions, which is not possible with existing approaches [Shiratori and Hodgins, 2008]. Furthermore, with ten participants in three game-like quantitative tasks and in qualitative questionnaires, we discover that our wave gesture control is more intuitive and more accurate for animation tempo control than a gamepad, but is more physically demanding. For applications, we interviewed three animation professionals, who found potential as a 'blocking' tool for early-stage content creation, for live performance, and for games.

Contributions

At a system level, we provide key features when compared to existing methods (Table 7.1). Our contributions to the literature are:

– A technique to robustly and accurately estimate amplitude, frequency, and

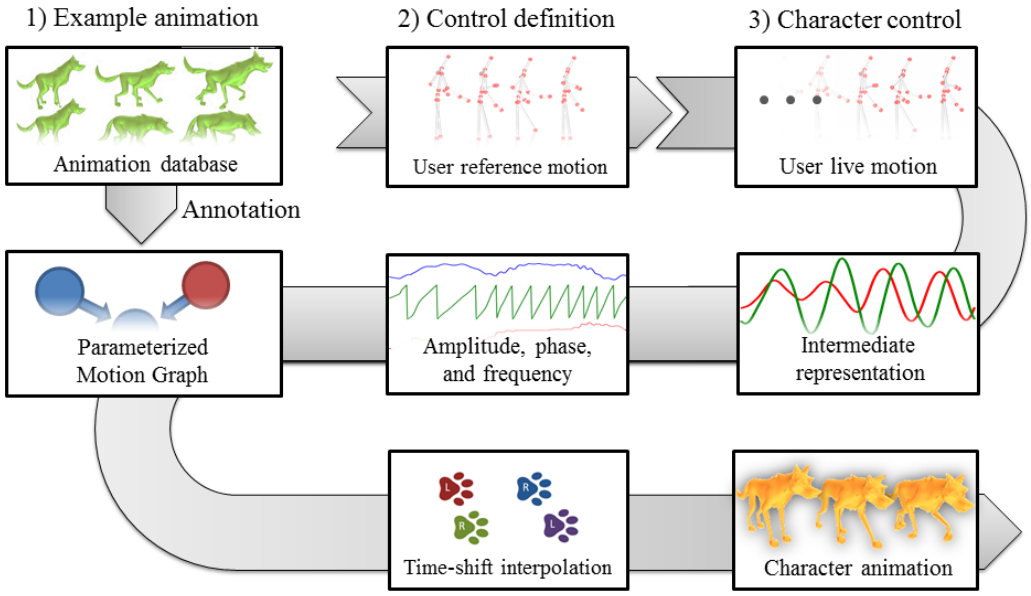


Figure 9.2: Pipeline: *1)* An animation database is created by an artist. *2)* Control definition: the user interactively performs one reference motion for each parametric motion class. *3)* Live character control: the virtual character is controlled by estimateing AFP parameters from independent intermediate representations, with animations synthesized by a new time-shift interpolation.

phase of simultaneous gestures in real time, generalized from a single user-defined reference motion.

- An interpolation method for motions with out-of-phase sub-motions that cannot be aligned by traditional time-warping.
- A live animation system, which couples wave gesture to parametric motion graphs and layers different input modalities.

9.1. Notation and overview

Three stages

Our goal is to generalize wave properties of motions from sparse examples for real-time character control. Three steps are required: *authoring*, *control definition*, and *live control* (Figure 9.2).

Overview

First, in the authoring step (Section 9.2), example character animations are arranged to form *parametric motion classes* as nodes in a motion graph, as would be typical for a game. Second, in a control definition step (Section 9.3), a *reference control motion* is specified for each motion class, from which we learn a mapping per node (Section 9.4). These motions would typically be predefined by a game designer, but as this is computationally fast it also allows for interactive end-user definition. Third, the user performs live *control motions* for real-time character control (Section 9.5). Simultaneously-performed motions are separated, and variations in motion AFP (amplitude, frequency, and phase) are extrapolated to generate new motion style variations. Animations are synthesized by database interpolation; for quadrupeds, we introduce a separate time and pose interpolation of individual limbs which improves animation quality and reduces foot-skating artifacts (Section 9.5.2).

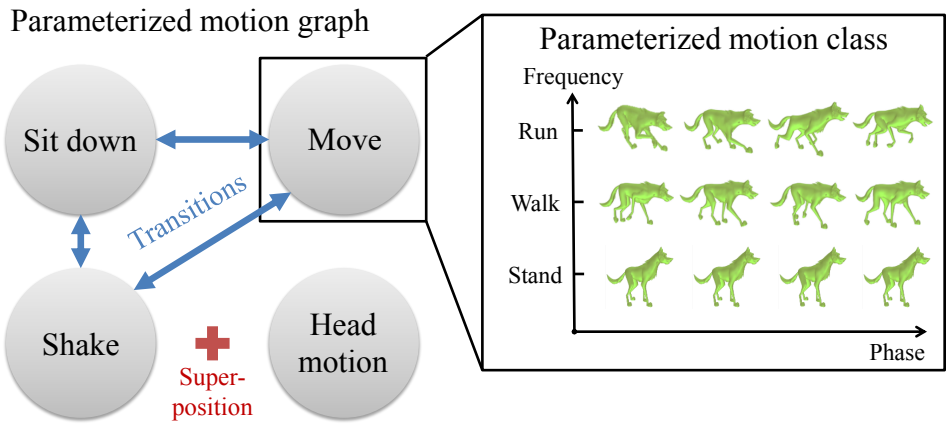


Figure 9.3: Parametrized motion graph for the dog character. Each node represents a parametrized motion class that synthesizes character animation from gesture AFP parameters. Edges mark transitions which are initiated by gesture activation. We also support superposition of secondary actions such as head motion which are additive.

9.2. Parametrized character representation

Parametrized motion graph

To create a semantic connection between user control motion variation and character animation, we arrange example animations into a vector space of parameters [Rose et al., 1998; Heck and Gleicher, 2007]. We use AFP parameters to generate live animations by interpolating the estimated and annotated AFP parameters. For instance, the user varying their leg height in a mimicking walk could be transferred to a dog character's stride length by annotating the database animations of standing and walking with amplitude 0 and 1, respectively. In a different application, amplitude could map to step height, for instance, for a horse dressage game character. The frequency parameter could trigger a transition from walk to run by the user speeding up the control motion, or could parametrize a shaking animation with varying centrifugal force. Depending on the tracker, these user motions can range from single finger movements, to facial expressions, up to full body motions. Additional annotation dimensions such as emotion (e.g., *happy*, *sad*) and heading rate (e.g., *left*, *right*) are also possible depending on the tracker.

Input and output notation

Formally, an animation Y of frame length T is a sequence of individual meshes $\mathbf{y}_t$, so $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$. Each $\mathbf{y}_t$ is a list of mesh vertex positions, i.e., no rig is required and any animation creation system can be used. T is an animation-specific variable, as different Y have different lengths. To build a vector space, each Y in a database $\mathcal{Y}$ is assigned parameters $\theta_Y = (a, f)$ to represent variations in frequency f (e.g., *fast*, *slow*) and amplitude a (e.g., *large*, *small*). As we focus on cyclic animations, the time index that specifies the current frame t of the example animation is parametrized in the cyclical domain $[0, 2\pi)$, invariant to the animation length T, by phase $\varphi := 2\pi t/T$. Y then exists in a two-dimensional vector space, with one dimension per parameter. These annotated database animations form parametric motion classes, which we combine into a motion graph (Figure 9.3).

9.3. Reference control motion definition

Control definition

Each parametric motion class requires one reference control motion X as a sequence $X = [\mathbf{x}_1, \ldots, \mathbf{x}_T]$ of poses as point positions. This could come from any tracking system. Each motion is defined by performance at an arbitrary steady speed for one period at maximum amplitude, which provides a kind of 'physical normalization' of amplitude to $[0, 1]$. For instance, one cycle of a mimicking walk, where the legs are raised as high as possible. Practically, the start frame $\mathbf{x}_1$ is marked by pressing a remote control. The end frame is automatically detected for cyclic motions by finding the pose most similar to the start pose which is at least $2/3$ periods away from the start. For non-cyclic motion classes, i.e., a dog sitting down, we require rest and extreme poses to be marked with a remote press, for instance, lowering the arm from horizontal to vertical position.

Generalization

Although we require only a single reference motion example, we are able to generalize or extrapolate to variations in the live animation step by analyzing the live control motion for differences in AFP. Furthermore, no manual mapping (e.g., Shiratori and Hodgins [2008]) or degree-of-freedom tagging (e.g., [Seol et al., 2013, Section 4.1]) is needed as it is automatically inferred by the regression method below, which makes it possible to define reference control motions in seconds.

User guidance

Furthermore, the automatic guidance method introduced in the previous chapter in Section 8.3 could be applied to support novice users in the control motion selection. As before, the suitability of the current input pose can be rated by the similarity to each frame of the already defined control motions.

9.4. Learning a user-to-character motion mapping

Live control

The goal of the live step is to map the stream V of control user poses $\mathbf{x}_t$ to the parametrized character representation according to the defined set of reference control motions $\mathcal{X}$. The live step accomplishes two tasks: separation (disambiguation) of simultaneously performed control motions for each reference motion X (Section 9.4.1), and estimation of AFP motion parameters θ_X for character animation (Section 9.4.2).

9.4.1. Separation of simultaneous gestures

Simultaneous control

Simultaneous input motions primarily occur in two situations: 1) For superposition effects, e.g. dog shaking while walking, with two or more simultaneous control motions; 2) At transitions between graph nodes like walking and jumping, future control motions may be started while current motions are gradually stopped, which we refer to as *intersecting* motions. Direct estimation of input AFP leads to interference between motions (Figure 9.15).

Intermediate representation

Instead, we separate high-dimensional input V into independent intermediate representations $Z_X = [\mathbf{z}_1, \ldots, \mathbf{z}_T]$ for each reference control motion X by linear pose mappings $\Phi_X : \mathbf{x} \rightarrow z$. Inspired by previous work in low dimensional circular

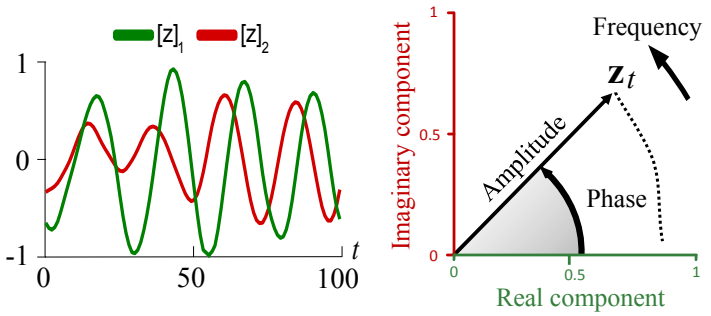


Figure 9.4: *Left:* Our intermediate representation as time-varying signal. *Right:* A polar plot with the intermediate representation as phase and amplitude. The frequency of sequence Z, and the current instantaneous sample $\mathbf{z}_t$, are represented by the dotted line.

embeddings [Lee and Elgammal, 2004] and frequency band decompositions [Akhter et al., 2010], we design Z_X as a complex sine wave (Figure 9.4):

$$\mathbf{z}_t = a_t \begin{pmatrix} \cos(\varphi_t) \\ \sin(\varphi_t) \end{pmatrix}. \tag{9.1}$$

Z_X forms a curve in polar coordinates which evolves counter-clockwise as t increases. It is a low-dimensional abstraction of input pose $\mathbf{x}_t$ which encodes phase as angle φ_t, amplitude as pointer magnitude a_t, and frequency f_t as change of phase over time.

Correspondence

Mappings Φ_X are learned by pairing X to one period of the complex sine wave $Z_X = [\mathbf{z}_1, \ldots, \mathbf{z}_T]$, with $\varphi_t = \varphi_{t-1} + 2\pi/T$, where T is set by $\mathbf{x}_1$ and $\mathbf{x}_T$. This matches the performance of one period of motion at maximum amplitude during control definition, $a_t = 1$, which sets the range of available amplitudes in the live motion control to $[0, 1]$. The initial phase φ_1 is set to the frame that is farthest from the mean. This mapping approach is similar to the one introduced in Chapter 8. The mapping is pose based, but instead of directly mapping to the target character pose (Section 8.4) the abstract intermediate representation is the target, which is further analyzed for motion properties.

Decorrelation

To separate simultaneous control motions, we enforce zero output for the remaining control motions by using them as negative examples and setting $\mathbf{z}_t = 0$. This forces Φ_X to depend on properties of the input that are unique to the reference. For instance, the two motions of bending a single finger and of bending all fingers at once can be distinguished and mapped to different motion classes without any labeling of body parts (Figure 9.15).

Ridge regression

Each map Φ_X is inferred by linear Gaussian process regression [Rasmussen and Williams, 2006], as described in Section 8.4. Each Φ_X is parametrized by a matrix M that is given as the mode of the posterior distribution:

$$p(M|X, Z_X) \propto \exp\left(-||MX - Z_X||_F^2 - \sigma_n||M||_F^2\right), \tag{9.2}$$

where $||.||_F$ is the Frobenius norm and σ_n is a regularization parameter. The linearity of Φ_X offers good extrapolation from the training sequence in that an amplified input motion leads to a proportionally larger amplitude of Z_X, and an increase in

motion speed leads to an increase of the frequency. This makes these parameters intuitively controllable. While, in general, more flexible non-linear maps can be adopted if necessary, the linear map proved to be sufficient in our experiments and was preferred over non-linear alternatives, as it is fast and it does not require tuning of additional hyperparameters.

Non-cyclic linear motions

1D linear motion

These have no notion of frequency, amplitude, and phase. Instead, we form a separate one-dimensional space z that represents the progression by $\varphi = 2\pi z, z \in [0, 1]$. During learning, the reference intermediate representation is defined as $z_t = t/T$ to map progression in the control motion linearly to φ.

9.4.2. Live estimate of motion properties

Time-frequency analysis

For notation ease, we explain how AFP motion properties are estimated for a single reference motion, and so we drop subscript X. We apply Gabor filtering, a variant of windowed Fourier analysis which has optimal time-frequency resolution [Feichtinger and Strohmer, 1998]. Gabor functions are sinusoids modulated by Gaussians $N(x; \mu, \sigma)$, where x is time, and μ and σ are the Gaussian center and standard deviation, respectively:

$$g(x; \mu, f) = \begin{pmatrix} \cos(2\pi f x) \\ \sin(2\pi f x) \end{pmatrix} N(x; \mu, \lambda/f). \tag{9.3}$$

We find the Gabor function that best fits (maximum response), and we adopt its phase, amplitude, and frequency as the instantaneous estimates $\hat{a}_t$, $\hat{f}_t$, and $\hat{\varphi}_t$. The Gabor response is the complex inner product

$$r_f = \langle [g(t-\tau; \mu, f), \ldots, g(t; \mu, f)], [z_{t-\tau}, \ldots, z_t] \rangle. \tag{9.4}$$

We filter a history of $\tau = 150$ frames (5 seconds) of Z with a series of 50 Gabor functions with wavelengths $1/f \in [5, 150]$ and mean μ fixed to the most recent frame t.

Alternative correlation estimation

Cross-correlation is one natural alternative (e.g. Shiratori and Hodgins [2008]); however, we chose Gabor filtering as phase has an analytic form $\hat{\varphi}_t = \text{atan2}([r_f]_2, [r_f]_1)$ which led to higher accuracy in our experiments (Figure 9.5; Section 9.6) vs. the required discrete phase sampling for cross-correlation.

Adaptive delay

Previous methods smooth the input signal temporally to overcome tracking noise and errors from user imprecision, which is essential for estimating velocity and acceleration. However, deciding smoothing window width is difficult: a large window strongly damps estimates, but small windows preserve high frequency noise. In our case, the Gabor window size adapts to the input motion speed: For slow motions, the window is large and high frequency noise is effectively ignored; for fast motions, a small window preserves rapid changes. The robustness-response trade-off is exposed by λ. We choose $\lambda = 2/3$, which smooths the response over most of one period (Figure 9.5, c & d). This is not a hard delay: response is immediate but small, increasing in magnitude over time.

