Retrieval-based Approaches for Tracking and Reconstructing Human Motions

Andreas Baak

Department 4: Computer Graphics
Max-Planck-Institut für Informatik
66123 Saarbrücken, Germany

Dissertation zur Erlangung des Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes, 13. November 2012

Betreuender Hochschullehrer / Supervisor:

Prof. Dr. rer. nat. Meinard Müller
Bonn University & MPI Informatik, Bonn, Germany

Gutachter / Reviewers:

Prof. Dr. rer. nat. Meinard Müller
Bonn University & MPI Informatik, Bonn, Germany

Prof. Dr.-Ing. Bodo Rosenhahn
Leibniz Universität Hannover, Germany

Prof. Dr.-Ing. Christian Theobalt
MPI Informatik, Saarbrücken, Germany

Dekan / Dean:

Prof. Dr. Wilhelm F. Maier
Universität des Saarlandes, Saarbrücken, Germany

Eingereicht am / Thesis submitted:

19. Juni 2012 / June 19th, 2012

Datum des Kolloquiums / Date of Defense:

9. November 2012 / November 9th, 2012

Andreas Baak
MPI Informatik, Geb. E1.4
Campus E1.4
66123 Saarbrücken
Germany
abaak@mpi-inf.mpg.de

Summary

Tracking, reconstructing, and analyzing human motions constitute central topics in computer vision and computer graphics. Although marker-less motion tracking has been an active research field for more than two decades, there are still major challenges, in particular when dealing with only few cameras, noise in the image data, occlusions, or fast motions. In this thesis, we introduce novel approaches for increasing the stability, accuracy, and efficiency of marker-less human motion tracking and 3D human pose reconstruction. As one common underlying concept, the presented approaches contain a retrieval component making use of database knowledge in the form of previously recorded marker-based motion capture (mocap) data. In particular, we contribute to three different areas dealing with various types of sensors including video cameras, optical mocap systems, inertial sensors, and depth cameras. Firstly, we introduce content-based retrieval techniques for automatically segmenting and annotating mocap data that is originally provided in form of unstructured data collections. Secondly, we show how such robust annotation procedures can be used to support and stabilize marker-less motion tracking. Thirdly, we develop algorithms for reconstructing human motions from noisy depth sensor data in real-time. In all these contributions, a particular focus is put on efficiency issues in order to keep the run time as low as possible.

Zusammenfassung

Die Analyse und Rekonstruktion von menschlichen Bewegungen aus Sensordaten stellt ein zentrales Thema in den Forschungsgebieten der Computer Vision und Computergrafik dar. Insbesondere die markerfreie Bewegungsschätzung aus Bilddaten weist trotz langjähriger Forschungsaktivitäten noch Defizite auf, die primär bei schnellen Bewegungen oder verrauschten und unvollständigen Sensordaten sichtbar werden. In dieser Arbeit führen wir neue Ansätze zur markerfreien Rekonstruktion menschlicher Bewegungen ein, welche den aktuellen Stand der Forschung im Hinblick auf Stabilität, Genauigkeit und Effizienz signifikant erweitern. Dazu entwickeln wir datengetriebene Methoden, die Vorwissen in Form von Bewegungsdatenbanken nutzen. Insbesondere tragen wir zu drei Teilgebieten bei, die sich mit der Analyse und der Rekonstruktion menschlicher Bewegungen aus Sensordaten von Videokameras, optischen Motion Capture-Systemen, Inertialsensoren und Tiefenkameras beschäftigen. Zunächst erforschen wir inhaltsbasierte Retrieval- und Annotationstechniken im Hinblick auf die Identifikation und Extraktion von geeigneten Ausschnitten aus unstrukturierten Sammlungen von Bewegungsdaten. Dann zeigen wir, wie Retrievaltechniken zur Stabilisierung von markerfreiem Tracking eingesetzt werden können. Schließlich entwickeln wir Algorithmen zur Rekonstruktion von menschlichen Bewegungen in Echtzeit aus verrauschten Tiefendaten. In allen Teilen dieser Arbeit spielen Effizienzaspekte eine große Rolle und es werden schnelle Algorithmen entwickelt und implementiert.

Acknowledgements

I am thankful to many people who supported me during the time of writing this thesis. First of all, I express deep gratitude to my girlfriend Jana Winterscheidt for being the center of my life, for tolerating occasional peaks in workload which sometimes flipped up my unpleasant side, for always being my resting point, and for conveying the important aspects of life: happiness, optimism, to love and being loved. I love you!

I would like to express gratitude to my supervisor Meinard Müller who opened up numerous opportunities for my research. Without you, this work would not have been possible! Thank you for the fruitful collaborations, for guiding and for showing the directions to go, and for leaving enough freedom for me to develop and implement new ideas. In this respect, I am also thankful to Hans-Peter Seidel who created a stimulating environment with excellent working conditions, exposing me to the influence of internationally leading researchers of their fields. In particular, I am thankful to Christian Theobalt who encouraged me to conduct research with time-of-flight cameras. Moreover, I would like to thank Bodo Rosenhahn and Gerard Pons-Moll for a close and fruitful collaboration which spurred my interest in computer vision.

A special word of thanks goes to the ladies of our secretary's office Sabine Budde and Ellen Fries. Thank you for being kind and capable helpers in all kinds of administrative circumstances.

I am thankful to Carsten Stoll for giving me the opportunity for an internship at Weta Digital and for teaching me different aspects of markerless motion capturing.

I would like to give a special word of thanks to my (office) mate and coauthor Thomas Helten for open and instructive discussions about technical peculiarities of different programming languages as well as OpenGL, and lots of technical stuff.

Thanks to my proof-readers Gaurav Bharaj, Nils Hasler, Thomas Helten, Kwang In 'Kimki' Kim, and Mohammed Shaheen, whose comments and suggestions helped to make this document more readable. Also, I would like to thank the anonymous reviewers of the corresponding papers for helpful comments and suggestions for improvement.

There are many people who made sure that I could use my spare time for recreation—in particular, I would like to thank Nils Hasler for always pushing me to my limits on the wall.

Parts of this work were supported by the German Research Foundation with the research plan named "SMART: Stabilized Motion Analysis through Retrieval Techniques" (DFG CL 64/5-1). I would like to thank Bodo Rosenhahn, Michael Clausen, Meinard Müller, and Daniel Cremers for their hard work of writing the corresponding research proposal. Furthermore, parts of the work in this thesis were supported by the Intel Visual Computing Institute.

Contents

Chapter 1

Introduction

1.1 Motivation

Commonly, motion capturing is referred to as the process of recording moving objects and creating three-dimensional digital representations of these motions [Gleicher, 2000]. The resulting motion capture data or simply mocap data constitutes the basis for applications in various fields such as gait analysis, rehabilitation, physical therapy, bio-mechanics research, sports sciences, or performance analysis. With increasing importance, mocap data is used in computer animation to create realistic motions for both movies and video games.

In general, we distinguish between *marker-based* and *marker-less* mocap. To date, mocap data for commercial purposes is predominantly generated using marker-based mocap systems. These systems rely on some sort of markers or other tracking devices that are attached to the actor's body. Although the use of markers might impair or even restrict the actor's movements, in many application scenarios the advantages of marker-based mocap systems greatly outweigh the disadvantages coming from markers being attached to the body.

The goal of marker-less motion tracking is to track the actor's movement without using any markers or other special tracking devices, thus allowing the actor to move in a more natural way. Marker-less motion tracking constitutes a central task in computer vision with many yet unsolved research problems. Most commonly, existing approaches rely on a calibrated set of optical cameras supplying temporally synchronized streams of image data. In order to describe the actor's motion, skeletal mocap data in terms of 3D positions, orientations, and angles of body joints are extracted from this data. To make the tracking task feasible, most of the current systems require further a-priori knowledge. Such knowledge may be given in form of a surface mesh modeling the actor's body, which is often obtained by using a body laser scanner. Another kind of a-priori knowledge may consist of a specific activity the actor is expected to perform. Here, additional knowledge in form of suitable example motions may be used to stabilize the tracking procedure. Although powerful approaches exist that permit stable tracking in studio setups, tracking algorithms still have to be improved in order to meet or possibly exceed the quality and efficiency of marker-based systems. In particular, in view of applications such as surveillance, rehabilitation, and electronic monitoring, there is a high demand for stable, accurate, and easy-to-use tracking algorithms that do not require the use of external markers.

1.2 Contributions and Organization

This thesis is motivated by the fact that existing algorithms for marker-less motion tracking still do not reach the robustness and efficiency of marker-based mocap systems. To stabilize and improve marker-less motion tracking, we investigate the combination of robust retrieval-based algorithms with existing tracking methods. In both the retrieval and the tracking steps of such data-driven algorithms, efficient concepts are investigated and implemented in order to reduce the overall run time of motion reconstruction. In this thesis, we explore multiple sensor types each exhibiting different noise characteristics. Among others, we show how one can analyze and reconstruct motions on the basis of data with problematic noise characteristics as obtained from inertial sensors or depth cameras.

Part I of the thesis deals specifically with retrieval and annotation of marker-based mocap data. There are many ways to generate marker-based mocap data using, *e. g.*, inertial, mechanical, magnetic, or optical systems. Each technology has its own strengths and weaknesses, and we refer to [Elson, 1994; Gleicher, 2000; Wikipedia, 2012] for overviews and discussions about the advantages and drawbacks of such systems. In Part I, we deal with mocap data coming from optical systems. Although such systems can record accurate mocap data very efficiently, the high cost of creating mocap data (coming from specialized and expensive technical equipment and the human resources in form of actors, technicians, *etc.*) motivates the reuse of existing collections of recorded mocap data. Therefore, automated methods for analyzing, structuring and organizing mocap data are needed. A further motivation for developing such methods comes from the fact that mocap data coming from a recording session often has to be segmented and annotated before it can be applied in, *e. g.*, the production of movies or computer games.

We present three major contributions in the first part of this thesis, working towards the goal of creating a robust and efficient framework for automated segmentation and annotation of mocap data. Here, one main difficulty is due to the fact that similar types of motion may exhibit significant spatial as well as temporal variations. To cope with such variations, existing algorithms often make use of computationally expensive warping and alignment techniques. After briefly summarizing in Chapter 2 the basic concepts as used throughout the thesis, we present a novel keyframe-based search algorithm that significantly speeds up the retrieval process and drastically reduces memory requirements (Chapter 3). In contrast to previous index-based strategies, our recursive algorithm can cope with temporal variations. In particular, the degree of admissible deformation tolerance between the queried keyframes can be controlled with an explicit stiffness parameter. While our algorithm works for general multimedia data, we concentrate on demonstrating the practicability of our concept by means of the motion retrieval scenario. Our experiments show that one can typically cut down the search space from several hours to a couple of minutes of mocap data within a fraction of a second.

As a second contribution, in Chapter 4 we introduce a genetic algorithm for automatically learning keyframes for a given motion category. The presented algorithm relies and builds on the efficient keyframe-based search algorithm as presented in Chapter 3. As the main contribution of Part I, we introduce an efficient approach to segment and label mocap data according to a predefined set of motion categories, each specified by a suitable set of positive example motions (Chapter 5). In our novel annotation procedure, the unknown motion data is segmented and annotated by locally comparing it with the available motion classes. In this framework, the keyframe-based search algorithm (Chapter 3) along with the learned keyframes (Chapter 4) are integrated yielding significant

improvements in the annotation quality and efficiency.

In **Part II** of the thesis, we combine retrieval techniques described in Part I with previously developed tracking techniques. In particular, we introduce a novel iterative framework for stabilizing marker-less human motion tracking in a data-driven manner (Chapter 6). In this framework, we start to track without applying prior knowledge to the tracking algorithm. The resulting 3D motion sequences, which may be corrupted due to tracking errors, are locally classified according to available motion categories using an algorithm similar to the one presented in Chapter 5. Depending on the classification result, a retrieval system supplies suitable motion priors, which are then used to regularize and stabilize the tracking in the next iteration step. Experiments with the HumanEVA-II benchmark show that tracking and classification are significantly improved after few iterations.

As a further contribution, in Chapter 7 we introduce a novel framework for automatically evaluating the quality of 3D tracking results obtained from marker-less motion tracking. In our approach, we use additional inertial sensors to generate suitable reference information. In contrast to previously used marker-based systems, inertial sensors are inexpensive, easy to operate, and impose comparatively weak additional constraints on the overall recording setup with regard to location, recording volume, and illumination. As our main contribution, we show how tracking results can be analyzed and evaluated on the basis of suitable limb orientations, which can be derived from 3D tracking results as well as from inertial sensors fixed on these limbs. Our experiments on various motion sequences of different complexity demonstrate that such limb orientations constitute a suitable mid-level representation for robustly detecting most of the tracking errors. In particular, our evaluation approach reveals also misconfigurations and twists of the limbs that can hardly be detected from traditional evaluation metrics.

In **Part III** of the thesis, we develop a data-driven approach for reconstructing human motions, where we use monocular input from a depth camera instead of multiview video streams. Depth cameras have become a widely available sensor type that captures depth images at real-time frame rates. A depth image can be imagined as a “pinpression”, where densely packed little iron nails are pushed forward to create a blueprint of a certain shape. In analogy to the pinpression, each pixel of a depth image stores the distance to the closest object in the scene along its viewing direction. Since this representation contains more depth information than a 2D color image and less information than a full 3D representation, such images are also referred to as 2.5D data. Even though recent approaches have shown that 3D pose estimation from monocular 2.5D depth images has become feasible, there are still challenging problems due to strong noise in the captured depth data and self-occlusions in the motions. In our data-driven approach, we make use of a previously recorded database of full-body poses to stabilize the motion reconstruction. As one main contribution, we develop an efficient algorithm for extracting semantically meaningful pose features from the depth data. These features are then used to retrieve stabilizing pose candidates from the database. By combining such a data-driven technique with an approach for marker-less tracking we achieve stable pose estimates even for complex motions and drift is effectively prevented. In our framework, we contribute with several technical improvements that lead to speed-ups of an order of magnitude compared to previous approaches. Our experiments show that the combination of the introduced techniques facilitates stable and accurate real-time tracking even for fast and complex motions, making it applicable to a wide range of interactive scenarios.

1.3 Publications of the Author

The core parts of this thesis are based on five main publications of the author. In the following, the publications are listed in chronological order, and their relations to this thesis are explained. In further publications, similar techniques and methods as developed in this thesis have been applied to related application scenarios. The corresponding publications will be sketched at the end of this section.

[Baak *et al.*, 2008] **Andreas Baak**, Meinard Müller, and Hans-Peter Seidel. An efficient algorithm for keyframe-based motion retrieval in the presence of temporal deformations. In *Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval (ACM MIR)*, pages 451–458, Vancouver, British Columbia, Canada, October 2008.

In this publication, a novel technique for keyframe-based motion retrieval is introduced. The developed search algorithm is described and evaluated in Chapter 3.

[Müller *et al.*, 2009] Meinard Müller, **Andreas Baak**, and Hans-Peter Seidel. Efficient and robust annotation of motion capture data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 17–26, August 2009.

In this work, a novel framework for automated annotation of mocap data is described. Moreover, a genetic algorithm for learning keyframes from a set of positive and negative example motions is sketched. In Chapter 4, this algorithm is described in more detail, before the framework for mocap annotation is explained and evaluated in Chapter 5.

[Baak *et al.*, 2009] **Andreas Baak**, Bodo Rosenhahn, Meinard Müller, and Hans-Peter Seidel. Stabilizing motion tracking using retrieved motion priors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1428–1435, September 2009.

In this contribution, retrieval techniques as developed in Part I of the thesis are used to retrieve motion priors for stabilizing marker-less motion tracking. The resulting data-driven tracking procedure is described in Chapter 6.

[Baak *et al.*, 2010] **Andreas Baak**, Thomas Helten, Meinard Müller, Gerard Pons-Moll, Bodo Rosenhahn, and Hans-Peter Seidel. Analyzing and evaluating marker-less motion tracking using inertial sensors. In *Proceedings of the 3rd International Workshop on Human Motion. In Conjunction with ECCV*, volume 6553 of *Lecture Notes of Computer Science (LNCS)*, pages 137–150. Springer, September 2010.

In this article, we develop a method that enables the automated evaluation of marker-less motion tracking also in outdoor scenarios by using data recorded from inertial sensors as reference information, see Chapter 7.

[Baak *et al.*, 2011] **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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1092–1099, November 2011.

In this publication, we introduce novel algorithms for reconstructing human motions from depth camera data. In Part III (Chapter 8), we significantly expand this article and present more algorithmic details and results.

Publications with related application scenarios which are not further detailed in this thesis:

[Pons-Moll *et al.*, 2010] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 663–670, June 2010.

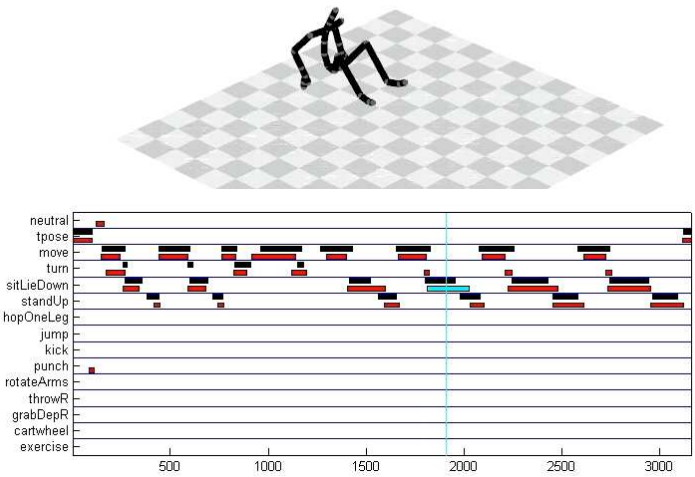
In this work, we designed and implemented an approach for fusing video data with orientation data obtained from inertial sensors to improve and stabilize full-body human motion tracking. To this end, a previously developed local optimization-based approach for tracking is enhanced and stabilized. The performance of the tracking is evaluated on indoor studio recordings.

[Pons-Moll *et al.*, 2011] 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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1243–1250, November 2011.

In this publication, we integrate orientation data obtained from inertial sensors into a particle filter framework for marker-less motion tracking. In contrast to the local optimization-based algorithm in [Pons-Moll *et al.*, 2010], the particle filter framework enables a much more stable tracking at the cost of higher run times. As one main technical contribution, we show how the recorded inertial sensor data can be used to reduce the dimensionality of the tracking task with an efficient analytic inverse kinematics approach. We demonstrate that complex motions can be tracked in an outdoor scenario with image data from just four unsynchronized consumer cameras and orientation data from just five inertial sensors.

Part I

Retrieval and Annotation of Motion Capture Data



Chapter 2

Basic Concepts

This chapter presents a brief review of basic concepts that are used throughout this part of the thesis. We begin with a brief summary of marker-based mocap data in Section 2.1. Then, we summarize relational features as introduced by [Müller *et al.*, 2005] and fix some notation in Section 2.2. Relational features are used as an underlying feature representation in this and the subsequent chapters. Such features transform mocap data into a space that is invariant to global translations and rotations. Moreover, by projecting the mocap data onto semantically meaningful relations, important and discriminative aspects of the motion are retained while a high degree of invariance to subtle and person-specific details in the motions is achieved. Using relational features as a basis, we outline in Section 2.3 the concept of motion templates, which capture the essence of a motion class in a semantically interpretable matrix [Müller and Röder, 2006]. In the subsequent chapters, we show how motion templates can be used in a robust and efficient manner for segmenting and annotating mocap data.

2.1 Marker-based Motion Capture Data

Modern marker-based mocap technology is capable of accurately tracking and recording human motions at high spatial and temporal resolutions. Such systems use cameras in order to record image data of the scene that contains markers. The markers either reflect (see, *e. g.*, [Vicon, 2012]),

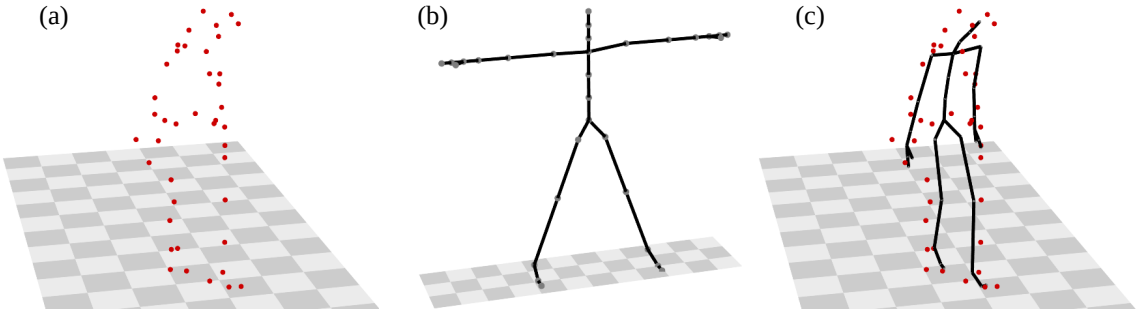


Figure 2.1. (a): 3D Marker positions as recorded from a commercial motion capture system. **(b)**: Skeletal kinematic chain with joints (gray) and bones (black). **(c)**: Reconstructed skeletal pose.

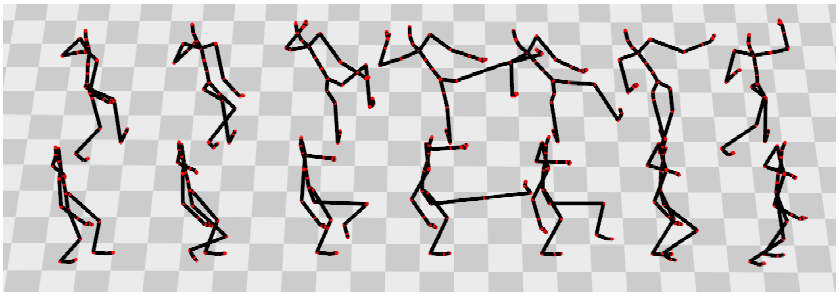


Figure 2.2. Seven poses of a side kick sequence (top) and a front kick sequence (bottom). Even though the two kicking motions are similar in some logical sense, they exhibit significant spatial and temporal differences. From Müller and Röder [2007].

or actively emit (see, *e. g.*, [PhaseSpace, 2012]) light. In order to ease the detection of the marker positions, some systems use infrared light sources and cameras that work in the infrared domain [Vicon, 2012]. From the recognized 2D pixel positions and the calibration of the cameras, 3D positions of the markers can be computed, see Figure 2.1 (a) for an example. From the 3D marker positions, the motion of the underlying skeleton can be reconstructed. To this end, the skeleton is modeled as a kinematic chain [Müller, 2007]. Although methods for automatically computing a suitable skeleton from just a sequence of 3D marker positions exist [Kirk *et al.*, 2005], a template skeleton (similar to Figure 2.1 (b)) is typically provided. In contrast to automatically generated skeletons, one gains full control over the admissible degrees of freedom of the movements by using a manually designed template skeleton. Using inverse kinematics optimization procedures [Murray *et al.*, 1994], joint angles of the skeleton can be determined from the 3D marker positions, see Figure 2.1 (c) for a resulting 3D pose of the skeleton. A temporal sequence of joint angles or joint positions is referred to as mocap data.

Mocap data is used in a variety of applications ranging from motion synthesis in data-driven computer animation to motion analysis in fields such as sports sciences, biomechanics, and computer vision [Kovar and Gleicher, 2004; Müller, 2007; Rosenhahn *et al.*, 2007c]. Although there is a growing corpus of free mocap data, *e. g.*, [CMU, 2003; Müller *et al.*, 2007; Tenorth *et al.*, 2009], there is still a lack of efficient motion retrieval systems that work in a purely content-based fashion without relying on manually generated annotations. Here, the main difficulty is due to the fact that similar types of motions may exhibit significant spatial as well as temporal variations [Kovar and Gleicher, 2004; Müller, 2007]. For example, the two kick sequences shown in Figure 2.2 are logically related even though they differ considerably with respect to motion speed as well as the direction, the height, and the style of the kick.

Most of the previous approaches to motion comparison are based on features that are semantically close to the raw data, using 3D positions, 3D point clouds, joint angle representations, or PCA-reduced versions thereof, see, *e. g.*, [Forbes and Fiume, 2005; Hsu *et al.*, 2005; Keogh *et al.*, 2004; Kovar and Gleicher, 2004; Sakamoto *et al.*, 2004; Wu *et al.*, 2003]. One problem of such features is their sensitivity to pose deformations which may occur in logically related motions. Furthermore, computationally expensive techniques such as dynamic time warping (DTW) are necessary to establish temporal correspondence between related frames [Kovar and Gleicher, 2004]. To cope with spatial variations, Müller *et al.* [2005] introduce the concept of *relational features*, which is based on the following observation. As opposed to other data types such as 3D shape, image, or video, 3D mocap data is explicitly based on a kinematic chain that models the human skeleton.

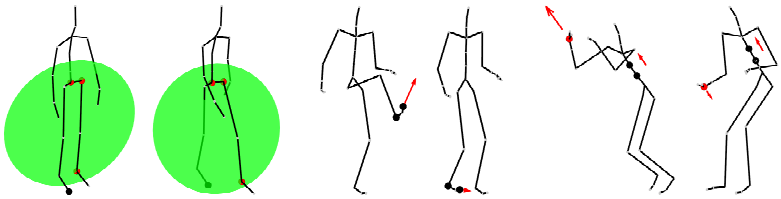


Figure 2.3. Various boolean relational features that encode spatial **(a)**, velocity-based **(b)**, as well as directional information **(c)** between the joints of a pose. From [Müller *et al.*, 2005].

This underlying model can be exploited by looking for semantically meaningful boolean relations between specified points of the body.

2.2 Relational Features

In the following, a stream of mocap data is modeled as a sequence $D = (P_1, P_2, \ldots, P_N)$ of poses $P_n \in \mathcal{P}$ for $n \in [1 : N] := \{1, 2, \ldots, N\}$ (*w.r.t.* a fixed sampling rate), where $\mathcal{P}$ denotes the set of all poses. Here, each pose consists of a full set of 3D coordinates describing the joint positions of a skeletal kinematic chain for a fixed point in time, see Figure 2.1. The idea of *relational features* as introduced by Müller *et al.* [2005] is to describe semantically interpretable, boolean aspects of a pose or a short sequence of poses expressing actions or interactions of certain body parts. Mathematically, a relational feature is a boolean function $F : \mathcal{P} \to \{0, 1\}$ that assumes only the values zero and one. As an example of a relational feature, consider the oriented plane determined by the center of the hip (the root), the left hip joint, and the left foot indicated by the green plane in Figure 2.3 (a). When the right foot lies in front of that plane, this relational feature, which we refer to as F^{15}, is defined to assume the value zero, otherwise one. Interchanging corresponding left and right joints in the definition of F^{15} and flipping the orientation of the resulting plane, we obtain another feature function denoted by F^{16}. Relational features may also encode velocity-based information. For example, one may check whether the absolute velocity of the right foot exceeds a certain velocity threshold, see Figure 2.3 (b). By checking the velocity of the right hand projected onto the direction determined by the belly and chest, one obtains a feature that tests whether the right hand is moving upwards or not, see Figure 2.3 (c).

Forming a vector of f boolean features for some $f \geq 1$, we obtain a combined feature $F : \mathcal{P} \to \{0, 1\}^f$ referred to as a *feature function*. Applying a feature function F with f components to a motion data stream D of length N in a pose-wise fashion yields a *feature matrix* $X \in \{0, 1\}^{f \times N}$, see Figure 2.4. The n^{th} column of X then contains the feature values of frame n and will be denoted by $X(n) := F(P_n)$, $n \in [1 : N]$.

In this thesis, we use relational features as outlined in Table 2.1. Here, the first 39 features are defined and described in [Müller and Röder, 2006; Müller, 2007; Müller *et al.*, 2005] to which we refer to for further details including the specification of various generic features and a discussion of threshold selection. The 40^{th} feature expresses whether the angular velocity of the root joint is high or not.

A feature function F with f components can be used to characterize semantic properties of a motion. As example, consider the feature function $F = (F^{15}, F^{16})$ which gives hints about the

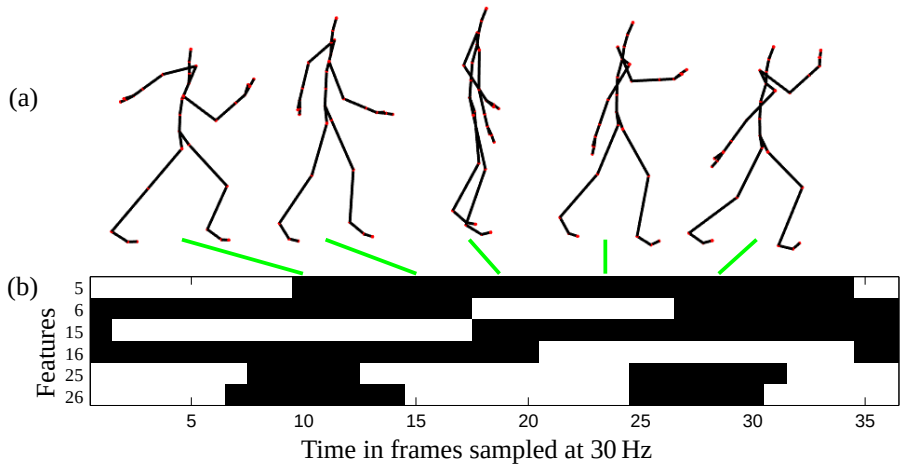


Figure 2.4. Skiing exercise motion. **(a)**: Poses of the motion at frame positions 10, 15, 19, 23, and 28, sampled at 30 Hz. **(b)**: Feature matrix of the skiing motion used in (a). The label numbers of the features correspond to the features used in [Müller and Röder, 2006]. Black encodes the feature value one, white encodes the value zero.

motion of the lower part of the body, see Figure 2.4, rows 15 and 16. Here, the feature values for one execution of a skiing exercise motion have been visualized, where black corresponds to feature value 0 and white corresponds to feature value 1. In the beginning phase of the motion (frame 10), the actor has the right foot in the back and the left foot in the front. This corresponds to $F^{15}(P_{10}) = 1$ and $F^{16}(P_{10}) = 0$. Subsequently, none of the feet are in the back in frame 19, corresponding to $F^{15}(P_{10}) = 0$ and $F^{16}(P_{10}) = 0$. Finally, the positions of the feet are interchanged in frame 28.

As another example, consider the features F^{25}/F^{26} which encode whether the right/left foot is fast. Clearly, during the phases in which the feet are moving in the air, these features show the value 1 in Figure 2.4. The values of the features F^5 (right hand moving upwards) and F^6 (left hand moving upwards) are also depicted in Figure 2.4.

In this manner, Müller and Röder [2006] defined a set of $f = 39$ features, see Table 2.1 for an overview. In addition to these features, in the Chapters 4 and 5 we will use an additional feature F_{40} that expresses whether the angular velocity of the root orientation is high or not. We chose to integrate such an additional feature in order to better discriminate motions with root rotation, *e. g.*, turning motions, from motions without root rotation. This feature set is specifically designed to focus on full-body motions. Note that even though relational features discard a lot of detail contained in the raw motion data, important information regarding the overall configuration of a pose is retained. Moreover, relational motion features are invariant under global orientation and position, the size of the skeleton, and local spatial deformations of a pose.

2.3 Motion Templates

We now review the concept of motion templates which was introduced by Müller and Röder [2006]. As underlying feature representation, we revert to relational features as described in Section 2.2.

ID	description
F_1/F_2	rhand moving forwards
F_3/F_4	rhand above neck
F_5/F_6	rhand moving upwards
F_7/F_8	relbow bent
F_9	hands far apart, sideways
F_{10}	hands approaching each other
F_{11}/F_{12}	rhand moving away from root
F_{13}/F_{14}	rhand fast
F_{15}/F_{16}	rfoot behind lleg
F_{17}/F_{18}	rfoot raised
F_{19}	feet far apart, sideways
F_{20}/F_{21}	rknee bent
F_{22}	feet crossed over
F_{23}	feet moving towards each other, sideways
F_{24}	feet moving apart, sideways
F_{25}/F_{26}	rfoot fast
F_{27}/F_{28}	rhumerus abducted
F_{29}/F_{30}	rfemur abducted
F_{31}	root behind frontal plane
F_{32}	spine horizontal
F_{33}/F_{34}	rhand lowered
F_{35}/F_{36}	shoulders rotated right
F_{37}	Y-extents of body small
F_{38}	XZ-extents of body large
F_{39}	root fast
F_{40}	root rotates around Y

Table 2.1. Description of the 40 relational features used in this part of the thesis. For details of the first 39 features we refer to [Müller and Röder, 2006]. The 40^{th} feature depicts whether the magnitude of the rotational velocity of the root joint is high.

Given a class C consisting of $\gamma \in \mathbb{N}$ example motions, such as the four motions from the class 'sitDownFloor' shown in Figure 2.5 (a), the goal is to automatically learn a motion class representation that grasps the essence of the class. One starts by computing the relational feature vectors for each of the γ motions. The corresponding feature matrices are shown in Figure 2.5 (b), where, for the sake of clarity, we display a subset comprising only eleven of the $f = 40$ features.

Next, a semantically meaningful average over the γ feature matrices is computed. To cope with temporal variations in the example motions, an iterative warping and averaging algorithm is employed which converges to an output matrix X_C referred to as *motion template* (MT) for the class C. The matrix X_C has real-valued entries between zero and one and has a length (number of columns) corresponding to the average length of the training motions. Figure 2.6 (a) shows a motion template obtained from $\gamma = 4$ motions of the class 'sitDownFloor'. The class MT constitutes a combined representation of all four input motions. The important observation is that black/white regions in a class MT indicate periods in time (horizontal axis) where certain features (vertical axis) consistently assume the same values zero/one in all training motions, respectively.

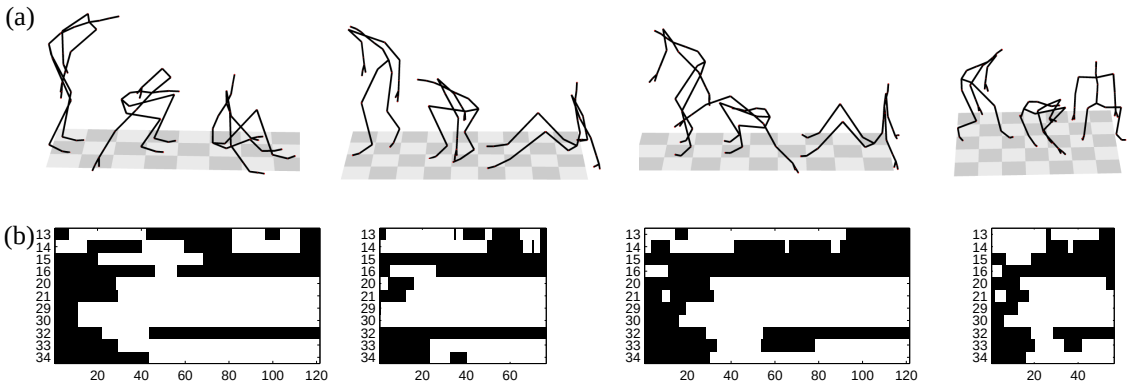


Figure 2.5. **(a)** Selected frames from four different motions of the class 'sitDownFloor'. **(b)** Resulting boolean feature matrices for selected relational features (numbered in accordance with the features defined in [Müller and Röder, 2006]). The columns represent time in frames (using 30 frames per second), whereas the rows correspond to boolean features encoded as black (0) and white (1).

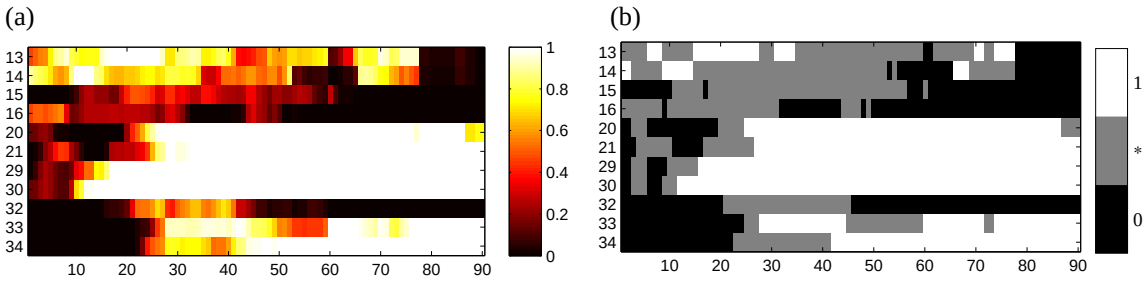


Figure 2.6. **(a)** Class MT for 'sitDownFloor' based on the $\gamma = 4$ training motions shown in Figure 2.5. **(b)** Corresponding quantized class MT.

By contrast, colored regions indicate inconsistencies mainly resulting from variations in the training motions (and partly from inappropriate temporal alignments). In other words, the black/white regions encode characteristic aspects that are shared by all motions, whereas the colored regions represent the class variations coming from different realizations. Finally, one obtains a *quantized MT* by replacing each entry of X_C that is below a quantization threshold δ by zero, each entry that is above $1 - \delta$ by one, and all remaining entries by a *wildcard character* $*$ indicating that the corresponding value is left unspecified, see Figure 2.6 (b).

In our experiments in Section 5.4, we use the threshold $\delta = 0.05$, which has turned out to yield a good trade-off between robustness to motion variations and discriminative power. Only in Chapter 4, where an algorithm for learning keyframes is described, we use the strict quantization threshold $\delta = 0$ in order to determine the features that do not show any variations among the training motions.

Chapter 3

Efficient Keyframe-based Retrieval

This chapter is based on the publication [Baak *et al.*, 2008] and constitutes one main contribution of this thesis. We introduce a novel algorithm for retrieving subsequences of mocap documents based on keyframes. The algorithm is inspired by the following observation. Consider the two kicking motions illustrated by Figure 2.2. Even though there may be large variations between different kicking motions, all such motions share some common characteristics: first the right knee is stretched, then bent, and finally stretched again, while the right foot is raised during this process. Afterwards, the right knee is once again bent and then stretched, while the right foot drops back to the floor. Therefore, by simply checking some characteristic poses in the temporal context, one can exclude all motions in the database that do not share the characteristic progression of relations. These characteristic poses are called *keyframes*. We use the term *keyframe query* to refer to a sequence of keyframes, where each keyframe is specified by a boolean feature vector that describes characteristic relations of a specific pose. Then, the general search strategy using the keyframe query is to extract all parts from the mocap database that exhibit feature vectors matching the keyframe feature vectors in the correct order within suitable time bounds. One important property of our search algorithm is that it allows us to explicitly control the degree of temporal deformations in the retrieval process. Intuitively spoken, the neighboring query keyframes are connected with elastic springs which can be expanded and compressed by a certain factor specified by what we refer to as *stiffness parameter*, see Figure 3.1. Even though our algorithm can handle temporal variations, it works with a standard inverted file index as used in text retrieval [Witten *et al.*, 1999]. Significantly speeding up retrieval and drastically reducing memory requirements, our strategy is ideally suited to cut down the search space in a preprocessing step before applying a more refined analysis to rank and further process the reduced dataset. We will demonstrate such a two-stage retrieval procedure by combining our keyframe-based search with the DTW-based retrieval strategy using motion templates as described in [Müller and Röder, 2006], see also Chapter 2 for a brief introduction.

The remainder of this chapter is organized as follows. We start by giving an overview about related work for this and the following chapters in this part (Section 3.1). Next, we give a motivating example for keyframe-based retrieval (Section 3.2). Then, we describe how we build up an inverted file index (Section 3.3). In Section 3.4, we introduce the query, hit, and match concepts, respectively. The details of the main algorithm and a discussion of its run time behavior are presented in Section 3.5, where we also illustrate the operation mode of our recursive algorithm by means of

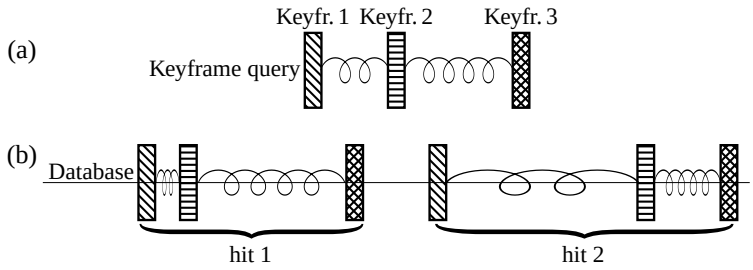


Figure 3.1. (a): In a keyframe query, the keyframes can be thought of as being connected by elastic springs. **(b)**: A database hit has to contain the queried keyframes in the same order and within specified time bounds controlled by a stiffness parameter. Note that the query and the two hits exhibit different temporal deformations.

an explicit example. We then describe our experiments in Section 3.6 and conclude in Section 3.7.