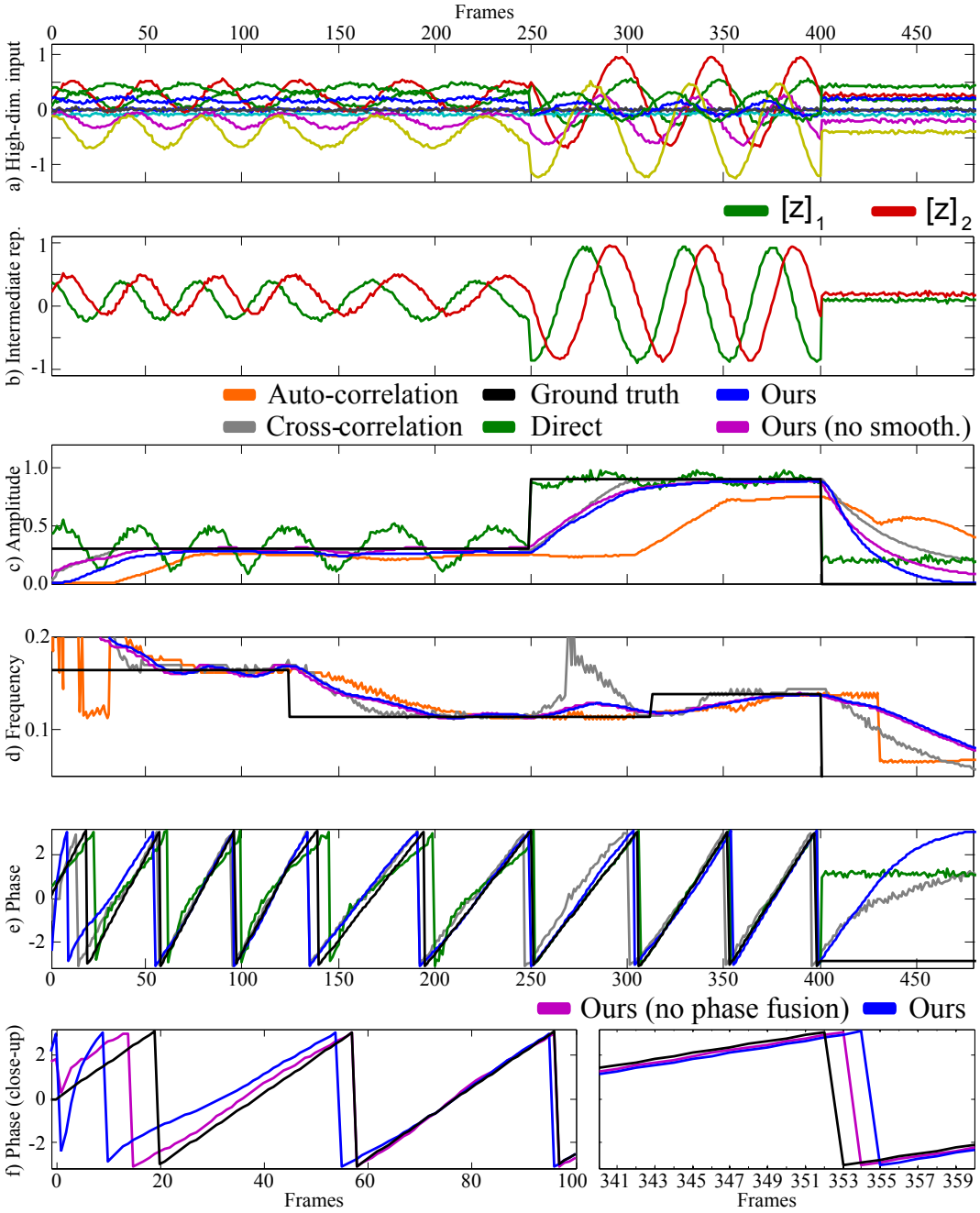


Figure 9.5: AFP parameter estimation on a synthetic sequence with ground truth: *a)* User input motion, displayed as trajectories over time, *b)* mapped onto the intermediate wave representation, *c–e)* Resulting amplitude, frequency, and phase estimates. Shiratori and Hodgins [2008] use cross-correlation (gray) for phase, which is generally noisier, and auto-correlation (orange) for frequency, which has delay of one period. Also direct estimates (green) of amplitude by $\|z\|$ and phase by $\text{atan2}([z]_2, [z]_1)$ are erroneous. *f)* Our phase fusion compensates phase discontinuities (first 15 frames) and preserves the signal otherwise (frames > 60).

Noise detection

Noise level

In preliminary experiments, the estimated amplitude was undesirably high for low signal-to-noise ratios, leading to unintended character actions. To reduce the amplitude in these cases, we exploit that noise corrupts the sinusoidal form of Z. Intuitively, given the best fit Gabor function, if Z is still not close to this perfect sinusoid shape, then the input is likely to be dominated by noise.

A good measure for how close Z is to a perfect sinusoid is the quotient of maximum Gabor response, r_f, and the total energy, n_f, apparent over the corresponding Gaussian window, with energy

$$n_f = \langle [N(t-\tau;\mu,\lambda/f),\ldots,N(t;\mu,\lambda/f)],[|z_{t-\tau}|,\ldots,|z_t|]\rangle. \tag{9.5}$$

If Z is not sinusoidal, i.e., $r_f/n_f < 5/6$, then the estimated amplitude, $\hat{a}$, is linearly damped to $a = \hat{a}2(s-1/3)$. We show in our experiments that this scaling effectively reduces the amplitude if the signal cannot be uniquely assigned to the control motion.

Discontinuity compensation

Frequency domain smoothing

As future input is unknown in our live setting, the Gabor function (and its Gaussian window) is one-sided: it is not smooth as it has a sharp edge. Hence, strong noise from partial tracking failure is possible, as are multiple local maxima within the filter response due to fast input motion frequency switches. These can lead to strong discontinuities in the estimated phase and frequency. To compensate for drastic changes, discovered frequency $\hat{f}_t$ and amplitude $\hat{a}_t$ parameters are smoothed over time to f_t and a_t by a small Gaussian of $\sigma = 200$ms, respectively. This is different from smoothing the input poses as high frequencies are preserved, and is closer in spirit to ease-in and ease-out effects.

Phase smoothing

To stabilize phase, we exploit the temporal dependency of frequency and phase. The instantaneous phase estimate $\hat{\varphi}_t$ is fused with its previous estimate φ_{t-1} and phase speed $2\pi f_t$:

$$\varphi_t = \frac{\hat{\varphi}_t + \gamma(2\pi f_t + \varphi_{t-1})}{1+\gamma}, \tag{9.6}$$

where $\gamma = 10$ was empirically set to balance integrated and estimated phase, and computation is in the circular domain $\varphi_t \mod 2\pi$. One could integrate φ_t directly from f_t and φ_{t-1} (i.e., $\gamma = \infty$), but this decouples the phase of the user control motion from the character animation, and so leads to less control. The fusion effect is visualized in Figure 9.5 e–f.

Under-constrained input

Ill-posed cases

For very simple input motions which show variation in a single dimension, such as raising and lowering an arm periodically, the mapping to the two-dimensional intermediate representation is under-constrained. We adopt a heuristic on the uncertainty of the prediction: The predictive variance ϑ_t corresponding to the

input $\mathbf{x}_t$ is estimated as the average pair-wise distance of the $n = 10$ reference intermediate representation frames that were assigned to the n nearest neighbors of $\mathbf{x}_t$ in the control definition step. This is a good estimator when the given input $\mathbf{x}_t$ is close to the training data [Kwon et al., 2015], as in our reference/live setup.

1D cyclical motion

The estimated variance reduces the influence of the potential unreliable component: we average variances over the Gabor filter window and weight the influence of the real and imaginary component of Z by their reciprocal, respectively. For complex motions both dimensions have full weight, as their variances are similarly low, which increases robustness to noise.

Emotion and direction

Additional dimensions

We estimate user emotion with a face tracker, which is an additional parameter dimension e for character control. Furthermore, the body orientation of the user with respect to the input device is transferred to control the character rate of turn β. This second additional parameter dimension improves the quality of animations such as bending the long body of a caterpillar turning.

9.5. Live character animation

Gesture activation

Given a user control motion stream $[\ldots, \chi_{t-1}, \chi_t]$ observed until t, parameters $\theta_X := (a, f, \varphi)$ are estimated for all control motions $X \in \mathcal{X}$ as described in Section 9.4.2. Here, we explain how θ_X is used to initiate transitions in the motion graph (Section 9.5.1) and to synthesize the actual animation of the character (Section 9.5.2).

9.5.1. Motion graph and motion transitions

Motion graph

Virtual characters often have their motion defined by an animation database, as is typical in video games. We organize an animation database into a *parametrized motion graph*. Multiple parametrized classes are connected into a motion graph, with edges marking possible transitions between classes.

Motion separation

By graph construction, nodes with an edge distance greater than two are independent, which increases gesture scalability. We trigger transitions along an edge by varying the activation of simultaneously-performed control motions, similar to the method of Ishigaki et al. [2009], which supports sequential but not simultaneous distinction. As simultaneous gestures are made independent by mapping to separate intermediate representations, we simply use the estimated amplitude a of θ_X to activate gesture X.

Transition activation

Transitions are initiated by increasing the amplitude α of the target node beyond 0.2 (maximum is 1). A transition is successful if the amplitude of the source node control motion is reduced to below 0.2. We abort the transition if the target activation sinks below 0.1. During the transition, we blend linearly over a fixed time window of half a second between the source and target. For non-cyclical motions, the progression parameter φ of θ_X is used instead of a as it measures the distance to the rest state. If desired, unrealistic transitions such as stopping during a jump

could be prevented by restricting transitions to specific points [Kovar et al., 2002] or windows [Heck and Gleicher, 2007].

Secondary animation

Superposition

Secondary control motions are superimposed onto the primary character animation via mesh deformations. The deformation is computed as the difference between the mesh animation controlled by the secondary control motion and the character rest pose. One example of this is lifting the torso of the dinosaur by raising the user's head, superimposed onto a simultaneously-controlled walk, or bending of the caterpillar body during turns caused by the user physically turning. Superimposing AFP mappings is also possible, for instance, shaking the dog's torso while walking.

9.5.2. Time-shift animation interpolation

NN-interpolation

Transitions and superpositions of multiple motion classes are synthesized by motion blending. To synthesize animations within a single motion class X with parameters equal to the most recent estimates θ_X, we build upon the radial basis function method of [Rose et al., 1998] and interpolate the nearest database animations with interpolation weights w_Y for each animation sequence Y set inversely proportional to the parameter distance $\|\theta_X - \theta_Y\|_W$, where W normalizes each dimension to $[0, 1]$. The timing of the animation, i.e., the time index into $Y = [\mathbf{y}_1, \ldots, \mathbf{y}_T]$, is given by the inferred phase φ of θ_X.

Motion alignment

We use linear derivative time warping [Keogh and Pazzani, 2001] to temporally align database animations during authoring. Even after temporal alignment, the naive interpolation of different quadruped locomotion states can cause strong artifacts, such as a leg stuck half-way during a transition between walks and runs. This is because the leg actually switches phase: it moves in the opposite direction in the run than in the walk (Figure 9.6). This cannot be solved by improving global alignment methods, nor by using different character representation such as skeletons, because the problem is inherent to the motion. While rarer, this issue is still possible in bipeds if arms and legs move synchronously and then asynchronously across motions.

Separation

This problem is related to the timing and interpolation of upper and lower body motions [Ashraf and Wong, 2000; Ha and Han, 2008], and to asynchronous time warping for horse gait transitions, where legs are blended separately and timing differences are compensated gradually [Huang et al., 2013; Sung, 2013]. However, these methods only work for transitions between two motions of boned characters, and not our more general mesh character case where transition length is unknown, and where we may interpolate between more than two animations (our most sophisticated example blends seven animations in a 4D space). We also experimented with dynamic time-warping of segments instead of linear time-warping. However, this led to unrealistic changes in the dynamics of the motion.

Time and space

We solve the problem with separate timelines for individual limbs that can shift for alignment, and by separating time and shape (mesh) interpolation.

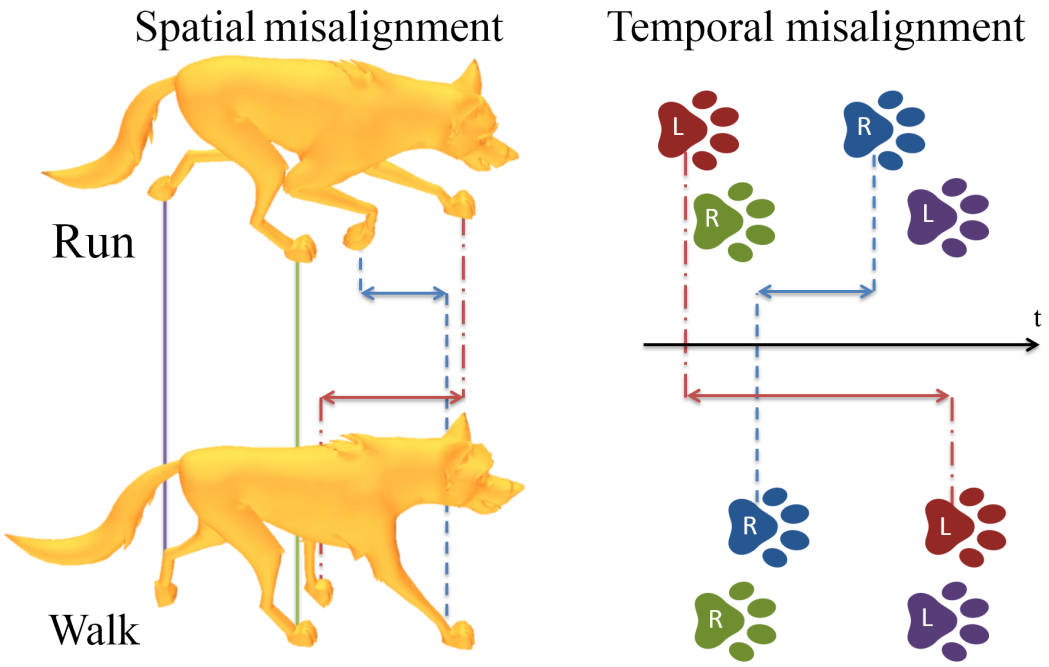


Figure 9.6: The quadruped leg interpolation problem. Interpolating between walk and run animations is difficult for many quadrupeds. *Left:* While the back legs align well spatially between the walk and run frames, the front legs switch phase. *Right:* Global linear time-warping cannot fix this temporal alignment problem as the order of foot placements switches (here, on the left-hand side).

Time interpolation

Segmentation

During database construction, the character mesh is segmented into torso and limb segments. The longest animation sequence is selected as reference and all other sequences are aligned by linear time warping, individually for each limb segment. This provides a phase offset $\nabla\varphi_{Y,i}$ for each motion Y and limb index i. During runtime, the time offsets (phase offsets) are interpolated separately for each i according to weights w_Y in the circular domain:

$$\varphi_i = \varphi + \sum_{Y \in \mathcal{Y}} w_Y \nabla\varphi_{Y,i}. \tag{9.7}$$

As each example motion Y has the same weight w_Y across all segments, the temporal dependencies between limbs are maintained implicitly if possible and are interpolated if they contradict (e.g. the order of foot plants).

Example

The timeline in Figure 9.7 exemplifies this temporal interpolation with three quadruped gaits of different foot placement beats. Feet which are temporally aligned are unchanged; only those feet that have different beat patterns are treated. This time interpolation prevents strong artifacts in the subsequent mesh interpolation, such as foot sliding and hanging limbs (Figure 9.8).

Time travel

Quick changes in weights (i.e., in user control) can cause very abrupt changes in phase, even going back in time, e.g., when switching from a positive to a negative phase offset. To prevent this, we bound the phase change of individual segments to

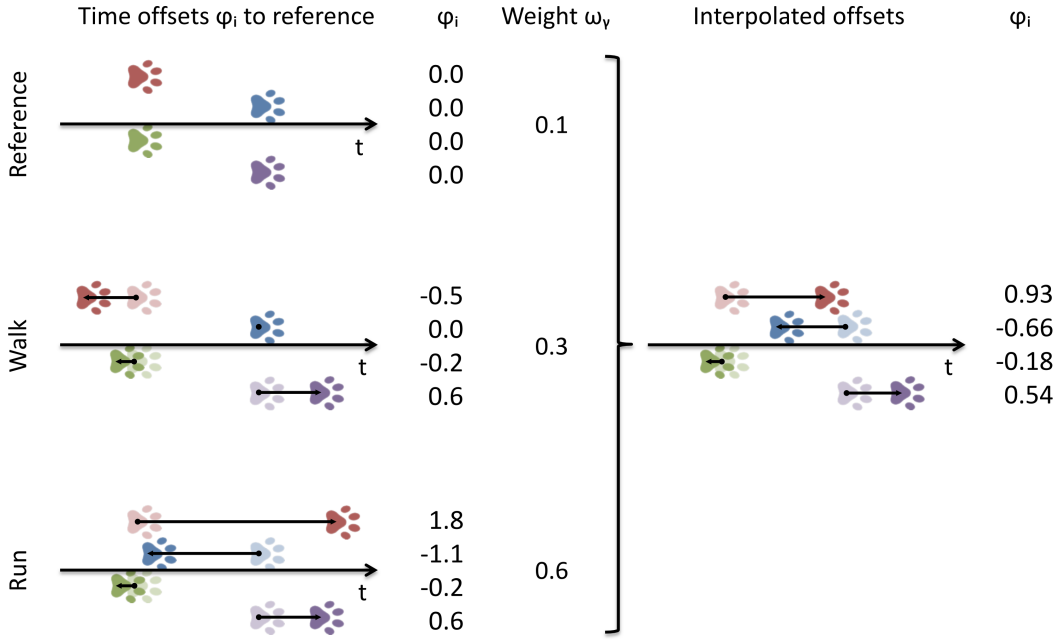


Figure 9.7: Timeline example of the interpolation of three quadruped gaits with different beats. Footplant timing differences of the four limbs (red, blue, green, violet) are shown as shifts along the timeline. Temporal interpolation is executed on the time offsets φ_i to the reference gait with interpolation weights ω_Y for gait Y.

the global animation speed by enforcing $(1-0.05)(\varphi_t - \varphi_{t-1}) < (\varphi_{i,t} - \varphi_{i,t-1}) < (1+0.05)(\varphi_t - \varphi_{t-1})$, where subscript t denotes time and i is the segment index. This effectively prevents negative phase changes and bounds the abruptness of changes to the current motion speed, i.e., during slow motions only slow changes in relative time differences are permitted, while quick motions allow for quick adaption.

Only after this temporal interpolation is the mesh interpolation performed at phases φ_i, with a differential coordinate reconstruction method which prevents seams between segments and includes global motion.

Mesh pose interpolation

Mesh representation

In this section, we provide details of the mesh interpolation given the previously-determined time offsets and interpolation weights w_Y. The interpolation of meshes is performed in a deformation space of per-triangle shear and rotation, with respect to a rest pose reference frame. A mesh segment A with F_A faces is represented by the set of shear matrices $\{\mathbf{S}_{A,i} \in \mathbb{R}^{3\times3}\}_{i\in F_A}$ and rotation vectors $\{\mathbf{R}_{A,i} \in \mathbb{R}\}_{i\in F_A}$ in axis-angle form, please consider Section 2.2.3 for details.

Mesh reconstruction

Assume that two mesh segments A, B are interpolated based on weights w_A, w_B. The triangle transformations are linearly interpolated by $S_{C,i} = w_A\mathbf{S}_{A,i} + w_B\mathbf{S}_{B,i}$ and $\mathbf{R}_{C,i} = w_A, \mathbf{R}_{A,i} + w_B\mathbf{R}_{B,i}$. From $\mathbf{S}_{C,i}$ and $\mathbf{R}_{C,i}$, a connected mesh is reconstructed from the differential coordinates of all segments by solving a Poisson system [Sumner and Popović, 2004], as explained in Section 2.2.3. This established approach

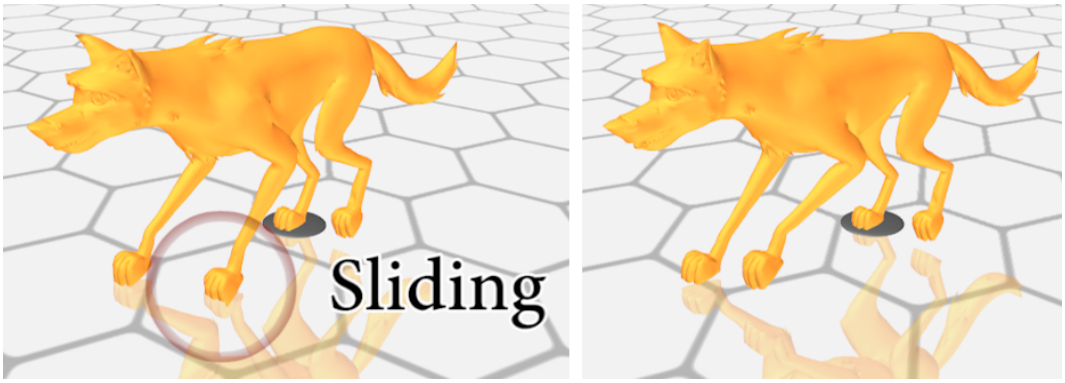


Figure 9.8: Mesh interpolation with and without time interpolation of limb segments. *Left:* Without time interpolation, the front left leg shows sliding and hanging limb artifacts, as mesh interpolation requires roughly-aligned poses. This is violated by the front leg moving forwards in one animation to interpolate and backwards in the other in the global time-warp-aligned database animations. *Right:* The proposed separate alignment and time interpolation overcomes this limitation.

is a non-linear interpolation in vertex coordinates space, which runs in real-time for meshes of $10k$ triangles on a standard PC.

Manual segmentation

The individual mesh segments are masked by painting on the character's mesh (Figure 9.10). We use masks with smooth boundaries and blend neighboring segments gradually to prevent seam artifacts. The interpolation in per-triangle rotation and shear space requires that rotations do not exceed $\pm 180°$. In our experiments this was only violated at sparse points on the shoulder of the dog character, and originates from flipped triangles in the artist created database animation. For all other characters the example animations are without flipped triangles and their interpolation shows no seam artifacts.

Global translation interpolation

World motion

Consistency between the global motion of the animated character and the ground plane is important for realistic animation. To enable this, we annotate each animation in the database that contains global motion with the character velocity in the ground plane $[\mathbf{u}_0, \ldots, \mathbf{u}_T]$. We separate the character animation frames $[\mathbf{y}_0, \ldots, \mathbf{y}_T]$ from their global velocities $[\mathbf{u}_0, \ldots, \mathbf{u}_T]$ automatically by orthogonal Procrustes analysis [Sorkine, 2009], which registers all frames against each other. During animation synthesis, we interpolate the velocities $u_{Y,\varphi}$ of all animation examples Y by the weights w_Y as before. The global position of the character is integrated from the interpolated velocities: it is the sum of all previous velocities multiplied by the respective changes in phase φ so as to accommodate for the current control frequency.

Foot-sliding cleanup

Sliding

The perceived realism of the synthesized animation also depends strongly on character feet not skating or sliding along the ground unexpectedly. To prevent this, we label database animation frames that show ground contact. During synthesis, we use the approach of Lee et al. [2010] and perform a weighted vote (contact=1, no

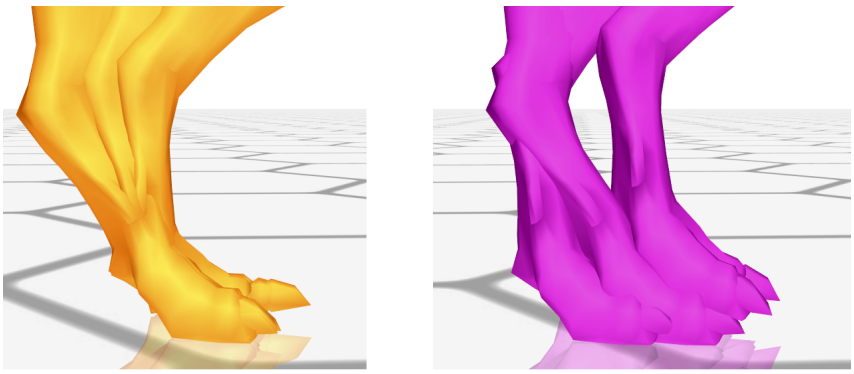


Figure 9.9: Overlay of three frames, spaced 10 frames apart, of the dinosaur's paw during a live-controlled walk. *Left:* The application of the foot-plant constraint effectively prevents foot-sliding. *Right:* In the unconstrained case, the foot slides back and forth.

contact=0) between all database motions with weights w_γ and per foot time indices from the time-shift interpolation (Section 9.5.2). The activation threshold is set to 0.9 in our experiments. During reconstruction, the constrained feet are pinned to the ground by additional vertex position constraints in our differential coordinate solver. This procedure is simple compared to more complex methods like MeshIK [Sumner, 2005], and effectively prevents sliding (Figure 9.9). Importantly, the foot constraint interpolation benefits significantly from our time shift solution to the quadruped leg interpolation problem.

Contact annotation

To determine areas of the character that should be on the ground during the locomotion cycles (i.e., vertices at the bottom of each foot), these regions are manually marked in the character mesh. By painting in 3D, this takes about 1 minute per character. Furthermore, a small area on the torso around the spine of the character location is marked, see Figure 9.10. This acts as an opposite constraint to maintain the character shape during mesh pose interpolation. Once completed, the interface propagates these vertices throughout each database sequence such that the ground contact can be annotated subsequently. This process additionally takes around one minute. For animations where the feet perfectly touch the ground during stance, the manual segmentation and annotation can be automated; however, this is usually not the case and the user must choose the desired planting behaviour when the feet are half-way down.

Contact activation

In the live phase, mesh interpolation is performed in two passes: In the first pass, the character mesh is reconstructed from the interpolated differential coordinates, without any additional constraints. Then, the second pass adds target locations for the vertices which are marked in the preprocessing. The activation of the contact constraints is determined for each foot by the discussed voting scheme. In addition, we activate the foot constraints if the foot is close to the floor and has low velocity. This proximity heuristic triggers only in very few cases and is not reliable enough to replace the voting scheme entirely.

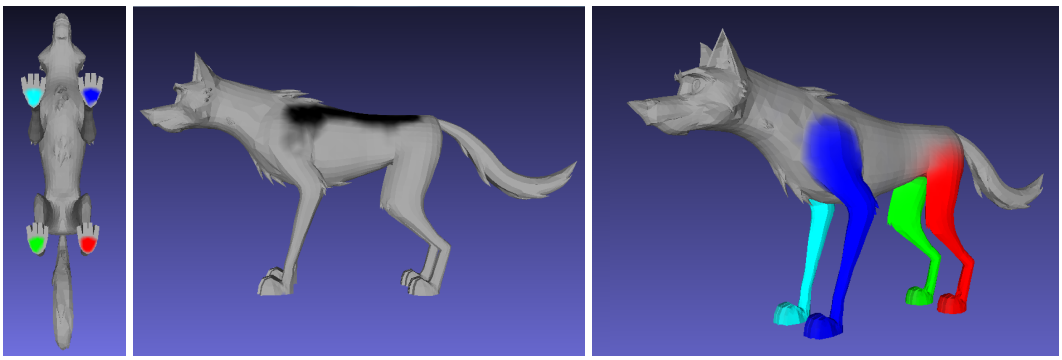


Figure 9.10: From left to right; marked footplant constraint areas, torso spine region, and segmented limbs for the dog character.

Contact constraints

All constraint vertices are positioned as follows. If the vertex constraint was activated at the previous frame, the vertex target is set to the previous location to pin the foot. If the constraint is newly activated at the current frame, the vertex target is set to the previous position with the vertical position set to the ground height to establish contact. The vertices of the feet which are in flight phase, i.e., are unconstrained, are set to their respective position from the first pass, unless they exhibit a large velocity due to a constraint in the previous frames. We clamp the velocity v of each vertex to $0.5v_{\text{first}} < v < 1.5v_{\text{first}}$, where v_{first} is the velocity of the respective vertex in the unconstrained animation from the first pass. This bounds the velocity to be similar to the unconstrained animation and effectively prevents temporal jitter of the legs in situations where the constrained foot strongly deviates from the unconstrained animation, for instance, due to rotation when running along an arc. To reduce the influence of the foot plant enforcement on the body pose, the spine segment is constrained to the first pass reconstruction at all times.

Contact enforcement

Each target position (for constrained vertices) is regarded as a hard constraint. In our implementation, we included this as soft-constraint into the Poisson solver, but with $10,000$ times higher weight than other constraints, which effectively renders it as a hard constraint.

9.6. Evaluation

5 characters real-time

We test our method on 5 characters (Table 9.1): dog, caterpillar, horse, human, and the dinosaur of Seol et al. [2013]. The experiments were performed on a Xeon CPU E5-1620 3.6 GHz at 30 FPS for models of 10k faces. Our results are best observed in the supplemental videos published alongside Rhodin et al. [2015b].

9.6.1. System setup

Simple hardware

Our system is simple to setup as it requires only cheap low-quality motion-capture sensors and is assembled within 5 minutes. In our experiments, full-body motion is captured as 3D joint locations by the Microsoft Kinect sensor [Shotton et al., 2011]. Hand articulation is captured as finger tip position by the Leap Motion

Table 9.1: Evaluation character databases, numerating the motion classes with characteristics and number of example animations.

Character	Dog	Caterpillar
Features	Transition, superposition, complex motion	Transition
Params.	Emotions, amplitude, frequency	Amplitude
Motion classes	Walk, sit, shake, look, wave, scratch, jaw	Walk, crawl, look, jump
# Animations	7 + 1 + 2 + 1 + 1 + 1 + 1	4 + 1 + 1 + 1
# DB frames	130	626

Character	Horse	Humanoid	Dinosaur
Features	Rich class control	Smooth locomotion	Transition, superposition
Params.	Emotions, frequency	Amplitude, frequency	Amplitude
Motion classes	Walk	Walk	Walk, jump, bend, stretch
# Animations	7	6	4
# DB frames	160	165	91

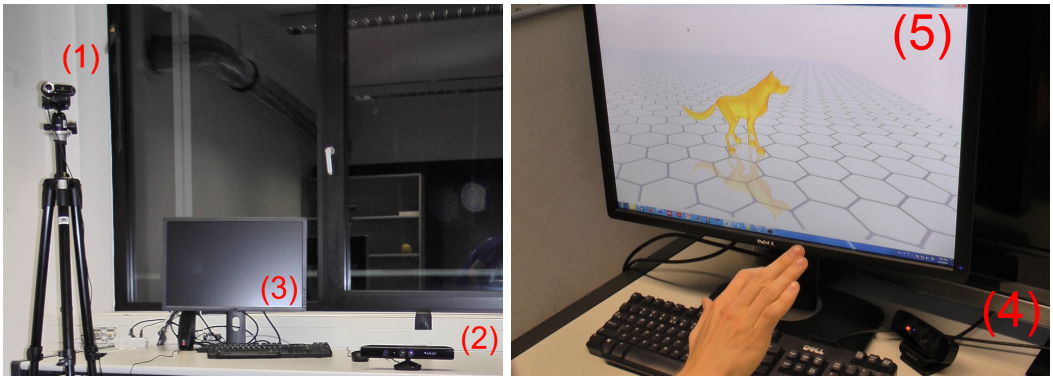


Figure 9.11: Camera Setup: Facial expressions are recorded by a conventional webcam (1), full body motion by the Microsoft Kinect (2), hand motion by the Leap Motion sensor (4). The live animation is displayed on a LCD screen in front of the user (3,5).

sensor [LeapMotion]. Facial motion is captured as 2D landmarks by a conventional webcam with the Intraface software (Figure 9.11). To control a new character the following steps are necessary:

Artist-created

Character animation. The controllable character motions are artist created and need to be acquired (we used 4-14 animations per character). Several commercial services offer game characters with basic motion cycles for free or low prices. As our system works on mesh representations any character that can be exported to a sequence of meshes can be used.

Game designer

Database construction (takes 30-60 min). Animations must be assigned to motion graph nodes, transition edges must be defined, and additive nodes must be marked (3 min). Each animation needs to be annotated by semantic amplitude and frequency parameters (5 min, including refinement for novice users) and by foot-plant timings (1-3 min). The foot-ground contact regions must be marked (2 min, once per

character) and quadrupeds need to be segmented into torso and limb segments by painting (5 min, once per character).

User-defined

Control definition (takes 1-15 min). Only a single reference control motion per motion graph node needs to be performed and recorded with the motion-capture sensor. The recording of all control motions takes about one minute. Typically, the set of control motions is slightly refined to establish natural control for a specific character and to avoid ambiguous motions (15 min). In a game scenario the user could also select existing control motions out of a dictionary of predefined control motions (1 min).

9.6.2. Character animation quality

Versatile input devices. Characters are animated with different user body, hand, and face control motions, captured by non-intrusive sensors that are suitable for VR applications (Figure 9.1). The body is tracked as 20 3D joint positions by Microsoft's Kinect and the hand as 9 3D fingertip and palm positions by Leap Motion. We regress facial expressions based on Intraface [Xiong and De la Torre, 2013] as a continuous value e between sad -1, neutral 0, and happy $+1$.

Intuitive control. Our mapping is able to generalize and distinguish between various user-defined control motions. Our users preferred human mimicking motions (swinging arms synchronously and asynchronously for walking and jumping, Figure 9.14) and abstract mimicking of character style (arm undulation for crawling, Figure 9.14).

Generalizing control. Figure 9.12 shows control of the dog locomotion class (Figure 9.3). From a single user-defined reference control motion of an on-the-spot run, and from three character animations, we can generalize to variations in frequency and amplitude affecting character stride length and step speed, and dynamics style changes from stand, to walk, and to run. Details such as eye blinks and speed-dependent ear wiggles are preserved from the artist animation. The parallel coordinates graph in Figure 9.13 also shows that our mapping is able to reach the majority of a four-dimensional parameter space, including character emotion.

Robust instantaneous estimates. The quality of estimates is compared to ground truth on a synthetic arm waving input motion with additive white Gaussian noise (SNR = 93.8). Our method quickly estimates changes in control speed and amplitude, gracefully smooths over discontinuities, and is more accurate compared to baseline methods (Figure 9.5).

Frequency invariance

One key advantage of our method is that it is reliable for a very large range of input motion speeds. We tested this with the human character, which has database animations of slow, medium and fast walks, as well as medium and fast runs. Our method smoothly cycles through slow walks of one quarter steps per second, up to a fast run of two steps per second.

Parametrized time-shift interpolation. With our time-shift approach, the artifacts with interpolating out-of-phase limb motions are significantly reduced, allowing effective foot-sliding prevention for bipeds and quadrupeds when varying between

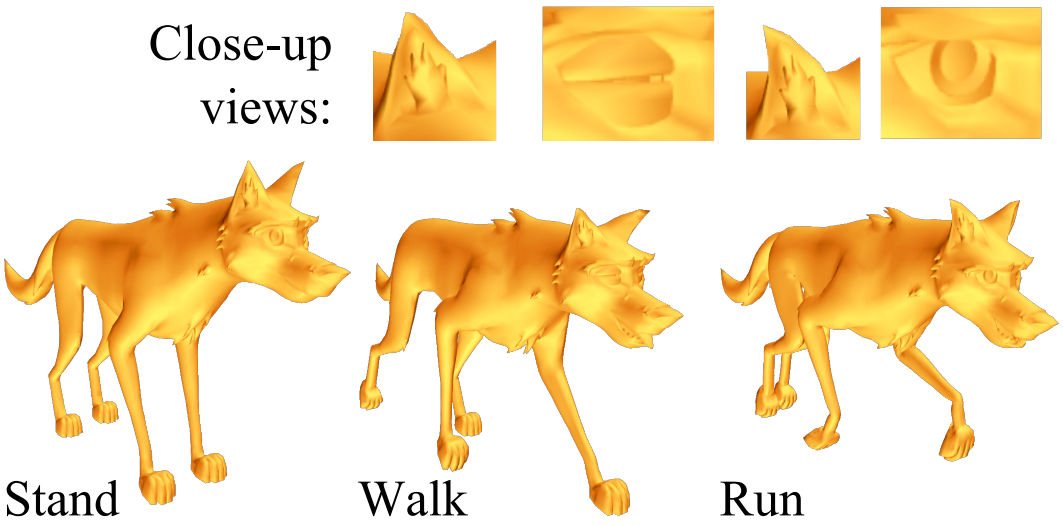


Figure 9.12: Frames from a dog animation generated by mimicking a walk. The close-up views highlight that temporally and spatially localized details and dynamics of the original animation are preserved by our system, such as an eye blink and flapping of the ear.

motion styles such as walk and run and different step sizes (Figure 9.9). This is visible in all our quadruped locomotion examples. In contrast, the pose mappings introduced Chapter 8 suffer from foot-sliding, as they do not provide the foot-placement annotations required for sliding prevention.