3.1 Related Work

Motion synthesis. The usage of prerecorded human 3D mocap data to create new, naturally looking motion sequences has become a standard procedure in computer animation. Here, *motion graphs* have become a popular tool which is used to efficiently combine fragments of existing motions, see [Kovar *et al.*, 2002; Chai and Hodgins, 2005; Shin and Oh, 2006; Heck and Gleicher, 2007; Lee and Lee, 2006; Safonova and Hodgins, 2007; Beaudoin *et al.*, 2007; Beaudoin *et al.*, 2008; Kovar *et al.*, 2008; Zhao *et al.*, 2009].

Many approaches for motion synthesis rely on morphable models or suitable blending strategies to create new motions from recorded sets of motion capture data [Giese and Poggio, 2000; Kovar and Gleicher, 2004; Arikan *et al.*, 2005; Mukai and Kuriyama, 2005; Hsu *et al.*, 2005; Zordan *et al.*, 2005; Zordan *et al.*, 2007; Lee *et al.*, 2009]. Other approaches rely on having specified keyframes, annotations, they use procedural rules, or physics-based simulation in order to synthesize motions [Pullen and Bregler, 2002; Lee *et al.*, 2002; Arikan *et al.*, 2003; Cooper *et al.*, 2007; Metoyer *et al.*, 2008; Mukai and Kuriyama, 2009; Lee *et al.*, 2010; Wei and Chai, 2011]. By contrast, Lau *et al.* [2009] and Min *et al.* [2010] follow a different line in which new motions are created by a generative model learned from a few example clips. For a further in-depth review of example-based motion synthesis we refer to the survey [Pejsa and Pandzic, 2010].

Current data-driven motion controllers allow us to generate a wide range of task-specific motion sequences satisfying additional spatial and temporal constraints. Most of the proposed controllers are built upon carefully compiled sets of prototype motions that cover the desired range of tasks and execution modes. For example, Rose *et al.* [1998] group similar example motions into "verb" classes to synthesize new, user-controlled motions by suitable interpolation techniques. For synthesizing new motions from motion graphs, Kovar *et al.* [2002] integrate annotation constraints given by a user. Acquisition, capturing and annotation of suitable motions for building up specialized datasets is a labor and cost intensive task [Cooper *et al.*, 2007]. Therefore, various strategies have been described to reuse previously recorded motions stored in a database. In this context, a *thorough and reliable annotation* of the stored motions is of great importance

and such a system will be developed and explained in the forthcoming chapters. Even though there is a rapidly growing corpus of freely available mocap data [CMU, 2003; Müller *et al.*, 2007; Eyes, JAPAN Co. Ltd., 2012; Ohio State University, 2012], there is still a lack of efficient systems that automate the annotation process without manual intervention. Here, one main challenge is to deal with significant spatial as well as temporal variations that may be present in semantically related motions [Kovar and Gleicher, 2004; Müller, 2007].

Indexing and DTW. Most existing retrieval and annotation approaches use indexing methods of some sort in order to speed up the retrieval process. Kovar and Gleicher [2004] propose to compute a so-called "match web" which is based on a self-similarity matrix of the database. This data structure describes potential matches between any pair of motion subsequences and can be used to efficiently search for similar motions. In their approach, results for a query are used as new queries in order to grow a set of similar motions from just a single example motion. However, the match web might be infeasible to build for large mocap databases and the query must be part of the mocap database.

Müller and his colleagues [Müller *et al.*, 2005; Müller and Röder, 2006; Demuth *et al.*, 2006] propose to use semantically motivated relational features to represent motion capture data. Using these features, they develop a fast index-based motion retrieval procedure [Müller *et al.*, 2005; Demuth *et al.*, 2006]. However, given a query motion, the user has to make an informed choice about the features used in the query in order to obtain satisfying retrieval results. In the same retrieval scenario, Gao *et al.* [2006a; 2006b] introduce a scene description language and a precomputation strategy that reduces the run time for processing a query. They also use retrieval techniques to build up a motion graph. Müller and Röder [2006] develop the concept of motion templates which capture consistent and varying aspects of a set of motions. With motion templates, retrieval can be performed without manually selecting feature functions for each query. However, since DTW is used in the retrieval step and no further indexing strategies are used, the method lacks efficient query processing for large databases. In the following chapters, we will show how keyframes can be learned and used in order to assist a motion template-based annotation procedure.

An indexing approach that proceeds in two stages was proposed by Wu *et al.* [2003]. First, start and end frames of possible candidate clips are identified utilizing a pose-based index and then the actual distance from the query is computed via DTW. However, the method of Wu *et al.* [2003] does not enable explicit control over temporal deformations of the keyframes which imposes a strong limitation on their preprocessing method. Liu *et al.* [Liu *et al.*, 2003] index mocap data using a hierarchical data structure that exploits the skeletal structure of mocap data. Based on the indexing method, mocap documents in the database are identified as candidate clips which are further analyzed by a DTW-based comparison. In their approach, the mocap database has to be segmented a priori, whereas our indexing and retrieval techniques handle unsegmented mocap sequences. As another related work, Chiu *et al.* [Chiu *et al.*, 2004] partition the skeleton into nine body parts and construct a separate index map for each body part using self-organizing map (SOM) clustering. Given a query motion, these maps are used in order to quickly identify candidate clips in a database which are further analyzed using DTW. The chosen indexing method uses a bag-of-postures representation of a motion and is thus invariant to the temporal order of the poses. By contrast, our method explicitly models the chronological order of and temporal deformation between keyframes. Similar to the work of Chiu *et al.*, Wu *et al.* [2009b;

2009a] use SOM clustering on the joint angle data in order to perform index-based mocap retrieval. A hierarchical indexing structure consisting of independent index trees each corresponding to a different sub-part of the body is used by Pradhan *et al.* [2009]. A hierarchical part-based representation is also used in the work of Deng *et al.* [2009]. In their method, a fast string matching algorithm on the discovered movement patterns is employed. Forbes and Fiume [2005] identify characteristic poses in both the database and query motions. These poses are used to constrain and speed up a DTW-based similarity computation.

Most of the above mentioned motion retrieval approaches try to cope with nonlinear temporal deformations in the motions with variants of DTW. By contrast, Keogh *et al.* [2004] observe that in many cases nonlinear temporal variations of motions do not have to be modeled in a retrieval scenario. Instead, they propose an indexing technique that accounts only for uniform scaling of mocap data in the temporal domain.

Dimension Reduction Techniques. In order to reduce the complexity and redundancy of the raw mocap data, several approaches employ dimension reduction techniques. Liu *et al.* [2005] represent a pose by a stacked vector of 3D marker locations. They reduce the dimensionality with principle component analysis (PCA) and represent a motion by a piecewise linear model. An indexing method that builds upon a clustering of the poses in a motion is employed in order to efficiently identify similar motions. In a followup work, they show how a piecewise linear model can be used to also reconstruct motions from a reduced marker set [Liu *et al.*, 2006]. Li *et al.* [2007] use singular value decomposition on a joint angle matrix of a motion clip to capture the major geometric structure of the matrix. A support vector machine is employed for classifying hand- as well as full-body motions. The ISOMAP dimension reduction technique is used in the work of Xiang *et al.* [2007] and Guo *et al.* [2011]. After compacting the mocap data, the subsequence DTW algorithm is employed for the computation of retrieval results.

Krüger *et al.* [2010] show that fast *kd*-tree-based nearest-neighbor searches along with viable medium-dimensional feature sets can lead to drastic speed-ups for several existing approaches to motion retrieval. Ren *et al.* [2011] precompute a BIRCH-based incremental clustering of the database. Then, they project each pose onto the nearest cluster center. Retrieval is performed using a variant of a longest common subsequence algorithm which is used to compare two sequences of cluster centers identifiers.

Several approaches to classification and recognition of motion patterns are based on Hidden Markov Models (HMMs), which are also a flexible tool to capture spatio-temporal variations, see, *e. g.*, [Brand and Hertzmann, 2000; Xiang, 2007; Wang and Lee, 2009]. Temporal segmentation of motion data can be viewed as another form of annotation, where consecutive, logically related frames are organized into groups, see, *e. g.*, [Barbič *et al.*, 2004].

Real-time approaches. Recently, real-time applications of motion retrieval and classification were discussed. For example, Deng *et al.* [2011] perform real-time recognition of dance motions. To this end, they subdivide the representation of a human body into five parts and project joint angle representations for each part onto cluster centers obtained from training data. Using a variant of DTW on the projected data, the input motion is continuously compared to template motions in real-time. Based on distance scores to template motions, a motion of a virtual dancing double is played back. Raptis *et al.* [2011] develop a real-time classification system for dance

motions. The classifier relies on an alignment of the performed motions to a given musical beat. Numaguchi *et al.* [2011] use an actual puppet equipped with rotational sensors in order to allow a user to intuitively formulate a query and obtain retrieved motions interactively.

Most of the above cited procedures use motion representations that are semantically close to the raw data. Here, problems occur when one has to cope with strong pose deformations within a class of logically related motions. Approaches such as [Liu *et al.*, 2005; Müller *et al.*, 2005] absorb spatial and temporal variations already on the feature level, which then facilitates a more robust and efficient motion comparison. In this thesis, we cope with spatial variations by using relational features as introduced in [Müller *et al.*, 2005]. As for temporal variations, we introduce an efficient keyframe-based indexing technique in the forthcoming chapters. As a further tool, we make use of the DTW algorithm for more refined temporal alignments.

3.2 Motivating Example

As a motivating example for the use of keyframes, consider a skiing exercise motion class as the one sketched in Figure 3.2 (a). Such a motion class is characterized by a diametrical backward and forward swinging of arms and legs coupled with a joint air and landing phase of the two feet. Note that by considering only the sketched poses or even a subset thereof, a human can easily distinguish this motion class from many other types of motion. Intuitively, we want to find a way how to efficiently encode and search for characteristic aspects of a motion class. These characteristic aspects correspond to *keyframes* which should carry the discriminative essence of a motion class. We now consider two different executions of the skiing exercise motion and plot the corresponding feature matrices in Figure 3.2 (b) and (c). Note that the two feature matrices disagree in length and the sequence of feature values because both executions of the motion differ in speed and style. In spite of this, the characteristic poses as sketched in Figure 3.2 (a) are represented using the same feature vectors in both executions of the motion, respectively. In the figure, we highlight the feature vectors of the corresponding poses using green boxes. Such feature vectors $\in \{0,1\}^f$, however, do not constitute a practical representation of a keyframe because no uncertainties in the poses are expressed. In fact, only two example motions are shown and only the most discriminative 6 out of the 39 features described in Section 2.2 are plotted in the figure. The remaining training motions as well as the other features can exhibit more variations and differences since the skiing motion class permits significant differences in the style of execution. For example, consider a feature that expresses whether the arm moves upwards. Among different executions of the motion, strong inconsistencies can be found in the values of this feature since some actors tend to lift the hands while moving forwards, and others keep the hands on the same height with respect to the body.

We express such uncertainties using the wildcard character $*$ as used in quantized motion templates, see Section 2.3. A quantized motion template for the skiing motion class trained from 15 example motions is depicted in Figure 3.2 (d). Inconsistencies in the executions of the training motions are reflected by the wildcard character $*$ (gray regions in the figure). For example, consider frame 16 of the motion template. In this phase of the motion, some actors raised the left hand and some actors kept the left hand at the same height. This is reflected by the value $*$ in the row corresponding to F_6 in the motion template.

As will be explained in the following section, in view of efficient retrieval using a standard inverted file index, we use boolean keyframe vectors as queries. In order to handle uncertainties represented

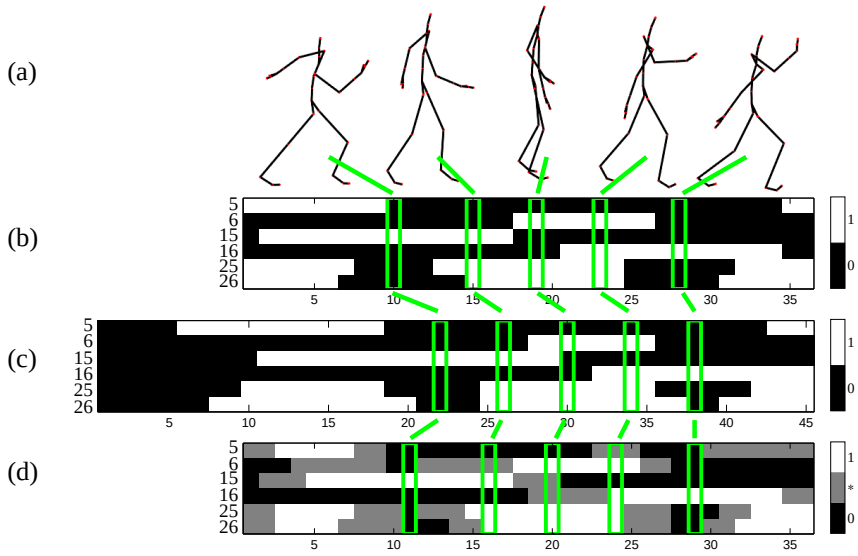


Figure 3.2. Skiing exercise motion. **(a)**: Poses of a motion at frame positions 10, 15, 19, 23, and 28 (30 Hz). **(b)**: Feature matrix of the skiing motion used in (a). The label numbers of the features correspond to the features used in [Müller and Röder, 2006]. Black encodes the feature value one, white encodes the value 0. **(c)**: Feature matrix of another execution of the skiing exercise. **(d)**: Quantized motion template of the skiing motion class trained with 15 training motions from the HDM05 [Müller *et al.*, 2007] motion database.

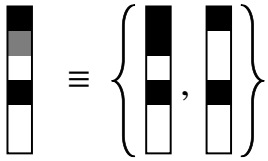


Figure 3.3. A keyframe with a wildcard character is handled by expanding the wildcard character and creating a set of keyframes.

by wildcard characters in the query vectors, we expand the wildcard characters and create a set of boolean keyframe vectors as indicated in Figure 3.3.

Recall from Section 2.3 that a motion template can be thought of as a generalized feature matrix which is obtained by suitably averaging the feature matrices of the training motions. Müller *et al.* [2006] suggest an MT-based motion retrieval method using a variant of dynamic time warping (DTW) to locally compare a motion template with the feature matrices of the unknown motion data. In the following section, we explain our keyframe-based algorithm and exemplify its capabilities by an application which speeds up an MT-based retrieval technique. Here, we first run our keyframe-based search algorithm to efficiently reduce the dataset. Then, we rank the reduced dataset by applying an MT-based retrieval component.

3.3 Indexing

Let $\mathcal{D} = (D_1, D_2, \ldots, D_I)$ denote a database consisting of mocap data streams or documents D_i, $i \in [1 : I]$. For simplicity, we may assume that the database $\mathcal{D}$ consists of one large document

$D = (P_1, \ldots, P_N)$. This can be achieved by concatenating the documents $D_1, \ldots, D_I$ while keeping track of document boundaries in a supplemental data structure. Note that due to their boolean nature, relational features are ideally suited for indexing. Let F be a fixed feature function having f relational features as its components, see Section 2.2. Then, for each feature vector $v \in \{0,1\}^f$ one stores the *inverted list* $L(v)$ consisting of the indices $n \in [1 : N]$ with $v = F(P_n)$. In other words, $L(v)$ shows which of the poses of D exhibit the feature vector v. In a preprocessing step, we construct a query-independent index structure $I_F^{\mathcal{D}}$ consisting of the 2^f inverted lists $L(v)$, $v \in \{0,1\}^f$. Note that one only has to store the non-empty lists. Furthermore, to control the number of index words, one can also split up the feature function into several feature functions and then work with the resulting smaller indices in parallel, see [Müller *et al.*, 2005]. The elements of the inverted lists are stored in ascending order, accounting for efficient union and intersection operations in the subsequent query stage. To further reduce the size of the index, the elements of each list $L(v)$ are run-length encoded. Using this encoding, only one entry in an inverted list is generated for a time section in a feature sequence in which the features do not change.

3.4 Query, Hit, and Match Concept

As mentioned in the introduction of this chapter, certain types of motions typically exhibit characteristic relations that already discriminate these motions from most other types of motions. For example, a cartwheel motion can be distinguished from most other motions simply by checking whether the body is upside down in the execution of the motion. The idea is to express the characteristic relations of a keyframe pose by a suitable feature vector $v \in \{0,1\}^f$ with respect to a fixed feature function F. Since using all components of F is often too restrictive, we allow an entire set

$$V \subseteq \{0,1\}^f \tag{3.1}$$

of alternative feature vectors to describe the characteristic relations. These sets can be derived from a quantized motion template as indicated in Figure 3.3. As an alternative, such keyframes can be generated in an automated way from example motions by employing a genetic algorithm, see Chapter 4. In the following, such a set V is simply referred to as *keyframe*.

A *keyframe query* of length K is a tuple $(\mathbf{V}, \mathbf{d})$ consisting of a sequence $\mathbf{V} = (V_1, \ldots, V_K)$ of keyframes $V_k \subseteq \{0,1\}^f$, $k \in [1 : K]$, and a sequence $\mathbf{d} = (d_1, \ldots, d_{K-1})$ of keyframe distances $d_k \in \mathbb{N}_0$, $k \in [1 : K-1]$. Here, d_k specifies the distance (in frames) of the neighboring keyframes V_k and V_{k+1}. To account for temporal deformations, we introduce a *stiffness parameter* $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{K-1})$, $\sigma_k \in [0,1]$, which controls the degree of expansion and compression allowed in the matching process.

A *hit* in the database document $D = (P_1, \ldots, P_N)$ with respect to the query $(\mathbf{V}, \mathbf{d})$ is a sequence $(n_1, \ldots, n_K)$ of increasing indices $1 \leq n_1 \leq \ldots \leq n_K \leq N$ such that the following two conditions are fulfilled:

$$\forall k \in [1 : K]: \qquad F(P_{n_k}) \in V_k \tag{3.2}$$

$$\forall k \in [1 : K-1]: \qquad \sigma_k \cdot d_k \leq n_{k+1} - n_k \leq \frac{1}{\sigma_k} \cdot d_k \tag{3.3}$$

Here, Condition (3.2) implies the occurrences of the characteristic keyframe poses and Condition (3.3) ensures that the distances of two consecutive keyframes are within the tolerated time

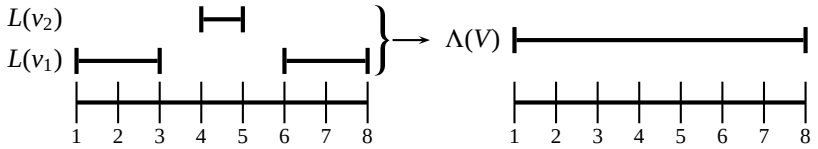


Figure 3.4. To compute the inverted keyframe list of a keyframe $V = \{v_1, v_2\}$, corresponding inverted lists are combined. The run-lengh-encoding step can reduce the total number of segments.

bounds specified by $\boldsymbol{\sigma}$ and keyframe distances in the query. Note that choosing $\sigma_k = 1$ implies that the keyframe distance of keyframes V_k and V_{k+1} within a hit have to coincide with the distances within the query (*i. e.*, there is no deformation tolerance). On the other hand, for $\sigma_k = 0$ (we then set $\frac{1}{\sigma_k} = \infty$) there are no deformation bounds—the keyframes within a hit simply have to appear in the order as specified by the query.

The number of different hits may explode with decreasing stiffness. For example, a small deviation in one of the keyframe positions already defines, in mathematical terms, a different hit. In applications, one is typically not interested in all hits but only in a set of representative hits. We therefore soften the frame-based notion of a hit and assume a segment-focused view. For each query keyframe V_k, $k \in [1 : K]$, we define an *inverted keyframe list*

$$\Lambda_k := \Lambda(V_k) := \bigcup_{v \in V_k} L(v). \tag{3.4}$$

Note that all inverted lists are sorted in lexicographic order which allows us to run an efficient merging algorithm in $O(\sum_{v \in V_k} |L(v)|)$. Further, note that each segment in an inverted list corresponds to a run of feature vectors in a document. Thus, by construction of the inverted lists, all considered segments are pairwise disjoint and we do not have to consider overlapping segments in the merging algorithm. We again look for maximal runs of consecutive indices in the merged list (similar to run-length encoding) and store the resulting inverted keyframe list, see Figure 3.4. Each such run is defined by a *segment* $[s : t]$ with integers $s \leq t$, where s denotes the start frame and t denotes the end frame of the segment. Then, one can encode the inverted keyframe list Λ_k by a sequence

$$\Lambda_k = ([s_{k,1} : t_{k,1}], \ldots, [s_{k,\ell_k} : t_{k,\ell_k}]) \tag{3.5}$$

of segments, where ℓ_k denotes the number of segments. Note that because of the maximality of the runs, one has

$$\forall i \in [1 : \ell_k - 1] : \quad t_{k,i} + 1 < s_{k,i+1} \tag{3.6}$$

Now, a sequence $M = (p_1, \ldots, p_K)$ with $p_k \in [1 : \ell_k]$, $k \in [1 : K]$, is called a *match* in D with respect to the query $(\mathbf{V}, \mathbf{d})$, if there exists a hit $H = (n_1, \ldots, n_K)$ with

$$\forall k \in [1 : K] : \quad s_{k,p_k} \leq n_k \leq t_{k,p_k}. \tag{3.7}$$

In this case, we also say that the match M *contains* the hit H. In other words, a match specifies a sequence of segments (rather than a sequence of frames) containing at least one hit. In the following, we think of p_k being a pointer to the segment $[s_{k,p_k} : t_{k,p_k}]$, see Figure 3.5. The motivation of this notion becomes clear in Section 3.5 when we describe the main algorithm.

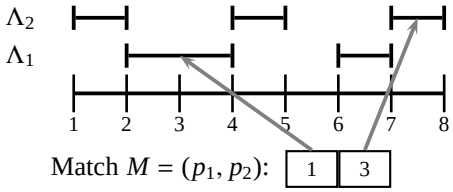


Figure 3.5. Two inverted keyframe lists Λ_1 and Λ_2. Here, p_1 points to the segment $\Lambda_1(p_1) = [2:4]$ and p_2 to the segment $\Lambda_2(p_2) = [7:8]$.

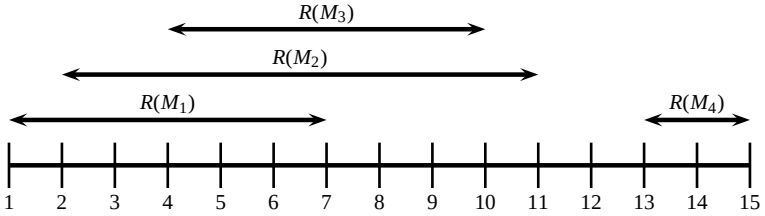


Figure 3.6. Ranges $R(M_i)$ (indicated by arrows) of four different matches M_i, $i \in [1:4]$. The ranges $R(M_1) = [1:7]$, $R(M_2) = [2:11]$ and $R(M_3) = [4:10]$ overlap, whereas $R(M_4) = [13:15]$ is disjoint to the other ranges.

Of course, a match may contain several hits. For a given match M let $R(M) = [s:t]$ be the segment (given by start frame s and end frame t) of minimal length that comprises all hits contained in M. We also refer to $R(M)$ as *hit relevant range* of M. For example, assume that the match $M = (p_1, p_2) = (1, 3)$ of Figure 3.5 contains exactly the three hits $H_1 = (3, 7)$, $H_2 = (4, 7)$, and $H_3 = (4, 8)$, then $R(M) = [s:t] = [3:8]$.

In the case that hit relevant ranges of several matches overlap, we consider, as a further reduction, the union of these ranges instead of the individual ranges. This is motivated by our strategy of running a multistage retrieval procedure. Here, in the first stage, for each document in the database it suffices to extract coarse candidates that contain the keyframe query at least once (as a kind of preselection). As an example, consider the four ranges shown in Figure 3.6. The ranges $R(M_1) = [1 : 7]$, $R(M_2) = [2 : 11]$, and $R(M_3) = [4 : 10]$ overlap, whereas $R(M_4) = [13 : 15]$ is disjoint to the other ranges. The union of the first three ranges defines the segment $[1 : 11]$. Note that the union does not change when considering only the first two ranges leaving out the range $R(M_3)$. We then say that M_3 is an *irrelevant match*. Our keyframe-based search algorithm to be presented next may actually leave out some matches, but for these one can show that they are irrelevant matches.

3.5 Main Algorithm

Before describing our main algorithm, we introduce some further notation. Generalizing the above notion, a *segment* is an element of the set

$$\mathcal{S} := \{[s:t] : s \in \mathbb{N}_0, t \in \mathbb{N}_0 \cup \{\infty\}, s \leq t\} \cup \{\emptyset\}. \tag{3.8}$$

Algorithm 1 Keyframe-based Search

Input: $(\mathbf{V}, \mathbf{d})$: keyframe query comprising K keyframes
σ: stiffness parameter
$I_F^{\mathcal{D}}$ inverted file index
Output: hitRanges: the union of all hit relevant ranges
Global: $(p_1, \ldots, p_K)$: pointers into the keyframe lists
hitRanges

1: **procedure** KEYFRAMEBASEDSEARCH($\mathbf{V}$, $\mathbf{d}$, σ, $I_F^{\mathcal{D}}$)
2: **for** $k \leftarrow 1$ **to** K **do**
3: $\Lambda_k \leftarrow \bigcup_{v \in V_k} L(v)$
4: $p_k \leftarrow 1$
5: **end for**
6: **for** $p_1 \leftarrow 1$ **to** ℓ_1 **do**
7: admissibleRange $\leftarrow \mu_1(\Lambda_1(p_1))$
8: RECURSIVESEARCH(2, admissibleRange)
9: **end for**
10: **end procedure**

11: **procedure** RECURSIVESEARCH(k, $[s:t]$)
12: **while** $p_k \leq \ell_k \wedge t_{k,p_k} < s$ **do**
13: $p_k \leftarrow p_k + 1$
14: **end while**
15: pointerIncremented $\leftarrow$ FALSE
16: intersection $\leftarrow \Lambda_k(p_k) \cap [s:t]$
17: **while** intersection $\neq \emptyset$ **do**
18: **if** $k = K$ **then**
19: hitRanges $\leftarrow$ hitRanges $\cup R((p_1, \ldots, p_K))$
20: **else**
21: admissibleRange $\leftarrow \mu_k$(intersection)
22: RECURSIVESEARCH($k + 1$, admissibleRange)
23: **end if**
24: $p_k \leftarrow p_k + 1$
25: pointerIncremented $\leftarrow$ TRUE
26: **if** $p_k > \ell_k$ **then**
27: intersection $\leftarrow \emptyset$
28: **else**
29: intersection $\leftarrow \Lambda_k(p_k) \cap [s:t]$
30: **end if**
31: **end while**
32: **if** pointerIncremented = TRUE **then**
33: $p_k \leftarrow p_k - 1$
34: **end if**
35: **end procedure**

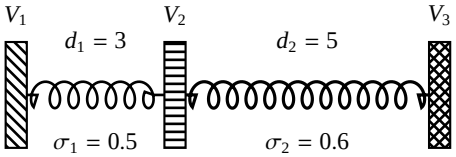


Figure 3.7. Keyframe query of our running example.

The intersection of two segments is defined as $\emptyset$ in case one of the segments is empty. Otherwise, for segments $[s_1 : t_1] \in \mathcal{S}$ and $[s_2 : t_2] \in \mathcal{S}$, we define

$$[s_1 : t_1] \cap [s_2 : t_2] := \begin{cases} \emptyset, \text{ if } \quad t_1 < s_2 \text{ or } t_2 < s_1 \\ [\max(s_1, s_2) : \min(t_1, t_2)], \text{ else.} \end{cases} \tag{3.9}$$

Our keyframe-based search algorithm (see Algorithm 1 on page 24) consists of a main procedure called KeyframeBasedSearch, and a recursive procedure called RecursiveSearch. The input consists of the inverted file index $I_F^{\mathcal{D}}$, a keyframe query $(\mathbf{V}, \mathbf{d})$ with $\mathbf{V} = (V_1, \ldots, V_K)$ and $\mathbf{d} = (d_1, \ldots d_{K-1})$, as well as a stiffness parameter $\boldsymbol{\sigma} = (\sigma_1 \ldots, \sigma_{K-1})$. Recall from Section 3.3 that the index $I_F^{\mathcal{D}}$ does not depend on the query. The algorithm outputs unions of hit relevant ranges which comprise all matches except for possibly some irrelevant ones. These hit ranges as well as the pointers $(p_1, \ldots, p_K)$ are given by global variables and are consistent in both procedures. In the following, we illustrate the functioning of our algorithm by means of an explicit example with three keyframes $\mathbf{V} = (V_1, V_2, V_3)$ and frame distances $\mathbf{d} = (3, 5)$. As for the stiffness parameter, we use $\boldsymbol{\sigma} = (0.5, 0.6)$, see also Figure 3.7.

The procedure KeyframeBasedSearch takes care of the initialization and the first step. In Line 3, the inverted keyframe lists $\Lambda_1, \ldots, \Lambda_K$ are computed, see Equation (3.4). Recall that each list Λ_k, $k \in [1 : K]$, consists of a sequence of ℓ_k segments, see Equation (3.5). In Line 4, the pointers $(p_1, \ldots, p_K)$ are all initialized to the value one, thus pointing to the first segments of the respective lists. For our running example, this state is also illustrated by Figure 3.8 (a). The three keyframe lists are

$$\begin{aligned} \Lambda_1 &= ([3:4], [6:7]), \\ \Lambda_2 &= ([1:2], [4:6], [9:10]), \quad \text{and} \\ \Lambda_3 &= ([1:3], [5:8], [11:12], [14:15]). \end{aligned} \tag{3.10}$$

Now, the **for**-loop in Line 6 sweeps over all segments $\Lambda_1(p_1)$, $p_1 \in [1 : \ell_1]$. Note that these segments exactly contain the database frames that match the first keyframe V_1. In other words, for each hit $H = (n_1, \ldots, n_K)$ one has $n_1 \in \Lambda_1(p_1)$ for some $p_1 \in [1 : \ell_1]$. Line 7 specifies an admissible search range $\mu_1(\Lambda_1(p_1))$ for the second keyframe V_2. More generally, given a segment of candidate frames for the k^{th} keyframe, μ_k computes the admissible range, which is specified by d_k and σ_k, for the $(k+1)^{\text{th}}$ keyframe. Here, the function $\mu_k := \mu_{\sigma_k, d_k} : \mathcal{S} \to \mathcal{S}$, $k \in [1 : K-1]$, is defined as

$$\begin{aligned} [s:t] &\mapsto \begin{cases} [s + \lceil \sigma_k \cdot d_k \rceil : t + \left\lfloor \frac{1}{\sigma_k} \cdot d_k \right\rfloor], & \text{if } \sigma_k > 0 \\ [s:\infty], & \text{if } \sigma_k = 0. \end{cases} \\ \emptyset &\mapsto \emptyset \end{aligned} \tag{3.11}$$

To illustrate this definition, we consider our running example for the case $p_1 = 1$. Then, $\Lambda_1(p_1) = [3:4]$ and

$$\begin{aligned}\mu_1([3:4]) &= \mu_{\sigma_1,d_1}([3:4]) && (3.12)\\ &= \mu_{0.5,3}([3:4]) && (3.13)\\ &= [3 + \lceil 0.5 \cdot 3 \rceil : 4 + \left\lfloor \frac{1}{0.5} \cdot 3 \right\rfloor] && (3.14)\\ &= [3 + 2 : 4 + 6] && (3.15)\\ &= [5:10]. && (3.16)\end{aligned}$$

In other words, if the first keyframe lies within the segment $[3:4]$, then the second keyframe must lie within the segment $[5:10]$ to fulfill Condition (3.3), see Figure 3.8 (a). Finally, Line 8 triggers the recursion starting with the second keyframe and the admissable range $\mu_1(\Lambda_1(p_1))$.

The procedure RecursiveSearch starts with fast forwarding the current pointer p_k (Line 13) until the list end is reached or until the current segment $\Lambda_k(p_k) = [s_{k,p_k} : t_{k,p_k}]$ does not lie entirely to the left of the admissible range $[s : t]$. In our example, this is the case for $p_2 = 2$, where $\Lambda_2(p_2) = [4 : 6]$. Line 16 calculates the intersection of the current segment and the admissible range. The intersection defines a segment of candidate frames that match keyframe V_k and fulfill the distance condition (3.3) for at least one frame of the previous segment $\Lambda_{k-1}(p_{k-1})$. In our example, the intersection is $[4:6] \cap [5:10] = [5:6]$.

In the **while**-loop, starting at Line 17, all segments in Λ_k that lead to a non-empty intersection with the admissible range $[s:t]$ are considered. Here, the increment of the pointer p_k and computation of the intersections is handled between Line 24 and Line 30. In the case $k = K$, each such intersection contributes to a hit (this directly follows from what was said above). Therefore, in Line 19, the hit relevant range of a resulting match is computed and the union is formed with the previously computed hit relevant ranges. An example for this step will be discussed later. In the case $k < K$, a new admissible range is computed (Line 21), and a recursion is triggered with the $(k+1)^{\text{th}}$ keyframe (Line 22).

We continue our example with $p_2 = 2$ and the non-empty intersection $[5 : 6]$. In Line 21, the admissible range $\mu_2([5:6]) = [8:14]$ is computed, see Figure 3.8 (b) for the state of the algorithm at this step. Line 22 triggers another call of RecursiveSearch for the third keyframe. At this recursion level, p_3 is incremented to $p_3 = 2$ (Line 13) and the intersection of the current segment $\Lambda_3(p_3)$ and the admissible range is $[5:8] \cap [8:14] = [8:8]$, see Figure 3.8 (c). Now, the condition $k = K$ is fulfilled. The pointers $(p_1, p_2, p_3) = (1, 2, 2)$ define a match and Line 19 extends the union of the hit relevant ranges by $R((1, 2, 2)) = [3:8]$. At this point we note that the hit relevant range $[s:t]$ of a given match can be computed efficiently. Here, the end frame of the intersection, calculated in Line 16, yields t. To calculate s, one has to backtrack from the intersection in the K^{th} keyframe list to the first keyframe list. Then, Line 24 sets $p_3 = 3$ and the resulting intersection is $[11:12] \cap [8:14] = [11:12]$ (Line 30). The pointers $(p_1, p_2, p_3) = (1, 2, 3)$ define another match with $R((1, 2, 3)) = [3:12]$, which is merged with the previously found hit relevant range by means of the union operator in Line 19. After incrementing to $p_3 = 4$, Line 27 sets the intersection to $[14 : 14]$ and $R((1, 2, 4)) = [3 : 14]$ is processed in Line 19. Now, incrementing the pointer p_3 in Line 24 exceeds the list boundary, so that the empty intersection as computed in Line 27 causes the **while**-loop to stop. In Line 33, the pointer p_3 is decremented to the previous value, where the intersection was non-empty. In our example, we then have $p_3 = 4$. Note that the decrementation

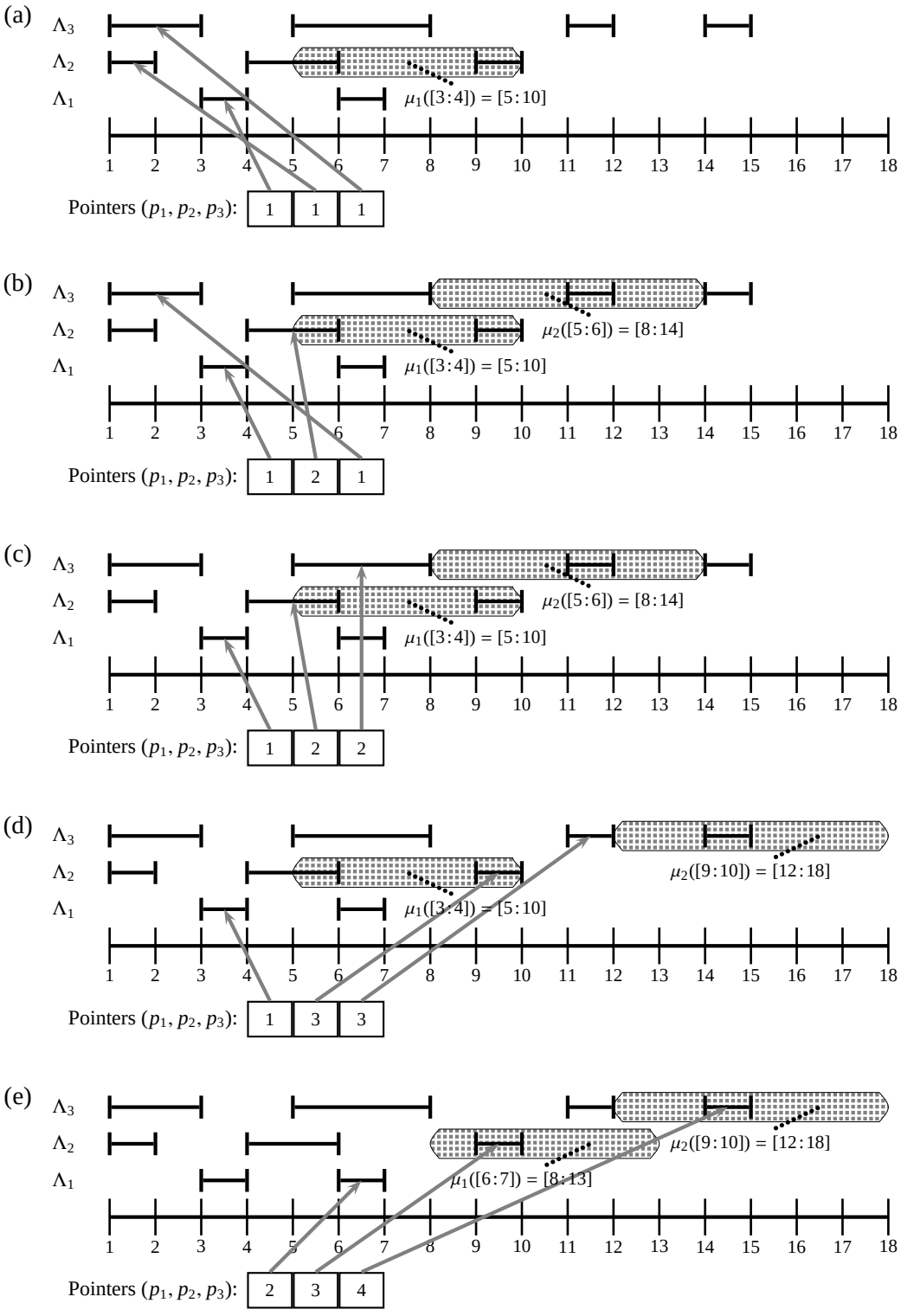


Figure 3.8. Keyframe lists Λ_1, Λ_2, and Λ_3 of our example keyframe query. **(a)**: The pointers (p_1, p_2, p_3) are initialized to point to the first segments of the respective list. The admissible range $\mu_1([3:4])$ is indicated by the gray dotted area. **(b)**–**(e)** show intermediate states of the algorithm, see the text for explanations.

M	Hits	$R(M)$
$(1,2,2)$	$(3,5,8)$	$[3:8]$
$(1,2,3)$	$(3,5,11),\ (3,6,11),\ (3,6,12),\ (4,6,11),\ (4,6,12)$	$[3:12]$
$(1,2,4)$	$(3,6,14),\ (4,6,14)$	$[3:14]$
$(1,3,4)$	$(3,9,14),\ (3,9,15),\ (4,9,14),\ (4,9,15),\ (4,10,14),\ (4,10,15)$	$[3:15]$
$(2,3,4)$	$(6,9,14),\ (6,9,15),\ (6,10,14),\ (6,10,15)$ $(7,9,14),\ (7,9,15),\ (7,10,14),\ (7,10,15)$	$[6:15]$

Table 3.1. Matches found by the proposed algorithm for our running example, hits that are contained in these matches, and their hit relevant ranges $R(M)$.

is necessary to find all hit relevant ranges which could possibly also include frame 15. Although the matches already comprise the last segment of the third inverted keyframe list, the hit relevant ranges of the matches found so far do not include frame 15. Decrementing p_3 and continuing the algorithm ensure finding the correct hit relevant ranges. The recursion returns to the point where RecursiveSearch(3, (8,14)) was called (Line 22). The pointer p_2 is incremented to $p_2 = 3$ (Line 24) and the intersection $= [9:10] \cap [5:10] = [9:10]$ is calculated (Line 30). The **while**-loop is repeated and in Line 21 the admissible range is set to $\mu_2([9:10]) = [12:18]$, see Figure 3.8 (d). The subsequent recursive call (Line 22) leads to the match $(1,3,4)$. Finally, the pointer p_1 is increased leading to another match $(2,3,4)$, see Figure 3.8 (e).

Table 3.1 shows all matches M found by our algorithm along with all hits contained in the respective match and the resulting hit relevant ranges $R(M)$. Actually, there are two matches, $(1,3,3)$ and $(2,3,3)$, which are not found by the algorithm. These matches, however, are irrelevant since $R((1,3,3)) = [3:12]$ and $R((2,3,3)) = [6:12]$ are contained in unions of hit relevant ranges of the other matches. Also recall that the actual output of the algorithm consists of the union of all $R(M)$, thus avoiding an explosion of the output size. In our example, this results in a single segment $[3:15]$.

The recurrence in our algorithm is a property that follows from the following consideration. When searching inside an admissible range, one has to set some frames as candidates for a hit and search in the next inverted keyframe list for suitable frames fitting to the next keyframe. After finishing the search in the next keyframe list, one still has to know the admissible range that held before the search was started. Therefore, a recursion is an appropriate way to express such a condition.

3.6 Experiments

Our keyframe-based search algorithm works for general time-dependent multimedia data and is designed for efficiently handling temporal deformations between the query and the database keyframes. We will demonstrate the practicability of our concept by means of the motion retrieval scenario, where one typically encounters such deformations between semantically related motion sequences. In Section 3.6.2, we describe some experiments showing that our algorithm is often able to cut down the search space from several hours to a couple of minutes of motion capture data within few milliseconds (ms). The so reduced dataset can then be ranked and analyzed by

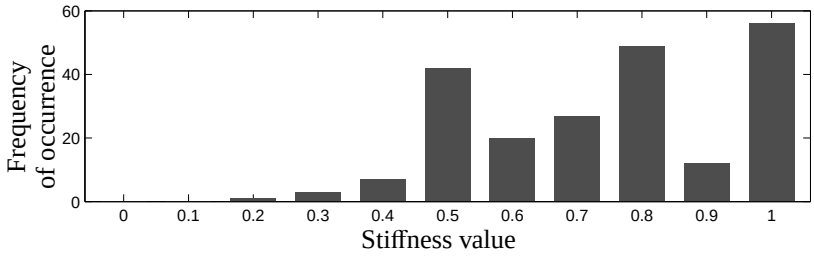


Figure 3.9. Histogram of stiffness values in the manually defined queries.

more refined alignment techniques. We will demonstrate such a two-stage retrieval strategy by using motion templates [Müller and Röder, 2006] for postprocessing the keyframe-based matches, see Section 3.6.3. Here, we also report on some experiments to demonstrate the effect of the stiffness parameter on the final retrieval result. Information on the experimental dataset and the used keyframes can be found in the following Section 3.6.1.

3.6.1 Experimental Data Set and Keyframes

For our experiments, we used the freely available HDM05 database [Müller *et al.*, 2007], which consists of 210 minutes of motion data contained in 324 files. Making up an average length of 39 seconds, each file consists of a sequence of different actions. A detailed description of the motion files can be found in [Müller *et al.*, 2007]. From the HDM05 database we cut out 1327 short motion clips which were organized into 57 motion classes, each containing 10 to 50 realizations executed by various actors. These motion classes were used to generate keyframes in a semi-automatic process. Here, using one half of the motion clips of each class as training data, we computed quantized motion templates based on the 39 relational features, see Section 2.2. These features were divided up into three feature sets: One upper body, one lower body, and one mixed set. We then selected 3 to 9 representative columns along with their distances of each class motion template as keyframes. For details of a similar procedure we refer to [Müller and Röder, 2006]. To avoid false negatives (at the expense of having more false positives), we manually post-processed the keyframes by adding or removing suitable feature vectors from the keyframes. For each query, we also manually define the vector of stiffness values, see Figure 3.9 for a histogram of these values. Among our 57 queries, stiffness values between 0.2 and 1.0 were used. Only 31% of the values occur in the more stiff range of $\sigma = 0.9$ or $\sigma = 1.0$ which shows that overall, a fair amount of temporal deformations is permitted in the keyframe queries. As further queries, we concatenated the motion class keyframes to generate longer and more complex queries describing sequences of different actions, see Section 3.6.3.

At this point, we note that the focus of this chapter is not on the fully automated learning of keyframes, but on the efficient and deformation-tolerant retrieval based on a given set of suitable keyframes. In order to fully automate the creation of keyframes, in Chapter 4 we employ a genetic algorithm to learn keyframes from positive as well as negative training motions. Once suitable keyframes are generated for a specific motion class, they can be used as queries to arbitrary databases.

To prove the applicability of our algorithm, we conducted several experiments. The presented algorithm has been implemented in a mixture of *C++* and MATLAB. All experiments were executed

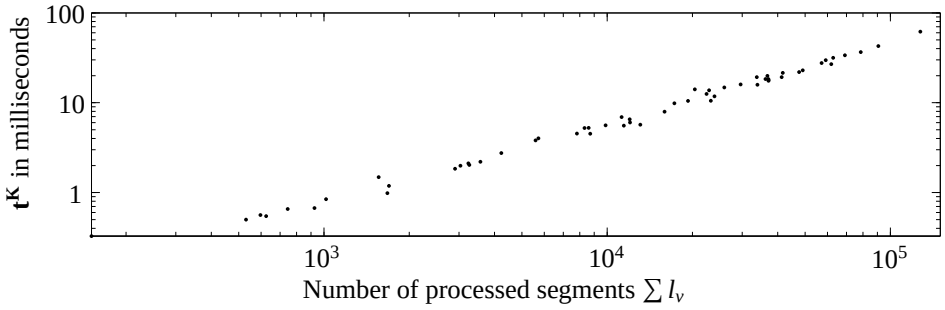


Figure 3.10. Run time of the keyframe-based search algorithm vs. the number of segments in the processed inverted lists, plotted on a log-log scale.

on an AMD Athlon 64 X2 5000+ (using only 1 core) with 3.5GB of RAM.

3.6.2 Data Reduction

As shown in Table 3.2, we used the generated keyframes as an input for our algorithm to reduce the search space for 57 query motion classes. For each query, K denotes the number of keyframes, and Avg. $\boldsymbol{\sigma}$ is the average of the stiffness vector used for this query. We give the number of segments in the inverted lists that have to be processed in the query with the value in the column corresponding to $\sum l_v$. To demonstrate the efficiency of our algorithm, $\mathbf{t^K}$ shows the keyframe search time in milliseconds, and %(**D**) shows the size of the reduced search space in percent with regard to the database size.

The time taken to reduce the search space using our algorithm amounts to only 12.5 ms on average. Effectively, the search space is reduced to less than 5% of the entire database on average. As the table shows, the search time depends on the size of the processed inverted lists. Small searching times, like 0.3 ms for query ID1, are due to queries that contain keyframes describing infrequent poses in the database. For query ID1, characteristic poses for a cartwheel, which occur only in few other motions in the database, were used as keyframes. In contrast to this, using the 39 full body motion features from [Müller and Röder, 2006], the class ID49 (turnRight) can not be distinguished from the standing pose. Here, using a total of 9 keyframes, more than 60000 segments have to be processed. Because of the unspecific keyframes, all standing poses in the database are contained in the reduced search space. It is important to notice that although the number of keyframes in this case is rather high, the search time does not explode. Actually, the run time grows linearly with the size of the processed keyframe lists as demonstrated by Figure 3.10. Here, we plot the run times $\mathbf{t^K}$ against the length of the processed lists $\sum l_v$ for all queries of Table 3.2.

As for the percental size of the reduced search space, for most of the classes, sizes of less than 3% can be achieved. For many classes, like ID6, even better reduction rates are reached, returning less than 1% of the HDM05 database. Unlike these results, some queries do not reduce the search space well. As already mentioned, the keyframes for query ID49 are rather unspecific. As a result, for this query more than 20% of the database is returned. Similarly, for queries ID50 to ID57, some of the reduction rates are not so good due to the large number of walking motions in HDM05 and due to some confusion between various walking styles.

As a further application, the keyframes can be combined to describe a query for an entire sequence

ID	Query class	K	Avg. σ	$\sum l_v$	$\mathbf{t^K}$	%(**D**)	t^R
1	cartwheelLHandStart1Reps	3	0.7	75	**0.3**	**1.2**	234
2	clap1Reps	5	0.9	39326	**36.6**	**14.0**	5094
3	clapAboveHead1Reps	5	0.8	11959	**11.8**	**1.7**	328
4	depositFloorR	5	0.5	18565	**17.5**	**6.0**	1063
5	depositHighR	7	0.6	20860	**21.5**	**2.2**	375
6	elbowToKnee1RepsLelbowStart	3	0.7	265	**0.5**	**0.6**	78
7	elbowToKnee1RepsRelbowStart	4	0.6	1631	**2.0**	**0.6**	63
8	grabFloorR	7	0.8	20664	**19.3**	**2.5**	344
9	grabHighR	6	0.7	18560	**17.9**	**2.9**	656
10	hopBothLegs1hops	5	0.7	23808	**22.0**	**1.0**	125
11	hopLLeg1hops	3	0.8	4358	**4.5**	**0.5**	31
12	hopRLeg1hops	4	0.8	16965	**15.8**	**1.1**	109
13	jogLeftCircle4StepsRstart	4	0.8	2115	**2.8**	**1.4**	203
14	jogOnPlaceStartFloor2StepsRStart	5	0.8	28583	**27.6**	**30.4**	8766
15	jogRightCircle4StepsRstart	3	0.6	1451	**1.8**	**1.3**	188
16	jumpDown	4	0.8	5726	**5.6**	**1.2**	188
17	jumpingJack1Reps	3	0.7	780	**1.5**	**0.9**	109
18	kickLFront1Reps	5	0.8	30918	**26.9**	**2.0**	250
19	kickLSide1Reps	4	0.9	1516	**2.0**	**1.1**	141
20	kickRFront1Reps	5	0.8	4302	**5.3**	**1.1**	188
21	kickRSide1Reps	4	0.9	11610	**10.5**	**1.2**	172
22	lieDownFloor	3	0.5	1784	**2.2**	**2.7**	813
23	punchLFront1Reps	5	0.7	11214	**12.5**	**1.4**	234
24	punchLSide1Reps	6	0.7	29549	**29.8**	**2.6**	375
25	punchRFront1Reps	6	0.7	16880	**19.3**	**2.5**	406
26	punchRSide1Reps	6	0.7	45343	**42.8**	**1.7**	266
27	rotateArmsBothBackward1Reps	3	0.6	837	**1.0**	**1.5**	172
28	rotateArmsBothForward1Reps	6	0.6	6009	**6.5**	**0.7**	78
29	rotateArmsLBackward1Reps	4	0.7	6021	**6.0**	**0.6**	78
30	rotateArmsLForward1Reps	3	0.7	508	**0.8**	**0.7**	94
31	rotateArmsRBackward1Reps	3	0.5	1618	**2.1**	**0.6**	63
32	rotateArmsRForward1Reps	3	0.6	298	**0.6**	**0.7**	78
33	runOnPlaceStartFloor2StepsRStart	5	0.9	24522	**22.9**	**0.4**	31
34	shuffle2StepsRStart	8	0.9	31395	**31.6**	**2.0**	359
35	sitDownChair	6	0.9	2793	**3.8**	**2.4**	422
36	sitDownFloor	5	0.7	3911	**4.5**	**4.7**	1109
37	sitDownKneelTieShoes	3	1.0	372	**0.7**	**2.2**	609
38	sitDownTable	6	0.8	18393	**19.9**	**29.3**	16766
39	skier1RepsLstart	5	0.6	4157	**5.2**	**0.8**	78
40	sneak2StepsLStart	4	0.7	4935	**5.6**	**1.5**	250
41	sneak2StepsRStart	6	0.8	18126	**18.3**	**2.0**	328
42	squat1Reps	4	0.6	848	**1.2**	**1.0**	141
43	staircaseDown3Rstart	5	0.8	9644	**10.5**	**6.5**	1469
44	staircaseUp3Rstart	5	0.5	2860	**4.0**	**1.7**	281
45	standUpLieFloor	3	0.6	463	**0.7**	**1.7**	375
46	standUpSitFloor	6	0.6	7969	**7.9**	**3.6**	672
47	throwBasketball	5	0.8	5624	**6.9**	**3.0**	563
48	turnLeft	7	0.9	34548	**33.8**	**10.6**	4063
49	turnRight	9	0.9	63819	**61.8**	**20.1**	9844
50	walk2StepsLstart	6	0.9	12956	**14.8**	**10.5**	4609
51	walk2StepsRstart	5	0.8	18547	**18.4**	**13.2**	5875
52	walkBackwards2StepsRstart	5	0.9	8641	**9.8**	**1.1**	141
53	walkLeft2Steps	3	0.9	312	**0.5**	**0.9**	141
54	walkLeftCircle4StepsRstart	6	0.7	11447	**13.8**	**13.5**	3953
55	walkOnPlace2StepsLStart	5	0.9	14792	**16.0**	**35.9**	17422
56	walkRightCircle4StepsRstart	7	0.6	10196	**14.1**	**11.5**	2719
57	walkRightCrossFront2Steps	3	0.8	6545	**5.7**	**4.0**	875
∅		4.8	0.75	12314	**12.5**	**4.8**	1657

Table 3.2. Retrieval results on the HDM05 database (3.5 hours of mocap data). K: Number of keyframes used in the query. Avg. $\boldsymbol{\sigma}$: average of the stiffness vector. $\sum l_v$: Number of segments in the processed inverted lists. $\mathbf{t^K}$: Keyframe search time in ms. %(**D**): Size of the reduced search space in percent (w.r.t. the size of HDM05). t^R: Time for motion template retrieval on the reduced search space in ms.

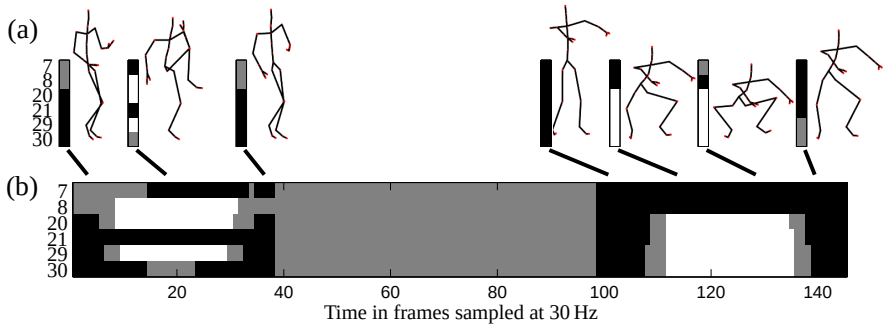


Figure 3.11. **(a)**: The two keyframe queries 'elbowToKnee' and 'squat' are combined to yield a new keyframe query. **(b)**: Corresponding combined MT.

ID	query class	K	Avg. σ	$\sum l_v$	$\mathbf{t^K}$	%(**D**)	t^R
58	ID 6 + ID 42	7	0.7	1114	**1.8**	**1.2**	281
59	ID 13 + ID 15	7	0.7	3566	**5.1**	**1.3**	438
60	ID 19 + ID 23, ID 26	16	0.7	63739	**72.7**	**1.8**	797
61	ID 22 + ID 22	6	0.7	3568	**4.2**	**2.1**	1656
62	ID 27 + ID 28	10	0.7	12030	**14.4**	**1.0**	219
63	ID 39 + ID 6	8	0.7	4423	**5.8**	**0.8**	234
64	ID 44 + ID 43	10	0.7	12505	**16.7**	**1.0**	281
65	ID 52 + ID 57	8	0.7	15186	**17.5**	**1.7**	594
66	ID 55 + ID 33	10	0.7	39314	**42.5**	**1.0**	391
∅		9.1	0.7	17272	**20.1**	**1.3**	543

Table 3.3. Retrieval results on the HDM05 database (3.5 hours of mocap data) using combined queries. The combined queries are used to search for a sequence of actions, see also Table 3.2.

of various actions. As an example, the concatenation of the queries elbow-to-knee and squat is shown in Figure 3.11 (a). By setting the distance d_3 and the corresponding stiffness parameter σ_3 between the last keyframe of the first motion class and the first keyframe of the second motion class, one can control the time that may elapse between the two actions. A similar approach to scene description has been sketched in [Müller *et al.*, 2005]. To demonstrate the applicability of this scenario, we have created nine combined queries. The retrieval results are documented in Table 3.3. Although more keyframes are used, the keyframe search time does not explode. For example, in query 60 comprising 16 keyframes, a total of 63 739 segments have been processed in 72.7 ms. In comparison to query ID49, which exhibits a similar number of processed segments, but a smaller number of keyframes, only a small increase in the search time can be observed. Additionally, for all combined queries the size of the reduced search space is very small. Generally, the more keyframes are used in a query, the less data fits to these keyframes.

3.6.3 Retrieval Quality

To show the effectiveness of our algorithm in a two-stage retrieval system, we apply motion template retrieval [Müller and Röder, 2006] as a ranking of the reduced search space. For queries ID1 to ID57, class motion templates have been used. For queries ID58 to ID66, the class motion templates have been combined as indicated in Figure 3.11 (b). In this example, the elbow-to-knee motion template and the squat motion template have been concatenated with a block of 0.5-values, assigning zero cost for this clipping during the ranking process. Then, the output of the MT-based

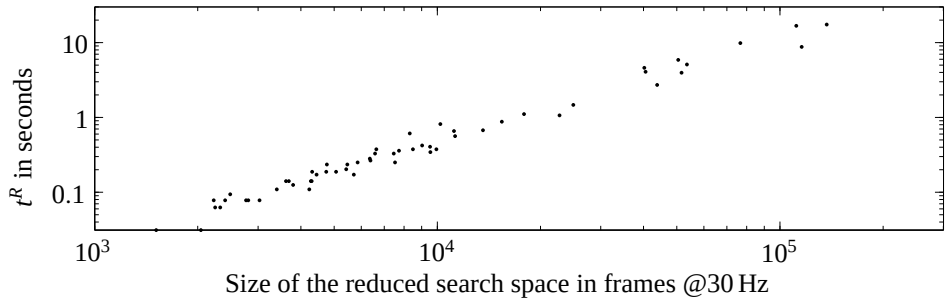


Figure 3.12. Run time of the MT-based ranking on the reduced search space vs. the size of the reduced search space on a log-log scale.

retrieval step is a set of ranked motion clips contained in the reduced search space.

In Tables 3.2 and 3.3, the columns t^R depict the time in ms required for the ranking step. Using a variant of DTW, the run time depends linearly on the product of the size of the reduced search space and the size of the motion template. Since in practice the size of the motion templates is similar among different queries, the run time can be assumed to grow linearly with the size of the reduced search space. This is also documented in Figure 3.12, where we plot the run time of the ranking step over the size of the reduced search space. Note that the ranking step takes less than a second for most of the queries. The speed-up with regard to motion template retrieval on the whole database is equal to the reduction rate of the keyframe based search.

To demonstrate the quality of the results, precision-recall diagrams for some queries are shown in Figure 3.13. For these queries, 15 to 75 relevant documents are contained in the HDM05 database. The recall is very high among all queries, which means that the reduced search space, obtained by our algorithm with stiffness values around 0.6, still contains most of the relevant documents. Unlike the other queries, on query ID36 (“sit down on floor”) false positives occur early in the hit list. Most of the false positives are motions of the class “lie down on floor”, which starts in most cases in the HDM05 database with a “sit down on floor”-phase.

To quantify the influence of the stiffness parameter, Figure 3.14 shows the results for the queries of the first two rows of Figure 3.13, where the stiffness parameter has been set to 1.0, thus creating rigid keyframe queries where any time deformation between the keyframes was prohibited. In comparison to the corresponding diagrams in the first two rows of Figure 3.13, many relevant documents were missed due to the denial of temporal deformations, since overall less parts of the database can be explained by the rigid queries. However, often, the remaining hits still contain true positives which might be motions that were executed at the same speed as the query. For example, nearly all jumping jack motions (query ID17) are missed when setting the stiffness to 1.0, but the remaining hits are all true positives. This suggests that most of these jumping jack motions have been performed with varying speed in the HDM05 motion database. By contrast, from query ID12 (jump on the right foot), only one relevant hit has been masked out in comparison to the original keyframe query. Here, the actors seem to have performed the motion in a rather constant speed. As a third example, consider query ID28. Again, the majority of the hits is missed when denying any temporal deformation.

A further experiment shows the effect of varying the stiffness parameter. Figure 3.15 (a) shows precision-recall diagrams for query ID59 with modified stiffness values. Some false positives

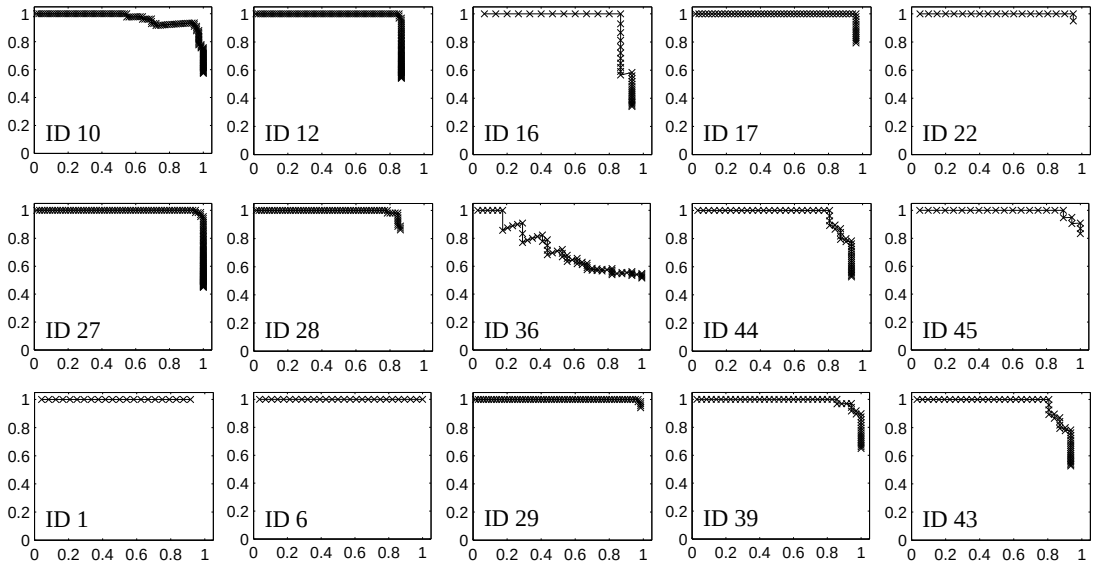


Figure 3.13. Precision-recall diagrams of hits obtained by queries with the indicated IDs (see Table 3.2).

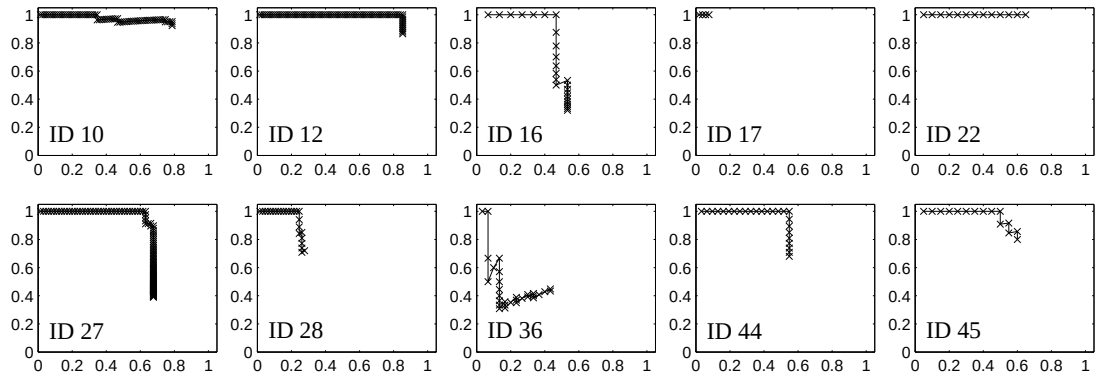


Figure 3.14. Precision-recall diagrams of some queries of Figure 3.13 wereas we modified the stiffness values to $\sigma \equiv 1.0$.

occur when setting $\sigma \equiv 0$, which are eliminated when using higher stiffness values. For $\sigma \equiv 0.6$, the precision-recall diagram is the same as the one with manually determined stiffness values. A further raise of the stiffness results in a loss of some relevant documents. A similar behavior is demonstrated for query ID60 in Figure 3.15 (b) and query ID61 in Figure 3.15 (c). Again, no difference in the $\sigma \equiv 0.6$-diagram can be noticed in comparison to the manually optimized query stiffness values. However, the sizes of the reduced search spaces and so the times for the ranking steps are smaller for queries ID59, ID60 and ID61 than for the $\sigma \equiv 0.6$-modified queries.

3.7 Conclusions

In this chapter, we introduced a novel algorithm for keyframe-based multimedia retrieval which can be used to drastically cut down the search space. In contrast to previous approaches, our index-based algorithm can cope with significant temporal deformations without resorting to com-

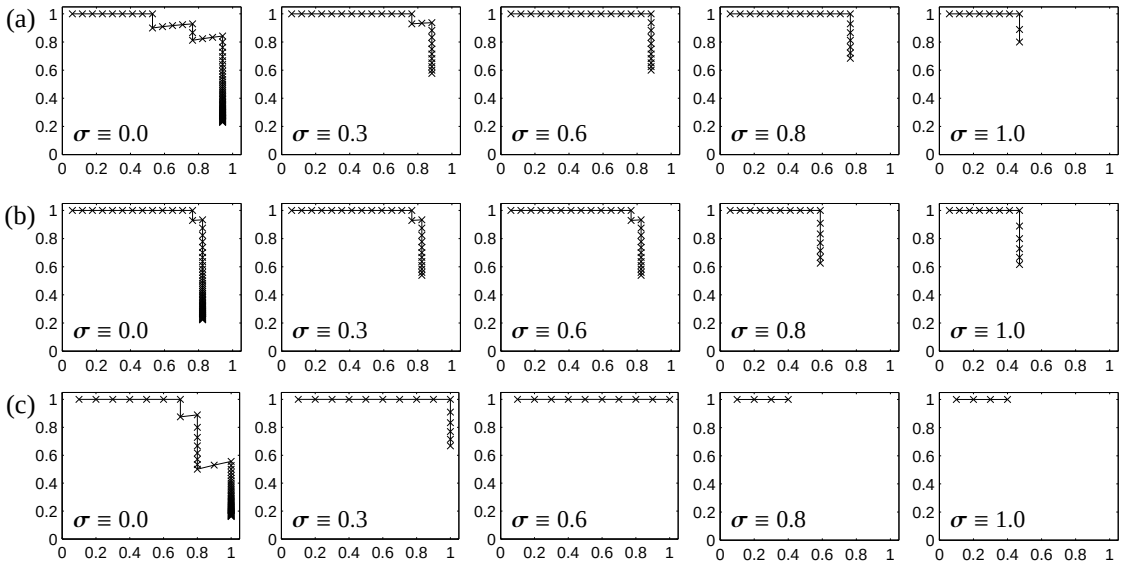


Figure 3.15. Precision-recall diagrams for **(a)** queries ID59, **(b)** ID60, and **(c)** ID61, whereas the manually determined stiffness values have been modified to the specified values.

putationally expensive techniques such as dynamic time warping. To prove its practicability, we applied our algorithm within a two-stage motion retrieval scenario, where the mocap database is pre-filtered in the first stage using the described search algorithm. As it turned out, the temporal flexibility introduced by our stiffness concept is necessary to avoid a large number of false negatives in the pre-filtering step. In our experiments, we showed that the run time of our algorithm scales well with the number of frames in the database. In particular, the keyframe-based search takes only a couple of milliseconds for a database comprising 3.5 hours of mocap data. In this chapter, the keyframes were generated in a semi-automatic process. In the next chapter, we will develop an algorithm for automatically learning keyframes from a given set of example motions, where the keyframe-based search will be executed repeatedly as a sub-component. The high efficiency of our keyframe-based search is one major reason why the run time of the learning algorithm is kept within acceptable bounds.

Chapter 4

A Genetic Algorithm for Learning Keyframes

As shown in the previous chapter, carefully selected and designed keyframe queries can lead to a successful and efficient retrieval. When designing keyframes, the following considerations should be taken into account. On the one hand, since the keyframes are used as hard constraints in a query, they should generalize well to avoid a large number of false negatives in the retrieval step. On the other hand, the keyframes must have a high discriminatory power to yield the desired pruning and data reduction capability. As the main contribution of this chapter, we describe how characteristic keyframe queries can be learned automatically from positive and negative example motions using a randomized genetic algorithm. Generally, such algorithms are population-based optimization techniques to find approximate solutions to optimization problems [Pohlheim, 1999]. This chapter is based on the publication [Müller *et al.*, 2009] and extends ideas from [Müller *et al.*, 2008]. One drawback of the approach described in Müller *et al.* [2008] is the modeling of temporal flexibility in the query, which is dependent on the chosen feature function. In our formulation based on keyframes and admissible temporal deformations between them, a more general model that is invariant to the chosen feature function is obtained. Further differences arise from the choice of our more general model and include changes in the initialization, the recombination and mutation steps of the genetic algorithm. Note that our work is fundamentally different to keyframe selection methods like [Assa *et al.*, 2005], where keyframes as used for visual summaries are computed. In contrast to our approach, such methods do not consider in how far such keyframes might be useful for discriminating motions in a retrieval scenario.

Based on the general paradigm of evolutionary algorithms [Pohlheim, 1999], we describe our keyframe learning algorithm in the subsequent sections. We first show how we model a population and its individuals (Section 4.1). Then, we describe how we model the quality or fitness of an individual (Section 4.2). After that, we show how we generate an initial population (Section 4.3) and discuss the main algorithm with its genetic operations (Section 4.4). Finally, we describe our experiments (Section 4.5). For further experiments in the retrieval context, we also refer to Chapter 5 where keyframes learned with the algorithm described in this chapter are used to assist an MT-based annotation procedure.

4.1 Modeling Individuals

A *population* consists of a set of *individuals* that represent candidate solutions for the optimization problem. In our scenario, an individual is represented by a keyframe query

$$\text{Ind} = (\mathbf{V}, \mathbf{d}') \,. \tag{4.1}$$

In the following, we describe how we compute such candidate solutions from a set of example motions. Recall the case of motion templates, where a set of positive example motions $\mathcal{T}^+$ was used to compute a representation of a motion class C. In addition to $\mathcal{T}^+$, we assume a set of negative example motions $\mathcal{T}^-$ that discriminate the class C to other motion classes. Then, the goal is to generate a keyframe query $(\mathbf{V}, \mathbf{d})$ yielding characteristic constraints shared by all motions belonging to C but not by motions from other classes. In other words, a keyframe search with $(\mathbf{V}, \mathbf{d})$ conducted on the set

$$\mathcal{T} := \mathcal{T}^+ \cup \mathcal{T}^- \tag{4.2}$$

should return exactly the motions contained in $\mathcal{T}^+$.

For the sake of clarity in the following notation, we slightly modify the representation of keyframes in comparison to the representation as introduced in Chapter 3. Instead of explicitly reverting to keyframe distances $\mathbf{d}$ and stiffness parameters $\boldsymbol{\sigma}$, we implicitly represent these values by the minimal and maximal admissible distances between keyframes

$$\mathbf{d}' = (d_1^{\min}, d_1^{\max}), \ldots, (d_{K-1}^{\min}, d_{K-1}^{\max})) \,. \tag{4.3}$$

with

$$\forall k \in [1 : K-1] : (d_k^{\min}, d_k^{\max}) \in \mathbb{N} \times \mathbb{N}, \quad d_k^{\min} \leq d_k^{\max} \,. \tag{4.4}$$

Note that as carried out also in the function μ, see Equation (3.11), one can easily map from the representation $(\mathbf{V}, \mathbf{d})$ to $(\mathbf{V}, \mathbf{d}')$ by

$$d_k^{\min} = \lceil \sigma_k \cdot d_k \rceil \tag{4.5}$$

$$d_k^{\max} = \begin{cases} \left\lfloor \frac{1}{\sigma_k} \cdot d_k \right\rfloor & \text{if } \sigma_k > 0, \\ \infty & \text{if } \sigma_k = 0, \end{cases} \tag{4.6}$$

for $k \in [1 : K-1]$. The inverse map from $(\mathbf{V}, \mathbf{d}')$ to $(\mathbf{V}, \mathbf{d})$ directly follows by

$$\hat{\sigma}_k = \begin{cases} \sqrt{d_k^{\min} / d_k^{\max}} & \text{if } d_k^{\max} \neq \infty, \\ 0 & \text{if } d_k^{\max} = \infty, \end{cases} \tag{4.7}$$

$$\hat{d}_k = \begin{cases} \sqrt{d_k^{\min} \cdot d_k^{\max}} & \text{if } d_k^{\max} \neq \infty, \\ \infty & \text{if } d_k^{\max} = \infty, \end{cases} \tag{4.8}$$

for $k \in [1 : K-1]$. Note that because of the rounding operations in Equations (4.5) and (4.6), $(\hat{d}_k, \hat{\sigma}_k)$ might not be equal to (d_k, σ_k). However, they are compatible in the sense that the same values $d_k^{\min}$ and $d_k^{\max}$ are computed from $(\hat{d}_k, \hat{\sigma}_k)$ and (d_k, σ_k) by means of Equations (4.5) and (4.6), respectively.

To simplify notation, we do not explicitly distinguish between both representations when it is clear from the context which representation is used.

4.2 Fitness Function

We measure the quality or the *fitness* of an individual in terms of precision and recall by evaluating the individual on the example motions. More precisely, let $\mathcal{H}(\text{Ind}) \subseteq \mathcal{T}$ denote the mocap documents retrieved by the keyframe query Ind. Then, we define precision $P(\text{Ind})$ through

$$P(\text{Ind}) := \frac{|\mathcal{H}(\text{Ind}) \cap \mathcal{T}^+|}{|\mathcal{H}(\text{Ind})|} \tag{4.9}$$

and recall $R(\text{Ind})$ through

$$R(\text{Ind}) := \frac{|\mathcal{H}(\text{Ind}) \cap \mathcal{T}^+|}{|\mathcal{T}^+|}. \tag{4.10}$$

Intuitively, the precision function measures the accuracy of a query, evaluating the fraction of relevant among the retrieved documents. On the contrary, the recall function constitutes a measure of completeness, indicating how many of the relevant documents are actually retrieved. Having both $P(\text{Ind}) = 1$ and $R(\text{Ind}) = 1$ describes the ideal query for which only the relevant documents are retrieved. In general, however, such queries are hard to design.

We now define a fitness value $\text{Fit}_\beta(\text{Ind})$ to an individual Ind with respect to a weighting parameter β. To this end, we make use of the weighted F-measure of Equation (4.9) and Equation (4.10) by

$$\text{Fit}_\beta(\text{Ind}) := \frac{(1+\beta^2) \cdot (P(\text{Ind}) \cdot R(\text{Ind}))}{\beta^2 \cdot P(\text{Ind}) + R(\text{Ind})}. \tag{4.11}$$

Typically, keyframes will be used to cut down the search space using, *e. g.*, our keyframe-based algorithm described in Chapter 3. The resulting search space will then be analyzed in a further refinement step, see Chapter 5 for an example in the context of automated annotation of mocap documents. In such a scenario, we want to avoid that hits are already excluded by the keyframe-based preprocessing, *i. e.*, we want to avoid false negative hits. We permit false negative hits even if it possibly comes at the expense of having less precision, meaning that we want to tolerate more false positive results. The main reason is that we assume that the resulting reduced search space will be processed further by a more refined analysis, where false positive results can still be eliminated. Therefore, we stress the recall value in our fitness function by setting $\beta = 2$.

4.3 Initialization

For the start of the optimization, we generate an initial population Π^1 consisting of M individuals. To this end, we first compute a quantized motion template $X \in \{0, 1, *\}^{f \times N}$ from $\mathcal{T}^+$ for a motion class C, see Section 2.3. Intuitively speaking, we then pick a small number of columns from X as a keyframe query. Here, we refer to the n^{th} column of X by $X(:, n)$, $n \in [1 : N]$. Following our strategy of trying to avoid false negative results, we use a strict quantization threshold $\delta = 0$ to compute the quantized motion template. This reveals also the slightest inconsistencies among the relational features of the training motions. As a consequence, each column corresponds to a keyframe which is guaranteed to have at least one match in each of the training motions. In other words, choosing one column from the motion template quantized with $\delta = 0$ will guarantee perfect recall on the training motions $\mathcal{T}$ given suitable keyframe distances $\mathbf{d}$.

Algorithm 2 Initialization of an Individual

Input: $X \in \{0, 1, *\}^{f \times N}$: quantized motion template
$\mathcal{N}(a, b)$: normally distributed random number generator with mean a and standard deviation b.
$\mathcal{U}(a, b)$: uniformly distributed random number generator that generates numbers in the interval $[a, b]$
Output: $(\mathbf{V}, \mathbf{d})$: Individual modeled by a keyframe query

1: $K \leftarrow \max(2, \mathcal{N}(3, 2))$
2: $\boldsymbol{P} \leftarrow K$ randomly selected frames from the range $[1 : N]$
3: sort($\boldsymbol{P}$)
4: **for** $k \leftarrow 1$ **to** K **do** ▷ Initialize keyframes
5: $V_k \leftarrow X(:, P(k))$
6: **end for**
7: **for** $k \leftarrow 1$ **to** $K - 1$ **do** ▷ Initialize admissible temporal variations
8: $\delta \leftarrow \boldsymbol{P}(k + 1) - \boldsymbol{P}(k)$
9: $d_{\min} \leftarrow \max(0, \delta/2 - \text{round}(\delta/2 \cdot \mathcal{U}(0, 2)))$
10: $d_{\max} \leftarrow \delta/2 + \text{round}(\delta/2 \cdot \mathcal{U}(0, 2))$
11: **end for**

Remember that a motion template X reveals the consistent aspects of the example motions and expresses characteristic properties of the class C. However, using the columns of X directly as keyframes does not account for the negative training examples in $\mathcal{T}^-$. As a consequence, the precision values of such keyframe queries might be comparatively low. Our idea is to use the motion template only for the initialization followed by a successive refinement of the keyframes, as summarized by Algorithm 2. To this end, for each of the M initial individuals we first choose a natural number K based on a normally distributed random number generator with a mean of 3 and a standard deviation of 2 (Line 1). The number K corresponds to the initial number of keyframes in the individual. We fixed the parameters of the random number generator by virtue of our experience with manually designed keyframe queries as evaluated in Chapter 3. Here, we found that using 4.8 keyframes on average yielded good retrieval results, see also Table 3.2. Note that our general strategy is to avoid false negative results. Thus, we set a slightly lower mean of 3 (instead of setting 4.8 as the mean), since using less keyframes leads to higher recall values in general.

After choosing K, we define the keyframe query $(\mathbf{V}, \mathbf{d})$. First, we randomly pick K columns of X to define the keyframes $V_1, \ldots, V_K$ (Lines 2 to 6). Then, the distance parameters are initialized based on the distances of the chosen keyframes admitting some randomly chosen tolerance, see Lines 7 to 11.