Motion graph and control superposition. Interpolating within a parametric motion class (stand to walk to run, Figure 9.1) is caused by changing motion speed and amplitude. Transitions across classes (crawl to jump, Figure 9.14) are caused by change of control motion. Secondary actions such as head motion and body bending are superimposed. The final dog example in the video published alongside [Rhodin et al., 2015b] combines everything: hand and face trackers for both cyclic and non-cyclic motions with wave-based and linear control. We extrapolate control within a shared locomotion and emotion space, plus linear control over head rotation, pawing at the dirt, shaking, mouth motion, begging, and sitting. This control example shows many of our advantages, most notably the independence of control motions, for instance, bending of the first finger is used in three separate control motions without interference from combinations with other fingers.

9.6.3. Comparison to related work

Table 7.1 compares the most related character control methods according to different criteria; the proposed approach is in favor for most categories.

AFP estimation [Shiratori and Hodgins, 2008]. A natural baseline for AFP estimation is auto- and cross-correlation as proposed by Shiratori and Hodgins [2008]. In our experiments, normalized cross-correlation between the reference control motion and input motion performs similarly in terms of delay and estimated values to Gabor filtering on our intermediate representation for either an independent control motion or for simultaneously-performed control motions which are spatially separated (e.g., left and right arm motion, Figure 9.5c–e). However, frequency and

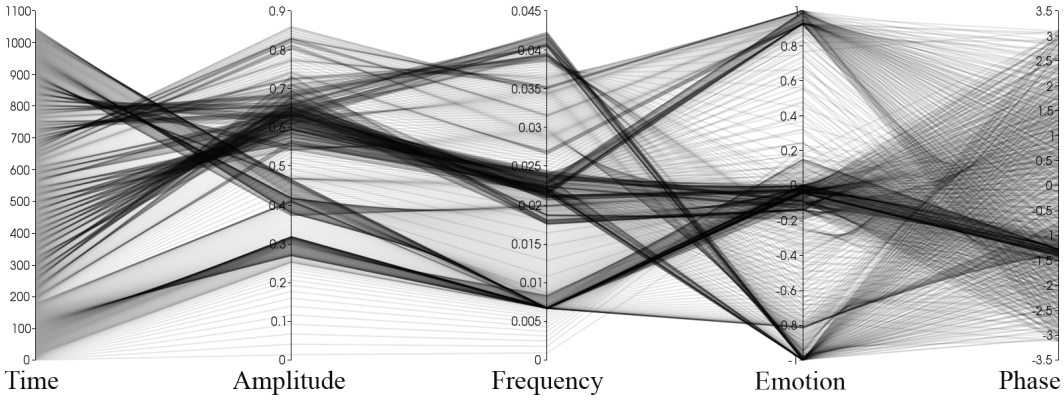


Figure 9.13: Parallel coordinate plot of the parameter space covered during the horse animation, where each line from left to right corresponds to one parameter configuration. Amplitude ranges from 0 almost to 1; frequency shows clusters around walk (0.025) and gallop (0.04) with transitions; emotion covers the full range from -1 (sad) to +1 (happy). Phase cycles between $-\pi$ and π. This demonstrates that our control scheme reaches wide parameter spaces.

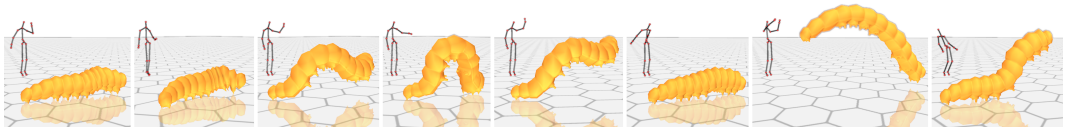


Figure 9.14: Animation of the caterpillar character, controlled by the user *(black skeleton)*: *Column 1:* Crawl by swinging the arms asynchronously. *Column 2:* Bending by turning user body. *Column 3-5:* Separate walk styles with large style variations by waving the left arm at different amplitudes. *Column 6-7:* Jump by swinging arms synchronously. *Column 8:* Rising the head by bending.

phase estimates were less reliable as the signal needs to be convolved for a discrete set of phase-frequency combinations. In contrast, the Gabor filter gives phase analytically and only requires to sample the frequency dimension. Shiratori and Hodgins [2008] use auto-correlation to measure the periodicity of a signal. It requires two motion periods for comparison, hence, introduces a lag of one period compared to cross-correlation and our approach. Moreover, it was less reliable in our experiments (Figure 9.5c-d). Overall, cross-correlation is an alternative to Gabor filtering, but was discarded due to its drawbacks.

Simultaneous analysis

Our main contribution to the AFP estimation problem is the separation of simultaneously performed motions into independent intermediate representations. We demonstrate its importance with the hand-controlled dog, where talking, shaking and scratching the paw are controlled by rotating the thumb and shaking the whole hand. Direct frequency decomposition methods (e.g., Unuma et al. [1995]) do not consider prior knowledge of a particular reference control motion, and would lead to permanent undesired control and superposition. Directly applying cross-correlation to the input motion (e.g., Shiratori and Hodgins [2008]) activates all character motions simultaneously when only one of the corresponding input motions is performed, as all involve motion of the thumb (Figure 9.15). This is effectively prevented by our separation method, as only very subtle interference is visible.

Gesture activation classification [Seol et al., 2013]. We compare our method to SVM

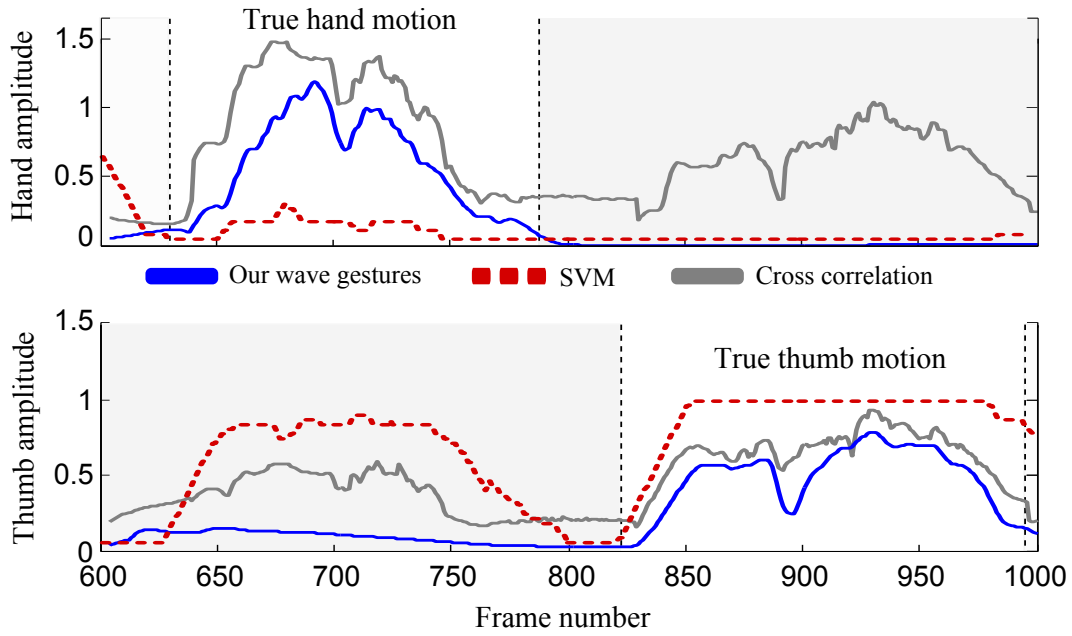


Figure 9.15: Analysis of control independence. Our Gabor filtering of independent wave representations separates rotation of the whole hand (*top*) from thumb rotation (*bottom*), with high amplitude in true motion areas and low otherwise. Cross-correlation (e.g. Shiratori and Hodgins [2008]) suffers from strong interference with high amplitude in all areas of motion. SVM classification (averaged over 30 frames as proposed by Seol et al. [2013]) falsely detects no hand motion, and also detects thumb motion during hand motion.

classification on the task of detecting the active control gesture. We train a Gaussian-kernel SVM on position, velocity, and acceleration features and average the activation of each gesture over 30 frames, as proposed by Seol et al. [2013]. For control motions which are spatially separated (e.g. left and right arm control of caterpillar), SVM and our method are equally robust, with a slightly lower delay for SVM. However, for hand input where the same finger is used during multiple gestures, SVM falsely detects thumb motion instead of hand rotation, and also fails to detect the hand rotation at all, because instantaneous velocity and acceleration features do not distinguish the performed quick hand motions reliably (Figure 9.15). In contrast, our method separates control motions correctly in this challenging case with very little interference.

Pose mappings. To see the effect of our motion mapping in contrast to existing pose mappings, we compare against a shared GPLVM with 15 latent dimensions (similar to Yamane et al. [2010]), a latent volume nearest neighbor (NN) pose mapping with 15 neighbors (similar to the mapping of Seol et al. [2013] for cyclic motions), and a linear pose mapping (similar to Rhodin et al. [2014], which was explained in Chapter 8, and Seol et al. [2013]). We train all systems with an arm swing control motion of 32 frames, corresponded to horse walk and horse stand animations.

Extrapolation capabilities

Our method is able to reproduce all details of the horse animations and extrapolates to smaller step lengths by reducing the control arm swing extent. No other compared method is able to generalize this change in control: The shared GPLVM method shows very strong jitter. The NN lookup cannot reproduce the temporal

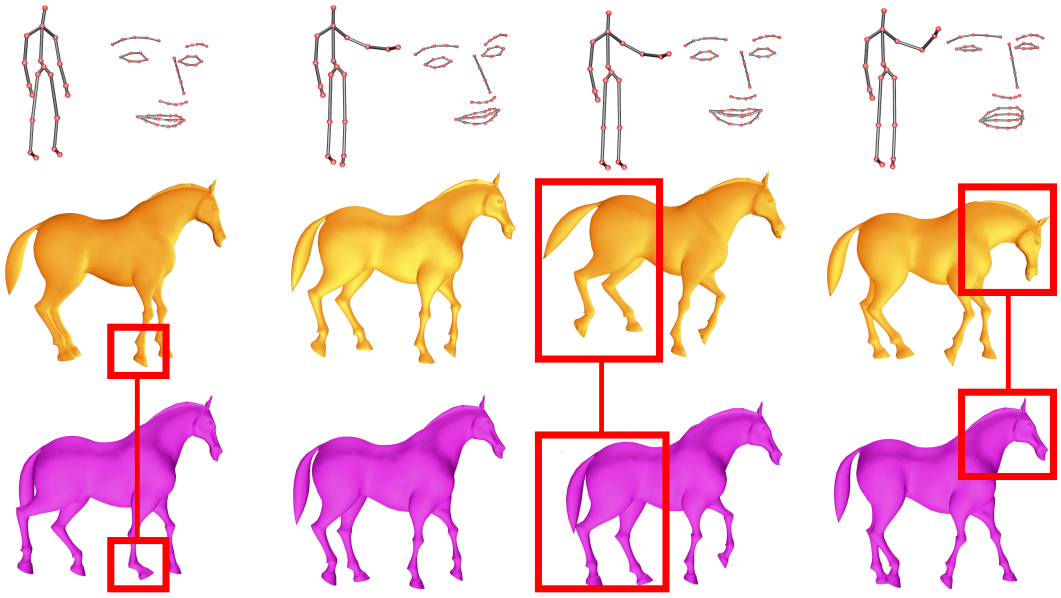


Figure 9.16: Comparison with Rhodin et al. [2014, Chapter 8]. *Top:* User body and face pose, used for both techniques. *Middle:* Our animation using the parametrized horse motion class; from left to right: stand, trot, happy gallop, and sad gallop. *Bottom:* Rhodin et al. shows distortions in the stand and is not able to generalize to a gallop when speeding up (column 3) nor to different moods (column 4).

evolution of the original animation exactly due to input noise and user imprecision, and also cannot generalize the transition from walk to stand. The linear map suffers from less jitter, but the animation detail is reduced. Moreover, at high control motion speed, it exhibits unnatural exaggerated rotations in the hooves, as the mapping was learned from a single example motion at fixed velocity and the used velocity features do not generalize to significantly different input speeds. None of these artifacts occur in our generated animation.

Pose mappings—Seol et al. [2013]. We directly compare to the dinosaur animation presented by Seol et al. [2013]. Cyclical motions are classified and mapped by NN maps, which are limited as they do not generalize to motions of varying amplitudes. Our method improves the control of cyclic motions as it is able to recover very slow and very fast motions, with independent control of dinosaur step length and step speed, all generalized from a single user motion training example. Moreover, we provide more controls (jumping and tail wiggle) within the same control sequence due to independence of controls. Finally, our control is more robust, and so the synthesized cyclic dinosaur walk contains less temporal jitter and no foot sliding.

Pose mappings—Rhodin et al. [2014] and Chapter 8. Compared on horse locomotion (Figure 9.16), our method provides improved control over motion style and emotion by mapping to a parametrized locomotion class of stand, trot, and gallop animations in happy, neutral, and sad emotion variations (seven animation examples). Neither the richness of control nor the extrapolation from a single example control motion is possible with the linear mapping used by Rhodin et al. [2014, Chapter 8].

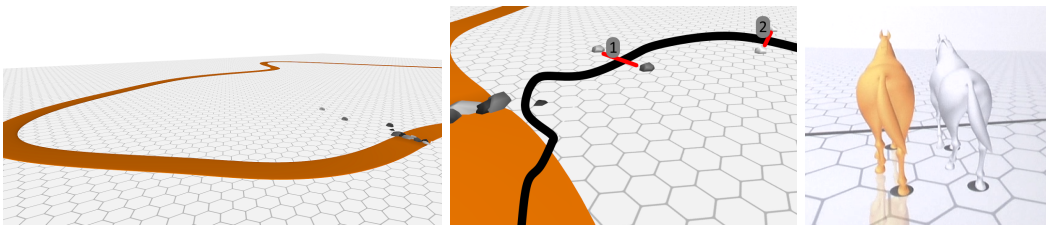


Figure 9.17: User study tasks. *Left:* Task 1. Follow the path with the dog character, trying to stay within the path width. *Center:* Task 2. Move through the terrain and gates, and switch motions with the caterpillar character when moving through each one. *Right:* Task 3. Mimic the white horse by matching the step frequency, stride length, and dynamics (walk or trot or gallop).

9.6.4. User evaluation

Gesture vs. game pad

We studied wave gestures with 10 participants. For fair comparison, we predefined control motions for all participants, as per a typical game setting. In a pilot study, we rejected NN mapping (similar to Seol et al. [2013]) as this did not generalize well to different user body proportions nor to variations in user-specific control motion characteristics. Direct pose mapping (similar to Rhodin et al. [2014, Chapter 8]) worked well for control; but output animations look stilted, with foot skating, floating, and temporal jitter. Specifically, very slow or fast control motions that differ strongly from the reference control motion lead to unpleasant artifacts. Thus, we chose the familiar gamepad as a baseline. Amplitude and frequency map to left/right analog triggers, heading to left thumb stick, and motion transitions to face buttons. Progression is obtained by integrating frequency.

Control tasks

We tested three game-like tasks: 1) Follow a curved path, to test world locomotion control; 2) Transition between motion classes at specific world points, to tests action or event response; 3) Control horse step length and frequency to match a reference, to test precise control. Figure 9.17 depicts the test scenarios. The path has width 4× character width and 10 turns with average maximum curvature 0.26× path width. After training of one minute for each task, all subjects were able to solve all tasks with both methods. We applied the following task descriptions and metrics.

Steering

Task 1: Follow the shown path with the dog character without leaving it.

Quantitative metric 1:

- Number of times the path was left by the controlled character during completion of the course.
- Duration in seconds until the controlled character returns to the path after first leaving the path.

Acting

Task 2: Control caterpillar through obstacles (stones), start crawling, move through small gap, switch to walk after first stone gate, lift head fully after second stone gate, reduce head lift to half the height, and finally jump three times in a row.

Quantitative metrics 2:

- Number of trials until successful transition.

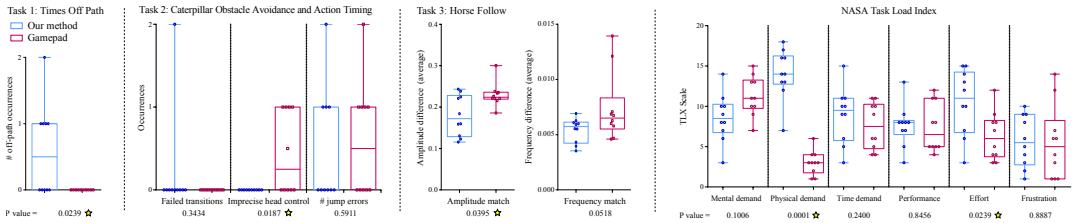


Figure 9.18: Box and whisker plots for our task and questionnaire evaluation of our approach against a familiar gamepad controller. Two-tailed P values are provided from paired Student's t-test analysis, which significant at the 95% confidence level ($P < 0.05$) marked with a star.

– Number of unintentional activated motions.
– Number of missed gates.
– Success of lifting head completely, then to half height.
– Difference from 3 jumps.

Reacting

Task 3: Follow the leading horse, try to mimic its gait rhythm/beat and stride length as close as possible.

Quantitative metrics 3:

– Absolute difference between reference amplitude and user controlled amplitude (The reference was created by performance as well.)
– Absolute difference between reference gait frequency and user controlled frequency.

Statistical evaluation

In summary, wave gestures are a competent alternative to a gamepad for our tasks. With the same control motions, all 10 participants were able to complete all tasks. This shows robustness to different body shapes and input motion variations, including to children. Figure 9.18 reports the main study results, it shows box plots with outlier rejection computed using ROUT at $Q = 1\%$, P values computed using paired Student's t-test analysis, and significance shown at the 95% confidence level.

Significance

In detail, for task 1, our method is slightly worse than the gamepad, straying from the path 0.6 times on average per participant and taking on average 4 seconds to get back on track, though this is somewhat expected as directional control is simple with the familiar gamepad. That said, users found it easier to adapt character speed to narrow or wide curves with wave gestures. Task 2 was comparable with both methods, but gamepad character animation looks less convincing with many abrupt changes in direction, amplitude, and frequency. For task 3, wave gestures were significantly more accurate (p-value = 3.95×10^{-2}), with frequency also showing improvements (p-value = 5.18×10^{-2}). We believe this is because frequency control is intuitive if performed with cyclic motions, as is change in stride length through amplitude control. We strengthened this belief in a post-task questionnaire, with wave gestures rated significantly more intuitive for stride length and step frequency control. The whole result is displayed in Figure 9.19.

Delay and demand

Users identified two significant limitations which apply broadly to gesture control: greater control delay vs. gamepad, and higher physical demand. Should a motion

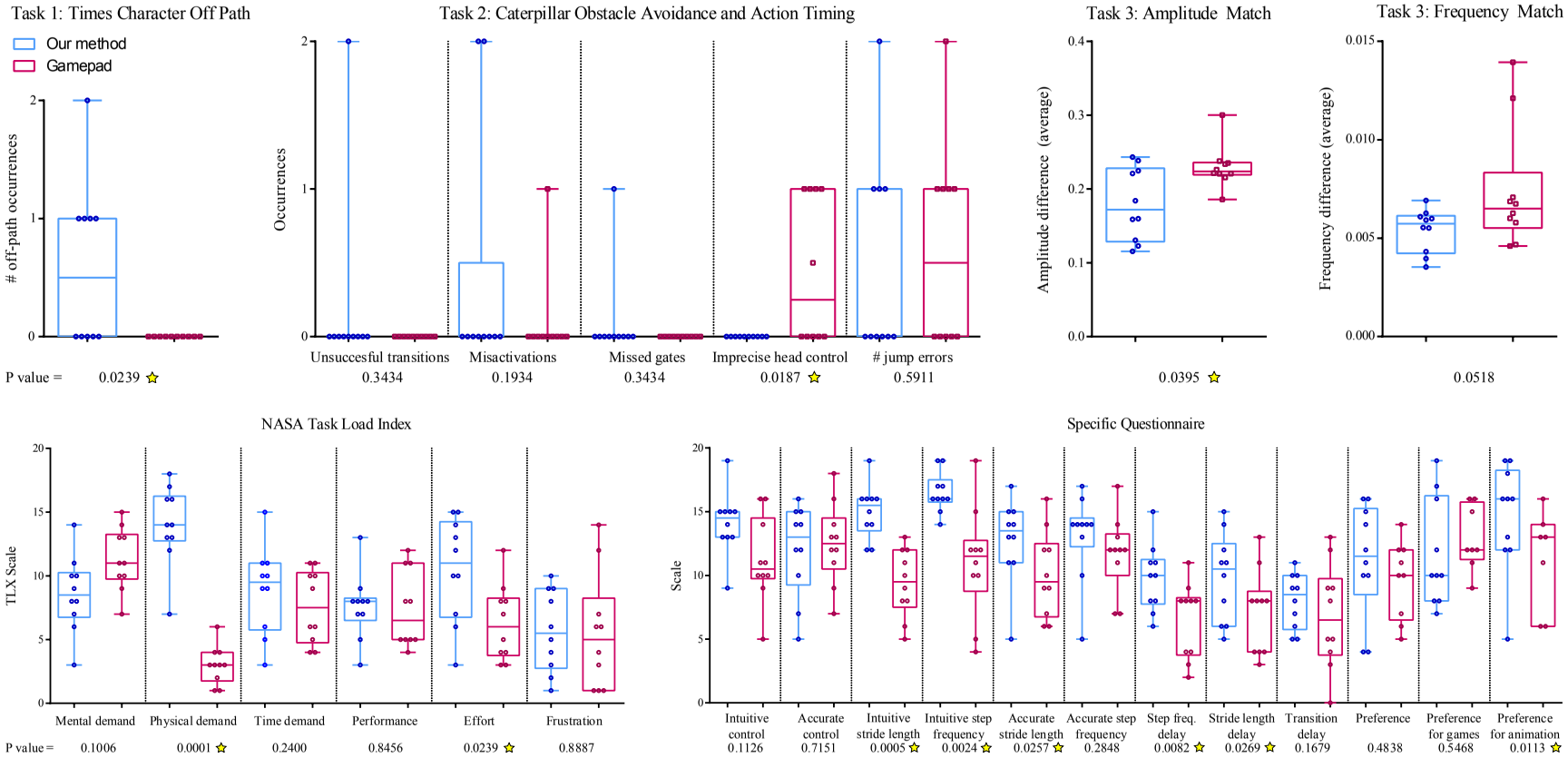


Figure 9.19: Box and whisker plots for our task and questionnaire evaluation of our approach against a familiar gamepad controller. Two-tailed P values are provided from paired Student's t-test analysis, which significant at the 95% confidence level ($P < 0.05$) marked with a star.

be uncomfortable, one benefit of our approach is that different control motions can be interactively defined in just a few seconds, with no required limb or part association. Our approach also allows different motion trackers to be swapped in easily as, after character authoring, the only input we require is 2D or 3D skeleton points. For example, we track the hand, which is more suitable for longer control sessions or desk work.

Coordination

One aspect that is untested in this study is the superposition of multiple motions, which is hard to map to a gamepad but is easily solved with wave gestures. The lack of more negative significant differences may be surprising given the familiarity of gamepads, though it is clear that both schemes have strengths and weaknesses.

9.6.5. Expert animation practitioners

We invited three professionals to asses our system: a live animator, an offline animator, and a game animation middleware developer.

Requires large motion dictionary

Live animator. Our trained performance animator was very enthusiastic about our system. They imagined a large potential for character control on stage through the mixing of pre-authored animations. However, the stage has demanding standards: while our approach is robust, slight delays when transitioning can cause inexperienced users to repeat actions, and the user must learn to trust this behavior in performance. She requested additional database animations, such as foot scratching and jumping; our system scales well to these additional animations by the motion graph.

For quick prototyping

Offline animator. Our offline animator saw the largest potential for our system at the prototyping stages of the animation pipeline, where quick and easy generation of animations would help communicate with the director. Furthermore, they saw potential during content creation as a blocking tool for creating an initial animation from a storyboard, which is then later refined offline in the standard animation pipeline.

For games and game creation

Game animation middleware developer. Our developer suggested that our approach fits many game requirements, notably the user flexibility, speed, and robustness. Our authoring pipeline is very similar, with games typically using animation blend trees (synonymous with parametric motion classes) and state machines (a simplified motion graph). Beyond real-time control, he saw a benefit for games companies who buy off-the-shelf motion databases, where our technique would allow animators to create new sequences in the style of the original database but with expanded variety, particularly for quadrupeds where existing data is rarer.

9.6.6. Controlling physical robots

From virtual to real worlds

Thus far, this performance-driven character animation approach has been evaluated solely on virtual characters. The problem was formulated as finding a mapping that transforms a stream of input poses to a stream of character poses, with the character pose represented by its mesh configuration. Existing approaches also used lower-dimensional rig parameters instead of direct mesh representations [Seol et al., 2013]. In principal, low-dimensional rigs could be the output too, as the

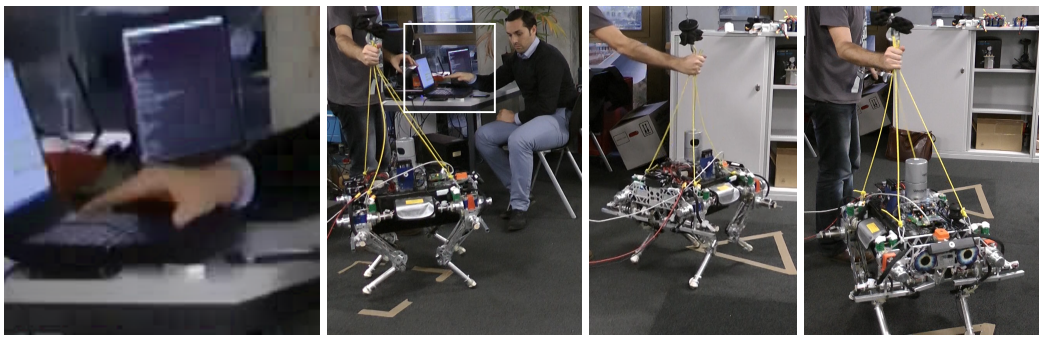


Figure 9.20: Intuitively controlling a physical robot with directional and cyclic hand gestures through a course of landmarks. From left to right: closeup on hand motion input, robot start position, moving towards the triangle mark, reaching triangle, and reaching second triangle. Please note, that a second person passively secured the robot to prevent fatal crashing in a potential failure.

inferred mapping does not depend on the target representation structure. In fact, there is no requirement that the target is a virtual character. It could also be any other parametric target domain, for instance, the tempo of music, or a physical robot parametrized by a space of control variables.

Four-legged robot

We probed performance-driven robot control on the four-legged *StarlETH* robot, which has parametric gait styles [Hutter et al., 2012]. A rotating finger reference motion, captured by the Leap motion sensor, is defined as input. It is mapped onto robot walk, where rotational speed maps onto robot step frequency, and rotation amplitude maps onto robot stride length. In addition, the hand orientation was configured to drive the robot direction and hand-sensor distance maps to robot crouching height. The translation from control variables to gait control was performed by the robot software, where only minor adaptation from the originally supported keyboard interface was necessary.

Intuitiveness

This mapping allowed to steer the four legged robot through predefined landmarks, see Figure 9.20. In particular, the control of locomotion and step frequency was intuitive, and allowed slowing down when approaching the target locations. Moreover, the direction of the whole robot could be controlled by changing the 3D direction of the hand. Such directional control could be useful in a remote setup, to control the view direction of the robot's primary camera, or to control an actuator to interact with the environment. We noticed that low delay was crucial to navigate safely. To reduce lag, we reduced the Gaussian filter window width λ to 0.5, which greatly improved controllability (Section 9.4.2). Different gait types, such as trot and gallop beat patterns, could also be controlled akin to different character motions. Once associated in a motion graph, a switch in control gesture would initiate a switch in robot gait style.

Prove of concept

While this preliminary study demonstrates the applicability of performance-driven character animation to physical robot control, additional investigations are necessary to explore and prove its use in real applications such as remote rescue scenarios.

9.7. Discussion and limitations

A trade-off Any method is a point in a design space with trade-offs. For real-time character control methods, these are typically expressiveness, learnability, flexibility, and robustness. Our choices focus on improving the last three of these attributes. However, we limit expressiveness, in contrast to retargeted 'one-to-one' mapping approaches, with a need for authored character animations. Having said that, our approach is a good fit for games and virtual worlds. One way to overcome the general expressiveness problem is to allow a gesture to switch into a retargeted animation mode as a node in the motion graph, e.g., when appropriate for direct interaction with an object. For environmental context triggering, our activation variables work similarly to Ishigaki et al. [2009], so moving in the world could trigger motion control changes.

Information content One might suspect that our 2D intermediate representation is too drastic a reduction in dimensionality. However, the intrinsic dimensionality of individual motions is low, and a drastic reduction is actually desirable: 1) To combat 'noise', as user shape and coordination skills vary greatly (e.g. children), with tracking inaccuracy also affecting the result; 2) Each parametric motion class has one reference control motion, and control through variations of this motion are inherently similar. A drastic reduction is also sufficient: If we attempt to reconstruct original control motions from our wave representation by the inverse linear map Φ^{+}, we measure reconstruction error as $\approx 0.3\times$ the standard deviation (over all dimensions) of the original signal. Empirically, our examples show that we compete with or exceed the flexibility, ease of control, and animation quality of alternative high-dimensional pose mapping approaches. In principle, the intermediate representation and filtering could also be extended to capture multiple harmonic frequency bands, which would allow multiple frequency motion detection, but this complicates user control and is harder to understand for novice users.

Delay The applied Gabor filtering infers AFP from ≈one period of motion, which limits the ability to control speed and amplitude of character motions within a single period. Control can become difficult at very abrupt changes such as quickly transitioning to high jumps. There is a fundamental trade-off between robustness and responsiveness; though different filtering techniques might adapt to this specific case, e.g. cubature Kalman filter [Arasaratnam and Haykin, 2009]. If desired, physical constraints could be incorporated to restrict unrealistic motions, such as stopping during jumps or flight phases.

Separability Our approach to distinguishing gestures is robust as competing reference control motions are used as negative examples in training. We show scalability to many different control motions; however, we have difficulty distinguishing gestures which are *very* similar, for instance, hand waving in a straight line vs. on an arc. Any control gestures which are more than two edges apart in a motion graph are independent, so 'scalability' must also consider *how many edges are typically needed per node?* For a game, this is governed by the available control DOFs. A typical gamepad has 10-12 binary buttons, two linear triggers, and two 2D linear thumbsticks. In our example of the dog controlled by the hand and face, we show independent linear or cyclic control of four locomotion, one emotion, and six action

parameters. Under this comparison, we fare comparably: we have less binary states, but we offer more expression per DOF. While the scalability problem is general to gestural approaches, we somewhat relieve these restrictions through the graph design.

Subjective use

The question of intuitive gestures for arbitrary characters is open-ended, involving issues such as learnability, comfort, and user preference. We demonstrate a flexible approach which allows interactive mapping of different controls learned with a single reference control motion from the user. Our user study shows that people of many shapes and sizes could quickly adapt to using the system with good accuracy.

Animation pipeline

Our approach is applicable to characters created by various animation tools because it is always possible to export any animation format to our mesh representation (e.g. rig or deformation cages), but it is harder to take our synthesized mesh animation and recover rig parameters to continue animation with traditional pipelines. Our core mapping is independent of character representation, as demonstrated with the legged robot control; however, demonstrating a mapping to rigged characters remains future work.

Extrapolation

Currently, we do not extrapolate outside the animation database. However, this would be possible by embedding into a latent space with extrapolation capabilities, such as, non-linear Gaussian process latent variable model (GPLVM) embeddings [Lee and Elgammal, 2004; Levine et al., 2012] and multi-dimensional scaling [Shin and Lee, 2006; Cashman and Hormann, 2012]. Our focus is instead on robust input generalization—the output of our algorithm could be used as input to these techniques to drive animation synthesis.

9.8. Summary

Robust and rich

We present an approach to decompose robustly and in real-time high-dimensional input motions into wave parameters of amplitude, frequency, and phase for a set of control motions. We interactively learn a mapping from single reference examples of a user defined control motion to an intermediate 2D sinusoid representation, which lets us generalize variations of wave parameters during live motion. This provides intuitive control variation, particularly for cycles, and produces higher-quality character animation than competing approaches. The result quality is greatly improved in comparison to the pose based methods introduced in Chapter 8, and dynamics are faithfully transferred to the target character. Furthermore, complex motions can be superimposed, and inferred global character motion is realistic due to foot-sliding prevention. For instance, when interpolating within a parametrized database, simply increasing the frequency of a gesture enables natural transitions from walking to running. The approach applies to arbitrary characters, and so for quadrupeds, we solve the locomotion interpolation problem with a time-shifted approach that separates temporal alignment from pose interpolation and partially decouples character segments (e.g. limbs). In a user study, we verified that our system was intuitive to learn and operate, and applies well to different users, control motions, and motion trackers. As such, it had the potential to be useful for situations where traditional controllers are inappropriate. It is particularly suited

Suited for VR and AR

for game and VR applications and, as we shall see, also serves the demands from other application fields.

Part IV

Conclusions

From acquisition to animation and beyond 10

The holy grail

We envision a future in which the complete human body is reconstructed at high temporal and spatial resolution from non-intrusive, easy-to-setup sensors, and in general unconstrained environments. Future reconstruction algorithms should be general enough to operate in casual living room environments, offices, and other workplaces, for quick athletic motions, and at medical precision standards. Besides metrically accurate reconstructions, semantics information should also be extracted, such as the intent of a motion or gesture. To attain these ideal properties, challenging algorithmic problems need to be solved.

Contribution generality

In this dissertation we have used performance-driven character animation in VR as our major example: for consumer adoption, it requires most if not all of the 'holy grail' target properties; yet is currently hampered by imprecise, invasive, and difficult to use systems which work only in constrained scenarios. Our contributions alleviate some of these problems, but their impact is not restricted to VR. The core algorithmic, mathematical, and conceptual contributions of this dissertation are general. To better place our contributions in the wider world of applications, and to see where new contributions must be made to reach the 'holy grail', we will discuss the applications of the augmented body, virtual interaction, sports coaching, and computer aid (see Figure 10.1).

Overview

In the following sections, we summarise the core contributions, subsequently discuss their merit in the four application scenarios, and sketch possible future solutions to remaining open challenges.