4.4 Genetic Operations

After the initialization, the three genetic operations referred to as *selection*, *recombination*, and *mutation* are used to iteratively breed a new population from a given population. Let Π^g,

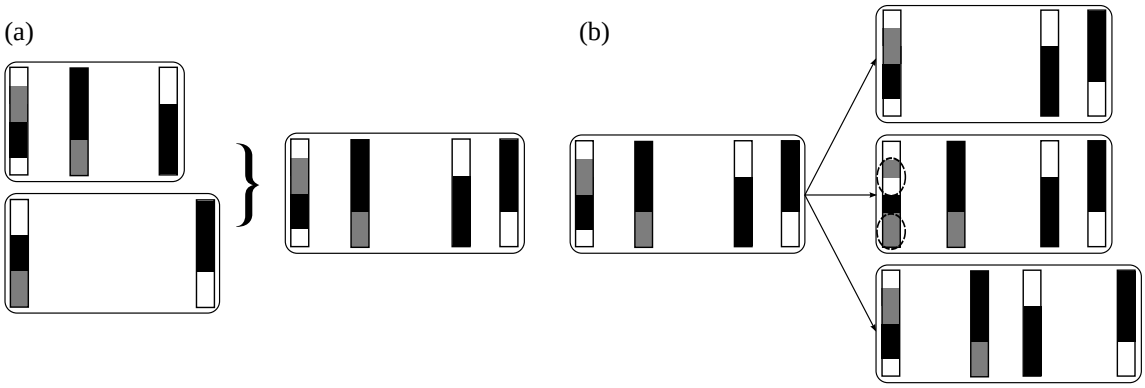


Figure 4.1. (a): Two parents are combined to generate an offspring. **(b)**: The offspring is mutated by either adding/removing a keyframe **(top)**, generalizing/specializing a keyframe **(middle)**, or by changing the admissible keyframe distances **(bottom)**.

$g \in \mathbb{N}$, denote the current population. Then, using the concept of universal stochastic sampling [Baker, 1987], we select r individuals from Π^g, which are referred to as *parents*. In the recombination step, the keyframes of any two of these parents are combined to derive new individuals, referred to as *offsprings*. To this end, we randomly choose a number of keyframes of each of the two parents and merge these keyframes to form a single keyframe sequence, see Figure 4.1 (a). The new distance parameters **d** are determined similarly to the initialization step. To avoid an early convergence of the optimization procedure towards a poor local optimum, additional modifications to the offsprings are applied by suitable random operations referred to as mutations, see Figure 4.1 (b). In our case, an offspring is mutated by randomly choosing and applying one of the following operations:

- Add or remove a randomly chosen keyframe.
- Specialize (*i. e.*, change $*$ to 0 or 1) or generalize (*i. e.*, change a value 0 or 1 to $*$) a randomly chosen keyframe.
- Randomly increase and decrease the values in **d**.

After the recombination and mutation steps, we obtain $\frac{r(r-1)}{2}$ offsprings. We arrange the M individuals of Π^g and the $\frac{r(r-1)}{2}$ offsprings in a sorted list with decreasing fitness. Finally, the new population Π^{g+1} is obtained by picking the M fittest individuals from this list. This entire procedure is iterated for $g = 1, \ldots, G$, where G denotes a fixed number of generations. The fittest individual of Π^G is the solution of the optimization procedure, see Algorithm 3 for an overview of the keyframe learning algorithm.

In our implementation the population size is set to $M = 50$, the number of parents to $r = 5$, and the number of generations to $G = 100$. The exact values of these parameters, which have been determined experimentally, are not of crucial importance for the final result. However, as typical for evolutionary algorithms, different runs of the overall procedure may result in significant differences between the keyframes of the various solutions. Therefore, for each motion class, we run the overall genetic algorithm several times (in our experiments 100-500 times) and then generate an individual with keyframes that most frequently occur in the solutions. To this end, we accumulate the resulting keyframes of each run of the genetic algorithm in a real-valued matrix,

Algorithm 3 Genetic Algorithm for Learning a Keyframe Query

Input: $\mathcal{T}^+$: positive training motions
$\mathcal{T}^-$: negative training motions
M: number of individuals in a population.
G: number of generations
r: number of individuals to select as parents
Output: Π^G: population of generation G

1: **for** $m \leftarrow 1$ **to** M **do** ▷ Initialize population Π^1
2: $\Pi^1(m)$ is generated with Algorithm 2 (page 40).
3: **end for**
4: **for** $g \leftarrow 1$ **to** G **do** ▷ loop over all generations of the population
5: **for** $m \leftarrow 1$ **to** size(Π^g) **do** ▷ loop over all individuals of one generation
6: Ind $\leftarrow (\mathbf{V}, \mathbf{d}) = \Pi^g(m)$
7: Obtain hits by querying $\mathcal{T} = \mathcal{T}^+ \bigcup \mathcal{T}^-$ with Ind using Algorithm 1 (page 24)
8: Each document in $\mathcal{T}$ with at least one hit is regarded as a retrieved document.
9: Evaluate Fit_2(Ind), see Equation (4.11).
10: **end for**
11: $\Pi^g \leftarrow$ The M fittest individuals of Π^g. ▷ Keep the population at a constant size
12: Select r individuals in Π^g using stochastic universal sampling.
13: Create $\frac{r(r-1)}{2}$ offsprings by recombining any two parents.
14: For each offspring, mutate it using a randomly chosen mutation operation.
15: $\Pi^{g+1} \leftarrow \Pi^g \bigcup$ {all mutated offsprings}.
16: **end for**

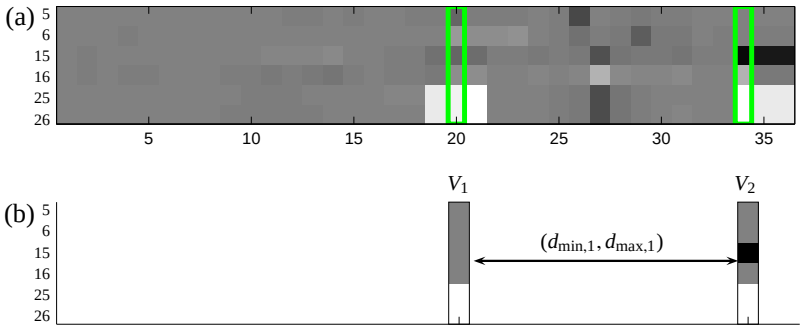


Figure 4.2. (a): After adding up the results of 300 runs of the genetic algorithm, frequently occurring keyframes stand out. **(b)**: The most frequently occurring keyframes are extracted.

whereas we represent the wildcard character with the value 0.5, see Figure 4.2 (a). Here, columns with white and black regions correspond to frequently occurring keyframes. Using this matrix representation, we then extract the most frequently occurring keyframes, see Figure 4.2 (b), and optimize **d** using a similar strategy as with the genetic algorithm.

4.5 Experiments

We implemented the learning algorithm in Matlab while passing time critical parts to subroutines implemented in C/C++. The computations were performed on an AMD Athlon X2 5000+ with 3.5 GB of RAM.

In our experiments, we used the described algorithm to generate keyframe queries for a set of $P = 69$ motion classes C_p, $p \in [1 : P]$. To this end, we took the motion classes as indicated in Table 3.2, and added additional classes. With the additional classes, we included motions like throwSittingHigh and throwSittingLow, as well as throwStandingHigh and throwStandingLow. These motions are very similar to each other despite from the height of the hands, respectively, and therefore pose a challenging test for generating keyframes. Moreover, we incorporate motions that are difficult to distinguish even for a human observer. For example, we included the classes grabFloorR and grabHighR. Notably, the only difference to the already existing classes depositFloorR and depositHighR (classes 4 and 5 in Table 3.2), is the end phase of the motion where the actor grabs or deposits a small object, respectively. Additionally, we included motion classes which describe grabbing and depositing motions of an object in a shelve at medium and low height, respectively.

We assembled a training database of 24 minutes length in total (42586 frames), which consists of nine example motions for each motion class, serving as $\mathcal{T}_p^+$, respectively. These example motions were manually cut out from documents of the HDM05 mocap database [Müller *et al.*, 2007]. In a first step, the relational features, which are needed for learning the motion templates as well as the keyframe queries, are computed and stored for the entire training examples (taking 137 seconds for the 24 minutes of data). From the features, we computed the quantized class motion templates using an iterative warping and averaging algorithm (see Section 2.3), which took roughly 3 seconds on average for each MT. To learn the keyframe queries, we also need negative example motions for each class. Here, we simply define $\mathcal{T}_p^-$ to be the union of all example motions that do not belong to the class C_p:

$$\mathcal{T}_p^- = \bigcup_{q \in [1:P] \setminus p} \mathcal{T}_q^+. \tag{4.12}$$

Applying the genetic operations in an iterative fashion leads to significant improvements in the discriminatory power of the keyframes. As illustration we refer to Figure 4.3 (a) which shows the discriminatory power of the learned keyframes over the iterations in terms of average precision, recall, and fitness (using Fit_2) on the training data. Here, averages are taken over the individuals of a population and over all motion classes. In particular, note that the recall values on the training set stay relatively stable at a high level around 1.0, whereas the precision values quickly rise to values above 0.8.

Further, we show how the queries generalize by means of additional mocap documents serving as a verification set. Similar to the training set, we select on average 14.6 additional motions per class, comprising 37 minutes of mocap data, or 66 749 frames. The corresponding performance of the learned keyframe keyframe queries is illustrated in Figure 4.3 (b). Note that although the overall performance on the verification set is lower than the performance on the training set, the fitness also on the verification set always increases with the number of iterations and stable values around 0.75 are reached.

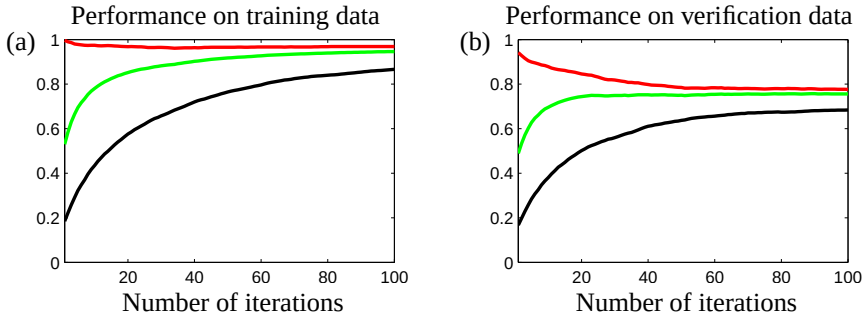


Figure 4.3. Average precision (black), recall (red), and fitness (green) of the learned keyframe queries evaluated on **(a)** the training data and **(b)** on the verification data as a function of the number of iterations used in the genetic algorithm.

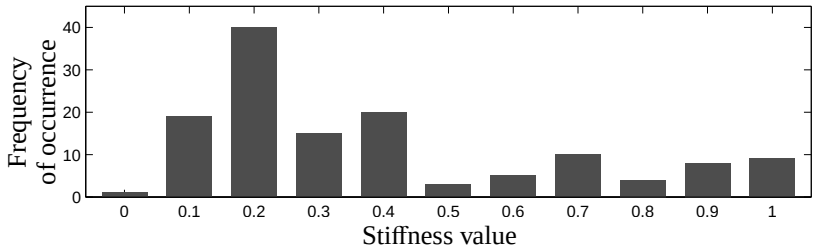


Figure 4.4. Histogram of stiffness values in the automatically learned queries.

Remember that for the initialization of the keyframe queries, we pick columns from a quantized MT. Using a quantization threshold $\delta = 0$ (being very strict to variations in the training data), this quantized MT typically contains many wildcard characters. On the one hand, the so-chosen queries have a recall close to one on the training data, but on the other hand, the discriminative power against other classes is low, yielding a small precision value. We chose this strategy to steer the generated keyframe queries to a high recall with the goal to avoid false negative hits already in the keyframe-based search. During the iterations, keyframe queries are refined and tuned towards a higher fitness. As summarized in Figure 4.3, a strong increase in the precision leads to the improvement in the fitness of the queries, at cost of a small decrease in recall.

In order to visualize in how far temporal variations are taken into account in the learned queries, we computed the histogram of the stiffness values occurring in all 57 queries, see Figure 4.4. More than 87% of the stiffness values occur in the range of stiffness values up to 0.8, which shows that temporal flexibility is needed in order to achieve a high recall. In comparison to the manually defined queries, see Figure 3.13, more stiffness values occur in the highly flexible range around $\sigma = 0.25$. One main reason for this behavior is that a decrease in the stiffness often leads to an increase in the recall on the training set which is preferred in the learning algorithm. Overall, despite of the low stiffness values, the automatically learned keyframes are well suited to cut down the search space as explained in Chapter 5. In the annotation scenario that will be described, we report a 15-fold speedup of the procedure when keyframes are used to cut down the search space.

As for the run times of the learning procedure, using the genetic algorithm with the parameters as specified in Section 4.4, it took roughly 10 seconds on average to learn a keyframe query for a given motion class. Since we run the overall genetic algorithm several times (100-500 times) to derive more characteristic keyframes, run time increases by a corresponding factor. The run

times are mainly determined by the operation that computes the fitness of an individual. Here, one needs to perform a keyframe-based retrieval on the 24-minute training database. On average, the retrieval took roughly 3 milliseconds for one query. This retrieval operation has to be performed several thousand times for each run of the genetic algorithm. Thus, the efficiency of the keyframe-based search algorithm as described in Chapter 3 constitutes one of the major reasons why we are able to reach acceptable run times for the presented genetic keyframe learning algorithm.

Chapter 5

Automated Annotation

This chapter is based on the publication [Müller *et al.*, 2009]. As a main contribution of this thesis, we describe a novel MT-based annotation procedure that is used to segment and label an unstructured mocap document on the basis of a given set of motion classes. Here, an assigned label corresponds to the motion class that best explains the respective motion segment. To solve this challenging task, several components described in the preceding chapters interact with a novel annotation procedure introduced in this chapter. In a preprocessing step, keyframes are learned using the genetic algorithm described in Chapter 4 and motion templates are generated. Then, in the annotation procedure, we first prune the unknown mocap document using the fast keyframe-based search algorithm described in Chapter 3. Hereafter, the novel MT-based annotation strategy is conducted only on a small subset of the document. Unlike previous work, our annotation procedure shows a high degree of robustness to large numerical differences that may exists between semantically related motions (*i. e.*, motions that belong to the same motion class). By employing a keyframe-based search, we efficiently narrow down the set of candidate motions related to a specific motion class and also improve the annotation quality by eliminating false positive matches, as shown in our experiments.

The remainder of this chapter is organized as follows. After giving a brief introduction (Section 5.1), we describe our novel MT-based annotation procedure (Section 5.2). Then, we show how efficiency and precision can be significantly improved by employing a keyframe-based preselection step (Section 5.3). In Section 5.4, we demonstrate the practicability of our overall annotation procedure by describing various experiments conducted on large mocap databases. Our conclusions are given in Section 5.5.

5.1 Introduction

In this chapter, we present a system for automatically and efficiently annotating large unstructured collections of mocap data. Given an unknown mocap document, the annotation task consists of segmenting the document into logical units and then classifying each segment according to a given set of motion classes. Note that the problem of *locally* annotating unknown motion data on the subsegment level is a much harder task than *globally* comparing and classifying motion data on the document level. In our annotation scenario, we assume that each motion class is specified by a

set of semantically related example motions which reflect the range of spatio-temporal variations appearing in valid motion realizations. As motion class representation, we revert to the concept of *motion templates* (MTs) as introduced by Müller *et al.* [2006], see also Section 2.3 for a short introduction. Such templates capture common as well as varying aspects of the underlying training motions in an explicit and semantically interpretable matrix representation.

Most related to our work, Arikan *et al.* [2003] propose a semi-automatic annotation procedure, where a user is required to annotate only a small portion of the database. The user's annotations are then generalized to the entire database in a framewise fashion using SVM classifiers. Our annotation approach differs from their approach in various ways. Firstly, our annotation strategy is segment-based instead of frame-based, thus resulting in semantically more meaningful units. Secondly, using concepts such as relational features and dynamic time warping, our approach is more robust to spatial and temporal variations than the one by Arikan *et al.* [2003], where normalized joint positions and fixed temporal windows are used. Finally, our strategy is to learn the necessary class representations (motion templates, keyframes) only once prior to the actual annotation step. Based on these representations, the annotation can then be performed very efficiently on large and arbitrary sets of mocap documents.

5.2 Annotation Procedure

As basis for our annotation procedure, we introduce a distance function that reveals all motion subsegments of an unknown mocap document D associated with a given motion class C. Let $X \in \{0, 1, *\}^{f \times N}$ be the quantized class MT of C of length N and $Y \in \{0, 1\}^{f \times L}$ the feature matrix of D of length L. We first define a cost measure c^Q for comparing the n^{th} column $X(n)$ of X and the ℓ^{th} column $Y(\ell)$ of Y, $n \in [1 : N]$, $\ell \in [1 : L]$. Let $X(n)_i$ denote the i^{th} entry of the n^{th} column of X. Now, for all columns $n \in [1 : N]$, we define the indices of the rows that do not contain a wildcard character as follows:

$$I(n) := \{i \in [1 : f] \mid X(n)_i \neq *\}, \tag{5.1}$$

Then, if $|I(n)| > 0$, we set

$$c^Q(n, \ell) = \frac{1}{|I(n)|} \sum_{i \in I(n)} |X(n)_i - Y(\ell)_i|, \tag{5.2}$$

otherwise we set $c^Q(n, \ell) = 0$. In other words, $c^Q(n, \ell)$ only accounts for the consistent entries of X with $X(n)_i \in \{0, 1\}$ and leaves the other entries unconsidered. Note that the wildcard entries in a motion template represent admissible variations of a motion class which we ignore in the distance measure by employing $I(n)$. Based on this cost measure, we define a distance function $\Delta : [1 : L] \to \mathbb{R} \cup \{\infty\}$ between X and Y using dynamic time warping (DTW):

$$\Delta(\ell) := \frac{1}{N} \min_{a \in [1:\ell]} \Big(\text{DTW}(X\,,\, Y(a : \ell))\Big), \tag{5.3}$$

where $Y(a : \ell)$ denotes the subsequence of Y starting at index a and ending at index $\ell \in [1 : L]$. Furthermore, $\text{DTW}(X, Y(a : \ell))$ denotes the DTW distance between X and $Y(a : \ell)$ with respect to the cost measure c^Q. To avoid degenerations in the DTW alignment, we use the modified step size condition with step sizes $(2, 1)$, $(1, 2)$, and $(1, 1)$ (instead of the classical step sizes $(1, 0)$, $(0, 1)$, and

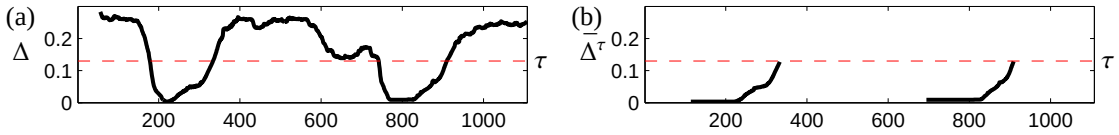


Figure 5.1. **(a)** Distance functions Δ based on the quantized class MT 'sitDownFloor' **(b)** Corresponding modified distance function $\bar{\Delta}^\tau$ for $\tau = 0.13$.

$(1, 1)$). Note that the distance function Δ can be computed efficiently using dynamic programming. For details on DTW and the distance function we refer to [Müller, 2007].

The interpretation of Δ is as follows: a small value $\Delta(\ell)$ for some $\ell \in [1 : L]$ indicates that the subsequence of Y starting at frame a_ℓ (with $a_\ell \in [1 : \ell]$ denoting the minimizing index in Equation (5.3)) and ending at frame ℓ is similar to the class MT X. Here, the starting frame index a_ℓ can be recovered by a simple backtracking within the DTW procedure. In other words, looking for all local minima in Δ below a suitable quality threshold $\tau > 0$ one can identify all subsegments of D closely correlating to the class C. As example, Figure 5.1 (a) shows a distance function based on the quantized MT for the class 'sitDownFloor'. Note that there are two local minima having a value close to zero that reveal the two 'sitDownFloor' subsegments contained in the mocap document.

Recall that a local minimum $\Delta(\ell)$ close to zero only indicates the *end frame* of a subsegment of D corresponding to the class C. We now modify the distance function in such a way that *all frames* $n \in [a_\ell : \ell]$ of the subsegment are distinguished by assigning to them the same distance value $\Delta(\ell)$. Furthermore, with the distance function we only want to consider those frames that closely correlate to C. To this end, we use the quality threshold $\tau > 0$ and iteratively define a modified distance function

$$\bar{\Delta}^\tau : [1 : L] \to \mathbb{R} \cup \{\infty\}. \tag{5.4}$$

First, we set $\bar{\Delta}^\tau(\ell) = \infty$ for all $\ell \in [1 : L]$. Then, iterating over all local minima $\ell \in [1 : L]$ of Δ below τ, we define $\bar{\Delta}^\tau(n)$ for $n \in [a_\ell : \ell]$ to be the minimum of the hitherto defined value $\bar{\Delta}^\tau(n)$ and $\Delta(n)$, see Figure 5.1 (b).

The basic idea of our annotation procedure is to locally compare a mocap document with the various class motion templates and then to annotate all frames within a suitable motion segment with the label of the motion class that best explains the segment. Let D be an unknown mocap document of length L and let $C_1, \ldots C_P$ be the motion classes used for the annotation, where $p \in [1 : P]$ denotes the label of class C_p. In our procedure, we compute a modified distance function $\bar{\Delta}^\tau_p$ for each class C_p as described above. We then minimize the resulting functions over all class labels $p \in [1 : P]$ to obtain a single function $\Delta^{\min} : [1 : L] \to \mathbb{R} \cup \{\infty\}$:

$$\Delta^{\min}(\ell) := \min_{p \in [1:P]} \bar{\Delta}^\tau_p(\ell), \tag{5.5}$$

$\ell \in [1 : L]$. Furthermore, we store for each frame the minimizing index $p \in [1 : P]$ yielding a function $\Delta^{\arg} : [1 : L] \to [0 : P]$ defined by:

$$\Delta^{\arg}(\ell) := \arg\min_{p \in [1:P]} \bar{\Delta}^\tau_p(\ell), \tag{5.6}$$

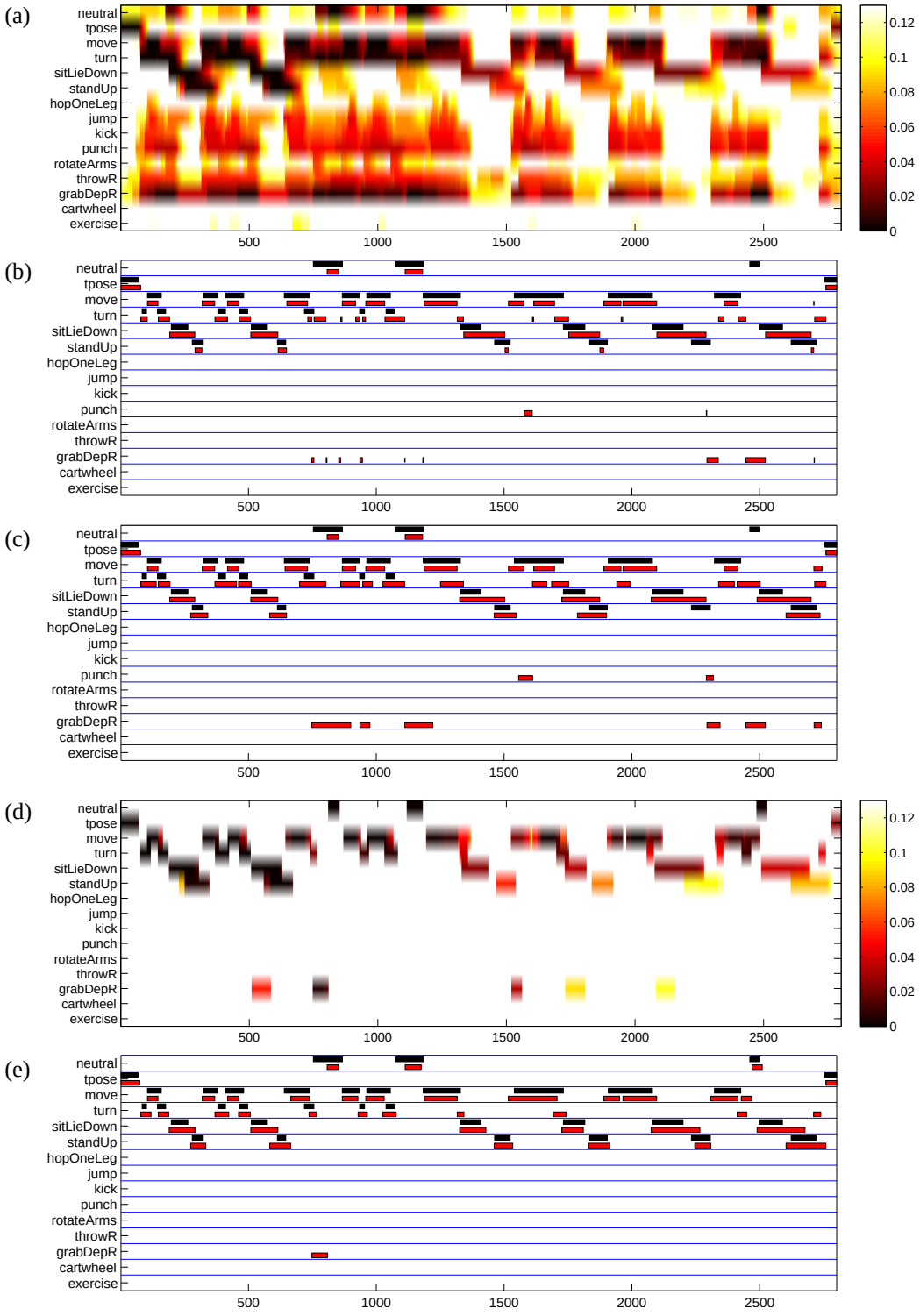


Figure 5.2. **(a)** Modified distance functions (color coded) for each class. White regions indicate distance values larger than $\tau = 0.13$. Time is sampled at 30 Hz. **(b)** Red blocks: Resulting annotations induced by Δ^{arg}. Black blocks: Ground truth annotations. **(c)** Result after extending (b). **(d)** Modified distance functions using keyframes as preprocessing step. **(e)** Annotation result using keyframes.

Class ID	class	description
C_1	neutral	stand in a neutral position, hands lowered
C_2	tpose	stand in t-pose, hands horizontally extended
C_3	move	2 steps (starting left or right, walk, jog, run, ...)
C_4	turn	turn around left or right
C_5	sitLieDown	sit down on chair or floor, kneel, lie down on floor
C_6	standUp	stand up from chair or floor
C_7	hopOneLeg	jump with left or right leg
C_8	jump	jump with both feet, jumping jack
C_9	kick	kick to front or side with left or right leg
C_{10}	punch	punch to front or side with left or right hand
C_{11}	rotateArms	rotate both or single arms front or back
C_{12}	throwR	throw an item with right hand, sitting or standing
C_{13}	grabDepR	grab or deposit with right arm high, middle, low
C_{14}	cartwheeel	cartwheel with left or right hand starting
C_{15}	exercise	elbow to knee, skier, squat

Table 5.1. Description of the 15 motion classes used in our experiments. Each class comprises various subclasses.

where $\Delta^{\mathrm{arg}}(\ell)$ is set to 0 in case $\Delta^{\min}(\ell) = \infty$ (and to the smallest class label number to break a tie). In principle, the function Δ^{arg} yields the annotation of the mocap document D by means of the class labels $p \in [1:P]$. Here, a value 0 means that the corresponding frame is left unannotated.

For a first illustrative example, we use the $P = 15$ classes indicated by Table 5.1, see Section 5.4 for a detailed discussion. Figure 5.2 (a) shows the resulting 15 modified distance functions $\bar{\Delta}_p^{\tau}$ with $\tau = 0.13$ in a color-coded form for a given mocap document D of length $L = 2800$ frames (≈ 93 seconds). The resulting annotations are shown in Figure 5.2 (b), where the color red corresponds to the automatically generated annotations induced by Δ^{arg} and the color black corresponds to manually generated ground-truth annotations. For a further discussion and evaluation of our annotation results, we refer to Section 5.4.

In the following, a maximal run of consecutive frames annotated by the same label is referred to as segment. Note that our procedure cuts the document D into disjoint segments, where some of these may be very short. For example, the 'standUp' annotation segment around frame 1500 comprises only 13 frames ($\approx 1/3$ of a second). This is due to the fact that the beginning of the actual 'standUp'-motion (actor is sitting) is annotated as 'sitLieDown'. This makes sense since the beginning of the 'standUp' motion semantically overlaps with the end of the previous 'sitLieDown'-motion, where the actor sits down. To enable overlapping annotations and semantically meaningful segments (*i. e.*, segments that represent a complete motion of the corresponding class), we further extend the annotated segments as follows. Suppose that the frames with indices $[s:t]$, $s,t \in [1:L]$, $s \leq t$, have been annotated with the class label $p \in [1:P]$:

$$\forall \ell \in [s:t] : \Delta^{\mathrm{arg}}(\ell) = p. \tag{5.7}$$

Then, let $r \leq s$ be the minimal index such that $\bar{\Delta}_p$ is monotonously increasing (or constant) on the interval $[r:s]$. Similarly, let $u \geq t$ be the maximal index such that $\bar{\Delta}_p$ is monotonously decreasing (or constant) on the interval $[t:u]$. Then all frames with indices in the interval $[r:u]$ will also be annotated with p, see Figure 5.2 (c).

5.3 Keyframe-based Preselection

As indicated by Figure 5.2 (c), our annotation procedure as explained so far may yield a number of false positive annotations. For example, the motion class ‘grabDepR’, which consists of right hand grabbing and depositing motions, causes a number of confusions with other classes. The reason is that grabbing and depositing motions are short motions and possess only few characteristic aspects—basically, the right hand is moving and nothing else happens in a consistent way. This leads to a rather unspecific class MT, which yields small distance values to many motion fragments that are actually part of other motion classes (note the overall low distance values in the ‘grabDepR’ row of Figure 5.2 (a) despite of the absence of such motions). To cope with this problem, we propose to integrate an additional keyframe-based preprocessing step. For example, for the class ‘grabDepR’ one may use a few keyframes enforcing that both feet do not move while the right hand moves to the front (before grabbing) and is then pulled back (after grabbing). By employing such additional keyframe constraints, we can eliminate a large number of false positive annotations and, additionally, significantly speed up the annotation procedure.

Recall that in our annotation procedure we want to assign class labels to an unknown mocap document D. In a preprocessing step, we learn the characteristic keyframe queries $(\mathbf{V}_p, \mathbf{d}_p)$ for each motion class p from positive and negative example motions as described in Chapter 4. Then, using the keyframe-based search algorithm as described in Chapter 3, we extract all motion segments from D that are relevant with respect to $(\mathbf{V}_p, \mathbf{d}_p)$. Finally, the distance function Δ_p is computed on the relevant segments only (setting the value to ∞ for the irrelevant frames). The resulting reduction is illustrated by comparing Figure 5.2 (d) with Figure 5.2 (a): the additional white regions in (d) correspond to irrelevant information masked out by the keyframe search. The annotations obtained from (d) are shown in Figure 5.2 (e). Note that the keyframe-based preselection has several benefits. Firstly, using additional constraints allows us to eliminate many false positive annotations. Furthermore, the index-based retrieval step is ideally suited to cut down the search space to relevant subsegments, thus significantly speeding up and drastically reducing memory requirements in the subsequent steps. For details on the keyframe-based search algorithm we refer to Chapter 3. The effect of the keyframe-based preprocessing step on the annotation quality and performance is discussed in Section 5.4.

5.4 Experiments

We implemented the annotation algorithm in MATLAB while passing time critical parts to subroutines implemented in C/C++. The computations were performed on an AMD Athlon X2 5000+ with 3.5 GB of RAM. For our experiments, we assembled an evaluation dataset consisting of 109 mocap documents having an average length of 40 seconds each. The total length amounts to roughly 74 minutes (133019 frames at 30 Hz). To illustrate the scalability of our annotation procedure, we used mocap data from two different sources: 60 minutes where drawn from the HDM05 database [Müller *et al.*, 2007] and 14 minutes from the CMU database [CMU, 2003]. We manually annotated all 109 documents on the subsegment level according to the 15 classes described in Table 5.1. These classes were assembled with respect to the actions performed in the HDM05 motion database. To illustrate the practicability of our annotation procedure, we used various kinds of classes including rather general motion classes such as ‘move’, more specialized classes such

as ‘cartwheel’, and rather uncharacteristic classes such as ‘grabDepR’. Here, the more general classes are assembled from various subclasses. For instance, four different subclasses (sit down on a chair, sit down on the floor, kneel, lie down on the floor) contribute to the class ‘sitLieDown’. To obtain the annotations on the class level, one can simply combine the annotations on the subclass level. At this point, we emphasize that the particular choice of the motion classes is not of crucial importance. The choice was driven by the availability of the mocap data and by our motivation to give a comprehensive demonstration of the algorithms’ performance (even in the presence of more critical classes such as ‘grabDepR’). The concepts presented in this chapter are generic in the sense that the underlying set of motion classes may easily be extended or modified to satisfy a user’s specific needs.

Prior to the actual annotation step, we learned the motion templates and keyframe queries for each of the classes C_p, $p \in [1:P]$, see Chapter 2 and Chapter 4 for an introduction to motion templates and a description of the keyframe learning algorithm. Having completed the preprocessing step, our annotation procedure facilitates efficient annotatation of arbitrary and large sets of unknown mocap documents according to the given set of motion classes (or subsets thereof). To automatically annotate our evaluation database (109 documents, total length of 74 minutes), we proceed as follows. First, we extract the relational features and index the mocap documents using a standard inverted file index [Müller *et al.*, 2005]. In our implementation, the feature extraction takes roughly 250 seconds, whereas the indexing takes 4 seconds. Using the purely MT-based annotation procedure as described in Section 5.2, it took 305 seconds to annotate the 109 documents (here, the index structure is not needed). Applying the keyframe-based preselection (Section 5.3), the run time of the overall annotation procedure decreased to 20 seconds, amounting to a 15-fold speed-up. Here, processing a single keyframe query on the 74-minute evaluation database takes on average only 4 milliseconds (using the index structure), which is negligible compared to the MT-based annotation step.

The keyframe-based preselection step not only yields a significant speedup of the overall annotation procedure, but also has a considerable impact on the final annotation quality. First of all, false positive annotations can be eliminated. As an example, consider the false positive annotations in Figure 5.2 (c) for the class ‘grabDepR’. After integrating the keyframe-based preselection, most of these false positives could be eliminated, see Figure 5.2 (e). As an additional benefit, false negative annotations have been corrected by the keyframe-based preselection as well. Consider the missing missing ‘standUp’ annotation (frames 2250 to 2300 in Figure 5.2 (c)). This annotation appears in the final annotation because the corresponding frames in the ‘sitDown’ class have been masked out by the keyframe search. A similar consideration leads to the presence of the previously missing ‘neutral’ annotation, frame 2500, where the false positive ‘turn’ annotation prevented the ‘neutral’ annotation from being found.

The observed effects are affirmed by our quantitative experiments. Here, we evaluated various variants of our annotation procedure. To this end, we compared the automatically generated annotations with manually generated ground truth annotations by means of two different performance measures. As first measure, we consider precision and recall values on the *frame level*. More precisely, for a given mocap document D of length L we define the sets

$$M(D) := \{(\ell, p) | \text{ frame } \ell \text{ manually annotated with class } p\} \quad \text{and} \tag{5.8}$$

$$A(D) := \{(\ell, p) | \text{ frame } \ell \text{ automatically annotated with class } p\}, \tag{5.9}$$

where $(\ell, p) \in [1:L] \times [1:P]$. In other words, the set $M(D)$ describes the manually generated

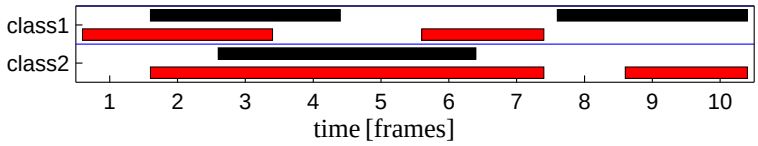


Figure 5.3. Example annotation result to illustrate our evaluation functions, see the text for an explanation.

or *relevant* annotations, whereas the set $A(D)$ describes the automatically generated or *retrieved* annotations produced by our procedure. Then, precision and recall of our annotation procedure are expressed by

$$P_1(D) := \frac{|M(D) \cap A(D)|}{|A(D)|} \quad \text{and} \tag{5.10}$$

$$R_1(D) := \frac{|M(D) \cap A(D)|}{|M(D)|}. \tag{5.11}$$

Furthermore, let

$$F_1(D) := \frac{2P_1(D)R_1(D)}{P_1(D) + R_1(D)} \tag{5.12}$$

be the resulting F-measure. Note that $P_1(D) = 1$ in case of all retrieved annotations being among the relevant annotations (no "false positives"), whereas $R_1(D) = 1$ in case of all relevant annotations being retrieved. The frame-based performance measure F_1 may be problematic, since the beginning and ending of a motion of a specific class is often ambiguous. For example, consider a mocap document showing a person who sits down on a chair and remains seated for a long time. Then, it is not clear where exactly to set the end frame when manually annotating the document with respect to the class 'sitDownChair'. Also certain motion transitions from one class to another (*e. g.*, from 'move' to 'turn') can often not be exactly specified. To account for such ambiguities, we use a second performance measure by considering precision and recall on the *segment level*. Here, we only check for overlaps of a manually annotated motion segment and an automatically generated segment both bearing the same class label p. We then define the segment-based precision $P_2(D)$, recall $R_2(D)$, and F-measure $F_2(D)$ analogously to the frame-based case.

For an illustrative example, we refer to Figure 5.3. Here, the manual ground truth annotation for the class corresponding to the label 'class1' consists of the segments $\{[2:4],[8:10]\}$. The corresponding automatic annotation is $\{[1:3],[6:7]\}$. The overall number of annotated frames amounts to $|M(D)| = 10$ for the manual annotations and $|A(D) = 13|$ for the automatic ones. In this example, the evaluation functions result to

$$P_1 = \frac{6 \text{ frames}}{13 \text{ frames}} \approx 0.46, \tag{5.13}$$

$$R_1 = \frac{6 \text{ frames}}{10 \text{ frames}} = 0.60, \tag{5.14}$$

$$P_2 = \frac{2 \text{ segments}}{4 \text{ segments}} = 0.50, \quad \text{and} \tag{5.15}$$

$$R_2 = \frac{2 \text{ segments}}{3 \text{ segments}} \approx 0.67. \tag{5.16}$$

As also shown by this example, the segment-based measures are more tolerant to smaller deviations in the annotations than the relatively strict frame-base measures, whereas the segment-based measure might sometimes tolerate also strong misalignments between the automatic and the

		P_1	R_1	F_1	P_2	R_2	F_2
total	without keyframes	0.48	0.78	0.60	0.57	0.91	0.70
	with keyframes	**0.69**	**0.79**	**0.74**	**0.78**	**0.88**	**0.82**
HDM	without keyframes	0.49	0.80	0.61	0.61	0.91	0.73
	with keyframes	**0.70**	**0.80**	**0.75**	**0.80**	**0.88**	**0.83**
CMU	without keyframes	0.41	0.75	0.53	0.39	0.90	0.54
	with keyframes	**0.66**	**0.74**	**0.70**	**0.65**	**0.91**	**0.76**

Table 5.2. Various performance measures for our MT-based annotation procedure without and with keyframe-based preselection.

ground truth annotations. Therefore, the actual annotation quality is described well by the range defined by the values $F_1(D)$ and $F_2(D)$.

To compute the performance measures on the entire evaluation database, we simply concatenated the 109 documents to form a single document and applied the above calculation steps, where we performed our annotation procedure without as well as with the keyframe-based preselection step. The results are shown in Table 5.2. For example, the precision P_1 without using keyframes is 0.48 and increases significantly to 0.69 when using our automatically computed keyframe queries. At the same time, the recall R_1 slightly increases from 0.78 to 0.79. While the increase in precision is expected when using keyframes, the increase in recall is somewhat surprising at first sight. Here, one reason is that by eliminating false positives, some of the relevant annotations that have previously been "overlayed" by false positive annotations emerge when using our minimization strategy, see Equation (5.5). This again demonstrates that the keyframe-based preselection step eliminates a large number of false positive annotations while not loosing (or even yielding) relevant annotations. Figure 5.4 shows some representative examples. For example, note that many of the false positive annotations from the rather unspecific class 'grabDepR' could be eliminated across all shown documents using the keyframes. Next, consider the segment between frames 50 and 150 in Figure 5.4 (a). Here, the actor shouts out having both hands raised in front of the mouth. As this motion is not related to any of the employed 15 classes, no manual annotation has been generated for these frames. Without using keyframes, our automatic procedure considers them most similar to either a 'throwR' or a 'grabDepR' motion. Using keyframes as hard constraints, these false positives are eliminated. As a consequence, the precision values as reported below each subfigure receive a significant boost. Next, consider the segment between frames 800 and 1000 in Figure 5.4 (b). Here, starting with both hands in front of the belly, the actor swings his arms sideways into a horizontal position and then lets them drop back to the belly again. This motion is repeated four times in a row. As this motion is not related to any of the employed 15 classes, no manual annotation has been generated for these frames. For this example, the keyframe-based preselection could mask out some false positive 'throwR' and 'grabDepR' annotations, however, a false 'punch' annotation still remains. In the same document, the described motion is followed by a combined walking and arm rotating motion, indicated by the simultaneous 'move' and 'rotateArms' ground truth annotation between frames 1100 and 1400. Note that the motion templates as well as the keyframes have been trained using full body motions that do not include the indicated combined motion. Despite of this, parts of the motion are annotated correctly and the integration of keyframes still improves the overall annotation quality. As another example for a combined motion, consider Figure 5.4 (c). Here, the actor conducts a 'skiing exercise' motion with the upper

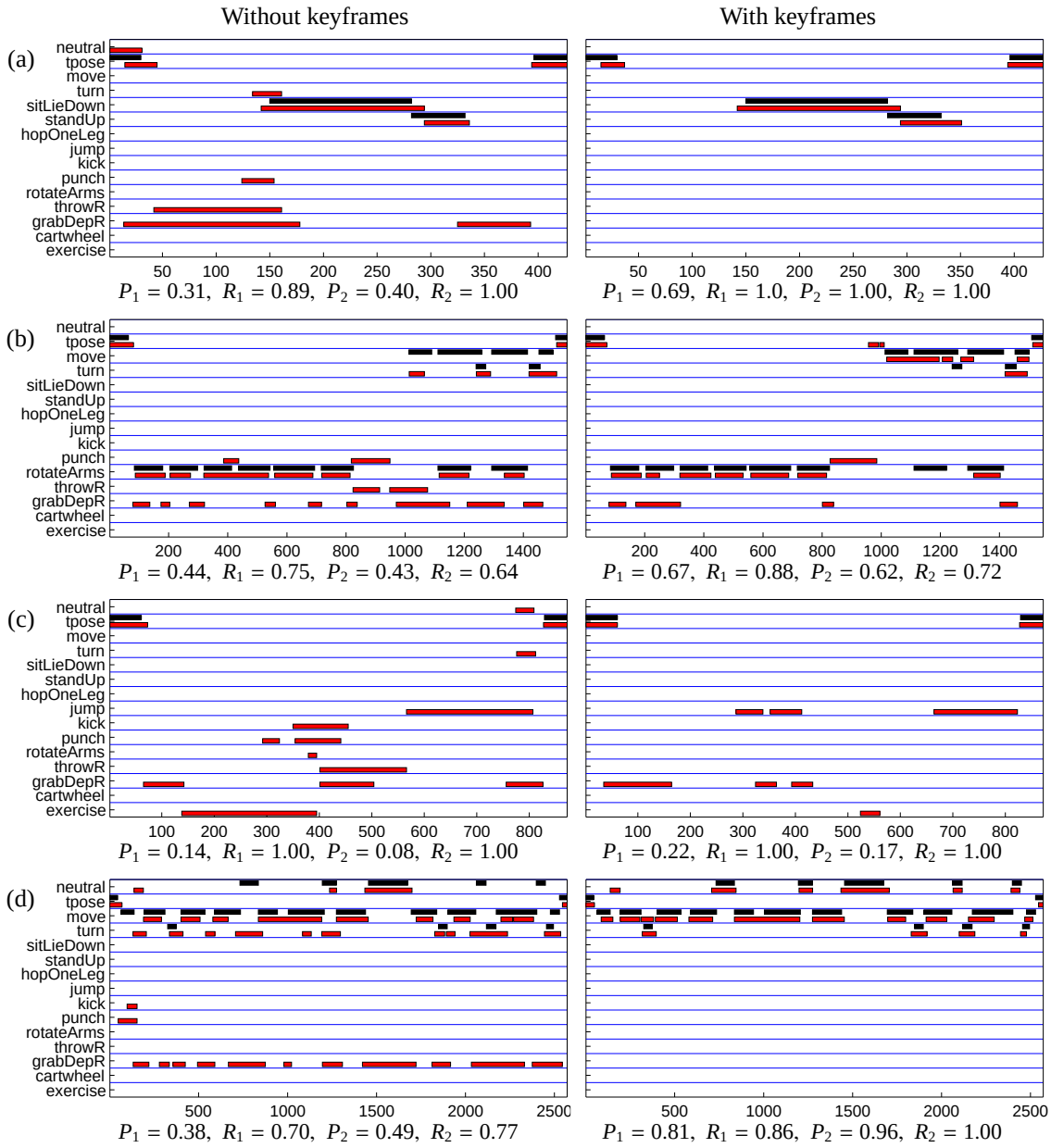


Figure 5.4. Influence of the use of keyframes on the overall annotation result. Left column: annotation result without keyframes. Right column: corresponding annotation result with keyframes.

body and a ‘jumping jack’ motion with the lower body, which is semantically most similar to the ‘exercise’ motion class. However, the manual annotator considered this motion to be too different to the indicated motion classes and did not give a ground truth annotation. Although even with included keyframes some false positive annotations remain, the overall annotation quality improves. As a final example, consider Figure 5.4 (d). Here, the integration of keyframes boosts the segment-based precision and recall to a nearly perfect result, as also confirmed by a manual qualitative inspection.

As expected, the segment-based precision and recall values are higher than the frame-based val-

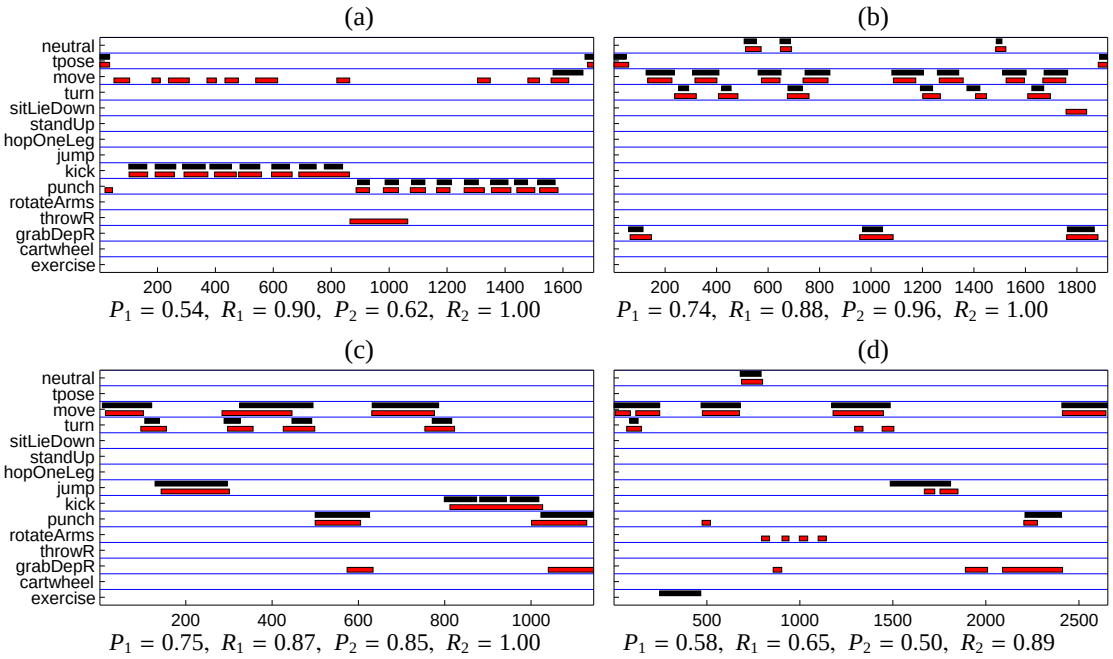


Figure 5.5. Representative annotation results for two HDM05 (**(a)**,**(b)**) and two CMU (**(c)**,**(d)**) documents.

ues, see Table 5.2. For example, using keyframes, one has $P_2 = 0.78$ (opposed to $P_1 = 0.69$). In other words, only 22% of the retrieved annotated segments are false positives. For the segment-based recall, one obtains $R_2 = 0.88$ (opposed to $R_1 = 0.79$). Here, only 12% of the relevant annotations are missing. Note that the frame-based performance measures are generally too strict whereas the segment-based ones are generally too tolerant. So, in summary, the actual performance of our overall annotation procedure can be described by the two F-measures $F_1 = 0.74$ (being pessimistic) and $F_2 = 0.82$ (being optimistic).

As was mentioned above, the HDM05 mocap data used for training is not contained in the evaluation data. However, the various motions corresponding to a specific class, even though performed by various actors executed with significant variations, are still somewhat controlled by general performance specifications. We therefore also evaluated our procedure on CMU documents containing at least some subsegments corresponding to our 15 classes. Table 5.2 shows the various performance measures separately for the HDM05 and CMU documents. Due to significant motion variations in the CMU data, some of which are not well reflected by the HDM05 training material, one has a decline in performance. For example, the F-measures of our overall procedure for the CMU data ($F_1 = 0.70$, $F_2 = 0.76$) are a bit lower than for the HDM05 data ($F_1 = 0.75$, $F_2 = 0.83$.

Figure 5.5 depicts representative annotation results for both HDM05 and CMU documents. Here, Figure 5.5 (a) shows a document containing energetic kicking and punching motions. In contrast to the high recall values as reported below the subfigures ($R_1 = 0.9$, $R_2 = 1.0$), the precision values seem to fall short ($P_1 = 0.54$, $P_2 = 0.62$) mainly due to false positive annotations in the ‘move’ class. A manual inspection of this document showed that in fact the actor moved two steps before each kick in order to gain momentum which gives another twist on the false positive annotations. Also, the ‘move’ annotations during the punches correspond to forward and backward motions with the feet. Finally, we inspected the false positive ‘throwR’ motion around frames 900 - 1100 which actually contains two punching motions. Note that throwing and punching motions share

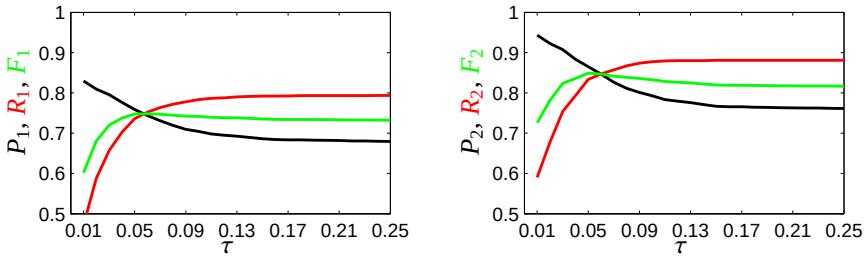


Figure 5.6. Impact of the quality threshold τ on the frame-based (left) and segment-based (right) performance measures (using the annotation procedure with keyframe-based preselection). Black: precision. Red: recall. Green: F-measure.

common similarities. We found that from the mocap data of the false positive annotation alone without being able to see whether an actual object has been thrown it was hard to judge to which class the motion actually belongs. Thus, in some cases the subjective quality of the annotations can be much higher than indicated by the precision and recall values.

An example which contains motions of the class 'grabDepR' is shown in Figure 5.5 (b). Although this class is rather unspecific, it has been singled out by the annotation procedure, yielding very good precision and recall values on the segment level.

We show a document from the CMU mocap database in Figure 5.5 (c). This document contains energetic jumping, kicking, and punching as well as moving and turning motions. Despite of the different style of execution in comparison to the HDM05 training data, these motions have been annotated correctly.

The annotation of a more problematic CMU document is shown in Figure 5.5 (d). Here, the motion segment around frame 1000 erroneously received the annotation 'rotateArms'. A manual inspection showed that this segment actually consists of several arm swings—a motion type that is not reflected in the 15 motion classes used for the annotation. Furthermore, an exercise motion (around frame 400) was not annotated. Here, it turned out that the motion did not satisfy the keyframe constraints learned from HDM05 data. The performed actions can be reviewed on our project homepage `http://www.mpi-inf.mpg.de/resources/MocapAnnotation` where we show videos along with the manual and automatic annotations of all 109 evaluation documents.

In all of the above experiments, we used the quality threshold $\tau = 0.13$. Actually, the choice of τ influences the quality of the overall annotation result. Note that a small value of τ poses a stronger condition on what to consider similar, thus leading to higher precision and lower recall, while a large value of τ has the opposite effect. To find a good trade-off of having high precision as well as high recall, we computed the various performance measures for different values of τ, see Figure 5.6. Our final choice of $\tau = 0.13$ is motivated by the request of having high recall values possibly at the expense of some additional false positive annotations.

5.5 Conclusions

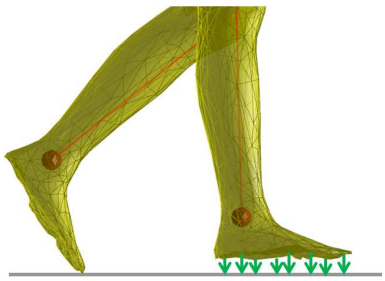
In this chapter, we presented a robust and efficient procedure for annotating large collections of motion capture documents. Using motion templates, we were able to identify logically related mo-

tions on the subsequence level even in the presence of significant numerical differences in the original raw mocap data. Using keyframe queries, we were able to efficiently prune the search space and to eliminate false positives. We developed a purely content-based framework where keyframes and motion templates were all generated automatically by means of user-supplied training motions. We reported on various experiments that demonstrated the practicability of our annotation procedure.

Our concept is generic in the sense that it allows a user to easily adapt and modify the annotation types simply by exchanging the underlying motion classes. Because of their explicit semantic interpretation, even a manual design or tuning of motion templates and keyframes is feasible in case no suitable example motions are available. As for future work, we plan to apply our concept for automatically annotating various types of gesture. From such annotations, statistical models of a person's particular gesture style could be learned in order to synthesize personalized gestures [Neff *et al.*, 2008]. As shown in the subsequent chapter, a procedure for mocap annotation can also be used to generate suitable prior knowledge which can then be used to stabilize and support human motion tracking.

Part II

Stabilizing and Analyzing Video-based Motion Tracking



Chapter 6

Stabilizing Tracking using Retrieved Motion Priors

As one main contribution of this thesis, we introduce in this chapter an iterative framework that makes use of a retrieval and annotation component similar to the one described in Chapter 5 in order to stabilize an algorithm for marker-less motion tracking. This work has been published in [Baak *et al.*, 2009]. A flowchart of our framework is given in Figure 6.1. As one component, we use an algorithm for makerless motion tracking that estimates the 3D motion of a human actor based on image data. The tracking system takes as input a multiview image sequence ('Video'), see Figure 6.2, and returns as output mocap data given as a sequence of joint angles over time ('3D Mocap'). Due to noise, occlusions, and other ambiguities in the image data, tracking may fail for parts of the sequence resulting in corrupted poses, see Figure 6.3. However, in spite of these errors, the overall rough course or at least parts of the motion may still be recognized to a reasonable degree. For example, a local tracking error in the arm during a walking motion may still permit an algorithm to recognize the motion class correctly. We use an annotation algorithm similar to the one presented in Chapter 5 to locally assign class labels to the tracked 3D mocap data based on available motion classes ('Database knowledge'). These assigned class labels represent an increased semantic knowledge about the tracked sequence that we exploit by allocating priors based on the obtained annotation results. For example, after recognizing a walking cycle in the tracked sequence, this increase of knowledge can be used to allocate a simple prior such as 'left foot moves to the front'. The allocated priors are integrated into the tracking procedure as regular-

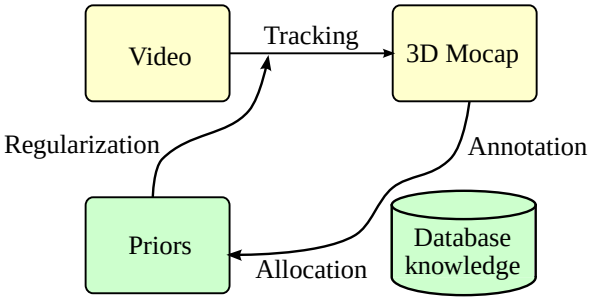


Figure 6.1. Iterative motion tracking framework. Based on an automated annotation of the tracking result, priors are allocated that regularize the tracking in the next iteration step.

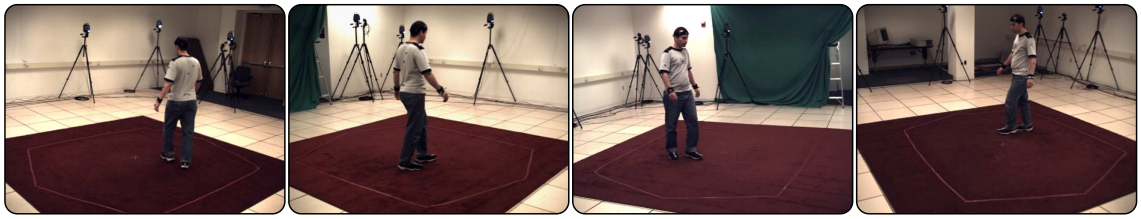


Figure 6.2. One frame in the HumanEVA multiview video footage as seen from the four different cameras.

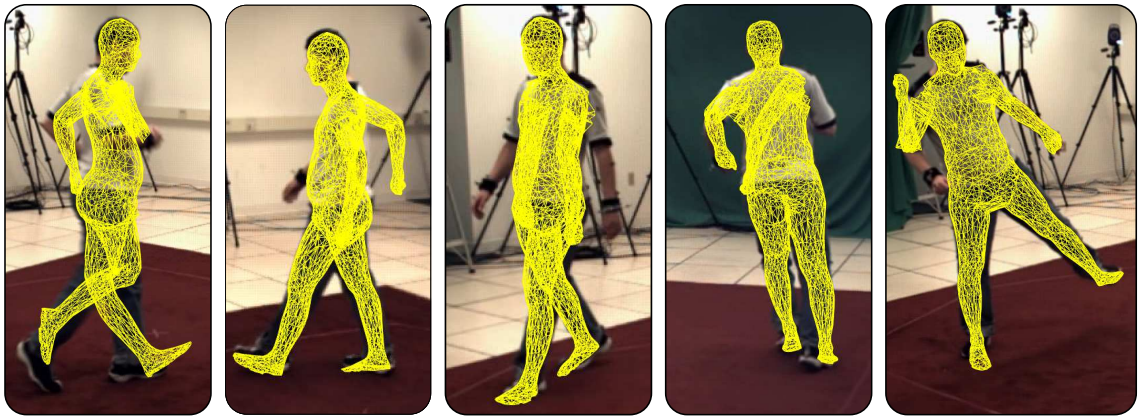


Figure 6.3. Tracking without priors may lead to invalid poses.

ization terms, and the tracking step is repeated to yield an enhanced tracking result. Iterating such a procedure, as shown by our experiments with the HumanEVA-II benchmark, improves the result significantly after few iterations.

The remainder of the chapter is organized as follows. We first give an overview about related work (Section 6.1) and summarize the tracking procedure (Section 6.2). After that, we briefly describe the retrieval component (Section 6.3). Then we explain how to fuse the retrieval results with our tracking procedure in Section 6.4. At the time of publication, such an iterative tracking procedure using retrieved motion priors had not been considered before to the best of our knowledge. Besides stabilization of 3D tracking we also gain an annotation, which bridges the gap between the low-level tracked 3D mocap data and the high-level symbolic representation of the underlying actions. Our experiments are presented in Section 6.5, before we conclude in Section 6.6.

6.1 Related work

Marker-less motion tracking is an active field of research in computer vision and graphics, see [Moeslund *et al.*, 2006; Poppe, 2010; Sigal *et al.*, 2010; Moeslund *et al.*, 2011] for in-depth reviews of the vast literature. The goal of marker-less mocap is to determine the moving and deforming 3D surface geometry of an actor from image data. Applications are mostly found in the game and and movie industry, in biomechanics, medicine, and sports sciences. In a tracking scenario, it is common to assume that the input consists of a sequence of multiview images of the performed motion as well as a static surface mesh of the actor's body as, *e. g.*, obtained via a body laser scanner. Commonly, a skeleton is used to drive the deformation

of the surface geometry, see Figure 6.4. Since the pose and joint parameters are usually unknown and have to be estimated from the image data, one typically has to cope with high-dimensional search spaces (often more than 30 dimensions for a full body model). In order to reliably estimate pose parameters, recent approaches such as [Gall *et al.*, 2010a; Liu *et al.*, 2011; Stoll *et al.*, 2011] either rely on global optimization methods (which are often computationally expensive and thus prohibitive for real-time applications) or need many cameras. To enhance the stability and reliability of the tracking procedure and to enable tracking also with a low number of cameras, further sources of information can be integrated into the motion tracking process, *e. g.*, by capturing light sources [Bălan *et al.*, 2007b] or by using physically-based models and forces arising from a ground plane [Brubaker *et al.*, 2007; Vondrak *et al.*, 2008]. Another strategy is to reduce the manifold of all virtually possible configurations to a lower-dimensional subspace. One possibility is to explicitly prevent self-occlusions and to impose fixed joint angle limits as suggested in [Herda *et al.*, 2004; Sminchisescu and Triggs, 2003]. Other options include to directly learn a mapping from the image or silhouette space to the space of pose configurations [Agarwal and Triggs, 2006; Shakhnarovich *et al.*, 2003], to learn a restricted motion model from training data [Li *et al.*, 2010], or to combine generative an discriminative models [Salzmann and Urtasun, 2010]. A very popular strategy for restricting the search space is dimensionality reduction, either by linear or by nonlinear projection methods. In [Sidenbladh *et al.*, 2002], the low-dimensional space is obtained via PCA and the motion patterns in this space are structured in a binary tree. In [Sminchisescu and Jepson, 2004] it has been suggested to learn a Gaussian mixture from pose configurations. Similarly, in [Urtasun *et al.*, 2006], a nonlinear projection is employed, in this case via a Gaussian process model.

A common problem with learning-based approaches is the need of suitable training data that reflects the statistics of the expected motions. For instance, if the user knows that the subject performs a walking pattern, suited training data is selected and integrated in the tracking system. Then, the tracking system might fail if the user does not restrict his motion to walking. As a further problem, current probabilistic learning approaches are limited in their ability to handle large training sets. Only recently, local regression methods have been proposed that allow for coping with a large number of motion patterns in a tracking scenario [Urtasun and Darrell, 2008]. In activity recognition, many approaches rely on 2D descriptors or image silhouettes, such as presented in [Liu and Shah, 2008; Tran and Sorokin, 2008]. An approach that performs simultaneous motion tracking and action recognition has been presented in [Chen *et al.*, 2009]. Recently, Gall *et al.* [2010b] presented a tracking approach conceptually related to the one presented in this chapter. Similar to our approach, action-specific priors are applied to the sequence to be tracked by means of an annotation scheme. However, instead of running an annotation on the 3D tracked mocap data, they use an approach for activity recognition on the 2D image data. As a further difference, in the annotation phase, our approach makes hard decisions whether an action has been detected or not, whereas Gall *et al.* [2010b] compute and use action probabilities in a particle-based approach. According to the probabilities, the particles, representing candidate poses, are spread within corresponding learned pose manifolds of the activities in order to stabilize the tracking procedure. Using the action probabilities with action-specific pose manifolds seems to be advantageous at first sight with respect to hard decisions, since false annotations might not have a strong effect on the tracking result. However, in the approach of Gall *et al.* [2010b], pose manifolds have to be created for all actions that occur in the sequence to be tracked, and actions that have not been learned may lead to tracking errors. As we will show in our experiments, we

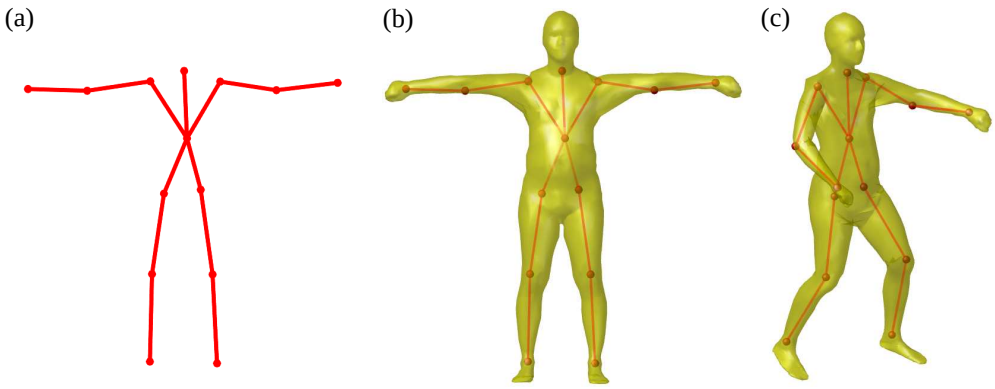


Figure 6.4. Model of the actor. **(a)**: Skeletal kinematic chain. **(b)**: Rigged mesh. **(c)**: Mesh in a new pose.

do not need prior knowledge for all actions present in the tracking sequence, since our annotation procedure does not assign priors to unknown actions. Moreover, we supply priors on a logical level like 'foot moves to the front'. Such logical priors generalize well to variations in actor size and motion styles. For example, both a small person with a staggering or shuffling gait as well as a large person with a normal gait will follow the same logical pattern of first moving one foot and then the other foot to the front. Such logical priors stand in contrast to numerical priors on the distribution of joint angles as supplied in Gall *et al.* [2010b].

6.2 Tracking Procedure

The input of our tracking procedure is a data stream of multi-view images (obtained from a set of calibrated and temporally synchronized cameras) as well as a surface mesh of the subject to be tracked (obtained by a body laser scanner). We further assume that the mesh is rigged so that all mesh points are associated in a fixed way to the joints of an underlying kinematic chain, see Figure 6.4. Then, the tracking problem consists of computing the configuration parameters (joint angles as well as root orientation and translation) of the kinematic chain from the given image data. Here, the surface mesh should be transformed with the configuration parameters in such a way that the projection of the mesh covers the observed subject in the images as accurately as possible.

In the remainder of this section, we briefly summarize the tracking procedure used in our framework. After describing the kinematic chain model for the human skeleton (Section 6.2.1), we summarize how we estimate its pose parameters based on image constraints (Section 6.2.2). Then, in Section 6.2.3, we show how such image constraints are obtained.

6.2.1 Kinematic Chains

The subject to be tracked is modeled by a so-called *kinematic chain*, which is generally used to model a flexibly linked rigid body such as a human skeleton [Bregler *et al.*, 2004], see Figure 6.4 (a) for an example. In the following, we use homogeneous coordinates to represent 3D points and exponential functions of twists to represent rigid body motions. The configuration of

a kinematic chain can then be described by a consecutive evaluation of exponential functions of twists, see [Bregler *et al.*, 2004]. More precisely, let $x \in \mathbb{R}^3$ be a 3D coordinate of a joint in the neutral configuration (standard pose) of the kinematic chain. Let $X = \binom{x}{1}$ be the respective homogeneous coordinate and define π as the associated projection with $\pi(X) = x$. Furthermore, let ξ be a rigid body motion, which can be represented as $\xi = \exp(\theta\hat{\xi})$ with a twist $\hat{\xi}$ and $\theta \in \mathbb{R}$. The overall configuration of the kinematic chain is specified by a rigid body motion $\xi = \exp(\theta\hat{\xi})$ encoding the root orientation and translation as well as a sequence $\xi_1 = \exp(\theta_1\hat{\xi_1}), \ldots, \xi_n = \exp(\theta_n\hat{\xi_n})$ of rigid body motions encoding the n joint angles. Note that the twists $\hat{\xi}_1, \ldots, \hat{\xi}_n$ are fixed for a specific kinematic chain. Thus, the configuration of a fixed kinematic chain is specified by the following $(6 + n)$ free parameters:

$$\chi := (\xi, \Theta) \quad \text{with} \quad \Theta := (\theta_1, \ldots, \theta_n). \tag{6.1}$$

In other words, the configuration parameter vector χ consists of the 6 degrees of freedom for the rigid body motion ξ and the joint angle vector Θ, see also [Rosenhahn *et al.*, 2008a]. Now, for a given point x on the kinematic chain, we define $\mathcal{J}(x) \subseteq \{1, \ldots, n\}$ to be the ordered set that encodes the joint transformations affecting x. Then, for a given configuration parameter vector $\chi := (\xi, \Theta)$, the point x is transformed according to

$$Y = \exp(\theta\hat{\xi}) \prod_{j \in \mathcal{J}(x)} \exp(\theta_j \hat{\xi}_j) X. \tag{6.2}$$

6.2.2 Pose Estimation

In our setup, the vector χ is unknown and has to be determined from the image data. In the following, instead of regarding points on the kinematic chain, we use points on the surface mesh. As the mesh is rigged, the mesh points are directly associated to a joint. Given a set of 3D surface mesh points x_i, $i \in I$, we assume for the moment that one knows corresponding 2D coordinates of these points within a given image, and we refer to Section 6.2.3 for a description of how such correspondences can be obtained. Furthermore, we represent each 2D point as a reconstructed projection ray given in 3D Plücker form $L_i = (n_i, m_i)$, see also [Murray *et al.*, 1994]. For pose estimation, the basic idea is to apply the (unknown) rigid body motions on 3D points x_i according to χ and to claim incidence with the reconstructed projection rays. Due to the properties of Plücker lines, this incidence can be expressed as

$$\Big(\pi\Big(\exp(\theta\hat{\xi}) \prod_{j \in \mathcal{J}(x_i)} \exp(\theta_j \hat{\xi}_j) X_i\Big) \times n_i\Big) - m_i = 0. \tag{6.3}$$

To simultaneously account for the incidences of all points x_i, $i \in I$, one minimizes the following term in a least-squares sense:

$$\arg\min_{\chi} \sum_i \left\| \Big(\pi\Big(\exp(\theta\hat{\xi}) \prod_{j \in \mathcal{J}(x_i)} exp(\theta_j \hat{\xi}_j) X_i\Big) \times n_i\Big) - m_i \right\|_2^2 \tag{6.4}$$

To solve for the unknown parameters in the exponential functions, we linearize each function by using the first two elements of the respective Taylor series: $\exp(\theta\hat{\xi}) \approx \mathbf{1} + \theta\hat{\xi}$. This leads to three linear equations with $6 + n$ unknowns for each exponential function. In case of many correspondences (*i. e.*, in case there are many mesh points x_i with correspondences), one obtains an over-determined linear system of equations, which can be solved in the least squares sense. The approximation errors introduced by the linearization step are handled by applying an iterative computation scheme, see [Rosenhahn *et al.*, 2008a] for details.

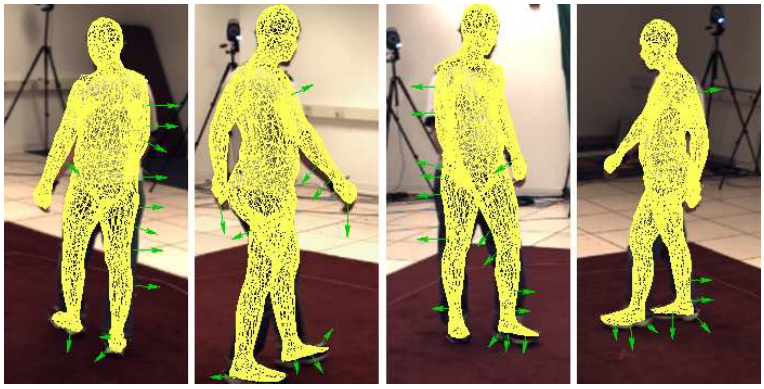


Figure 6.5. Example forces (enlarged force vectors, green) acting on the contour line of the projected surface mesh.

6.2.3 Region-based Pose Tracking

The pose estimation procedure described in Section 6.2.2 requires known correspondences between 2D image points and 3D surface mesh points. In the following, we briefly describe how one can generate such correspondences from suitable foreground and background statistics derived from the image data. Having correspondences, the sought pose parameters can be computed by solving Equation (6.4). In our framework we use the region-based tracking approach as presented in [Schmaltz *et al.*, 2007], to which we refer for details. However, alternatively, one could also use other techniques as presented in [Bray *et al.*, 2006; Dambreville *et al.*, 2008; Rosenhahn *et al.*, 2007b; Stoll *et al.*, 2011].

The concept is to estimate pose parameters χ such that the projection of the resulting surface mesh optimally splits the image into a foreground (subject) and a background region. Here, the splitting is regarded as optimal if suitable image features (color, texture) are maximally dissimilar in the two regions with regard to estimated density functions, see [Schmaltz *et al.*, 2007]. Starting with a first estimate χ, the transformed mesh points y_i (see Equation (6.2)) are projected onto image points p_i yielding correspondences in a natural way. One then considers only the visible image points p_i that lie on the contour line separating foreground and background. Next, these points are shifted inwards or outwards (orthogonal to the contour line) according to force vectors so that the resulting points, say q_i, better explain the color distributions of the foreground and background regions, see Figure 6.5. Finally, using the points q_i with the corresponding mesh points x_i (obtained from the transformed mesh points y_i) we apply the minimization (6.4) to obtain an improved estimation of the pose parameters. The entire process is iterated until convergence, see [Schmaltz *et al.*, 2007] for details.

6.3 Retrieval Component

In this section, we describe the retrieval component that is used to automatically assign priors to a tracked sequence. We use an annotation procedure similar to the one described in Chapter 5. However, we do not integrate a keyframe-based search for two reasons. Firstly, motions to be tracked with marker-less mocap methods are comparatively short sequences of up to a minute in

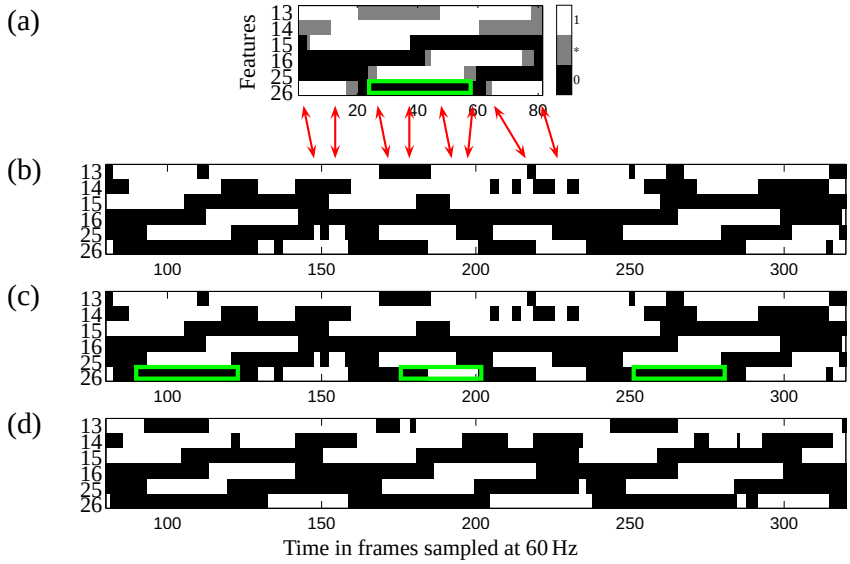


Figure 6.6. **(a)**: Marked motion template for motion class 'walk2StepsLstart'. The feature numbers correspond to the features used in [Müller and Röder, 2006], see also Table 2.1 on page 13. The annotation for the relational or logical constraint 'leftFootOnGround' is indicated by the green rectangle. **(b)**: Feature matrix for the subsegment of D_{EVA} consisting of the first three walking cycles. The second walking cycle (frames 160 to 240) has not been tracked correctly. **(c)**: Feature matrix (b) with allocated priors (green rectangles). **(d)**: Feature matrix after regularized tracking using the priors of (c).

length. For such sequences, DTW-based retrieval can be performed within less than a second and therefore no additional techniques to speed up the annotation procedure are needed. Secondly, we want the annotation procedure to be robust to tracking errors, where, *e. g.*, a foot or a hand might constantly assume incorrect poses. With hard-constraints coming from keyframes, such sequences containing tracking errors might be excluded from the set of hit candidates if keyframe-based preprocessing is used.

In the following, we briefly describe the annotation procedure as used in this chapter. In our scenario, each motion category is given by a class C consisting of a set of logically related example motions. For each class, we first compute a quantized MT X_C as summarized in Chapter 2, see Figure 6.6 (a) for a resulting template.

Let $C_1, \ldots C_P$ be the available motion classes, where $p \in [1:P]$ denotes the class label of class C_p. Then, given a mocap sequence D of length L, the annotation task is to identify all motion subsegments within D that belong to one of the P classes. To this end, we compute a distance function $\Delta_p := \Delta_{C_p}$ for each class C_p, see Equation (5.3) in Chapter 5 on page 48, and minimize the resulting functions over $p \in [1:P]$ to obtain a single function $\Delta^{\min} : [1:L] \to \mathbb{R} \cup \{\infty\}$ for a sequence comprising L frames:

$$\Delta^{\min}(\ell) := \min_{p \in [1:P]} \Delta_p(\ell) \tag{6.5}$$

for $\ell \in [1:L]$, see Figure 6.7 for an example based on $P = 2$ motion classes. Furthermore, we store for each frame the minimizing index $p \in [1:P]$ yielding a function $\Delta^{\arg} : [1:L] \to [1:P]$ defined by:

$$\Delta^{\arg}(\ell) := \arg\min_{p \in [1:P]} \Delta_p(\ell). \tag{6.6}$$

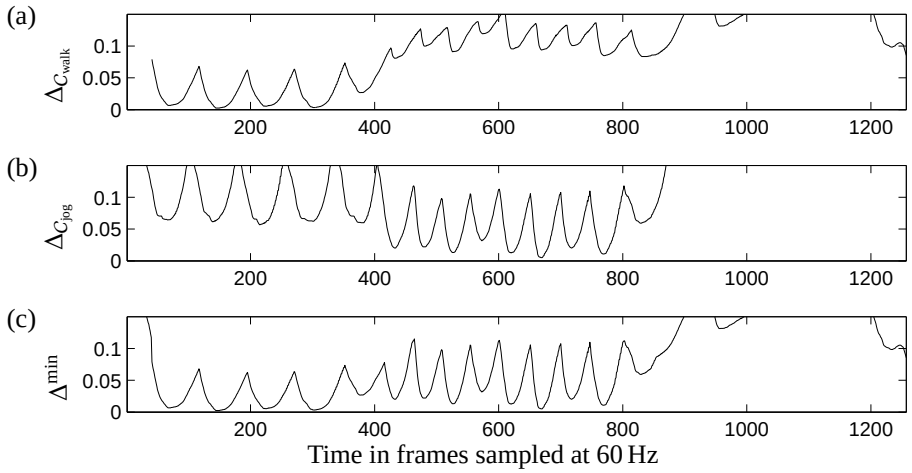


Figure 6.7. Distance function Δ_C for D_{EVA} with respect to the class **(a)** 'walk2StepsLstart' and **(b)** 'jog2StepsLstart'. **(c)**: Combined distance function $\Delta^{\min}$ obtained by minimizing (a) and (b).

The function $\Delta^{\arg}$ yields an annotation of the mocap sequence D for every frame by means of the class labels $p \in [1 : P]$.

6.4 Allocation and Integration of Priors

In this section, we first explain how to generate suitable motion priors based on an annotation of the tracked sequence (Section 6.4.1). Then, we show how two types of allocated priors are integrated into the subsequent tracking iteration as soft constraints (Sections 6.4.2 and 6.4.3).