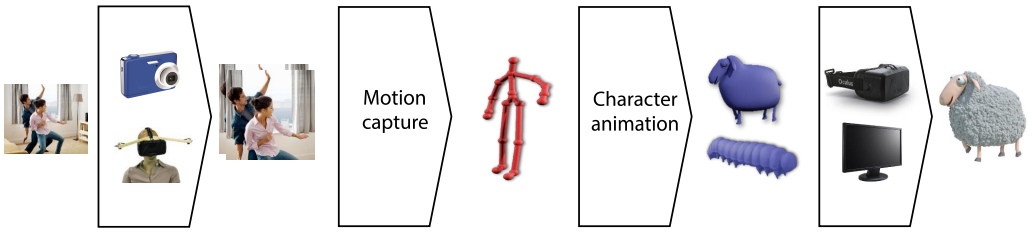
10.1. Contribution summary

Smooth volumetric model for pose estimation

The volumetric scene model handles occlusions in a smooth and differentiable way. The developed theory is applicable to diverse problems, such as rigid object tracking, shape estimation, and articulated pose estimation. For pose estimation, we demonstrated that the advanced occlusion handling yields higher reconstruction quality compared to existing solutions, in particular when using few input views.

Volumetric contours for shape estimation

The volumetric contour model directly aligns object boundaries with image edges, avoiding the typical and error-prone silhouette extraction. It is particularly useful for the proposed actor model initialization, where no prior appearance model is available. This is in contrast to methods using segmentation and a foreground color model, such as GrabCut segmentation.



Features	a) Sensing	b) Reconstruction	c) Processing	d) Display
	1. Egocentric rig	5. Occlusion handling, full-body	9. User-defined, intuitive	13. Compatible with HMD
	2. Lightweight	6. Fully automatic	10. Real-time mapping	
	3. Free roaming	7. Real-time	11. Gesture generalization	
	4. Low-cost	8. Automatic database creation	12. Robustness	

Applications	Computer aid	Virtual sports coach	Virtual interaction	The augmented body
	Easy-to-use (6, 9)	Low-cost, lightweight (2, 4)	Intuitive (6,7,9,10)	Intuitive (6,7,9,10), HMD (1,13)
	Robustness (5, 8, 11, 12)	Semantic analysis (11)	Medium detail (5)	Fully automatic, (6,7,9,10)
	Free roaming (2, 3)	Non-intrusive (2,3,6)	Semantic analysis (11)	Full-body capture (5)
	Varying illumination	Free roaming, full-body (3,5)	Fine detail	Social acceptance
	Very high accuracy	Interactive (6,7,9,10)	Loose clothing	High frame rate, low delay

Figure 10.1: This dissertation advances motion-capture and character-animation algorithms to enable new levels of performance-driven character animation. This figure lists the main features of the proposed algorithms (center) and relates them to diverse applications (bottom). The attained advances are general and have merit in augmented reality and virtual interaction, and also in fields outside virtual reality, such as sports coaching and computer aid. Some remaining open challenges are marked in light gray.

Optical inside-in motion capture

The egocentric tracking approach allowed inside-in motion estimation without using external sensors and reference objects. Its design avoids occlusion problems in crowded scenes and reduces setup time compared to outside-in arrangements. Furthermore, it provides free roaming, while maintaining minimal instrumentation. The proposed automatic annotation and augmentation strategy eases the creation of large databases, which are required for machine learning methods—for both outside-in and inside-in approaches.

Guided interactive control

The interactive pose mapping with automated guidance lowered the entry barrier and enabled performance-driven character animation for novice non-technical users. The mathematical formulation as a general mapping between well chosen input and character representations proved state-of-the-art generalization to various character types, and the use of unlabeled data in a semi-supervised learning framework was crucial for regularization.

Cyclical gesture processing

The wave gesture generalization is designed to handle cyclical motions, a key ingredient to enable rich control of the important motion class of locomotion. Working in the time-frequency domain increased robustness to sensor, control, and mapping inaccuracies, which significantly advanced the quality of real-time character animation. Together with improved quadruped interpolation, the quality of performance-driven character animation was greatly improved.

From motion capture to interactive virtual worlds

In summary, the pipeline from sensing, motion capture, and character animation is advanced significantly. The algorithmic improvements attain new theoretical properties and have practical merit in virtual reality applications and beyond, as we outline below.

10.2. Future applications

10.2.1. The augmented body

Dynamic body augmentation

We argued that using the user's real body motion has high potential for natural human-computer interaction. A precise reconstruction of the user enables new immersive experiences, for instance, by augmenting the user's appearance with an avatar that precisely follows the user motion. With a VR or AR head mounted display, it is be possible to place the user into a virtual surrounding and into a virtual character. By exactly overlaying the avatar on the user's position, the user gets the impression of embodying the avatar. For instance, to embody a super hero or to wear a new clothing style.

Real-time

Body augmentation requires extremely low delay and high frame rate pose estimation Even a slight delay would lead to ghosting artifacts. In EgoCap we used a ConvNet to estimate pose. For (discriminative) neural network-based reconstruction, the network depth is the limiting factor, since the execution time is limited by the sequential dependency of layers. In our EgoCap project, 100 layers are used and lead to frame rates of about 5 fps. Attaining similar reconstruction quality with less layers is an open problem. In our framework, the discriminative neural network is combined with a generative model. It might be bearable to use the discriminative part at lower frame rate for reinitialization only. The proposed gradient-based generative optimization scales well to higher input frame rates. For a constant motion, pose changes between frames become proportionally smaller with increasing frame rate. That means, that higher input frame rates reduce the required step size and the number of gradient iterations per frame can be reduced.

Low-cost lightweight

Besides technical challenges, the design and manufacturing of HMDs and motion capture equipment that is small and non-intrusive is difficult. However, aesthetics are very important for social acceptance. The current commercial HMD solutions are relatively bulky and are intended for home use. The proposed egocentric setup requires cameras mounted at small extensions, to provide a sufficient viewing angle and reduced occlusions of the lower body. While these extensions are lightweight and do not hamper most applications, they likely pose a problem for social acceptance outside the living room. Together with future miniaturization of HMDs, these extensions should be replaced by alternative on-body camera positioning. The proposed algorithm is not specific to head-mounted inside-in capture. Miniature cameras are available and could be placed at different body locations, such as at the shoe and by integration into digital smart clothes. It is an open challenge where to best place cameras to reduce intrusion, while maintaining reconstruction quality.

Easy to use

The proposed egocentric pose estimation is not yet ready for real-time augmentation, but we outlined possible advances towards this goal. EgoCap was designed to require little manual initialization and can be pre-calibrated, yet it provides full-body motion capture. The proof-of-concept VR application successfully demonstrated that the generative component has real-time performance. It creates a solid foundation for future consumer-level pose estimation for augmented reality.

10.2.2. Virtual interaction

High resolution

One limitation of virtual interaction is the lack of haptic feedback, such as when engaging in an handshake. Nevertheless, most of the every day interaction is audio-visual. A realistic avatar with high temporal and spatial resolution is demanded to facilitate natural visual interaction, i.e., a perfect visual replication of the interacting users.

From medium to high resolution

In this dissertation, we focused on advances of skeleton motion estimation and body shape estimation. The developed algorithms provide a good overall reconstruction, but no fine detail such as cloth wrinkles. However, this medium-scale detail forms a good starting point for refinement methods, for instance the method of Robertini et al. [2016] could be used to reconstruct detail of deforming cloth. To enable high-detail visual interaction, exiting refinement methods need to be extended. Among others it is a challenge to increase accuracy in general environments, to support the egocentric perspective, to reconstruct loose clothing, and to support topology changes. Intriguing is also the utilization of today's high-resolution cameras, which allow to observe fine details, such as fine-grained hand articulation, even from distant cameras. Increased resolution creates new challenges: it requires good runtime scalability and closeup views show new characteristics and ambiguities. For instance, fabric strains and skin pores form repetitive patterns which make it hard to find reliable correspondences between frames and multiple views. Nevertheless, they provide structure that can be used for fine-scale geometry reconstruction

Semantic analysis

While a very realistic reconstruction would enable rich human-human interaction in virtual worlds, human-computer interaction requires to interpret performance capture data semantically. For instance, to estimate the intent of a motion [Ishigaki et al., 2009]. In Chapter 9, we proposed to separate simultaneously performed gestures and to estimate continuous motion parameters. Besides controlling virtual characters, it would be interesting to extend these approaches and to use them in general interfaces, such as for navigating in menus, browsing (web) content, and giving instructions.

Full-body detail

Detailed information on the user's hand is necessary for accurate interaction with the virtual world, such as picking up items, placing objects, and turning control knobs. Promising for the future are combinations with existing hand tracking algorithms, for instance, the color camera based work of Wang and Popović [2009] and Sridhar et al. [2014]. However, adaptation for the egocentric placement and fisheye camera distortion is required. As a first step, coarse discrete information could be extracted, for instance, for picking up an item it is sufficient to determine whether the hand is open or closed.

Inverse mapping, character to human

Haptic feedback would provide an additional communication channel. While it is already used in dedicated flight simulators and some VR applications, in the context of performance-driven character animation, a stronger feedback loop could be explored. This would include integrating physics to simulate character-environment collisions and providing the user with corresponding haptic feedback. While we developed While we developed a human to non-human character mapping, this would require us to map in the inverse direction: to translate character collisions to

semantically equivalent haptic force on the human user.

10.2.3. Virtual sports coach

The virtual trainer

Professional sports exploits motion pattern analysis to run faster, throw further, and harmonize team play. To train athletes, 2D video analysis is an established analysis tool in professional sports. Motion capture technology, as developed in this dissertation, can provide further 3D cues and insights [Bregler, 2012]. For instance, comparing multiple reconstructions allows us to automatically analyze the edge an olympic athlete has on his competitors, possibly exposing details not visible to the human eye. Such advanced supervision and coaching is currently only accessible to professionals. With future consumer level hardware and software, even amateur athletes could be coached by virtual trainers. Autonomous assessment of a performed action would be possible through highlighting the difference of the performed motion to the ideal motion pattern of a professional. More informed analyses could suggest the better timing of actions and overall body posture.

Free-roaming non-intrusive

For such a coaching analysis, unconstrained motion reconstruction in large environments is required, and the methods developed in Part II are a good match. Additionally, semantic analysis requires accurate advanced motion property estimation, which can be attained with the real-time estimation methods proposed in Part III. Together with new augmented reality devices, a system is conceivable that gives immediate and non-stop feedback during training.

10.2.4. Computer aid

Supervision and service robots

One long lasting hope of digitalization is the compensation of handicaps by assistance. This requires to understand human needs, and among others the reconstruction of human motion. For instance, real-time motion analysis would allow us to detect the fall of elderly people living alone, and to alert the ambulance. It could also be used to support medical treatment, for instance, to count Parkinsons syndromes during a day and to adjust medication doses accordingly [Dai et al., 2015]. Long-term recording and supervision over weeks or months could also estimate convalescence, and guide rehabilitation activities. Across populations, the analysis of groups of people over extended periods could statistically identify the ideal treatment for a specific illness.

Robustness

The proposed egocentric approach allows recording in general scenes for several minutes. It is a first step towards day-long recordings, which requires high robustness. It is an open question of how to best tolerate strong illumination changes, with one approach being to integrate shading estimation into the objective, such as proposed by [Wu et al., 2012]. Discriminative methods can generalize well, so long as the provided training data includes examples of sufficient variation. The automatic augmentation and annotation method proposed in Chapter 6 makes it easy to extend databases to additional subjects, environment conditions, occlusion cases, and clothing.

Intuitive fully automatic

Moreover, gestural control could allow people suffering from immobility and other physical handicaps to interact and navigate by simple gestures, e.g., using simple

hand gestures to control a service robot, as demonstrated in Section 9.6.6. It would be interesting to further advance human-robot collaboration with natural motion interfaces.

Conclusion Through predicting future applications, we have described promising research directions in which to advance. Like any prediction, the proof of time will measure their true value; however, we hope that the contributions in this thesis—the new algorithms, representations, and devices—further our ability to understand and support humans via natural motion interfaces.

Bibliography

Naveed Ahmed, Edilson de Aguiar, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Automatic generation of personalized human avatars from multi-view video. In *ACM Symposium on Virtual Reality Software and Technology*, pages 257–260, 2005.

Ijaz Akhter, Yaser Sheikh, Sohaib Khan, and Takeo Kanade. Trajectory space: a dual representation for nonrigid structure from motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(7):1442–1456, 2010.

Ijaz Akhter, Tomas Simon, Sohaib Khan, Iain Matthews, and Yaser Sheikh. Bilinear spatiotemporal basis models. *ACM Transactions on Graphics*, 31(2):1–12, 2012.

M. Alexa and W. Müller. Representing animations by principal components. *Computer Graphics Forum (Proceedings of Eurographics)*, 19(4):411–418, 2001.

Benjamin Allain, Jean-Sébastien Franco, and Edmond Boyer. An efficient volumetric framework for shape tracking. In *CVPR*, pages 268–276, 2015.

Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 22(3):587–594, 2003.

Sikandar Amin, Mykhaylo Andriluka, Marcus Rohrbach, and Bernt Schiele. Multi-view pictorial structures for 3D human pose estimation. In *BMVC*, 2013.

Mykhaylo Andriluka, Stefan Roth, and Bernt Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *CVPR*, pages 1014–1021, 2009.

Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *CVPR*, 2014.

Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. SCAPE: shape completion and animation of people. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):408–416, 2005.

I. Arasaratnam and S. Haykin. Cubature Kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, 2009.

Okan Arikan and D. a. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 21(3):1–8, 2002.

G. Ashraf and K. C. Wong. Generating Consistent Motion Transition via Decoupled Framespace Interpolation. *Computer Graphics Forum*, 19(3):447–456, 2000.

Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *ICCV*, 2011.

Alexandru O Bălan and Michael J Black. The naked truth: Estimating body shape under clothing. In *ECCV*, pages 15–29, 2008.

Alexandru O Bălan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed human shape and pose from images. In *CVPR*, 2007.

Luca Ballan and Guido Maria Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *Proceedings of the International Symposium on 3D Data Processing, Visualization, and Transmission*, 2008.

Luca Ballan, Aparna Taneja, Juergen Gall, Luc Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, 2012.

Ilya Baran, D Vlasic, E Grinspun, and J Popović. Semantic deformation transfer. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 28(3):36:1–36:6, 2009.

Connelly Barnes, David E Jacobs, Jason Sanders, Dan B Goldman, Szymon Rusinkiewicz, Adam Finkelstein, and Maneesh Agrawala. Video puppetry: a performative interface for cutout animation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 27(5):124, 2008.

Alan H Barr. Global and local deformations of solid primitives. *Computer Graphics (Proceedings of SIGGRAPH)*, 18(3):21–30, 1984.

Sumit Basu, Irfan Essa, and Alex Pentland. Motion regularization for model-based head tracking. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 611–616, 1996.

Thabo Beeler, Bernd Bickel, Paul Beardsley, Bob Sumner, and Markus Gross. High-quality single-shot capture of facial geometry. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 29(4):40, 2010.

Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab, and Slobodan Ilic. 3D pictorial structures for multiple human pose estimation. In *CVPR*, pages 1669–1676, 2014.

Gaurav Bharaj, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. Automatically rigging multi-component characters. *Computer Graphics Forum (Proceedings of Eurographics)*, 30(2):755–764, 2011.

Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 187–194, 1999.

Federica Bogo, Michael J Black, Matthew Loper, and Javier Romero. Detailed full-body reconstructions of moving people from monocular RGB-D sequences. In *ICCV*, pages 2300–2308, 2015.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *ECCV*, 2016.

Mario Botsch, Mark Pauly, Leif Kobbelt, Pierre Alliez, Br uno Lévy, Stephan Bischoff, and Christian Rossl. Geometric modeling based on polygonal meshes. In *ACM SIGGRAPH Course Notes*, 2007. Award: Best course notes for a revised course.

Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):99:1–99:9, 2008.

Matthieu Bray, Pushmeet Kohli, and Philip H. S. Torr. POSECUT: simultaneous segmentation and 3D pose estimation of humans using dynamic graph-cuts. In *ECCV*, pages 642–655, 2006.

Christoph Bregler. Manhattan mocap. http://manhattanmocap.com/olympics2012, 2012. Accessed: 2016-08-08.

Christoph Bregler and Jitendra Malik. Tracking people with twists and exponential maps. In *CVPR*, pages 8–15, 1998.

Christoph Bregler, Aaron Hertzmann, and Henning Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR*, volume 2, pages 690–696, 2000.

Christoph Bregler, Lorie Loeb, Erika Chuang, and Hrishi Deshpande. Turning to the masters: Motion capturing cartoons. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 21(3):399–407, 2002.

Thomas Brox, Bodo Rosenhahn, and Joachim Weickert. Three-dimensional shape knowledge for joint image segmentation and pose estimation. In *Joint Pattern Recognition Symposium*, pages 109–116, 2005.

Thomas Brox, Bodo Rosenhahn, Uwe G. Kersting, and Daniel Cremers. Nonparametric density estimation for human pose tracking. In *Proceedings of Pattern Recognition*, pages 546–555, 2006.

Chris Budd, Peng Huang, Martin Klaudiny, and Adrian Hilton. Global non-rigid alignment of surface sequences. *International Journal of Computer Vision*, 102(1-3):256–270, 2013.

Magnus Burenius, Josephine Sullivan, and Stefan Carlsson. 3D pictorial structures for multiple view articulated pose estimation. In *CVPR*, 2013.

Cedric Cagniart, Edmond Boyer, and Slobodan Ilic. Probabilistic deformable surface tracking from multiple videos. In *ECCV*, pages 326–339, 2010.

Neill D. F. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. In *BMVC*, pages 530–539, 2007.

The Captury. Captury Studio. http://www.thecaptury.com/.

Dan Casas, Margara Tejera, Jean-Yves Guillemaut, and Adrian Hilton. 4D parametric motion graphs for interactive animation. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 103–110, 2012.

Thomas J. Cashman and Kai Hormann. A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Computer Graphics Forum (Proceedings of Eurographics)*, 31(2pt4): 735–744, 2012.