6.4.1 Allocation of Priors

We now explain how to generate suitable motion priors, which can then be used to regularize the tracking process. Recall that a class motion template X_C explicitly encodes characteristic motion aspects (corresponding to black/white regions) that are typically shared by motions of class C. We select some of these aspects by marking suitable entries within the template X_C. These entries are also referred to as *MT priors*. As an example, consider Figure 6.6 (a), where the entries of row 26 between columns 22 and 58 are marked by the green rectangle. Feature 26 expresses whether the right foot rests (black, value 0) or assumes a high velocity (white, value 1). Since all entries have the value 0 within the green rectangle, this MT prior basically expresses that the right foot rests (stays on the ground) during this phase of the motion. Note that the MT priors are part of the database knowledge and do not depend on the sequence to be tracked.

Now, let D be a mocap sequence of length L obtained from some tracking procedure and let $Y \in \{0,1\}^{f \times L}$ be the corresponding feature matrix of relational features, see Figure 6.6 (b). The goal is to automatically transfer suitable MT priors to the tracked sequence to obtain what we refer to as *tracking priors*. Let $C_1, \ldots C_P$ be the available motion classes with corresponding motion templates $X_p = X_{C_p}$, $p \in [1 : P]$, each equipped with suitable MT priors. We compute the functions $\Delta^{\min}$ and $\Delta^{\arg}$ as described in Section 6.3. Recall that a local minimum $\ell \in [1 : L]$

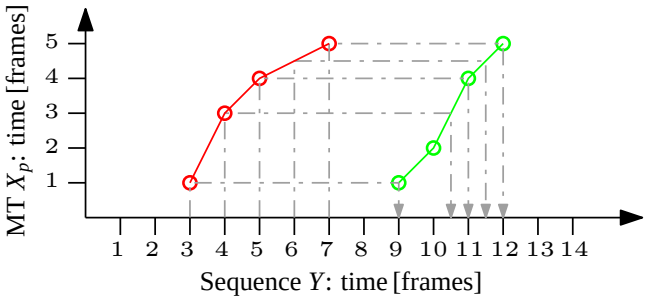


Figure 6.8. Pose priors are allocated by looking for all subsequences in Y that align to the same MT X_p.

of $\Delta^{\min}$ close to zero indicates the presence of a motion subsegment of D (starting at a suitable frame index $a_\ell < \ell$ and ending at frame index ℓ) that corresponds to motion class $\Delta^{\arg}(\ell) \in [1 : P]$. Now, we fix a quality threshold $\tau > 0$ and look for all essential local minima $\ell \in [1 : L]$ with $\Delta^{\min}(\ell) < \tau$. Here, *essential* means that we only consider one local minimum within a suitable temporal window to avoid local minima being too close to each other. Using the same DTW procedure that yields the distance function Δ_p in Section 6.3, we then derive an alignment between the motion template X_p with $p := \Delta^{\arg}(\ell)$ and the feature subsequence of Y ranging from a_ℓ to ℓ. Figure 6.6 shows an example, where the alignment is indicated by the red arrows. Note that such an alignment establishes temporal correspondences between semantically related frames and thus allows us to automatically transfer the MT priors within X_p to corresponding regions within Y, see Figure 6.6 (c). These regions, in the following referred to as *tracking priors*, are then used for regularization in the tracking procedure as explained in the following sections.

As an additional stabilizing factor, we take further advantage of cases where several subsequences of Y align to the same MT X_p. The idea is to use X_p as a kind of mediator to generate additional priors from the multiply aligned subsequences. We explain this idea by means of a simple example consisting of two subsegments as indicated by Figure 6.8. Here, each circle denotes a correspondence (x_ℓ, ℓ) between frame x_ℓ in X_p and frame $\ell \in [1 : L]$ of Y. In this example, essential local minima were found for frames 7 and 12 (matching to the subsequences ranging from frames 3 to 7 and from 9 to 12). Now, suppose that the green alignment has a cost close to zero ($\Delta^{\min}(12) \approx 0$). In practice, such an alignment corresponds to a subsequence of Y that does not contain tracking errors. By contrast, suppose that the subsequence corresponding to the red alignment contains some tracking errors resulting in higher alignment cost. Then, the idea is to use the poses of the "green subsequence", referred to as *pose priors*, to stabilize the tracking of the "red subsequence". The correspondence of poses between the subsequences is established via the alignments to X_p, see Figure 6.8. For example, frame 9 of Y yields a pose prior for frame 3 of Y since both frames are aligned to the first frame of X_p (indicated by the dashed arrow line). To put it in simple words, we first detect the presence of repetitions within Y by means of the MT-based local classification and then generate pose priors from the established correspondences.

6.4.2 Integration of Tracking Priors

Tracking priors provide information about certain logical movement behaviors of body parts within a certain motion context. As an example, we consider the tracking prior "left foot should be on the floor at a certain point in time". We use soft constraints to integrate this information in

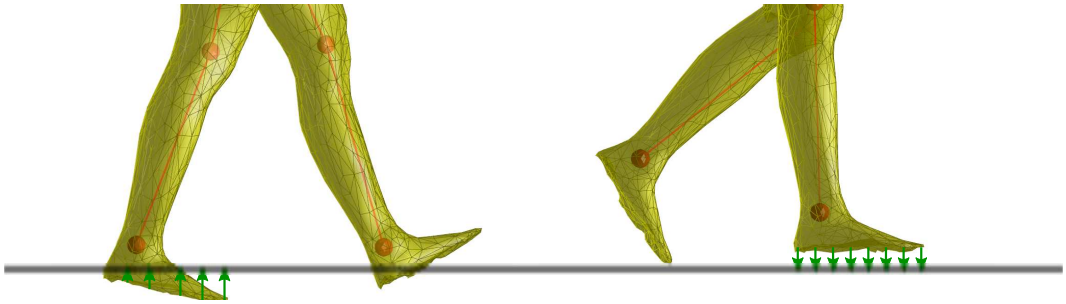


Figure 6.9. Integration of the constraint *foot on floor*. Points on the sole are pushed onto the floor.

the tracking framework, where the amount of influence of a prior can be controlled by a weighting parameter. In particular, soft constraints are formulated as additional equations that are included in the minimization step depicted in Equation (6.4). To implement the example tracking prior, all points $y_s, s \in S \subset I$, on the sole of the foot (the plantar) are projected onto the ground plane, yielding the points z_s. Then, we claim incidence

$$y_s - z_s = 0, \qquad s \in S \tag{6.7}$$

to push the sole onto the ground plane, see Figure 6.9. Related to this floor constraint is the implementation as carried out in [Rosenhahn *et al.*, 2008b]. In their work, the constraint can only be applied when penetrations of the foot with the ground plane are detected. As an extension, in our work, the constraint also has an effect if the foot is above the ground plane, thus reducing the effect of "flying above the ground plane".

Using Equation (6.2) to express y_s by the underlying kinematic chain, we integrate the set of equations

$$\pi\Big(\exp(\theta\hat{\xi}) \prod_{j \in \mathcal{J}(y_s)} \exp(\theta_j \hat{\xi}_j) X_s\Big) - z_s = 0, \quad s \in S \tag{6.8}$$

into the minimization step (6.4). Note that the unknowns are the same as for Equation (6.4) as z_s are considered as constants for one frame. In a similar manner it is straightforward to integrate motion dynamics like arms swinging forward or backwards.

6.4.3 Integration of Pose Priors

Unlike tracking priors, *pose priors* denote that a certain joint angle configuration Θ at frame $\ell_1 \in [1 : L]$ should also be assumed in frame $\ell_2 \in [1 : L] \setminus \{\ell_1\}$. For example, consider Figure 6.10 where the generated pose prior suggests to take Θ of frame $\ell_1 = 157$ for frame $\ell_2 = 310$. To this end, equations similar to Equation (6.8) can be integrated into the minimization step (6.4) to regularize the joint angle configuration at frame ℓ_2 towards Θ in the subsequent tracking iteration.

6.5 Experiments

In our experiments, we used the Human EVA-II benchmark dataset [Sigal *et al.*, 2010]. Here, a surface model, calibrated multiview image sequences of four cameras, and background images

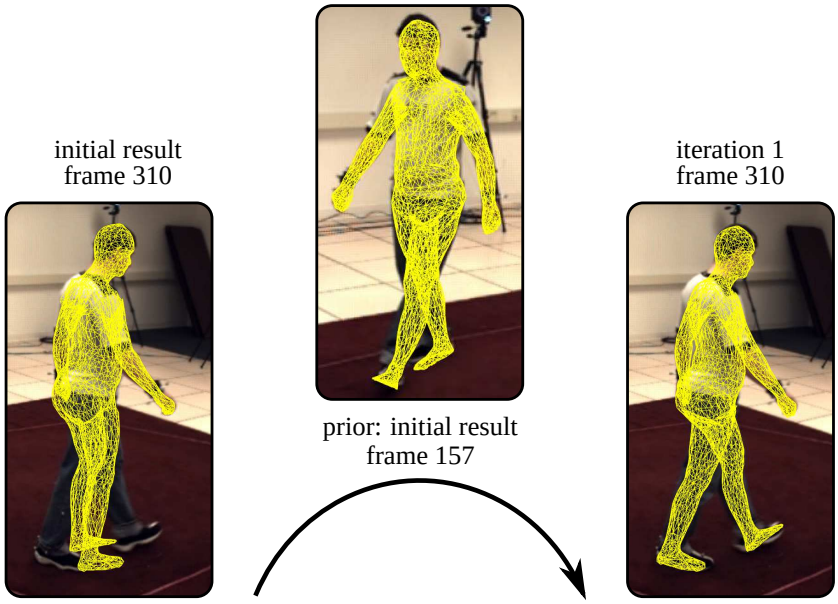


Figure 6.10. A well tracked frame (middle) is used as a prior for a frame with tracking errors (left). After the subsequent tracking iteration, the error has been resolved.

are provided. Note that our region-based pose tracking does not rely on background subtraction and therefore the background information is not used in our method. Instead, we rely on the image data, projection matrices and a mesh model. Due to color similarities of foreground and background as well as the sparse number of cameras, tracking is challenging and the results are likely to be corrupted if no priors are involved. Tracking results (as 3D marker positions) can be uploaded to a server at Brown University for evaluation. As the sequence has been captured in parallel with a marker-based tracking system [Vicon, 2012], an automated script can evaluate the accuracy of a tracking result in terms of relative errors in millimeters. In the Human EVA-II sequence $S4$, three different actions are performed consecutively, lasting for 6.7 s (400 frames at 60 Hz) each. A non-professional actor walks in a circle, jogs in a circle, and then balances on each foot. We chose this sequence for several reasons: Firstly, it is a publicly available benchmark dataset, which allows us to quantitatively compare to other existing approaches. Secondly, the sequence contains three different patterns and we want to test whether our system is able to classify and single out the involved motions correctly (walking and jogging). Thirdly, walking and jogging are similar patterns, which allows us to get a good feeling about the sensitivity of our approach in classifying similar patterns. Fourthly, for the balancing part of the sequence, we do not have appropriate database knowledge. This means that the algorithm should not assign a class label in the annotation stage, so that the tracking is only driven from the image data without any priors. All these aspects can be covered by this sequence.

The database knowledge that is used by the retrieval system is generated in a preprocessing step. To this end, we assembled a total of 232 short 3D mocap clips, which we manually cut out from the freely available HDM05 mocap database [Müller *et al.*, 2007] (obtained from a Vicon system). The mocap clips of an average length of 1.1 s were categorized into $P = 6$ different motion categories, which are ‘walk two steps’, ‘jog two steps’, and ‘change from walk to jog’, each for starting with the left and right foot, respectively.

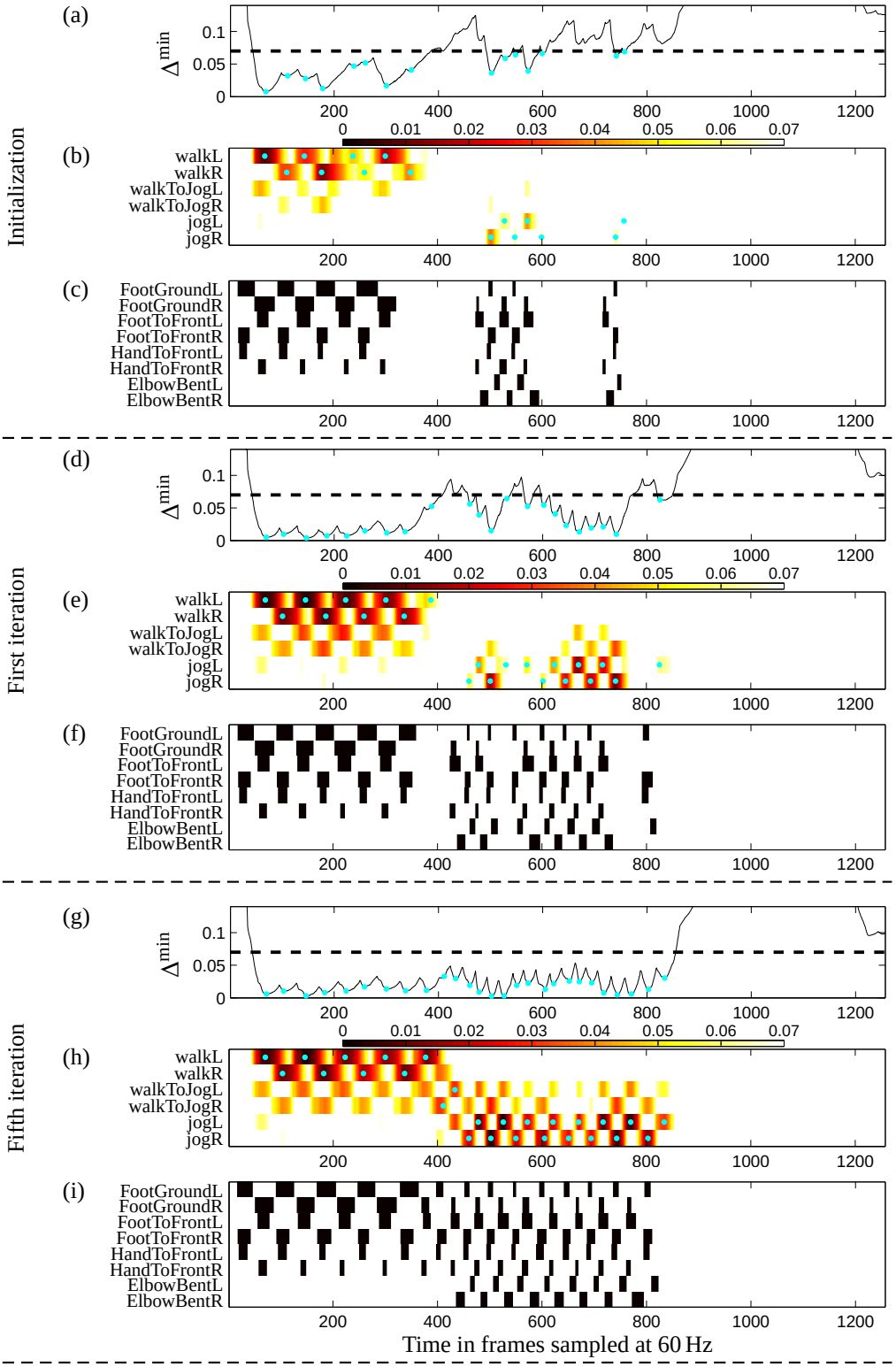


Figure 6.11. **(a)**: $\Delta^{\min}$ for the initial tracking result. Essential local minima are marked by a cyan dot. **(b)**: Corresponding distance functions Δ_C for six MTs shown in a color coded fashion. Values greater than $\tau = 0.7$ are drawn in white. **(c)**: Allocated tracking priors. **(d)-(f)**: Corresponding plots after the first iteration. **(g)-(i)**: Corresponding plots after the fifth iteration.

	no additional noise			+40 px image noise		
	∅	σ	max	∅	σ	max
Initial step	79.1	26.2	165.9	75.2	28.2	184.0
Iteration 1	51.8	18.5	134.1	65.0	24.2	144.4
Iteration 3	47.9	12.8	105.8	51.0	15.5	139.0

Table 6.1. Improvements of the tracking quality over various iterations. Average errors, standard deviations, and maximal errors (in millimeters) over all 1257 frames of the sequence are shown.

After extracting the relational features for each example motion at a sampling rate of 60 Hz, we computed a quantized motion template for each of the P motion classes and marked suitable regions in the quantized MTs as MT priors, see Figure 6.6 (a). In our scenario, we marked MT priors corresponding to 'left/right foot is on ground', 'left/right foot moves to front', 'left/right hand moves to front', and 'left/right elbow is bent'. Note that the set of motion templates along with the MT priors, which constitutes our database knowledge, is independent of the sequence to be tracked and has to be generated only once.

In the initialization step, tracking is performed without using any regularizing priors. The resulting tracked sequence is then locally classified according to the precomputed MTs. In our experiments, a quality threshold of $\tau = 0.07$ used in the allocation step (Section 6.4.1) turned out to be a robust choice. In Figure 6.11 (a), we show the resulting distance function $\Delta^{\min}$. Essential local minima below τ are marked by a cyan dot. Note that for the walking part of the benchmark sequence (frames 1 to 400), $\Delta^{\min}$ assumes lower values than for the jogging part (frames 400 to 800), which indicates that the walking part contains less tracking errors than the jogging part. Note also that for the balancing part (frames 830 to 1200), $\Delta^{\min}$ is far above τ revealing a strong difference to walking or jogging patterns. The function $\Delta^{\arg}$ assigns the essential local minima to appropriate motion categories, see Figure 6.11 (b). Furthermore, each motion subsequence induced by a local minimum is aligned to the corresponding MT. Based on these local alignments, suitable tracking priors are allocated for the next iteration, see Figure 6.11 (c). Figure 6.11(d)–(f) show the corresponding distance functions and allocated tracking priors after the first iteration. The minima in $\Delta^{\min}$ have already received a significant qualitative boost. Note that the walking cycles have been stabilized as visible in the lower distance values in Figure 6.11 (d) during the first 400 frames. Moreover, many more instances of the jogging motion are detected, see Figure 6.11 (e). As a result, more priors could be allocated for the subsequent iteration, Figure 6.11 (f). The distance functions and priors after the fifth iteration are shown in Figure 6.11(g)–(i). The detections have received an additional qualitative boost, and all occurring motions, including the transition between walking and jogging, have been annotated correctly. Furthermore, the occurrence of the different motion categories are revealed in a much more distinctive way in the fifth iteration, compare (h) and (e) of Figure 6.11. This all indicates a stabilization of the tracking procedure over the iterations.

We now discuss the actual improvements in the tracking results achieved by our novel iterative approach. Figure 6.12 shows representative poses overlaid with the tracking result (indicated by the yellow meshes) after the initial step, the first iteration, the third, and the fifth iteration. As seen, the initial tracking result contains various serious tracking errors such as a swap of legs or an incorrect angle in the elbow joint. These errors are corrected within few iterations. As also visible, only minor changes can be detected between the third and the fifth iteration. We

Figure 6.12. Improvements obtained by our iterative tracking procedure. The frames from left to right show examples of the walking, jogging, and balancing part (frames 210, 750 and 1170), iterations from top to bottom show results after the initialization, after the first, the third, and the fifth iteration. Several tracking errors (see arms and legs) are corrected.

show further comparisons between tracking results of the initialization and the fifth iteration in Figure 6.13. Here, the poses undergo a substantial qualitative improvement. Note that also the arm tracking error in the balancing part of the sequence (rightmost images) has been corrected although we did not assign any priors to this motion class. This can be explained as follows. In the initialization, the arm tracking error already appeared in the jogging motion that precedes the balancing. The tracking procedure could not recover from that error and continued to track the balancing with a wrong angle in the elbow joint. After a couple of iterations of our framework, the jogging motion has been corrected and the tracking starts the balancing motion with the correct arm configuration. With the correct initialization in the beginning of the balancing, the tracking

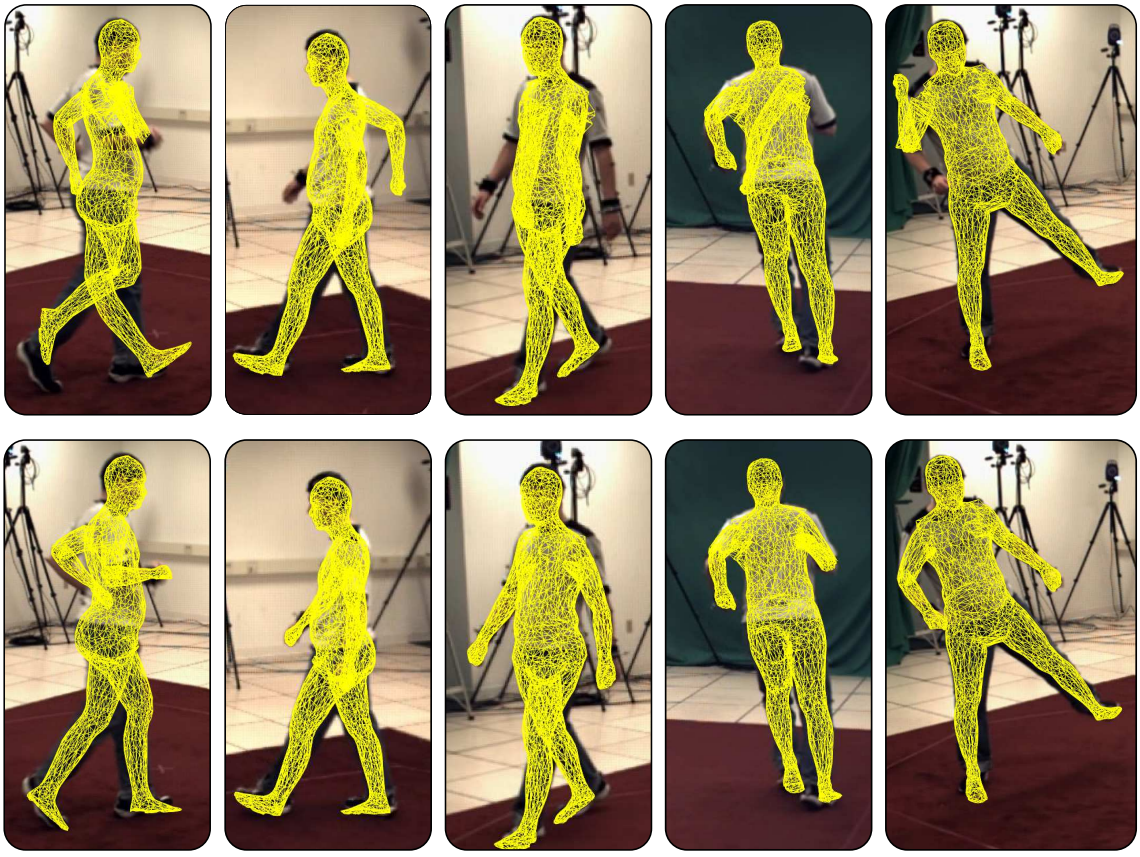


Figure 6.13. Example poses with tracking errors after the initialization (top row) and corrected poses after fifth iteration (bottom row).

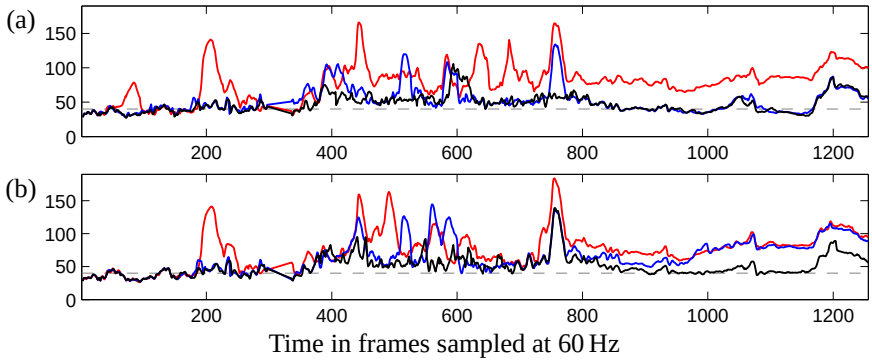


Figure 6.14. Framewise tracking error (in millimeters) after the initialization (red), the first (blue), and the third iteration (black). **(a)**: Without image noise. **(b)**: With Gaussian noise (40 pixels standard deviation) added to each frame.

procedure could correctly follow the motion of the actor throughout the last part of the sequence.

As quantitative evaluation, the absolute difference of the 3D joint positions of the tracking result and the ground truth positions are indicated by Table 6.1 and by Figure 6.14. These numbers were obtained by the automated evaluation system supplied by Brown University [Sigal *et al.*, 2010].

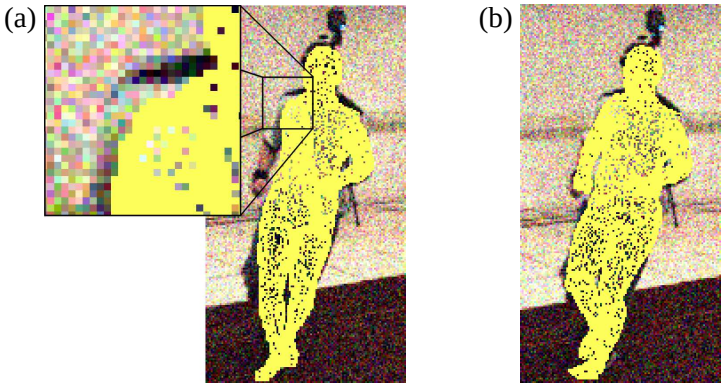


Figure 6.15. Gaussian noise has been added to each image. Pose overlay of frame 500 after the initial tracking **(a)** and after the third iteration **(b)**. During the iterations, several tracking errors (see right arm and both legs) are corrected.

During the iterations, the average error is reduced from 79 mm to 48 mm after few iterations, see Table 6.1. The significant improvements are also indicated by Figure 6.14 (a), which shows the framewise tracking errors of the initial tracking (red), the first iteration (blue), and the third iteration (black).

In another experiment, we added Gaussian noise to each frame such that each color channel of each pixel is the sum of the true pixel value and a random, Gaussian distributed noise value with standard deviation amounting to 15% of the full range of color values. Two example frames after the initialization and after the third iteration are shown in Figure 6.15. During the iterations, the average error dropped from 75 mm to 51 mm, see Table 6.1 and Figure 6.14 (b). Also, Figure 6.14 (b) shows the framewise tracking errors during the iterations. Note that due to the image noise the convergence is slower than in the first experiment. These results demonstrate the stabilizing effects achieved by our iterative tracking approach. Note that our framework requires that a sequence is tracked several times. Currently, our tracking implementation requires 7 s per frame resulting in 2.5 h run time for the entire 1257 frames. After tracking all frames, the annotation and allocation steps require only 15 s in total.

6.6 Conclusions

In this chapter, we introduced an iterative tracking approach that dynamically integrates motion priors retrieved from a database to stabilize tracking. Intuitively, our idea is to pursue a combined bottom-up and top-down strategy in the sense that we start with a rough initial tracking which is then improved by incorporating high-level motion cues. These motion cues are allocated based on an automated local annotation of the initial tracking result. In addition to stabilization, the local annotation also equips the tracked sequence with semantic motion class labels. By means of the HumanEVA-II benchmark, we showed that even simple motion priors lead to significant improvements in the tracking.

There are still limitations in our approach. In particular, the presence of strong tracking errors may lead to a confusion in the local annotation. Misallocated priors may then worsen the tracking error. As for future work, we plan to develop techniques that can cope with such situations, *e. g.*, by

integrating statistical confidence measures for the annotation and by simultaneously considering alternative motion priors.

Chapter 7

Evaluating Markerless Motion Tracking

In this chapter, we present a novel framework for automatically evaluating the quality of 3D tracking results obtained from marker-less motion tracking. In previous approaches, suitable reference information for evaluation was generated by means of an additional marker-based motion capture system. However, such systems are expensive and restricted to indoor-use. Moreover, the fact that markers are often visible in the video footage may interfere with the requirements of marker-less motion tracking. In contrast to previous approaches for evaluation, we use additional inertial sensors to generate suitable reference information. Inertial sensors are inexpensive, easy to operate, they can be hidden under clothing, and they impose comparatively weak additional constraints on the overall recording setup with regard to location, recording volume, and illumination. On the downside, acceleration and rate of turn data as directly obtained from such inertial systems are very local with respect to the temporal dimension and are therefore not directly suited for detecting the temporal extend of tracking errors. As our main contribution, we show how tracking results can be analyzed and evaluated on the basis of suitable limb orientations, which can be derived from 3D tracking results as well as from inertial sensors fixed on these limbs.

This chapter, which is based on our publication [Baak *et al.*, 2010], is organized as follows. We first give a motivation (Section 7.1) and discuss related work (Section 7.2). After summarizing basics on rotations and orientations (Section 7.3), we then describe how to obtain orientation data from the tracking result and from the inertial sensors (Section 7.4). We show that the tracking orientations and inertial orientations cannot be compared immediately. Being a special case of the hand-eye calibration problem as known in robotics, we introduce a robust optimization method for making this data comparable with only small calibration requirements (Section 7.5). Next, we present our evaluation framework and report on experiments conducted on the basis of 24 motion sequences using a marker-less tracking system in Section 7.6. Finally, we discuss the necessity of a calibration procedure in Section 7.7 and conclude in Section 7.8.

7.1 Motivation

Marker-less mocap with the objective to estimate 3D pose information of a human actor from image data is a traditional field of research in computer vision [Sidenbladh *et al.*, 2002; Bregler *et al.*, 2004; Brox *et al.*, 2006; Schmaltz *et al.*, 2007; Vlasic *et al.*, 2008]. Even though

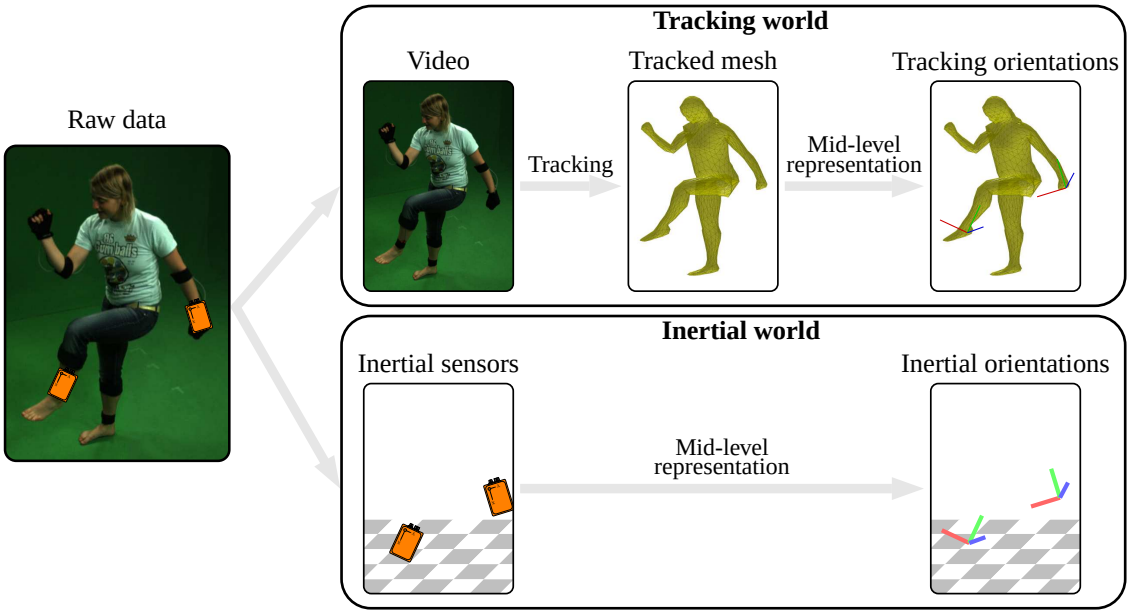


Figure 7.1. Raw data is recorded from video cameras and inertial sensors. In the tracking world, orientations are obtained by tracking. In the inertial world, orientations are derived from the sensor data.

motion capturing has been an active research field for more than two decades [Lowe, 1980; Moeslund and Granum, 2001; Moeslund *et al.*, 2006; Poppe, 2010; Moeslund *et al.*, 2011], recent tracking procedures still tend to produce many tracking errors. In particular, when dealing with involved settings like only few cameras, difficult lighting conditions, or challenging motion sequences, tracking errors are likely to occur.

In the process of developing and improving tracking algorithms, the analysis and evaluation of tracking results play a crucial role. In practice, the tracking results are often evaluated by manually inspecting the reconstructed 3D motion sequences or by looking at the differences between the 2D projections of these sequences and the original image data [Deutscher and Reid, 2005]. Obviously, such manual evaluations are tedious and prohibitive for large datasets. Furthermore, depending on the visual cues used for the analysis, such evaluations tend to be unreliable, subjective, and problematic in particular when one wants to compare the results of different tracking approaches. To automate the evaluation process and to make it objective, independent 3D ground truth information is needed in addition to the image sequences. So far, only few benchmark datasets with non-synthetic data such as [Sigal *et al.*, 2010; Tenorth *et al.*, 2009] are publicly available making a fully automated evaluation possible. Such benchmark datasets are generated by running a marker-based optical motion capturing system as a reference, which enables an accurate estimation of ground truth 3D positions of markers placed on the actor's body. However, the high cost of marker-based mocap systems, inconvenient setup as well as time consuming postprocessing of the obtained marker data may be among the reasons why publicly available benchmarks are rare. Also, many of the available marker-based mocap systems are vulnerable to bright lighting conditions and have to be run under low illumination. This is in discrepancy to the requirements of marker-less motion tracking, where one typically requires balanced and bright illumination. Furthermore, marker-based mocap systems typically pose additional constraints on the recording volume and environment (*e. g.*, indoor studios). As an alternative to recording human motions

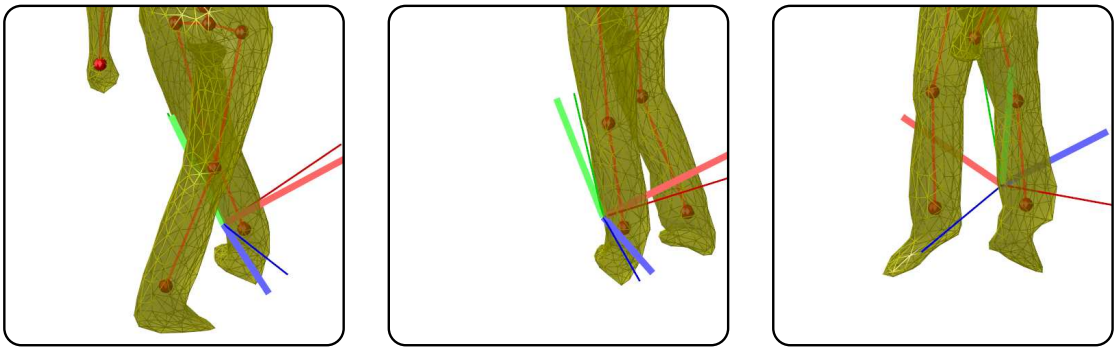


Figure 7.2. Snapshots of a tracking result at the given timestamps of the tracked sequence. Basis axes of the limb coordinate systems of the left lower leg are drawn, once extracted from the tracking result (thin axes, dark colors), and once from an enhanced inertial sensor (bold axes, light colors).

with real cameras, rendering software can be used to generate synthetic semi-realistic images, yielding a ground truth representation in a natural way [Agarwal and Triggs, 2004]. However, such image data as generated from standard rendering packages such as Poser [Software, 2012] still looks unrealistic. Moreover, in such video footage, low-level computer vision methods like feature detection might benefit from the synthetic data.

As the main contribution of this chapter, we present a novel approach for automatically analyzing and evaluating 3D tracking results using an inertial sensor-based system to generate suitable reference information. In the following, to clearly distinguish between these two types of data, we speak of the *tracking world* to refer to data derived from marker-less motion tracking, and we speak of the *inertial world* to refer to data derived from an inertial system, see also Figure 7.1. In contrast to marker-based reference systems, inertial sensors impose comparatively weak constraints on the overall recording setup with regard to location, recording volume, and illumination. Furthermore, inertial systems are relatively inexpensive as well as easy to operate and maintain. On the downside, the acceleration and rate of turn data obtained from such inertial systems cannot be directly compared with the tracking result which is given in form of 3D positional data or joint angles. There seem to be two obvious ways to make the inertial data (acceleration data, rate of turn data) comparable with the tracking results (3D positional data, joint angle data). Firstly, one could integrate the inertial data to obtain 3D positional data. This, however, is not practical since inertial data is prone to noise leading to very poor positional data when being integrated [Thong *et al.*, 2004]. Secondly, one could differentiate the 3D positional data of the tracking result to obtain velocities and accelerations. Such data, however, is very local in nature with respect to the temporal dimension. For example, local deviations in only few frames on the acceleration level may lead to long-lasting significant deviations on the positional level. This makes the evaluation process, as also shown by our experiments, very susceptible to short-time artifacts and unwanted outliers.

In this chapter, we introduce a novel inertial sensor-based evaluation framework, where we use orientation data as a common mid-level representation. The idea is as follows. In the tracking world, one obtains for each frame the estimated pose parameters of the underlying 3D model of the human actor. From this information, one can easily derive the 3D orientation of certain limbs (*e. g.*, the lower legs), which we refer to as *tracking orientations*. On the other hand, we use inertial sensors rigidly attached to some of the actor's limbs, from which we can also derive 3D orientations of the respective sensors referred to as *inertial orientations*. Now, in case of marker-

less motion tracking as well as the inertial measurement having worked correctly, the derived tracking orientations and inertial orientations should agree. However, as soon as tracking errors occur, the two types of orientations should exhibit significant differences, which can be easily captured. This fact is illustrated by Figure 7.2. As a central result presented in this chapter, we show that measuring the distance between the two types of limb orientations yields a simple and robust method for detecting the occurrence as well as the duration of tracking errors. In contrast to using velocities and accelerations, our orientation-based approach particularly suits this purpose since typical tracking errors stem from misconfigurations of certain limbs that effect the tracking result over an entire period of time rather than occurring at certain instances of time. Standard error metrics are based on Euclidean distances between positions of joints or markers which reflect positional errors fairly well. However, orientation errors, in particular misestimated rotations of cylindrical limbs, can lead to small deviations in the Euclidean distance metric. Moreover, these tracking errors are difficult to spot from visual cues. By contrast, our evaluation approach reveals twists of rotationally symmetric body parts by an orientation-based distance metric. As a further contribution, we introduce a robust calibration scheme that enables the direct comparison of the inertial and the tracking world.

7.2 Related Work

In this chapter, we show how data obtained from inertial sensors can be used to detect tracking errors. A natural complementary approach is to investigate how such data can be used to stabilize tracking. Such approaches have been investigated in Pons-Moll *et al.* [2010], where we use orientation data obtained from a small set of inertial sensors attached to the outer extremities in order to stabilize a local optimization-based marker-less motion tracking approach. In a followup work, Pons-Moll *et al.* [2011] use a similar setup of inertial sensors in a much more challenging outdoor tracking scenario, where we integrate inertial sensor data in a particle filter-based tracking framework. Within this framework, pose candidates are sampled directly from the space of inertial sensor-compatible poses. This space is efficiently generated by means of analytic inverse kinematics using the inertial sensor data. To account for uncertainties in the sensor data, a noise model that is based on the von Mises-Fisher distribution is employed.

To the best of our knowledge, this is the first approach for evaluating marker-less tracking using inertial sensors. However, there are several papers that deal with the estimation of the 3D position of a camera. In this context, inertial sensors attached attached to the camera are used to stabilize the estimates of the position. Works in this field have in common that the relative offset between both systems has to be obtained as a sub-task. Starting with works in robotics [Shiu and Ahmad, 1989; Park and Martin, 1994; Daniilidis, 1999; Strobl and Hirzinger, 2006], this task has also been approached in the vision community, *e. g.*, [Seo *et al.*, 2009]. Also, Hol *et al.* [2008] identifies the task with the gray-box problem in the area of system identification. Application scenarios include the estimation of an offset between a robot's end effector and a visual sensor attached to it [Shiu and Ahmad, 1989; Strobl and Hirzinger, 2006], or between an inertial sensor and a camera [Hol *et al.*, 2008; Seo *et al.*, 2009]. Analytically, both scenarios can be described by the *hand-eye calibration* equation $AX = XB$, to which we relate our work in Section 7.5.