Ufuk Celikcan, Ilker O Yaz, and Tolga Capin. Example-based retargeting of human motion to arbitrary mesh models. *Computer Graphics Forum*, 34(1):216–227, 2014.

Eva Cerezo, Frederic Pérez, Xavier Pueyo, Francisco J. Seron, and François X. Sillion. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328, 2005.

Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):686–696, 2005.

Manmohan Chandraker and Vikas Dhiman. Continuous occlusion models for road scene understanding. In *CVPR*, 2016.

Subrahmanyan Chandrasekhar. *Radiative Transfer*. Dover Publications Inc, 1960.

Jiawen Chen, Shahram Izadi, and Andrew Fitzgibbon. KinÊtre: animating the world with the human body. In *UIST*, pages 435–444, 2012.

Wenzheng Chen, Huan Wang, Yangyan Li, Hao Su, Dani Lischinsk, Daniel Cohen-Or, Baoquan Chen, et al. Synthesizing training images for boosting human 3d pose estimation. *arXiv preprint arXiv:1604.02703*, 2016.

Stelian Coros, Sebastian Martin, Bernhard Thomaszewski, Christian Schumacher, Robert Sumner, and Markus Gross. Deformable objects alive! *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):69:1–69:9, 2012.

Yan Cui, Will Chang, Tobias Nöll, and Didier Stricker. KinectAvatar: Fully automatic body capture using a single Kinect. In *Proceedings of ACCV Workshops*, pages 133–147, 2012.

Houde Dai, Pengyue Zhang, and Tim C Lueth. Quantitative assessment of parkinsonian tremor based on an inertial measurement unit. *Sensors*, 15(10):25055–25071, 2015.

Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics*, 27(3):98, 2008.

Edilson de Aguiar, Leonid Sigal, Adrien Treuille, and Jessica K. Hodgins. Stable spaces for real-time clothing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 29(3):106:1–106:9, 2009.

Martin de La Gorce, Nikos Paragios, and David J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *CVPR*, 2008.

Amaël Delaunoy and Emmanuel Prados. Gradient flows for optimizing triangular mesh-based surfaces: Applications to 3D reconstruction problems dealing with visibility. *International Journal of Computer Vision*, 95(2): 100–123, 2011.

Jonathan Deutscher and Ian Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205, 2005.

Jonathan Deutscher, Andrew Blake, and Ian Reid. Articulated body motion capture by annealed particle filtering. In *CVPR*, pages 126–133, 2000.

Abdelaziz Djelouah, Jean-Sébastien Franco, Edmond Boyer, François Le Clerc, and Patrick Pérez. Sparse multi-view consistency for object segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9): 1890–1903, 2015.

Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 409–416, 2003.

EgoCap. EgoCap dataset. http://gvv.mpi-inf.mpg.de/projects/EgoCap/, 2016.

A. Elhayek, C. Stoll, K. I. Kim, and C. Theobalt. Outdoor human motion capture by simultaneous optimization of pose and camera parameters. *Computer Graphics Forum*, 2014.

Ahmed Elhayek, Edilson de Aguiar, Arjun Jain, Jonathan Tompson, Leonid Pishchulin, Micha Andriluka, Chris Bregler, Bernt Schiele, and Christian Theobalt. Efficient ConvNet-based marker-less motion capture in general scenes with a low number of cameras. In *CVPR*, pages 3810–3818, 2015.

Alireza Fathi, Ali Farhadi, and James M. Rehg. Understanding egocentric activities. In *ICCV*, 2011.

Joao Fayad, Chris Russell, and Lourdes Agapito. Automated articulated structure and 3D shape recovery from point correspondences. In *ICCV*, pages 431–438, 2011.

Hans G Feichtinger and Thomas Strohmer. *Gabor analysis and algorithms: Theory and applications*. Springer Science & Business Media, 1998.

Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

Andreas Fender, Jörg Müller, and David Lindlbauer. Creature teacher: A performance-based animation system for creating cyclic movements. In *Proceedings of the Symposium on Spatial User Interaction*, pages 113–122, 2015.

Wei-Wen Feng, Byung-Uck Kim, and Yizhou Yu. Real-time data driven deformation using kernel canonical correlation analysis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):91:1–91:9, 2008.

Juergen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *CVPR*, pages 1746–1753, 2009.

Juergen Gall, Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision*, 87(1–2):75–92, 2010.

Pau Gargallo, Emmanuel Prados, and Peter Sturm. Minimizing the reprojection error in surface reconstruction from images. In *ICCV*, 2007.

Dariu M Gavrila and Larry S Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *CVPR*, pages 73–80, 1996.

Oliver Glauser, Wan-Chun Ma, Daniele Panozzo, Alec Jacobson, Otmar Hilliges, and Olga Sorkine-Hornung. Rig animation with a tangible and modular input device. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):144, 2016.

Michael Gleicher. Retargetting motion to new characters. In *Proceedings of SIGGRAPH*, pages 33–42, 1998.

Peng Guan, Alexander Weiss, Alexandru O Bălan, and Michael J Black. Estimating human shape and pose from a single image. In *ICCV*, pages 1381–1388, 2009.

Jean-Yves Guillemaut and Adrian Hilton. Joint multi-layer segmentation and reconstruction for free-viewpoint video applications. *International Journal of Computer Vision*, 93(1):73–100, 2011.

Yu Guo, Xiaowu Chen, Bin Zhou, and Qinping Zhao. Clothed and naked human shapes estimation from a single image. In *Computational Visual Media*, pages 43–50, 2012.

Dongwook Ha and JungHyun Han. Motion synthesis with decoupled parameterization. *The Visual Computer*, 24(7-9):587–594, 2008.

Sehoon Ha, Yunfei Bai, and C Karen Liu. Human motion reconstruction from force sensors. In *Proceedings of SCA*, 2011.

Miles Hansard. Stochastic visibility in point-sampled scenes. In *BMVC*, pages 89.1–89.12, 2015.

N Hasler and C Stoll. A statistical model of human pose and body shape. *Computer Graphics Forum (Proceedings of Eurographics)*, 2009.

Nils Hasler, Bodo Rosenhahn, Thorsten Thormahlen, Michael Wand, Jürgen Gall, and Hans-Peter Seidel. Markerless motion capture with unsynchronized moving cameras. In *CVPR*, pages 224–231, 2009.

Nils Hasler, Hanno Ackermann, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *CVPR*, pages 1823–1830, 2010.

Rachel Heck and Michael Gleicher. Parametric motion graphs. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 129–136, 2007.

Chris Hecker, Bernd Raabe, Ryan W. Enslow, John DeWeese, Jordan Maynard, and Kees van Prooijen. Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):27:1–27:11, 2008.

Robert Held, Ankit Gupta, Brian Curless, and Maneesh Agrawala. 3d puppetry: A kinect-based interface for 3d animation. In *UIST*, pages 423–434, 2012.

Thomas Helten, Andreas Baak, Gaurav Bharaj, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Personalization and evaluation of a real-time depth-based full body tracker. In *3DV*, pages 279–286, 2013.

Antonio Hernández-Vela, Stan Sclaroff, and Sergio Escalera. Poselet-based contextual rescoring for human pose estimation via pictorial structures. *International Journal of Computer Vision*, 118(1):49–64, 2016.

NJ Higham. Computing the polar decomposition-with applications. *SIAM Journal on Scientific and Statistical Computing*, 1986.

Adrian Hilton, Daniel Beresford, Thomas Gentils, Raymond Smith, and Wei Sun. Virtual people: Capturing human models to populate virtual worlds. In *Proceedings of Computer Animation*, pages 174–185, 1999.

M. B. Holte, C. Tran, M. M. Trivedi, and T. B. Moeslund. Human pose estimation and activity recognition from multi-view videos: Comparative explorations of recent developments. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):538–552, 2012.

Ting-Chieh Huang, Yi-Jheng Huang, and Wen-Chieh Lin. Real-time horse gait synthesis. *Computer Animation and Virtual Worlds*, 24(2):87–95, 2013.

Marco Hutter, Christian Gehring, Michael Bloesch, Mark A Hoepflinger, C David Remy, and Roland Siegwart. Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion. In *15th International Conference on Climbing and Walking Robot*, 2012.

T. Igarashi, T. Moscovich, and J. F. Hughes. Spatial keyframing for performance-driven animation. In *Proceedings of SCA*, pages 107–115, 2005.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computing*, 25(2):328–373, 2013.

Slobodan Ilic and Pascal Fua. Implicit meshes for surface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):328–333, 2006.

Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. DeeperCut: A deeper, stronger, and faster multi-person pose estimation model. In *ECCV*, 2016.

Catalin Ionescu, Joao Carreira, and Cristian Sminchisescu. Iterated second-order label sensitive pooling for 3d human pose estimation. In *CVPR*, pages 1661–1668, 2014.

Satoru Ishigaki, Timothy White, Victor B. Zordan, and C. Karen Liu. Performance-based control interface for character animation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 28(3):61:1–61:8, 2009.

Alec Jacobson, Ilya Baran, Ladislav Kavan, Jovan Popović, and Olga Sorkine. Fast automatic skinning transformations. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):77:1–77:10, 2012.

Alec Jacobson, Daniele Panozzo, Oliver Glauser, Cédric Pradalier, Otmar Hilliges, and Olga Sorkine-Hornung. Tangible and modular input device for character articulation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4):82, 2014.

Arjun Jain, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. Moviereshape: Tracking and reshaping of humans in videos. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 29(6):148, 2010.

Arjun Jain, Jonathan Tompson, Mykhaylo Andriluka, Graham W. Taylor, and Christoph Bregler. Learning human pose estimation features with convolutional networks. In *ICLR*, 2014.

Arjun Jain, Jonathan Tompson, Yann LeCun, and Christoph Bregler. MoDeep: A deep learning framework using motion features for human pose estimation. In *ACCV*, 2015.

Wenzel Jakob, Christian Regg, and Wojciech Jarosz. Progressive expectation–maximization for hierarchical volumetric photon mapping. *Computer Graphics Forum (Proceedings of Eurographics Symposium on Rendering)*, 30 (4):1287–1297, 2011.

A. Jalobeanu, F. O. Kuehnel, and J. C. Stutz. Modeling images of natural 3D surfaces: Overview and potential applications. In *Proceedings of CVPR Workshops*, pages 188–188, 2004.

Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):399–407, 2005.

Hao Jiang and Kristen Grauman. Seeing invisible poses: Estimating 3D body pose from egocentric video. arXiv:1603.07763, 2016.

Hao Jiang and Lei Zhang. Interactive animating virtual characters with the human body. In *Pacific Rim Conference on Multimedia*, pages 611–620, 2015.

Ming Jin, Dan Gopstein, Yotam Gingold, and Andrew Nealen. Animesh: interleaved animation, modeling, and editing. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6):207, 2015.

Michael Patrick Johnson, Andrew Wilson, Bruce Blumberg, Christopher Kline, and Aaron Bobick. Sympathetic interfaces: Using a plush toy to direct synthetic characters. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 152–158, 1999.

Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. In *CVPR*, 2011.

Andrew Jones, Graham Fyffe, Xueming Yu, Wan-Chun Ma, Jay Busch, Ryosuke Ichikari, Mark Bolas, and Paul Debevec. Head-mounted photometric stereo for performance capture. In *CVMP*, 2011.

Michael J Jones and Tomaso Poggio. Model-based matching by linear combinations of prototypes. A.I. Memo 1583, MIT, 1996.

Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, 2015.

Ioannis A. Kakadiaris and Dimitri Metaxas. Three-dimensional human body model acquisition from multiple views. *International Journal of Computer Vision*, 30(3):191–218, 1998.

Takeo Kanade, PJ Narayanan, and Peter W Rander. Virtualized reality: Concepts and early results. In *Proceedings of ICCV Workshops*, pages 69–76, 1995.

Takashi Kanai, Yutaka Ohtake, and Kiwamu Kase. Hierarchical error-driven approximation of implicit surfaces from polygonal meshes. In *ACM International Conference Proceeding Series*, volume 256, pages 21–30, 2006.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *Proceedings of the Symposium on Interactive 3D Graphics and Games*, pages 39–46, 2007.

Ladislav Kavan, Peter-Pike Sloan, and Carol O'Sullivan. Fast and efficient skinning of animated meshes. *Computer Graphics Forum (Proceedings of Eurographics)*, 29(2):327–336, 2010.

Roland Kehl, Matthieu Bray, and L Van Gool. Markerless full body tracking by integrating multiple cues. In *ICCV Workshop on Modeling People and Human Interaction*, 2005.

EJ Keogh and MJ Pazzani. Derivative dynamic time warping. In *Proceedings of SIAM SDM*, pages 5–7, 2001.

David Kim, Otmar Hilliges, Shahram Izadi, Alex D. Butler, Jiawen Chen, Iason Oikonomidis, and Patrick Olivier. Digits: Freehand 3D interactions anywhere using a wrist-worn gloveless sensor. In *UIST*, 2012.

Hansung Kim and Adrian Hilton. Influence of colour and feature geometry on multi-modal 3D point clouds data registration. In *3DV*, pages 202–209, 2014.

YoungBeom Kim and JungHyun Han. Bulging-free dual quaternion skinning. *Computer Animation and Virtual Worlds*, 25(3-4):321–329, 2014.

Kris M Kitani, Takahiro Okabe, Yoichi Sato, and Akihiro Sugimoto. Fast unsupervised ego-action learning for first-person sports videos. In *CVPR*, 2011.

Henner Kollnig and Hans-Hellmut Nagel. 3D pose estimation by fitting image gradients directly to polyhedral models. In *ICCV*, pages 569–574, 1995.

Ilya Kostrikov and Juergen Gall. Depth sweep regression forests for estimating 3d human pose from images. In *BMVC*, page 5, 2014.

L Kovar, M Gleicher, and F Pighin. Motion graphs. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, pages 473–482, 2002.

Jaka Krivic and Franc Solina. Contour based superquadric tracking. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 1180–1186, 2003.

P.G. Kry, L. Reveret, F. Faure, and M.-P. Cani. Modal Locomotion: Animating Virtual Characters with Natural Vibrations. *Computer Graphics Forum (Proceedings of Eurographics)*, 28(2):289–298, 2009.

Younghee Kwon, Kwang In Kim, James Tompkin, Jin Hyung Kim, and Christian Theobalt. Efficient learning of image super-resolution and compression artifact removal with semi-local Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1792–1805, 2015.

W Michael Lai, David H Rubin, David Rubin, and Erhard Krempl. *Introduction to continuum mechanics*. Butterworth-Heinemann, 2009.

Joe Laszlo, Michael Neff, and Karan Singh. Predictive feedback for interactive control of physics-based characters. *Computer Graphics Forum (Proceedings of Eurographics)*, 24(3):257–265, 2005.

Joseph Laszlo, Michiel van de Panne, and Eugene Fiume. Interactive control for physically-based animation. In *Proceedings of SIGGRAPH*, pages 201–208, 2000.

A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. *NIPS*, pages 329–336, 2004.

LeapMotion. Leapmotion. http://www.leapmotion.com/. Accessed: 2016-08-08.

CS Lee and A. Elgammal. Gait style and gait content: bilinear models for gait recognition using gait re-sampling. *Proceedings of the International Conference on Automatic Face and Gesture Recognition*, pages 147–152, 2004.

Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 21(3): 491–500, 2002.

Mun Wai Lee and Ram Nevatia. Human pose tracking using multi-level structured models. In *ECCV*, pages 368–381, 2006.

Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. Motion fields for interactive character locomotion. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 29(6):138:1–138:8, 2010.

V. Lempitsky, Y. Boykov, and D. Ivanov. Oriented visibility for multiview reconstruction. In *ECCV*, 2006.

Sergey Levine, Jack M. Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 31(4):1–10, 2012.

J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: A unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of SIGGRAPH*, pages 165–172, 2000.

Sijin Li and Antoni B Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *ACCV*, pages 332–347, 2014.

Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rossi, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling Applications*, pages 181–190, 2004.

Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 24(3):479–487, 2005.

Noah Lockwood and Karan Singh. Finger walking: Motion editing with contact-based hand performance. In *Proceedings of SCA*, pages 43–52, 2012.

Matthew Loper, Naureen Mahmood, and Michael J Black. MoSh: Motion and shape capture from sparse markersol. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 33(6):220, 2014.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. SMPL: a skinned multi-person linear model. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6):248, 2015.

William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (Proceedings of SIGGRAPH)*, 21(4):163–169, 1987.

David G Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Minghuang Ma, Haoqi Fan, and Kris M. Kitani. Going deeper into first-person activity recognition. In *CVPR*, 2016.

Ziyang Ma and Enhua Wu. Real-time and robust hand tracking with a single depth camera. *The Visual Computer*, 30(10):1133–1144, 2014.

Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *Proceedings of SIGGRAPH*, pages 369–374, 2000.

Abhimitra Meka, Michael Zollhöfer, Christian Richardt, and Christian Theobalt. Live intrinsic video. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 35(4):109:1–14, 2016.

Alberto Menache. *Understanding Motion Capture for Computer Animation*. Morgan Kaufmann, 2nd edition, 2010.

Clément Ménier, Edmond Boyer, and Bruno Raffin. 3D skeleton-based body pose recovery. In *3rd International Symposium on 3D Data Processing, Visualization and Transmission*, pages 389–396, 2006.

Anton Milan, Stefan Roth, and Konrad Schindler. An analytical formulation of global occlusion reasoning for multi-target tracking. In *Proceedings of the International IEEE Workshop on Visual Surveillance*, 2011.

Jianyuan Min and Jinxiang Chai. Motion graphs++: A compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 31(6):153–153, 2012.

Thomas B. Moeslund, Adrian Hilton, and Volker Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2–3):90–126, 2006.

Thomas B. Moeslund, Adrian Hilton, Volker Krüger, and Leonid Sigal. *Visual Analysis of Humans: Looking at People*. Springer, 2011.