For activity recognition, Kunze and Lukowicz [2008] evaluate how sensor displacement on a certain body limb influences recognition performance. They propose a heuristic for improving detec-

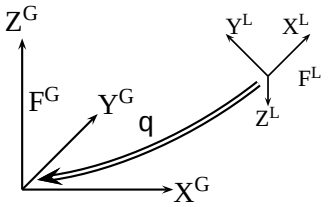


Figure 7.3. A global right-handed orthonormal coordinate system F^G and a local right-handed orthonormal coordinate system F^L related by a rotation q.

tion results when the exact sensor position on the limb is not known. Recently, de la Torre *et al.* from Carnegie Mellon University made a multi-modal activity database publicly available that also contains inertial data [de la Torre *et al.*, 2008]. In biomedics, Dejnabadi *et al.* [2006] use inertial sensors fixed on a lower leg to reconstruct the one-dimensional knee angle in the sagittal plane. They compare the reconstructed angles with ground truth angles computed by using a reference ultrasonic motion measurement system. To study biomechanical properties of outdoor activities, GPS information can be combined with inertial sensors [Brodie *et al.*, 2008]. Using a combination of inertial, magnetometer and GPS information, Foxlin [2005] shows that accurate position estimates for pedestrians can be obtained. By fusing the data modalities, positions are estimated even if GPS information is not available. However, as the authors report, the proposed techniques using so-called zero velocity updates are not applicable to sensors mounted on other limbs than the feet. Tao *et al.* [2007] reconstruct the motion of an arm model using inertial sensors. Slyper and Hodgins [2008] retrieve motions from a database using few inertial sensor signals to obtain a full body motion. Using only inertial and magnetic sensors, Roetenberg *et al.* [2006] show that a full body motion can be reconstructed. Having many sensors in a custom motion capture suit, a plausible motion model in everyday surroundings can be reconstructed [Vlasic *et al.*, 2007]. For home entertainment, inertial sensors have been used actively in the recent years, for example, in the Nintendo Wii game console [Nintendo, 2012]. User interfaces based on such sensors have been studied, *e. g.*, in [Shiratori and Hodgins, 2008].

7.3 Basics

Suppose a fixed global coordinate system F^G that is represented by a right-handed orthonormal basis (like all coordinate systems in this thesis). Furthermore, suppose a local coordinate system F^L that moves for a static observer in F^G. The relative orientation of F^L with respect to F^G can be modeled as a rotation, see Figure 7.3. Given the basis vectors X^L, Y^L, and $Z^L \in \mathbb{R}^{3\times1}$ of F^L in coordinates of F^G, the rotation is defined by a rotation matrix R by

$$R = \left(X^L, Y^L, Z^L\right). \tag{7.1}$$

In the following, we represent a rotation (or orientation) by a unit length quaternion $q \in \mathbb{R}^4, \|q\|_2 = 1$, which is a more compact representation than rotation matrices, see [Grassia, 1998; Shoemake, 1985]. The composition of two rotations represented by q_1 and q_2 is then given as the composition $q_2 \circ q_1$. Furthermore, the inverse rotation of q is given by the quaternion conjugate

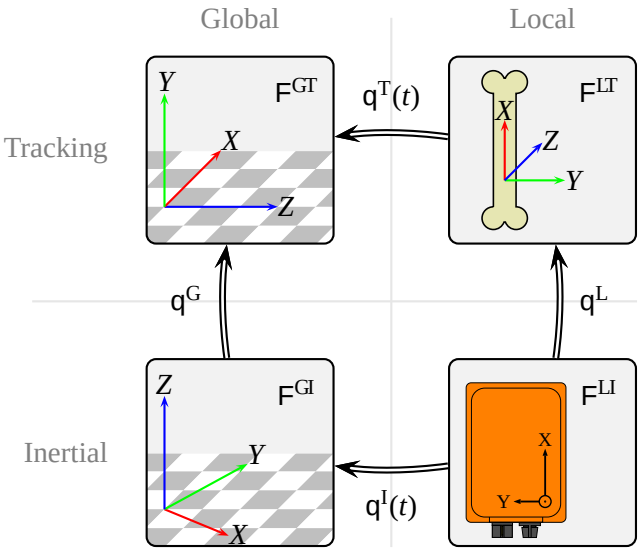


Figure 7.4. Relation between the orientation of coordinate systems.

$\overline{\mathsf{q}}$. Let $d_{\text{quat}} : \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}$ denote the following distance function:

$$d_{\text{quat}}(\mathsf{q}_1, \mathsf{q}_2) = \frac{360}{\pi} \cdot \arccos \left\| \langle \mathsf{q}_1, \mathsf{q}_2 \rangle \right\|_2 , \tag{7.2}$$

which expresses the angle in degrees between the rotations defined by q_1 and q_2, see [Huynh, 2009] for a proof. We use the notation

$$\mathsf{F}^{\mathrm{A}} \xRightarrow{\mathsf{q}} \mathsf{F}^{\mathrm{B}} \tag{7.3}$$

to describe a transformation of coordinate systems F^{A} to F^{B} using the rotation defined by q. For time dependent quantities we append a discrete frame index (t) and assume that co-occurring quantities are subject to the same sampling rate.

7.4 Obtaining Two Types of Orientation Data

We now describe how to obtain orientation data in the inertial as well as in the tracking world. In the *inertial world*, as described in [Harada *et al.*, 2007], an orientation estimation device can be used to measure its orientation in a static global coordinate system $\mathsf{F}^{\mathrm{GI}} = (\mathrm{X}^{\mathrm{GI}}, \mathrm{Y}^{\mathrm{GI}}, \mathrm{Z}^{\mathrm{GI}})$. In this coordinate system, the Z^{GI} axis points to the negative gravity direction, the X^{GI} direction is the orthogonalized direction of the magnetic North, and Y^{GI} is chosen to form an orthonormal right-handed basis. Measurements of accelerometers, gyroscopes, and a magnetic field sensor, as described in Appendix A, are fused in a Kalman filter method, which provides drift free estimates of the sensor's orientation $\mathsf{q}^{\mathrm{I}}(t)$. This orientation maps from the sensor's local coordinate system F^{LI} to F^{GI}, see Figure 7.4. We refer to $\mathsf{q}^{\mathrm{I}}(t)$ with the term *inertial orientation*. In our experiments, we use an orientation estimation device MTx provided by Xsens [Xsens, 2012].

In the *tracking world*, a global coordinate system F^{GT} is defined by camera calibration. Tracking results are typically given by a mesh-based surface representation for every frame in coordinates

of F^{GT}. To obtain the orientation of a certain limb in the surface mesh, one needs to define a local coordinate system F^{LT} that is rigidly attached to the limb. By selecting three non-collinear vertices of the limb, an orthonormal basis of F^{LT} can be build. To ensure that the coordinate system is well defined, one has to claim one-to-one vertex correspondence throughout the entire motion sequence. In many cases, tracking results are given as joint angles of a skeletal kinematic chain which drives the animation of the mesh surface. In this case, without having to resort to the vertices of the mesh surface, a local coordinate system for every limb can be defined by forward kinematics [Murray *et al.*, 1994]. This way, a *tracking orientation* $\mathsf{q}^{\mathrm{T}}(t)$ can be obtained, see Figure 7.4.

In order to make $\mathsf{q}^{\mathrm{I}}(t)$ and $\mathsf{q}^{\mathrm{T}}(t)$ comparable, one needs a correspondence between the global coordinate systems F^{GI} and F^{GT} as well as between the two local coordinate systems F^{LI} and F^{LT}. These correspondences, however, are generally not known. The global inertial coordinate system F^{GI} is defined by physical quantities, whereas F^{GT} is defined by an arbitrary placement of a calibration cube in the recording volume. Let q^{G} denote the resulting offset, see Figure 7.4. Furthermore, the local coordinate system F^{LI} is defined by the placement of the sensor on a limb of the human actor, whereas F^{LT} is defined either by mesh vertices or by means of a kinematic chain. Let q^{L} denote the resulting offset. The estimation of q^{G} and q^{L} is referred to as calibration, which is a tedious and error-prone task when done manually. Therefore, automated calibration is an important concern that we deal with in Section 7.5.

7.5 Calibration and Error Measure

In this section, we present a robust and efficient solution for the calibration problem, namely how q^{L} and q^{G} can be obtained. We show that the described problem is closely related to the prominent *hand-eye calibration* task in robotics [Tsai and Lenz, 1988]. The orientation $\mathsf{q}^{\mathrm{I}}(t)$ can be described by two distinct compositions of rotations in the diagram of Figure 7.4, once with tracking and once with inertial orientations:

$$\mathsf{F}^{\mathrm{LI}} \xrightarrow{\mathsf{q}^{\mathrm{L}}} \mathsf{F}^{\mathrm{LT}} \xrightarrow{\mathsf{q}^{\mathrm{T}}(t)} \mathsf{F}^{\mathrm{GT}} \xrightarrow{\overline{\mathsf{q}^{\mathrm{G}}}} \mathsf{F}^{\mathrm{GI}} \qquad \mathsf{F}^{\mathrm{LI}} \xrightarrow{\mathsf{q}^{\mathrm{I}}(t)} \mathsf{F}^{\mathrm{GI}} \tag{7.4}$$

With quaternion algebra, this equality can be expressed as

$$\mathsf{q}^{\mathrm{I}}(t) = \overline{\mathsf{q}^{\mathrm{G}}} \circ \mathsf{q}^{\mathrm{T}}(t) \circ \mathsf{q}^{\mathrm{L}} \ . \tag{7.5}$$

Now, we can express the rotation that is needed to transform F^{LI} at frame s to F^{LI} at frame t. In Figure 7.4, there are two distinct compositions of rotations, starting at t in F^{LI} and ending at s in F^{LI}:

$$\mathsf{F}^{\mathrm{LI}} \xrightarrow{\mathsf{q}^{\mathrm{L}}} \mathsf{F}^{\mathrm{LT}} \xrightarrow{\mathsf{q}^{\mathrm{T}}(t)} \mathsf{F}^{\mathrm{GT}} \xrightarrow{\overline{\mathsf{q}^{\mathrm{G}}}} \mathsf{F}^{\mathrm{GI}} \xrightarrow{\mathsf{q}^{\mathrm{G}}} \mathsf{F}^{\mathrm{GT}} \xrightarrow{\overline{\mathsf{q}^{\mathrm{T}}(s)}} \mathsf{F}^{\mathrm{LT}} \xrightarrow{\overline{\mathsf{q}^{\mathrm{L}}}} \mathsf{F}^{\mathrm{LI}} \qquad \mathsf{F}^{\mathrm{LI}} \xrightarrow{\mathsf{q}^{\mathrm{I}}(t)} \mathsf{F}^{\mathrm{GI}} \xrightarrow{\overline{\mathsf{q}^{\mathrm{I}}(s)}} \mathsf{F}^{\mathrm{LI}} \tag{7.6}$$

Here, tracking orientations in the upper path and inertial orientations in the lower path are used. The equality of the paths can be expressed with quaternion algebra, where the offset q^{G} cancels out:

$$\overline{\mathsf{q}^{\mathrm{I}}(s)} \circ \mathsf{q}^{\mathrm{I}}(t) = \overline{\mathsf{q}^{\mathrm{L}}} \circ \overline{\mathsf{q}^{\mathrm{T}}(s)} \circ \mathsf{q}^{\mathrm{T}}(t) \circ \mathsf{q}^{\mathrm{L}} \ . \tag{7.7}$$

Substituting $\mathsf{q}^{\mathrm{A}} := \overline{\mathsf{q}^{\mathrm{I}}(s)} \circ \mathsf{q}^{\mathrm{I}}(t)$, $\mathsf{q}^{\mathrm{B}} := \overline{\mathsf{q}^{\mathrm{T}}(s)} \circ \mathsf{q}^{\mathrm{T}}(t)$, and $\mathsf{q}^{\mathrm{X}} := \overline{\mathsf{q}^{\mathrm{L}}}$, we get

$$\mathsf{q}^{\mathrm{A}} \circ \mathsf{q}^{\mathrm{X}} = \mathsf{q}^{\mathrm{X}} \circ \mathsf{q}^{\mathrm{B}} \,. \tag{7.8}$$

In robotics, a more general equation of the same form, in which homogeneous transformations are used instead of sole rotations, describes the hand-eye calibration problem. Manifold solutions to this problem have been published, see, *e. g.*, Strobl and Hirzinger [2006] and references therein. Unique solutions can be found as soon as two measurements of q^{A} and q^{B} are available. However, in the presence of noise, an approximate solution using many measurements is preferable to diminish the influence of measurement errors. Therefore, we suggest to use $N \gg 2$ measurements based on a calibration tracking result. The solution of

$$\arg\min_{\mathsf{q}^{\mathrm{X}}} \sum_{n\in[1:N]} \|\mathsf{q}_n^{\mathrm{A}} \circ \mathsf{q}^{\mathrm{X}} - \mathsf{q}^{\mathrm{X}} \circ \mathsf{q}_n^{\mathrm{B}}\| \tag{7.9}$$

yields a best approximate solution under the Euclidean norm. Park and Martin [1994] present an efficient and easy to implement solution for this subproblem of the hand-eye calibration using exponential coordinates, which we adapt for our needs. Denoting the real part of a quaternion q with q_{w} and the imaginary part with $\mathsf{q}_{\mathrm{xyz}}$, the quaternion logarithm is defined as

$$\log(\mathsf{q}) := 2\arccos(\mathsf{q}_{\mathrm{w}}) \frac{\mathsf{q}_{\mathrm{xyz}}}{\|\mathsf{q}_{\mathrm{xyz}}\|} \in \mathbb{R}^{3\times 1} \,. \tag{7.10}$$

Intuitively, $\log(\mathsf{q})$ extracts a representation for rotations in which the direction of $\log(\mathsf{q})$ denotes the axis and the length denotes the angle of the rotation. Then, we define the matrix $M \in \mathbb{R}^{3\times 3}$ as

$$\alpha_n := \log(\mathsf{q}_n^{\mathrm{A}}) \tag{7.11}$$

$$\beta_n := \log(\mathsf{q}_n^{\mathrm{B}}) \tag{7.12}$$

$$M := \sum_{n\in[1:N]} \beta_n \cdot \mathrm{trans}(\alpha_n), \tag{7.13}$$

where $\mathrm{trans}(\alpha)$ is the transpose of α. The solution to Equation (7.9) as a rotation matrix is given by

$$\mathrm{M}^X := (\mathrm{trans}(\mathrm{M}) \cdot \mathrm{M})^{-1/2} \cdot \mathrm{trans}(\mathrm{M}) \,. \tag{7.14}$$

To convert M^X to the quaternion q^{X}, we refer to [Shoemake, 1985]. Using this formulation, the offset q^{L} can be found efficiently from Equation (7.7). Analogously, one can also regard the dual equation

$$\mathsf{q}^{\mathrm{I}}(s) \circ \overline{\mathsf{q}^{\mathrm{I}}(t)} \;\; = \overline{\mathsf{q}^{\mathrm{G}}} \circ \mathsf{q}^{\mathrm{T}}(s) \circ \overline{\mathsf{q}^{\mathrm{T}}(t)} \circ \mathsf{q}^{\mathrm{G}} \tag{7.15}$$

to find a solution for the global offset q^{G}. After alignment, we use a thresholding strategy based on Equation (7.5) to detect whether a tracking error in frame t occurs by evaluating

$$d_{\mathrm{quat}}\left(\mathsf{q}^{\mathrm{I}}(t), \overline{\mathsf{q}^{\mathrm{G}}} \circ \mathsf{q}^{\mathrm{T}}(t) \circ \mathsf{q}^{\mathrm{L}}\right) > \tau \,, \tag{7.16}$$

where the threshold parameter τ can be set to trade-off between the number of correctly detected tracking errors and false detections, see Section 7.6.4.

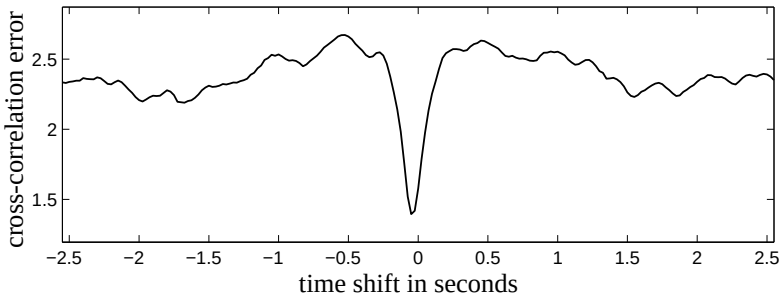


Figure 7.5. Cross-correlation error on absolute acceleration data of the inertial and the tracking world for the sequence used in Figure 7.2. The sharply peaked minimum reveals the temporal offset used to synchronize the inertial and the tracking data.

7.6 Experiments

In this section, we give an overview of the proposed pipeline for the detection of tracking errors. After describing the acquisition of data (Section 7.6.1), we sketch the tracking procedure used for obtaining the tracking orientations (Section 7.6.2), and describe our calibration method (Section 7.6.3). Finally, we evaluate the proposed framework in detail on a large dataset of tracked motions (Section 7.6.4).

7.6.1 Data Acquisition

For our experiments, we recorded image sequences using eight calibrated and temporally synchronized cameras as well as inertial data for five different body parts using MTx devices manufactured by the company Xsens [2012]. By systematically recording two human actors performing various actions including motion classes such as walk, sit down, stand up, hop, jump, cartwheel, rotate arms, and throw, we obtained 24 takes with a total length of 14 131 frames or 353 seconds of data.

We selected different body points where we fixed the sensors. Firstly, to represent body limbs that are influenced by a small number of degrees of freedom, we selected the lower legs as mounting position. Secondly, to represent body limbs that are influenced by a larger number of degrees of freedom, we selected the hands as mounting positions. Thirdly, the fifth sensor was fixed on the upper torso. This way, we fixed the sensors at points corresponding to different kinematic levels of the skeleton. Finally, we needed to temporally align the inertial data and the video data. To this end, we first obtained absolute acceleration data from both worlds. Then, we used a cross-correlation method in order to obtain a robust estimate of the temporal offset between both data streams. Note that absolute acceleration data is invariant to the spatial calibration of both systems and therefore suited to derive a temporal offset. However, the acceleration data obtained from the tracking might be impaired by tracking errors. In our sequences, such tracking errors occurred only temporally local and thus did not influence the accuracy of a constant temporal offset estimated over a whole tracked sequence. As an example, Figure 7.5 shows the cross-correlation error for the sequence which is also used in Figure 7.2. Despite of the strong tracking error in the leg, the cross correlation measure reveals the temporal offset with a sharp peak. All data streams were sampled at 40 Hz. For research purposes, we made the whole dataset publicly available in [Pons-Moll *et al.*, 2008].

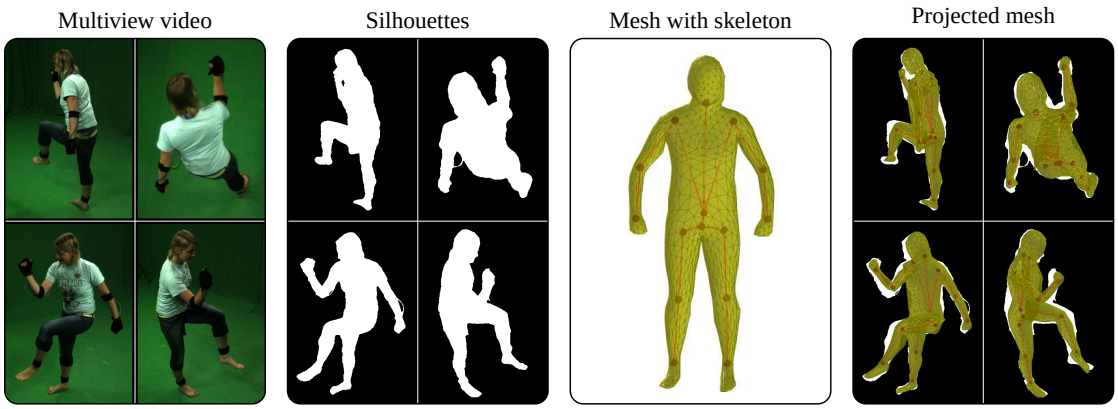


Figure 7.6. Overview of the silhouette-based tracking procedure [Rosenhahn *et al.*, 2007a]. Starting from multiview video, silhouettes are extracted by chroma keying. A skeleton-enhanced 3D model of the actor is then fit to the silhouettes based on optimization of joint angle parameters as well as the root orientation and translation.

7.6.2 Tracking

Our framework is thought for evaluating tracking results independent of the specific tracking method. In our experiments, we exemplarily used a tracking algorithm similar to [Rosenhahn *et al.*, 2007a], see Figure 7.6. First, we extract silhouettes from captured images by chroma keying. We generate a surface mesh of the actor using a 3D body scanner and fit a skeletal kinematic chain to it. Then, the surface deformation of the mesh is defined by joint angle parameters as well as root orientation and translation of the kinematic chain. Using a local optimization-based approach, pose configuration parameters are determined to minimize the distance between the transformed 3D mesh projected back onto the 2D images and the silhouettes. This way, we generated tracking results for all 24 takes, which are then evaluated in our experiments.

7.6.3 Calibration

To compare orientation data from different worlds, the global coordinate system offset q^{G} and local offsets $\mathsf{q}_s^{\mathrm{L}}$ have to be estimated for each of the sensors $s \in [1:5]$ as explained in Section 7.5. For this purpose, we propose a solution using a calibration take. There are only two requirements for the calibration take that are easy to meet in practice. Firstly, the orientations of the limbs should be represented reasonably well by the tracking result. Secondly, to obtain unambiguous offsets, the take should contain poses in different orientations. To this end, we selected a take containing relatively slow motions which are rather easy to track. Since the offset for the local and global orientations are constant for each actor, local tracking errors do not have a significant impact on the final estimations.

7.6.4 Automatic Evaluation

In our experiments, we resort to a studio setup for the multiview recordings. For tracking outdoor recordings, a more advanced tracking method than the one we used would be required. However,

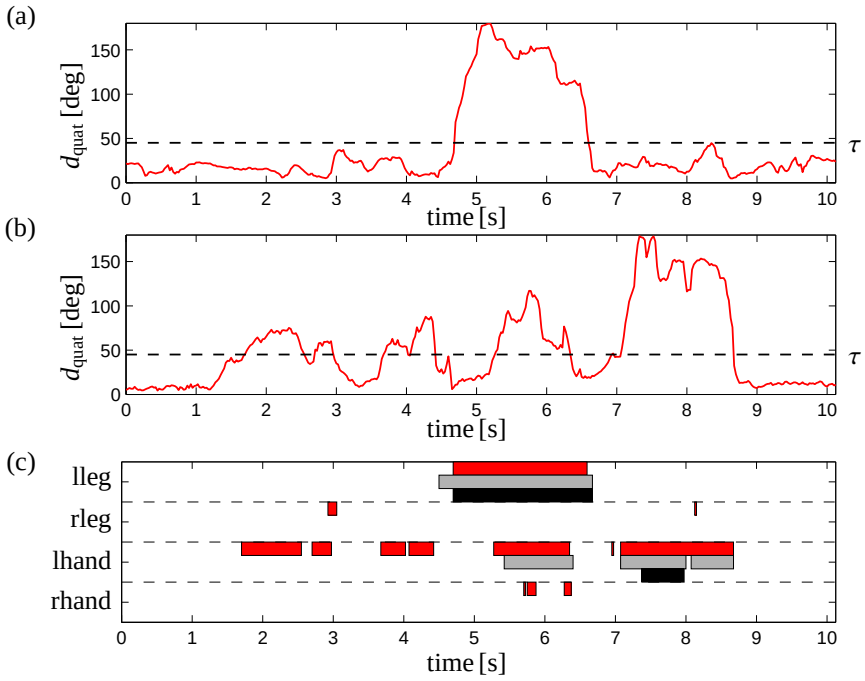


Figure 7.7. Distance measure d_{quat} and threshold τ used to reveal tracking errors in an example tracking sequence for **(a)** the left leg and **(b)** the left hand. Using these curves, automatically detected tracking errors are marked by red boxes, see **(c)**. Manual annotations conducted by two subjects are marked with gray and black boxes, respectively.

our evaluation concepts transfer without modification to more advanced tracking scenarios. In particular, inertial sensors do not depend on a studio setup and are applicable for outdoor settings.

To automatically detect tracking errors, we evaluate the distance measure of Equation (7.16) for every limb and frame. In this equation, the calibrated offsets q^{L} and q^{G} are used to make the inertial orientations as measured by an inertial sensor and the tracking orientations as estimated from the tracking result comparable. In Figure 7.7, the quaternion distance functions for (a) the left leg and (b) the left hand are drawn. In Figure 7.7 (c), the detected tracking errors for the body segments are marked with red boxes, which we refer to as *automatic annotations*. In our experiments, we chose the quality threshold $\tau = 45°$ (dashed line), which turned out to be a suitable trade-off between error detection capability and robustness. The threshold selection will be discussed later, see also Figure 7.12.

Since we aim to assess the quality of our procedure for tracking error detection, we asked two people (hereafter referred to as A1 and A2) of our working group to manually annotate each frame of the tracking results according to tracking errors in the limbs, see Figure 7.7 (c). We refer to these annotations as *manual annotations*, the gray and black rectangles show the manual annotations of A1 and A2, respectively. For this task, the annotators were provided with the original multiview videos as well as with a tool to view the reconstructed 3D mesh from arbitrary viewpoints. As it turned out, both annotators did not notice any tracking errors in the torso. This is also reflected by our distance measure, which stays well within a small range of 14.4° mean and 7.7° standard deviation. Therefore, we only regard the other four sensors in our evaluation below.

In Figure 7.7 (a), high distance values correspond to a tracking error in the left leg. The correspond-

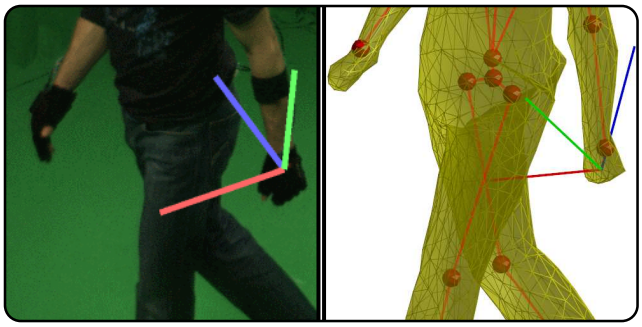


Figure 7.8. Left: Calibrated inertial orientation for point in time 4.2 s of the example tracking sequence. **Right:** Tracking orientation. A tracking error can be detected means of orientation distances.

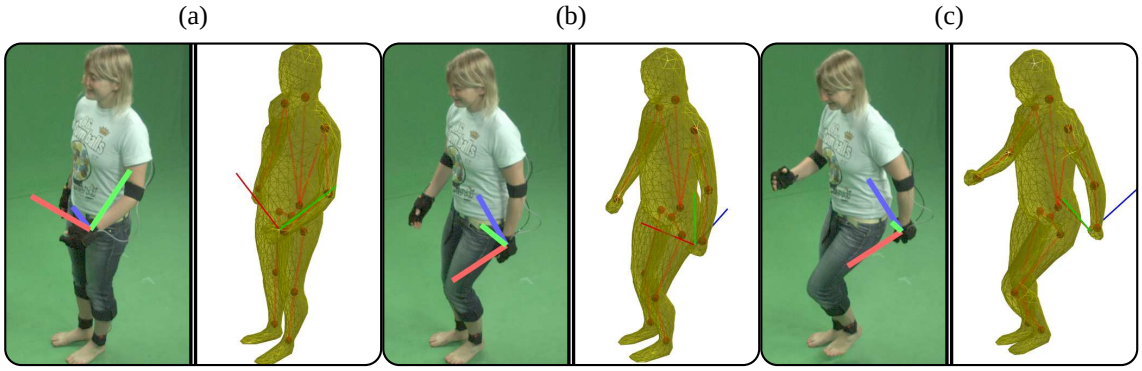


Figure 7.9. Tracking error in the left hand for a running motion.

ing motion sequence is also indicated by Figure 7.2. Here, both annotators as well as the automatic annotations agree. However, we found that the automatic annotation procedure generally marked more frames as erroneous than the annotators did. For an annotator A, one needs to distinguish between false positives (automatic annotations, where A has not seen an error), and false negatives (A has seen a tracking error, but the automatic annotation procedure did not detected it). In fact, by examining the false positives in more detail, we found that they often correspond to subtle tracking errors that are hardly visible when looking at the reconstructed mesh. For instance, in the example sequence at 4.2 s, the procedure has marked a tracking error in the left hand. Figure 7.8 shows that the palm faces the actor's hip, represented by the blue axis of the calibrated inertial orientation. In the 3D reconstruction (right), however, the palm faces backwards. A similar tracking error can be observed in Figure 7.9. Here, the actress performed a run-on-place-motion. In the tracking, the orientation of the hand slowly drifted towards a false orientation which becomes apparent when looking at the differences of the inertial and the tracked orientations in Figure 7.9 (b) and (c).

At this point we emphasize that such a tracking error might appear subtle and unimportant, because it is hardly noticeable in the visual appearance of an untextured 3D mesh. However, when using a textured mesh in a rendered scene, this kind of orientation error will lead to unwanted and visually annoying artifacts. Such an error is not well reflected by previous evaluation metrics like the ones presented in [Sigal *et al.*, 2010; Bălan *et al.*, 2005]. In these metrics, ground truth marker trajectories are compared to trajectories extracted from the 3D mesh, where such an error results in only negligible differences on the positional level. Yielding similar results, other works evaluate joint location errors in the 2D image domain [Lee and Nevatia, 2009]. With the proposed method

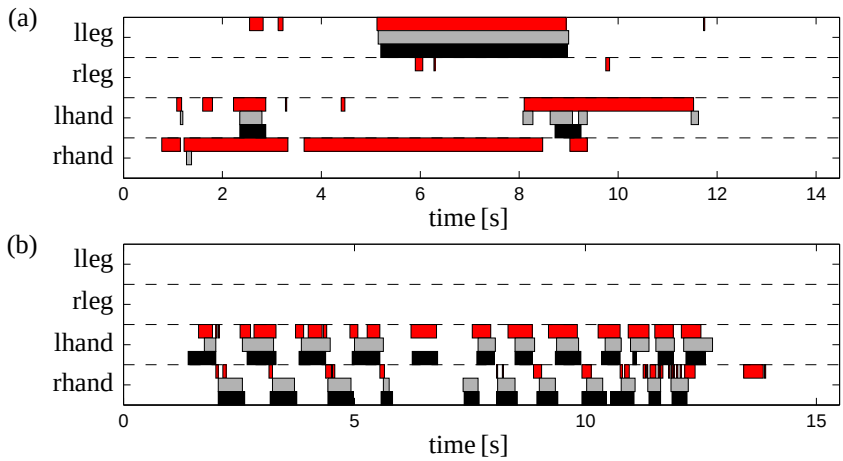


Figure 7.10. Comparison of automatic (red) and manual annotations (gray, black) of **(a)** cartwheels and **(b)** locomotion.

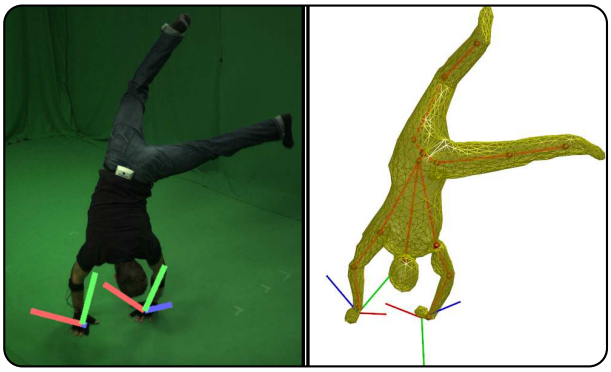


Figure 7.11. In a cartwheel sequence, both hands show tracking errors.

	A1		A2	
	P	R	P	R
Legs	0.65	0.91	0.64	0.96
Hands	0.36	0.78	0.45	0.69

Table 7.1. Precision and recall values for $\tau = 45°$.

based on orientation data, however, this error can be revealed.

Figure 7.10 (a) shows the annotations of a take containing cartwheels. As an example for a false positive, consider the point in time 2.5 s. Both annotators agreed on a tracking error in the actor's left hand. In Figure 7.11, this error is visible even without the additionally drawn inertial orientations (left) and tracking orientations (right), since the left hand points into the wrong direction. By contrast, the tracking error in the right hand is much less visually apparent. In fact, the orientation of the whole arm is estimated incorrectly, coming from a misconfiguration in the shoulder joint. This error is revealed by the orientation error of the end-effector in the kinematic chain. Again, this error could not be captured well with traditional metrics.

To evaluate the accuracy on all takes, we calculated precision and recall values, taking each of the manual annotations as baseline. We separately report on the values for the hands and the legs

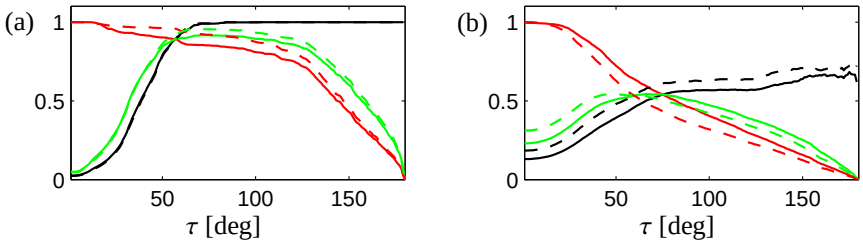


Figure 7.12. Precision (black), recall (red) and F-measure (green) over variations of τ for **(a)** legs and **(b)** hands. Solid and dashed lines represent values belonging to A1 and A2, respectively.

representing two kinematic levels, see Table 7.1. For both the legs and hands, the automatic annotations show relatively small precision values of around 0.65 and 0.36, respectively. As discussed above, the low precision is coming from a large amount of automatically detected tracking errors that the annotators did not see. This shows that the manual evaluation of tracking results is not sufficient to find all tracking errors. By contrast, the recall values for the legs are quite high, showing that the automatic annotation procedure detected nearly all manually annotated errors. The hands, however, have a lower recall in comparison to the legs. Note that this is mainly due to the per-frame annotations we pursued. In case of short tracking errors that mainly occur in the tracking results of the hands, small misalignments in the results lead to low recall values, see Figure 7.10 (b). Although most of the boxes coming from manual annotations have a certain overlap with an automatic annotation, the automatic annotations achieve a low recall. Here, segment-based rather than frame-based values may be better suited.

As for quantitative evaluations, a combined recording setup with a marker-based optical motion capture system would have been beneficial. In our setup we did not have a marker-based reference system at hand. Different sources of errors like sensor noise and bias, calibration errors, sensors getting out of place, or errors due to the approximation of the human body with a rigged surface mesh are thus difficult to quantify. However, our experiments show that the influence of all sources of noise are small. For example, the distance measure of the upper torso sensor over all 14 131 frames of our evaluation data stays within a small error range with a mean of 14.4° and a standard deviation of 7.7°, and the manual inspection shows that there are no noticeable tracking errors in the torso region. This observation suggests that the overall noise lies within this small order of magnitude. In particular, it follows that the accuracy of the obtained inertial orientations is high enough for a quantitative evaluation of tracking results. Moreover, our experiments show that the proposed distance metric is able to cover most of the manually observed tracking errors, which is supported by high recall values. Finally, a manual inspection showed that the false positive detections correspond to tracking errors that were difficult to perceive for the manual annotators. This supports the statement that our orientation-based distance measure is well suited for detecting tracking errors.

To evaluate the influence of the threshold parameter τ, we computed precision, recall, and F-measure for variations of τ, see Figure 7.12. Selecting a low τ leads to a high recall, since many parts of the evaluated takes are annotated. However, also many parts unrelated to tracking errors are annotated, yielding a low precision. Our final choice of $\tau = 45°$ is motivated by the request of having high recall values without having too many false detections.

As described in Section 7.4, orientation data from the inertial world is obtained by combining

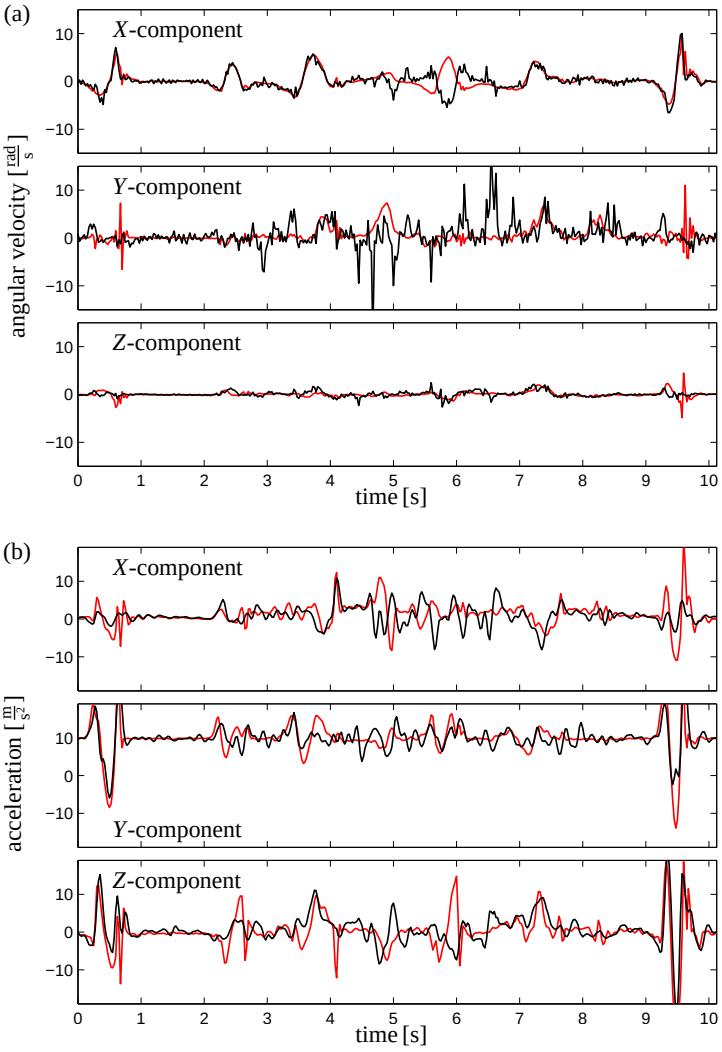


Figure 7.13. Calibrated inertial data (red) and tracking data (black) for the left leg of the example sequence used in Figure 7.7 (a). **(a)**: Local rate of turn data. **(b)**: Local acceleration data. The presence and duration of tracking errors is difficult to determine from this data.

different sensors. These sensors naturally provide 3D acceleration and rate of turn data, as further explained in Appendix A. Thus, a method comparing these types of data with corresponding data generated from the tracking world could also reveal tracking errors. In practice, however, this does not work well. In Figure 7.13, we show a comparison of the rate of turn data (a) and the acceleration data (b) corresponding to the left leg for the example tracking sequence also used in Figure 7.7 (a). The severe tracking error in the left leg occurred from 4.6 s to 6.6 s. During this period of time, the X- and Y-components of the angular velocity show some deviations, and also in the acceleration some differences between the inertial and the tracking data can be revealed. However, on the basis of this data, it is difficult to isolate the tracking error from spurious detections coming from noise in the signals. Three more properties make this kind of data not suited for our task of tracking error detection. Firstly, these quantities are very local in nature with respect to the temporal dimension. This makes it hard to detect the duration as well as the temporal starting and ending point of an error. Secondly, filtering techniques necessary to determine meaningful

acceleration and rate of turn data may not only suppress the sensor noise but may also smooth out peaks coming from actual tracking errors. Thirdly, slowly moving limbs generate low amplitudes in these quantities, which makes it hard, if not infeasible, to detect errors for such motions. With orientation data, as shown in this chapter, these considerations do not hold, thus yielding a robust procedure for tracking error detection.