Jean-Sébastien Monzani, Paolo Baerlocher, Ronan Boulic, and Daniel Thalmann. Using an intermediate skeleton and inverse kinematics for motion retargeting. *Computer Graphics Forum*, 19(3):11–19, 2000.

P. Moulon, P. Monasse, and R. Marlet. Global fusion of relative motions for robust, accurate and scalable structure from motion. In *ICCV*, 2013.

Armin Mustafa, Hansung Kim, Jean-Yves Guillemaut, and Adrian Hilton. General dynamic scene reconstruction from multiple view video. In *ICCV*, 2015.

Michael Neff, Irene Albrecht, and Hans-Peter Seidel. Layered performance animation with correlation maps. *Computer Graphics Forum*, 26(3):675–684, 2007.

Richard A Newcombe, Dieter Fox, and Steven M Seitz. DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time. In *CVPR*, pages 343–352, 2015.

Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. *arXiv preprint arXiv:1603.06937*, 2016.

Katsunori Ohnishi, Atsushi Kanehira, Asako Kanezaki, and Tatsuya Harada. Recognizing activities of daily living with a wrist-mounted camera. In *CVPR*, 2016.

BH Olstad, C Zinner, D Haakonsen, J Cabri, PL Kjendlie, R Meeusen, J Duchateau, B Roelands, M Klass, B De Geus, et al. 3d automatic motion tracking in water for measuring intra cyclic velocity variations in breast-stroke swimming. In *17th annual congress of the European college of sport science.*, 2012.

Sageev Oore, Demetri Terzopoulos, and Geoffrey Hinton. A desktop input device and interface for interactive 3D character animation. *Proceedings of Graphics Interface*, pages 133–140, 2002.

Hyun Soo Park, Eakta Jain, and Yaser Sheikh. 3D social saliency from head-mounted cameras. In *NIPS*, 2012.

Hyun Soo Park, Takaaki Shiratori, Iain Matthews, and Yaser Sheikh. 3D trajectory reconstruction under perspective projection. *International Journal of Computer Vision*, 115(2):115–135, 2015.

Sang Il Park and Jessica K. Hodgins. Data-driven modeling of skin and muscle deformation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):96:1–6, 2008.

Sungheon Park, Jihye Hwang, and Nojun Kwak. 3d human pose estimation using convolutional neural networks with 2d pose information. *arXiv preprint arXiv:1608.03075*, 2016.

Tomas Pfister, James Charles, and Andrew Zisserman. Flowing ConvNets for human pose estimation in videos. In *ICCV*, 2015.

Leonid Pishchulin, Stefanie Wuhrer, Thomas Helten, Christian Theobalt, and Bernt Schiele. Building statistical shape spaces for 3D human modeling. *arXiv preprint arXiv:1503.05860*, 2015.

Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter Gehler, and Bernt Schiele. DeepCut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, 2016.

Ralf Plänkers and Pascal Fua. Tracking and modeling people in video sequences. *Computer Vision and Image Understanding*, 81(3):285–302, 2001.

Ralf Plankers and Pascal Fua. Articulated soft objects for multi-view shape and motion capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):63–83, 2003.

Martin Poirier and Eric Paquette. Rig retargeting for 3d animation. In *Proceedings of Graphics Interface*, pages 103–110, 2009.

Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. Multisensor-fusion for 3D full-body human motion capture. In *CVPR*, 2010.

Gerard Pons-Moll, Andreas Baak, Juergen Gall, Laura Leal-Taixé, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. Outdoor human motion capture using inverse kinematics and von Mises-Fisher sampling. In *ICCV*, 2011.

Gerard Pons-Moll, David J. Fleet, and Bodo Rosenhahn. Posebits for monocular human pose estimation. In *CVPR*, 2014.

K. Pullen and C. Bregler. Animating by multi-level sampling. In *Proceedings of Computer Animation*, pages 36–42, 2000.

Michalis Raptis, Darko Kirovski, and Hugues Hoppe. Real-time classification of dance gestures from skeleton animation. In *Proceedings of SCA*, pages 147–156, 2011.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

Nicholas Rhinehart and Kris M. Kitani. Learning action maps of large environments via first-person vision. In *CVPR*, 2016.

Helge Rhodin, James Tompkin, Kwang In Kim, Kiran Varanasi, Hans-Peter Seidel, and Christian Theobalt. Interactive motion mapping for real-time character control. *Computer Graphics Forum (Proceedings of Eurographics)*, 33(2), 2014.

Helge Rhodin, Nadia Robertini, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. A versatile scene model with differentiable visibility applied to generative pose estimation. In *ICCV*, 2015a.

Helge Rhodin, James Tompkin, Kwang In Kim, Edilson de Aguiar, Hanspeter Pfister, Hans-Peter Seidel, and Christian Theobalt. Generalizing wave gestures from sparse examples for real-time character control. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 34(6), 2015b.

Helge Rhodin, Christian Richardt, Dan Casas, Eldar Insafutdinov, Mohammad Shafiei, Hans-Peter Seidel, Bernt Schiele, and Christian Theobalt. Egocap: Egocentric marker-less motion capture with two fisheye cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 35(8), 2016a.

Helge Rhodin, Nadia Robertini, Dan Casas, Christian Richardt, Hans-Peter Seidel, and Christian Theobalt. General Automatic Human Shape and Motion Capture Using Volumetric Contour Cues. In *ECCV*, 2016b.

N. Robertini, D. Casas, H. Rhodin, H.-P. Seidel, and C. Theobalt. Model-based outdoor performance capture. In *3DV*, 2016.

Grégory Rogez and Cordelia Schmid. Mocap-guided data augmentation for 3d pose estimation in the wild. *arXiv preprint arXiv:1607.02046*, 2016.

Grégory Rogez, Maryam Khademi, James Steven Supancic, III, J. M. M. Montiel, and Deva Ramanan. 3D hand pose detection in egocentric RGB-D images. In *Proceedings of ECCV Workshops*, 2014.

Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

Bodo Rosenhahn, Christian Perwass, and Gerald Sommer. Pose estimation of 3D free-form contours. *International Journal of Computer Vision*, 62(3):267–289, 2005.

Bodo Rosenhahn, Thomas Brox, Uwe Kersting, Andrew Smith, Jason Gurney, and Reinhard Klette. A system for marker-less motion capture. *Künstliche Intelligenz*, 1(2006):45–51, 2006.

Benjamin Sapp and Ben Taskar. MODEC: Multimodal decomposable models for human pose estimation. In *CVPR*, 2013.

Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A toolbox for easily calibrating omnidirectional cameras. In *IROS*, 2006.

Yeongho Seol, Carol O'Sullivan, and Jehee Lee. Creature features: online motion puppetry for non-human characters. In *Proceedings of SCA*, pages 213–221, 2013.

Hyun Joon Shin and Jehee Lee. Motion synthesis and editing in low-dimensional spaces. *Computer Animation and Virtual Worlds*, 17(3-4):219–227, 2006.

Hyun Joon Shin and Hyun Seok Oh. Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of SCA*, pages 291–298, 2006.

Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Transactions on Graphics*, 20:67–94, 2001.

Takaaki Shiratori and Jessica K. Hodgins. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 27(5):123:1–123:9, 2008.

Takaaki Shiratori, Hyun Soo Park, Leonid Sigal, Yaser Sheikh, and Jessica K. Hodgins. Motion capture from body-mounted cameras. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 30(4):31:1–10, 2011.

Takaaki Shiratori, Moshe Mahler, Warren Trezevant, and Jessica K Hodgins. Expressing animated performances through puppeteering. In *IEEE Symposium on 3D User Interfaces*, pages 59–66, 2013.

Jamie Shotton, Andrew W. Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from single depth images. In *CVPR*, pages 1297–1304, 2011.

Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

Hedvig Sidenbladh and Michael J Black. Learning the statistics of people in images and video. *International Journal of Computer Vision*, 54(1–3):183–209, 2003.

Hedvig Sidenbladh, Michael J. Black, and David J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *ECCV*, pages 702–718, 2000.

Leonid Sigal, Sidharth Bhatia, Stefan Roth, Michael J Black, and Michael Isard. Tracking loose-limbed people. In *CVPR*, pages I–421, 2004.

Leonid Sigal, Alexandru O Bălan, and Michael J Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1):4–27, 2010.

Leonid Sigal, Michael Isard, Horst Haussecker, and Michael J Black. Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation. *International Journal of Computer Vision*, 98 (1):15–48, 2012.

Cristian Sminchisescu and Alexandru Telea. Human pose estimation from silhouettes. a consistent approach using distance level sets. In *WSCG*, volume 10, 2002.

Cristian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. *The International Journal of Robotics Research*, 22(6):371–391, 2003.

Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Learning joint top-down and bottom-up processes for 3d visual inference. In *CVPR*, volume 2, pages 1743–1752, 2006.

O Sorkine. Least-squares rigid motion using SVD. Technical report, Technical notes, ETHZ, 2009.

O Sorkine, D Cohen-Or, and Y Lipman. Laplacian surface editing. In *Proceedings of SGP*, pages 175–184, 2004.

Olga Sorkine. Differential representations for mesh processing. *Computer Graphics Forum (Proceedings of Eurographics)*, 25(4):789–807, 2006.

Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, 2007.

Srinath Sridhar, Helge Rhodin, Hans-Peter Seidel, Antti Oulasvirta, and Christian Theobalt. Real-time hand tracking using a sum of anisotropic Gaussians model. In *3DV*, 2014.

Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *CVPR*, 2015.

J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27:21–31, 2007.

Jonathan Starck and Adrian Hilton. Model-based multiple view reconstruction of people. In *ICCV*, pages 915–922, 2003.

Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of Gaussians body model. In *ICCV*, pages 951–958, 2011.

D. J. Sturman. Computer puppetry. *IEEE Computer Graphics and Applications*, 18(1):38–45, 1998.

Yu-Chuan Su and Kristen Grauman. Detecting engagement in egocentric video. In *ECCV*, 2016.

Erik B Sudderth, Michael I Mandel, William T Freeman, and Alan S Willsky. Visual hand tracking using nonparametric belief propagation. In *Proceedings of CVPR Workshops*, pages 189–189, 2004.

Yusuke Sugano and Andreas Bulling. Self-calibrating head-mounted eye trackers using egocentric visual saliency. In *UIST*, 2015.

Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 23(3):399–405, 2004.

Robert Walker Sumner. *Mesh modification using deformation gradients*. PhD thesis, Massachusetts Institute of Technology, 2005.

Mankyu Sung. Fast motion synthesis of quadrupedal animals using a minimum amount of motion capture data. *ETRI Journal*, 35(6):1029–1037, 2013.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *ICCV*, pages 517–524, 1998.

Luis M. Tanco and Adrian Hilton. Realistic Synthesis of Novel Human Movements from a Database of Motion. *Workshop on Human Motion*, 2000.

Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics*, 30(3):18, 2011.

Bugra Tekin, Isinsu Katircioglu, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Structured prediction of 3d human pose with deep neural networks. *arXiv preprint arXiv:1605.05180*, 2016a.

Bugra Tekin, Artem Rozantsev, Vincent Lepetit, and Pascal Fua. Direct prediction of 3D body poses from motion compensated sequences. In *CVPR*, 2016b.

S. C. L. Terra and R. A. Metoyer. Performance timing for keyframe animation. In *Proceedings of SCA*, pages 253–258, 2004.

Christian Theobalt, Edilson de Aguiar, Carsten Stoll, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from multi-view video. In *Image and Geometry Processing for 3-D Cinematography*, pages 127–149. Springer, 2010.

Bill Tomlinson, Marc Downie, Matt Berlin, Jesse Gray, Derek Lyons, Jennie Cochran, and Bruce Blumberg. Leashing the alphawolves: Mixing user direction with autonomous emotion in a pack of semi-autonomous virtual characters. In *Proceedings of SCA*, pages 7–14, 2002.

Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, pages 1799–1807, 2014.

Jing Tong, Jin Zhou, Ligang Liu, Zhigeng Pan, and Hao Yan. Scanning 3D full human bodies using Kinects. *IEEE Transactions on Visualization and Computer Graphics*, 18(4):643–650, 2012.

Lorenzo Torresani, Danny Yang, Gene Alexander, and Christoph Bregler. Tracking and modeling non-rigid objects with rank constraints. In *CVPR*, pages 493–500, 2001.

Alexander Toshev and Christian Szegedy. DeepPose: Human pose estimation via deep neural networks. In *CVPR*, 2014.

T. Tung, S. Nobuhara, and T. Matsuyama. Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo. In *ICCV*, pages 1709–1716, 2009.

Martin Tyler and Michael Neff. Interactive quadruped animation. In *Proceedings Motion in Games*, pages 208–219, 2012.

Munetoshi Unuma, Ken Anjyo, and Ryozo Takeuchi. Fourier principles for emotion-based human figure animation. In *Proceedings of SIGGRAPH*, pages 91–96, 1995.

R Urtasun, DJ Fleet, and P Fua. 3D people tracking with Gaussian process dynamical models. *CVPR*, 2006a.

Raquel Urtasun, David J Fleet, and Pascal Fua. Monocular 3d tracking of the golf swing. In *CVPR*, pages 932–938, 2005.

Raquel Urtasun, David J Fleet, and Pascal Fua. Temporal motion models for monocular and multiview 3D human body tracking. *Computer Vision and Image Understanding*, 104(2):157–177, 2006b.

Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popović. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 26(3):35, 2007.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 27(3):97, 2008.

A Vögele, M Hermann, B. Krüger, and R Klein. Interactive steering of mesh animations. In *Proceedings of SCA*, pages 53–58, 2012.

Stefan Wachter and Hans-Hellmut Nagel. Tracking of persons in monocular image sequences. In *Nonrigid and Articulated Motion Workshop*, pages 2–9, 1997.

Jack M. Wang, David J. Fleet, and Aaron Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008.

Jing Wang, Yu Cheng, and Rogerio Schmidt Feris. Walk and learn: Facial attribute representation learning from egocentric video and contextual data. In *CVPR*, 2016a.

Li Wang, Franck Hétroy-Wheeler, and Edmond Boyer. A hierarchical approach for regular centroidal voronoi tessellations. *Computer Graphics Forum*, 35(1):152–165, 2016b.

Robert Y Wang and Jovan Popović. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 28(3):63, 2009.

Rui Wang, Kun Zhou, John Snyder, Xinguo Liu, Hujun Bao, Qunsheng Peng, and Baining Guo. Variational sphere set approximation for solid objects. *The Visual Computer*, 22(9–11):612–621, 2006.

Tinghuai Wang, John Collomosse, and Adrian Hilton. Wide baseline multi-view video matting using a hybrid Markov random field. In *Proceedings of the International Conference on Pattern Recognition*, pages 136–141, 2014.

Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016.

Xiaolin Wei, Peizhao Zhang, and Jinxiang Chai. Accurate realtime full-body motion capture using a single depth camera. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 31(6):188:1–12, 2012.

Thibaut Weise, Sofien Bouaziz, Hao Li, and Mark Pauly. Realtime performance-based facial animation. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 30(4), 2011.

DJ Wiley and JK Hahn. Interpolation synthesis of articulated figure motion. *IEEE Computer Graphics and Applications*, 17(6):39–45, 1997.

Christopher Richard Wren, Ali Azarbayejani, Trevor Darrell, and Alex Paul Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

Chenglei Wu, Kiran Varanasi, and Christian Theobalt. Full body performance capture under uncontrolled and varying illumination: A shading-based approach. In *ECCV*, pages 757–770, 2012.

Chenglei Wu, Carsten Stoll, Levi Valgaerts, and Christian Theobalt. On-set performance capture of multiple actors with a stereo camera. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6):161, 2013.

Xuehan Xiong and Fernando De la Torre. Supervised descent method and its applications to face alignment. In *CVPR*, pages 532–539, 2013.

Katsu Yamane, Yuka Ariki, and Jessica Hodgins. Animating non-humanoid characters with human motion data. In *Proceedings of SCA*, pages 169–178, 2010.

Yi Yang and Deva Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, 2013.

Hashim Yasin, Umar Iqbal, Björn Krüger, Andreas Weber, and Juergen Gall. A dual-source approach for 3d pose estimation from a single image. In *CVPR*, 2016.

Anthony Yezzi and Stefano Soatto. Stereoscopic segmentation. *International Journal of Computer Vision*, 53(1): 31–43, 2003.

KangKang Yin and Dinesh K Pai. Footsee: an interactive animation system. In *Proceedings of SCA*, 2003.

Haruka Yonemoto, Kazuhiko Murasaki, Tatsuya Osawa, Kyoko Sudo, Jun Shimamura, and Yukinobi Taniguchi. Egocentric articulated pose tracking for action recognition. In *Proceedings of Machine Vision Applications*, 2015.

Peizhao Zhang, Kristin Siu, Jianjie Zhang, C Karen Liu, and Jinxiang Chai. Leveraging depth cameras and wearable pressure sensors for full-body kinematics and dynamics capture. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia)*, 33(6):221:1–14, 2014.

K Zhou, W Xu, Y Tong, and M Desbrun. Deformation transfer to multi-component objects. *Computer Graphics Forum (Proceedings of Eurographics)*, pages 319–325, 2010.

Kun Zhou, Qiming Hou, Minmin Gong, John Snyder, Baining Guo, and Heung-Yeung Shum. Fogshop: Real-time design and rendering of inhomogeneous, single-scattering media. *Computer Graphics Forum (Proceedings of Pacific Graphics)*, pages 116–125, 2007.

Xingyi Zhou, Xiao Sun, Wei Zhang, Shuang Liang, and Yichen Wei. Deep kinematic pose regression. *arXiv preprint arXiv:1609.05317*, 2016.

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics (Proceedings of SIGGRAPH)*, 33(4): 156, 2014.