7.7 Discussion

As one contribution of this chapter, we described a calibration framework that was used to make orientation data from both worlds comparable. In this section, we motivate why the calibration step is important and needs to be performed prior to evaluating distance measures such as d_{quat} on orientation data from the inertial and the tracking world. To this end, we first define a distance curve $\Delta_{\mathsf{q}_a,\mathsf{q}_b} : [1 : T] \mapsto [0°, 360°]$ based on Equation (7.16), which is parametrized by two calibration offsets q_a and q_b:

$$\Delta_{\mathsf{q}_a,\mathsf{q}_b}(t) := d_{\text{quat}}\left(\mathsf{q}^{\text{I}}(t), \mathsf{q}_b \circ \mathsf{q}^{\text{T}}(t) \circ \mathsf{q}_a\right). \tag{7.17}$$

For a given sequence of inertial orientations $\mathsf{q}^{\text{I}}(t)$ and a sequence of corresponding tracking orientations $\mathsf{q}^{\text{T}}(t)$, we can use Equation (7.17) to evaluate the effect of wrong calibration offsets by shifting q_a and q_b away from the true offsets. It is tempting to think that a modification of the offsets q_a and q_b just leads to a shift in the distance function by a fixed $\delta \in \mathbb{R}$. If that was true in general, then we could avoid the calibration by relying on the assumption that most frames of the sequence were tracked correctly. In that case, the most frequently occurring distance value δ^* would represent the correctly tracked frames. Then, the distance curve could be shifted by δ^* in order to obtain the correctly calibrated distance curve. However, modifying the offsets in Equation (7.17) leads to a non-linear distortion of the distance curve due to the non-linearity of the quaternion multiplication and the arccos function within Δ, see Equation (7.2).

In the following, we will show the effect of missing or wrong calibration offsets by means of an explicit example. To this end, we first generate a synthetic sequence of $T = 640$ unit quaternions $\mathsf{q}^{\text{I}}(t)$. Then, to simulate the tracked sequence $\mathsf{q}^{\text{T}}(t)$, we copy $\mathsf{q}^{\text{I}}(t)$ and introduce two tracking errors from frames 50 to 200 and from frames 400 to 600 by modifying $\mathsf{q}^{\text{T}}(t)$. Now, $\mathsf{q}^{\text{I}}(t)$ represents the measured inertial orientations and $\mathsf{q}^{\text{T}}(t)$ represents the tracked orientations. For these streams of orientations, the correct offsets are both the quaternion identity q^{Id}. The corresponding ground truth distance curve is obtained by evaluating $\Delta_{\mathsf{q}^{\text{Id}},\mathsf{q}^{\text{Id}}}$, see Figure 7.14 (a). In this curve, the temporal extend and the magnitude of the errors are correctly represented. Now, suppose that only one offset is known and the other offset is unknown or wrong. This setting is simulated by evaluating $\Delta_{\mathsf{q}^{\text{Id}},\mathsf{q}_b}$ for a fixed quaternion q_b, see Figure 7.14 (b). As for the correctly tracked regions, one can see by comparing with Figure 7.14 (a) that the error curves are shifted by a certain amount of degrees. However, in the frames containing tracking errors, two major distortions are visible. Firstly, the magnitude of the tracking errors are no longer represented correctly. For example, the ground truth error in frame 200 is 70°. In the same frame in Figure 7.14 (b), the difference to the level of correctly tracked frames amounts to only 40°. Secondly, some frames with tracking errors cannot be distinguished from correctly tracked frames. For example, consider the region around frames 485. Here, the same error 100° as for the correctly tracked frames is computed, rendering the detection of a tracking error in these frames impossible. The situation becomes even more in-

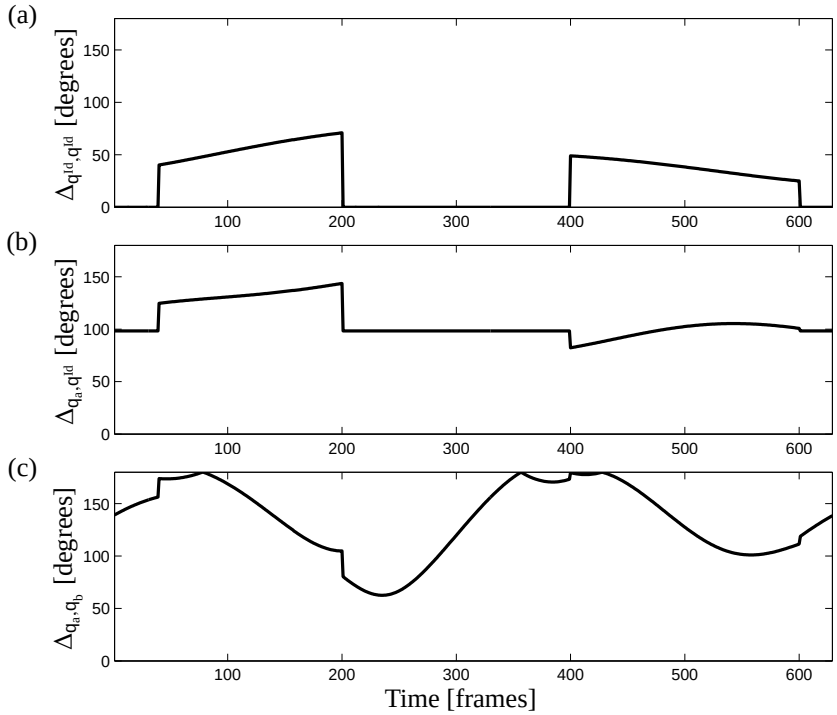


Figure 7.14. A synthetic example for quaternion distance curves with different calibration offsets. The ground truth distance curve is shown in **(a)** (both calibration offsets are correct). **(b)** and **(c)** show the curves when one offset and both offsets are wrong, respectively.

tricate if both offsets are wrong. We simulate this setting by evaluating Δ_{q_a,q_b}, see Figure 7.14 (c). Detecting the tracking errors from this representation seems to be impossible.

However, there is a solution which could circumvent the need for calibration. Recall that in practical scenarios, the two calibration offsets to be estimated correspond to the global coordinate system offset q^{G} between the two global coordinate systems and the local offset q^{L} between the local coordinate systems of the sensor and the bone. In such a scenario, the calibration for the global offset q^{G} can be avoided already in the preprocessing step where the cameras are calibrated. In this step, the object used for calibrating the cameras can be placed manually such that the tracking coordinate system F^{GT} coincides with F^{GI}. Once the global offset is known (in the described case, it is the identity), the local offsets q^{L} can be computed from a tracked frame of a sequence where the pose of the actor is already estimated. For example, if the first frame of the tracking is manually initialized, this frame could be used to obtain the offset q^{L}. While this scenario represents a practicable way of avoiding the calibration steps, the errors made by manually aligning the camera calibration object and by manually initializing the tracking in one frame might lead to distortions of the error function. By contrast, our calibration procedure is fully automatic and yields a robust solution by taking a whole sequence of frames into account for the calibration.

7.8 Conclusions

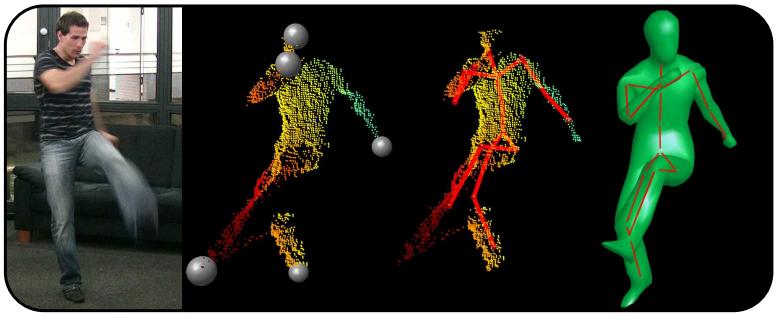
As a main result of this chapter, we showed that limb orientations constitute a suitable mid-level representation for detecting tracking errors in marker-less motion tracking. In contrast to con-

ventional evaluation techniques with marker-based optical systems, the usage of inertial sensors provides an unobtrusive and affordable way to generate ground truth data. Furthermore, inertial sensors impose comparatively weak additional constraints on the overall recording setup with regard to location, recording volume, and illumination. We showed that our procedure can reveal even subtle orientation mistakes which are hard to detect from a visual analysis of the tracking result or from previously used evaluation metrics based on positional information.

Further applications of inertial sensors can be found in sports sciences where marker-based mocap technology is sometimes difficult to apply. For example, obtaining marker-based mocap data from an athlete performing trampoline jumps is problematic due to the large recording volume and self-occlusions during the motions. Moreover, the rapid motions and trampoline contacts can cause optical markers to fall off. To make automated motion analysis in such a scenario possible, Helten *et al.* [2011] investigate an approach for automated segmentation and annotation of trampoline jumps using data obtained from inertial sensors.

Part III

Real-time Motion Reconstruction from Depth Images



Chapter 8

A Hybrid Approach for Reconstructing Human Motion

The 3D reconstruction of complex human motions from 2D color images constitutes a challenging and sometimes intractable problem. The pose estimation problem becomes more feasible when using streams of 2.5D monocular depth images as provided by a depth camera. However, due to low resolution of and challenging noise characteristics in depth camera images as well as self-occlusions in the movements, the pose estimation task is still far from being simple. Furthermore, the reconstruction task becomes even more challenging in real-time scenarios, where the usage of computationally expensive global optimization strategies is generally not possible.

In this chapter, we introduce a data-driven hybrid strategy that combines local pose optimization with global retrieval techniques to facilitate reconstruction of full-body human motions from a single depth image stream. In contrast to the offline method for stabilizing marker-less motion tracking presented in Chapter 6, we focus in this chapter on online real-time tracking. In all steps including the feature extraction, the retrieval, and the tracking steps, we develop and implement efficient algorithms in order to achieve real-time frame rates. In order to also obtain a robust reconstruction of the performed motions, we combine tracking with retrieval techniques. The final pose estimate for each frame is then determined from tracked and retrieved pose hypotheses which are fused using a fast voting scheme. Our algorithm reconstructs complex full-body motions in real-time and effectively prevents temporal drifting, thus making it suitable for various real-time interaction scenarios.

This chapter is based on the publications [Baak *et al.*, 2011b] and [Baak *et al.*, 2013][1]. After giving a motivating introduction (Section 8.1), we discuss related work (Section 8.2). We then briefly introduce depth cameras and the basic processing steps of the captured data (Section 8.3). As main contribution of this chapter, in Section 8.4 we describe in detail the different steps of our framework for motion reconstruction. Our quantitative and qualitative experiments as well as limitations are discussed in Section 8.5, before we conclude in Section 8.6.

[1] Reproduced with kind permission of Springer Science and Business Media.

8.1 Introduction

In recent years, several approaches for marker-less human pose estimation from multiple video streams have been presented [Bregler *et al.*, 2004; Bălan *et al.*, 2007a; Deutscher and Reid, 2005; Gall *et al.*, 2009; Stoll *et al.*, 2011]. While multi-view tracking already requires solving challenging non-linear optimization problems, monocular pose estimation puts current technology to its limits since, with intensity images alone, the problem is underconstrained [Moeslund *et al.*, 2006; Poppe, 2010; Bo and Sminchisescu, 2010]. In order to have a chance to reconstruct human movements, non-trivial inference or optimization steps are needed in combination with strong priors. In general, real-time reconstruction of complex human motions from monocular intensity image sequences can still be considered an open problem.

New depth sensors, such as time-of-flight (ToF) cameras or the Microsoft Kinect sensor, provide depth images at video frame rates. In such images, each pixel stores a depth value instead of a color value. Since this representation of a scene stands somewhere in the middle between a pure 2D color-based representation and full 3D scene geometry, depth images are also referred to as 2.5D data [Kolb *et al.*, 2010]. It turns out that with depth cameras, the 3D reconstruction of human motion from a single viewpoint becomes more feasible [Bleiweiss *et al.*, 2009; Friborg *et al.*, 2010; Ganapathi *et al.*, 2010; Knoop *et al.*, 2009; Pekelny and Gotsman, 2008; Shotton *et al.*, 2011; Zhu *et al.*, 2008], see also our discussion of related work in Section 8.2. In this chapter, we present a tracking framework that yields robust motion reconstruction from monocular depth image sequences. Moreover, our framework enables significant speed-ups of an order of magnitude compared to most of the previous approaches. In fact, we reach similar run time behavior as the algorithm implemented in the Microsoft Kinect [Shotton *et al.*, 2011], whereas we do not need GPU implementations.

Our procedure follows a hybrid strategy combining generative and discriminative methods, which is an established paradigm for pose reconstruction and tracking problems. While local optimization strategies [Knoop *et al.*, 2009] have proven to yield high frame rates, such techniques tend to fail for fast motions. Algorithms using global optimization techniques provide more reliable pose estimates, but are typically slow and prohibitive for real-time scenarios. Various data-driven approaches have also been suggested to overcome some of these issues, enabling fast yet robust tracking from intensity image streams, see [Okada and Stenger, 2008; Rosales and Sclaroff, 2000; Shakhnarovich *et al.*, 2003; Wang and Popovic, 2009]. These approaches rely on databases that densely cover the range of poses to be tracked, and fail on poses that are not contained in the database. Moreover, due to the high variability of general human motion, constructing such a database might become intractable. Hybrid strategies that combine generative and discriminative methods have proven to be a suitable methodology for pose estimation and tracking procedures, see Chapter 6 or [Demirdjian *et al.*, 2005; Ganapathi *et al.*, 2010; Rosales and Sclaroff, 2006; Salzmann and Urtasun, 2010; Sigal *et al.*, 2008; Ye *et al.*, 2011]. In these works, the main idea is to stabilize generative optimization algorithms by a discriminative component based on a database lookup or a classification scheme. Using this strategy, the risk of getting stuck in local minima is significantly reduced, while time-consuming global optimization methods are avoided.

In our approach, we employ a data-driven hybrid strategy conceptually similar to the work of Demirdjian *et al.* [2005], where local optimization is combined with global retrieval techniques, see Figure 8.1 for an overview. In our scenario, an actor may perform even complex and fast motions in a natural environment facing a single depth camera at a reasonable distance. Similar to

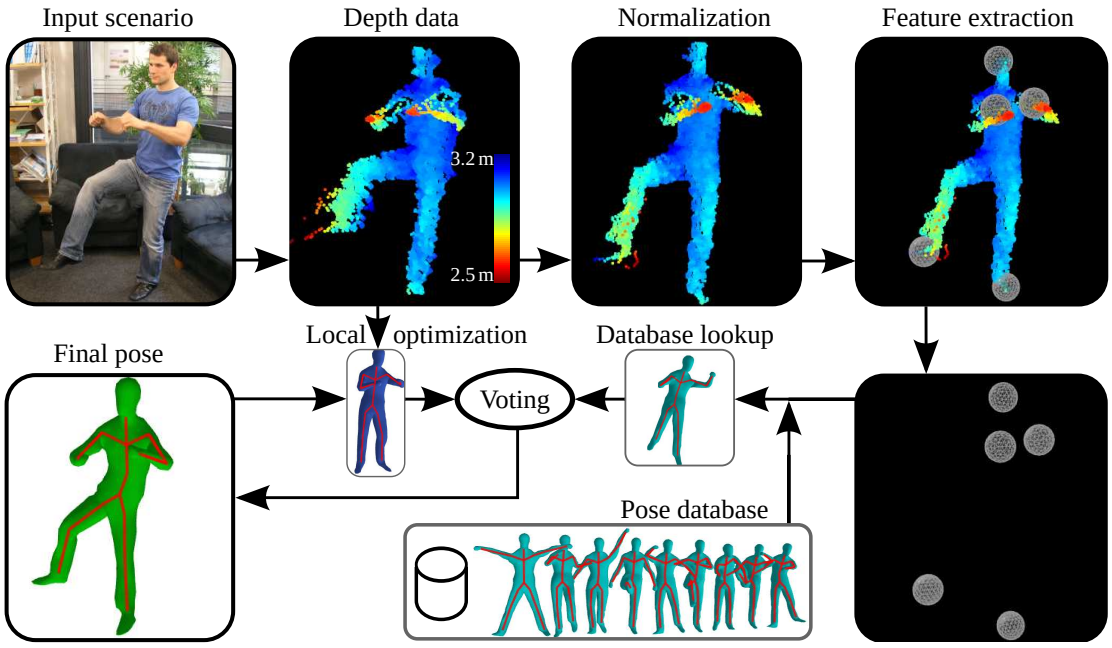


Figure 8.1. Overview of our proposed framework for reconstructing motions from depth data. Using features extracted from the raw depth data, we retrieve a pose hypothesis using a database lookup scheme. An additional hypothesis is obtained by running a local optimization algorithm that is initialized with the final pose of the previous frame. Then, a hypothesis voting decides for the either of the candidates.

Demirdjian *et al.* [2005], we retrieve a pose hypothesis from a large database of 3D poses using sparse features extracted from the depth input data. Additionally, a further hypothesis is generated based on the previously tracked frame. After a local optimization of both hypotheses, a late-fusion voting approach combines the hypotheses to yield the final pose. While the overall procedure is inspired by previous work [Demirdjian *et al.*, 2005; Ganapathi *et al.*, 2010], we introduce a number of novel techniques which add robustness and significantly speed up computations at various stages including efficient feature computation, efficient database lookup, and efficient hypothesis voting. In our experiments, we also compare our reconstructed motions to previous work using the publicly available benchmark dataset [Ganapathi *et al.*, 2010]. We gain significant improvements in accuracy and robustness (even for noisy ToF data and fast motions) while achieving frame rates of up to 100 fps (opposed to 4 fps reported in [Ganapathi *et al.*, 2010]).

Contributions. In this chapter, we present a system for full-body motion reconstruction from monocular depth images that requires only 10 to 16 milliseconds per frame on a standard single-core desktop PC, while being able to track even fast and complex full-body motions. Following a data-driven hybrid strategy that combines local pose estimation with global retrieval techniques, we introduce several technical improvements. Firstly, in the feature extraction step, we introduce a variant of Dijkstra's algorithm that allows us to efficiently compute a large number of geodesic extrema. Secondly, in the retrieval step, we employ an efficient database lookup scheme where semantic labels of the extrema are not required. Thirdly, we describe a novel late-fusion scheme based on an efficiently computable sparse and symmetric distance measure. It is the combination of all these techniques that avoids computational bottlenecks while providing robust tracking results.

8.2 Related Work

Intensity-image-based tracking. Monocular 3D human pose reconstruction from intensity images has become an important research topic. In order to deal with challenges coming from occlusions and missing 3D information, different approaches have been pursued making use of statistical body models [Guan *et al.*, 2009], physical constraints [Wei and Chai, 2010], object interaction [Romero *et al.*, 2010], or motion capture data [Rosales and Sclaroff, 2006]. For example, Guan *et al.* [2009] fit a statistical model of human body pose and shape to a single image using cues such as silhouettes, edges, and smooth shading. In a similar vein, Hasler *et al.* [2010] present a method for estimating human body pose and shape from single images using a bilinear statistical model. Physics-based constraints are used in Brubaker *et al.* [2010], where a physically-based modeling of the lower body helps to track walking motions from monocular images. In [Wei and Chai, 2010], the authors propose to annotate parts of image sequences with 2D joint positions, bone directions, and environmental contacts. From such annotations and the image data, they compute physically realistic human motions. As a different type of constraint, the interaction with objects can be exploited as demonstrated in the the work of Romero *et al.* [2010]. Furthermore, some approaches derive a direct mapping from image features to a dataset of admissible poses recorded with a marker-based system [Okada and Soatto, 2008; Okada and Stenger, 2008; Rosales and Sclaroff, 2000; Shakhnarovich *et al.*, 2003; Wang and Popovic, 2009]. With such discriminative approaches, poses that are not contained in the database are difficult to recover. The combination of generative and discriminative approaches can yield robust and smooth monocular motion estimates [Fossati *et al.*, 2010; Rosales and Sclaroff, 2006; Salzmann and Urtasun, 2010; Sigal *et al.*, 2008].

Depth-image-based tracking. 3D human motion reconstruction based on a single depth image stream has received increasing attention in the last years. While at first sight it appears simpler than its corresponding problem with monocular color images, one still has to deal with noise in the input data, low resolution sensors, lack of color information, and occlusion problems. Nowadays, commercial packages [Bleiweiss *et al.*, 2009] or software libraries exist that can compute joint positions from depth images for multiple people in real time (Microsoft Kinect SDK [Microsoft, 2012], Primesense NITE middleware [Primesense, 2012]). While the algorithm behind the NITE middleware is not revealed to the public, Microsoft published the approach that is implemented in the Kinect SDK in [Shotton *et al.*, 2011]. The authors use randomized decision forests trained on a huge set of various body poses and shapes in order to hypothesize joint locations from features on the raw depth input data. The approach was recently combined with a regression scheme to predict joint locations more accurately [Girshick *et al.*, 2011]. Also, positions of occluded joints can be estimated.

Some approaches use *global optimization* methods to solve the motion reconstruction task. For example, Friborg *et al.* [2010] use a GPU-accelerated particle filter to fit a surface mesh consisting of rigidly connected generalized cylinders to stereo depth data. However, even with GPU implementations, such approaches are often not real-time capable. Pure *local optimization* strategies have also been explored, which are implemented as variants of the iterative closest points (ICP) method [Besl and McKay, 1992]. For example, Pekelny and Gotsman [2008] simultaneously track and reconstruct the shape of limbs through depth images. Knoop *et al.* [2009] show that a combination of ToF and stereo data enables full-body motion reconstruction at real-time frame rates.

Also using ICP, Grest *et al.* [2007] combine depth and silhouette data to track articulated motion with ten degrees of freedom in real time. Although yielding high frame rates, such methods often fail due to noise and motion blur present in the depth data. In particular with fast motions, local optimization easily gets stuck in erroneous poses which are hard to recover from.

In order to yield a more robust tracking, many approaches stabilize the optimization algorithms using additional *prior knowledge*. For example, Schwarz *et al.* [2010] include a database of motions as prior knowledge for a particle filter-based optimization method. However, motions not present in the database cannot be tracked. As a complementary technique for stabilization, many approaches *detect features in a bottom-up fashion* directly from the depth input data. Here, geodesic distances are used in some works in order to detect anatomic landmarks. Following such a scheme, Ganapathi *et al.* [2010] classify geodesic extrema features extracted from depth images according to the class labels 'hand', 'foot', and 'head'. With these detections, the search space of a particle filter is constrained. Integrating constraints into similar optimization techniques, anatomical landmarks are identified using feature tracking or heuristics in [Azad *et al.*, 2008; López-Méndez *et al.*, 2011; Siddiqui and Medioni, 2010]. Using object detectors to estimate the position of the head and the hands, Gall *et al.* [2011] stabilize a local optimization-based algorithm for tracking the upper body from depth data. Also, constrained inverse kinematics has been used on anatomical landmarks in [Schwarz *et al.*, 2011; Zhu *et al.*, 2010]. In our approach, we also make use of bottom-up detected features in order to stabilize a local optimization approach. As for the feature extraction, we build on the idea of accumulative geodesic extrema [Plagemann *et al.*, 2010] and contribute with an efficient feature computation strategy.

Depth cameras seem to be an ideal type of sensor to facilitate intuitive human computer interaction based on full-body motion input. Therefore, many approaches focus on achieving *real-time* performance and try to find efficient algorithms for the motion reconstruction task. Although efficiency is clearly one of the key aspects to make motion reconstruction applicable for home use, most approaches with a focus on robustness reach only interactive run times around 10 FPS [Ganapathi *et al.*, 2010; Grest *et al.*, 2007; Zhu *et al.*, 2010]. Only recently, methods have been published that perform robust motion reconstruction within just a couple of milliseconds per frame [Girshick *et al.*, 2011; Shotton *et al.*, 2011]. Such approaches for motion reconstruction are interesting from a practical point of view since they leave enough CPU cycles free for applications or games that use the reconstructed motion as input. Exceeding the performance of most published methods, we can report nearly 100 FPS for full body motion reconstruction. Apart from the methodology of combining discriminative and generative models, the key to our efficient and stable motion reconstruction procedure is a compound of efficient feature computation, efficient database lookup, and an efficient voting strategy.

8.3 Acquisition and Data Preparation

In this section, we first summarize the concept of depth cameras while fixing basic notation (Section 8.3.1). After that, we describe the model of the person to be tracked (Section 8.3.2). Our pose database is introduced in Section 8.3.3. Finally, we describe how we normalize the depth data (Section 8.3.3). Such a normalization is important in order to facilitate efficient computations in the forthcoming steps.

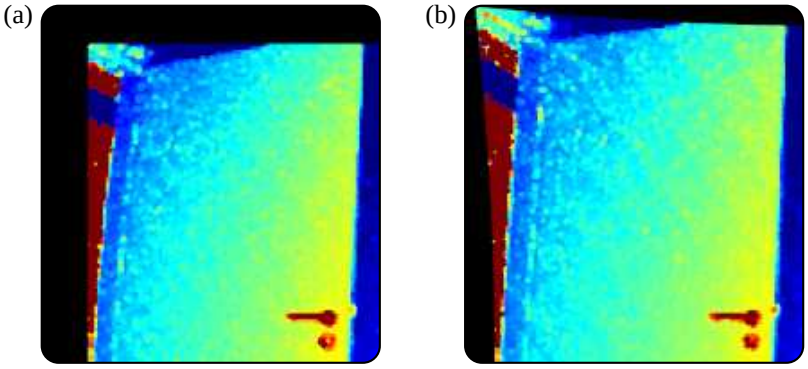


Figure 8.2. Point cloud obtained by a ToF camera **(a)** without and **(b)** including a method for removing lens distortion effects. Note the straightening effect on the left edge of the door.

8.3.1 Depth data

Before ToF cameras and the Kinect sensor became popular, stereo cameras were predominantly used to obtain depth data in real-time. In order to compute depth values, such passive stereo systems need to identify corresponding features in the two images captured at every time step. However, computing and matching such features is computationally expensive and often fails for objects without texture or for objects with repeating texture.

Current depth cameras based on active illumination with infrared light overcome these limitations. Moreover, current ToF cameras are robust to background illumination and yield stable distance values independent of the observed textures. In principle, ToF cameras capture depth/distance data at video frame rates by measuring the round trip time of infrared light emitted into and reflected from the scene. Several successive measurements have to be made in order to estimate the phase shift of the infrared light from which the round trip time is derived [Kolb *et al.*, 2010]. Moreover, further measurements are taken over a longer period of time in order to reduce noise in the measurements. For static scenes, this process leads to measurements with high accuracies in the range of millimeters. For dynamic scenes with moving objects, however, this process can lead to errors in the estimation of depth values. Problematic are edges that separate an object from another, more distant object, resulting in strongly corrupted depth measurements, also called *mixed pixels*. Furthermore, low resolution, strong random noise and a systematic bias [Kolb *et al.*, 2010] lead to data that is difficult to handle.

A depth camera returns a distance image $I := \mathbb{Z}^2 \rightarrow \mathbb{R}$ with $\mathbb{Z}^2$ being the pixel domain. Since the camera also produces an amplitude image in the infrared domain, we use a standard pattern-based camera calibration [Matlab, 2012] to recover the camera matrix and parameters for the lens distortion. To remove lens distortion effects, we apply the method of Heikkila and Silven [1997] which yields stable and accurate metric distance values, see Figure 8.2 as an example. We do not calibrate for systematic bias of the camera, since for full-body motion reconstruction slight constant deviations in the measurements do not play an important role. For a recent method that calibrates for systematic bias using an intensity-based approach we refer to [Lindner *et al.*, 2010]. Using the calibration information, we transform the per-pixel distances into a metric 3D point cloud $\mathcal{M}_I \subseteq \mathbb{R}^3$ for every input frame of our online motion reconstruction framework, see Figure 8.3 (a). We then perform background subtraction using a static prerecorded background model and delete contour pixels to mitigate the influence of mixed pixels. Finally, a 3×3 median filter is

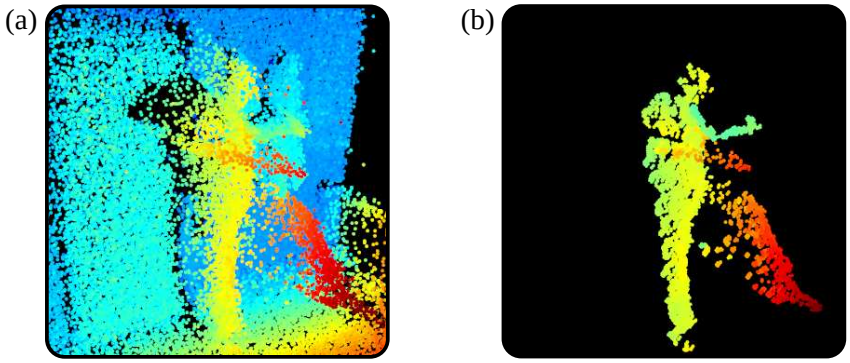


Figure 8.3. (a): Original depth point cloud. **(b)**: Point cloud after background subtraction and filtering. Some mixed pixel artifacts remain, see, *e. g.*, the left leg.

used to reduce noise in the measurements, see Figure 8.3 (b).

In contrast to a ToF camera, the Microsoft Kinect depth sensor uses an active stereo approach. More specifically, a camera records an image of a projected structured light pattern in the infrared domain. Then, from the recorded pattern, a depth map is derived. In contrast to a ToF camera, only one image is analyzed in every time step. Thus, the Kinect camera is less susceptible to mixed pixels in dynamic scenes. However, the data also exhibits significant noise. In particular, artifacts like holes in the data appear when the projected pattern cannot be recognized. Moreover, the coarse depth quantization limits the accuracy in the far field from the camera, where, for example, a 2.5 cm quantization gap occurs at 3 meters distance to the camera. The presented algorithms in this article have been applied to depth data coming from ToF cameras as well as data coming from the Microsoft Kinect. Without changing or tuning the proposed algorithms, the final pose estimates with each of the cameras are qualitatively very similar as shown in the video [Baak *et al.*, 2011a] accompanying the corresponding paper.

8.3.2 Model of the actor

The body of the actor is modeled as a kinematic chain [Murray *et al.*, 1994]. We use $J = 20$ joints that are connected by rigid bones, where one distinguished joint is defined as the root of the kinematic chain, see Figure 8.4 (a). A pose is fully determined by the configuration of a kinematic chain specified by a pose parameter vector χ containing the position and orientation of the root joint as well as a set of joint angles. Through forward kinematics [Murray *et al.*, 1994] using χ, 3D joint positions represented by a stacked vector $P_\chi \in \mathbb{R}^{3\cdot J\times 1}$ can be computed. Using linear blend skinning [Lewis *et al.*, 2000], we attach a surface mesh with a set of 1170 vertices $\mathcal{M}_\chi$ to the kinematic chain to model the deforming body geometry, see Figure 8.4 (b). Initializing the body model to the shape of a specific actor is beyond the scope of this article. Methods exist to solve this task using image data and a large database of scanned humans, see, *e. g.*, [Guan *et al.*, 2009; Hasler *et al.*, 2010]. Recently, Weiss *et al.* [2011] have shown that the body shape of a person can be determined using depth images from four different views. As shown in our experiments (Section 8.5), even with a fixed body model we can track people for a range of different body sizes.

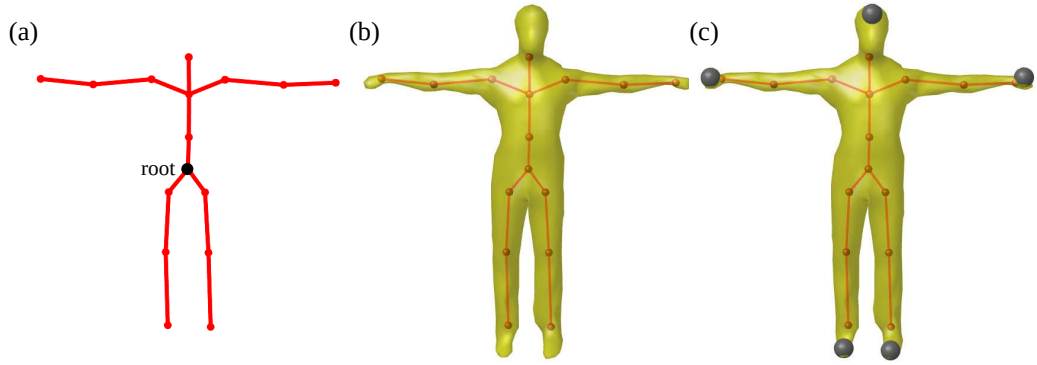


Figure 8.4. (a) Skeletal kinematic chain with a root joint. **(b)** Rigged mesh. **(c)** Highlighted end effectors (hands, feet, and head).

8.3.3 Pose database

Motion tracking approaches based on local optimization update the model parameters (in our case the joint angles) by optimizing a specified cost function, where convergence only to a near local minimum can be guaranteed. Although such methods typically run very fast, they fail when the initialization is too far away from the actual pose. For such a failure case we say that the algorithm loses track. This is often the case for fast motions where body parts can move far from frame to frame. One strategy to overcome such limitations is to reinitialize the local optimization when the track is lost. In the proposed algorithm, we use global pose estimates derived from database knowledge for such reinitializations. To this end, we create a database of human full body poses obtained with a marker-based motion capture system. The actor performs a variety of motions including hand gestures and foot motions to span a large range of different poses. To enable invariance under global transformations, the obtained poses χ_i are then normalized according to the positions of the root joint and the viewing direction. Furthermore, to maximize the variety and minimize the number of poses in the database, we select a subset of the recorded poses using a greedy sampling algorithm [Wang and Popovic, 2009]. In this algorithm, the distance of two poses specified by χ_1 and χ_2 is measured based on the distance of the corresponding joint positions

$$d_P(\chi_1, \chi_2) := {}^1/_J \cdot \|P_{\chi_1} - P_{\chi_2}\|_2. \tag{8.1}$$

In contrast to Wang and Popovic [2009], we truncate the sampling as soon as the minimal distance between all pairs of selected poses reaches a certain threshold. Using the truncated sampling, we obtain roughly 25 000 poses in which any two selected poses have a pose distance d_P larger than 1.8 cm. For each selected pose, we then consider end effector positions of the left/right hand, the left/right foot, and the head, modeled as $E^5_\chi := (e^1_\chi, \ldots, e^5_\chi) \in (\mathcal{M}_\chi)^5$, see Figure 8.4 (c).

The following three reasons motivate the use of end effector positions as features. Firstly, end effector positions can be efficiently estimated for a large set of different poses even from depth data alone, see Sect 8.4.2. Secondly, for many poses these positions are characteristic, thus yielding a suitable representation for cutting down the search space. Thirdly, they lead to low-dimensional feature vectors which facilitate the usage of efficient indexing methods. Thus, end effector positions constitute a suitable mid-level representation for full-body poses that on the one hand abstract away most of the details of the noisy input data, yet on the other hand retain the discriminative power needed to cut down the search space in the pose estimation procedure.

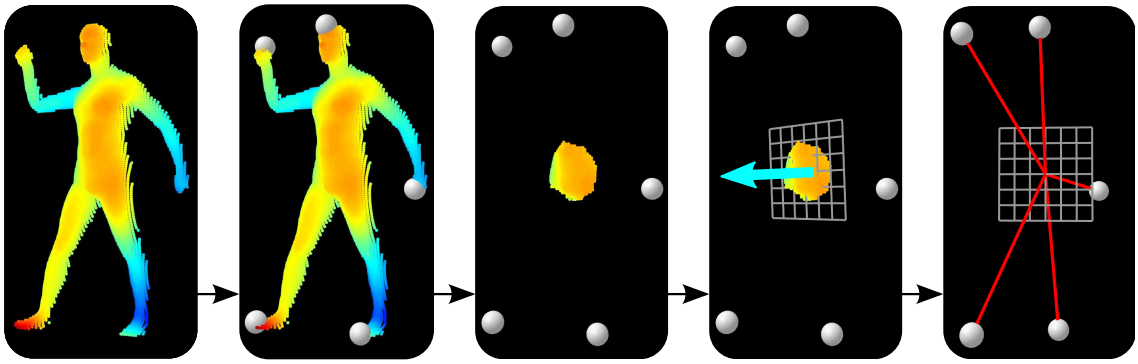


Figure 8.5. Normalization of the geodesic extrema with respect to a computed viewing direction.

For indexing, we use a *kd*-tree [Cormen *et al.*, 2001] on the 15-dimensional stacked vectors E_χ^5 since they provide logarithmic search time in the size of the database and have turned out to be an efficient search structure for low-dimensional feature vectors. Since the size of the skeleton (*e. g.*, body height or arm span) varies with different actors, the pose database has to be adapted to the actor. While not implemented in the system presented, this task can be solved using a retargeting framework. Even without retargeting, by manipulating the depth input point cloud $\mathcal{M}_I$ we are able to track motions of people if body proportions are not too far off the database skeleton, see Section 8.5.

8.3.4 Normalization

In the proposed tracking framework, we allow the actor to move freely within the field of view of the camera, while we restrict variations of the viewing direction to the range of about ±45° rotation around the vertical axis with respect to the frontal viewing direction. Recall that in our database all poses have been normalized with regard to the position of the root joint and the viewing direction. Thus, in order to query the database in a semantically meaningful way, we need to cope with variations in global position and orientation of the actor. We normalize $\mathcal{M}_I$ with respect to global position by means of a 3D ellipsoid fit around $\mathcal{M}_I$ using a mean-shift algorithm similar to [Wang and Popovic, 2009]. To cope with global rotations, one could augment the database to contain pose representations from several viewing directions [Demirdjian *et al.*, 2005; Shakhnarovich *et al.*, 2003; Wang and Popovic, 2009]. In this case, the retrieval time as well as the risk of obtaining unwanted poses would increase. Instead, in our framework, we normalize the depth input point cloud according to an estimated viewing direction. To this end, we compute a least-squares plane fit to points corresponding to the torso, which we assume to be the points that are closer than 0.15 m to the center of $\mathcal{M}_I$, see Figure 8.5. The normal of the plane, as indicated by the cyan arrow in Figure 8.5, corresponds to the Eigenvector with the smallest Eigenvalue of the covariance matrix of the points. The viewing direction is its projection onto an imagined horizontal ground plane. We then rotate the positions of the geodesic extrema about the vertical axis through the center such that the normal of the rotated plane points to front. To cope with frames in which the viewing direction cannot be estimated because, *e. g.*, the torso is occluded, we adaptively smooth the estimated directions over time. We detect whether the torso is occluded by inspecting the Eigenvalues of the above mentioned covariance matrix. Here, occluding body parts often lead to a stronger curvature in the regarded points (smallest Eigenvalue is relatively large)

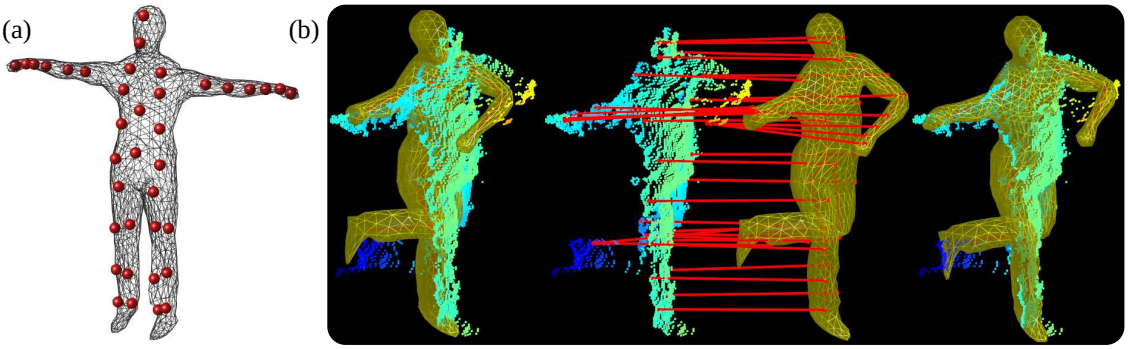


Figure 8.6. (a): Subset of vertices $C_\chi \subseteq \mathcal{M}_\chi$. **(b)**: From pose χ (left), correspondences for mesh vertices in C_χ are estimated (middle). Local optimization using the correspondences yields an updated pose χ' (right).

or a less circular fit (largest Eigenvalues are not similar). Then, we minimize the influence of the estimated normal. As a consequence, the detected viewing direction remains stable even if the arms occlude the torso or the center of $\mathcal{M}_I$ does not correspond to the torso.

8.4 Motion Reconstruction Framework

As explained in the previous section, in the offline preprocessing phase, the camera matrix is obtained and the background model is created. We now describe our proposed online framework, see also Figure 8.1. At a given frame t, the first steps are to compute the point cloud $\mathcal{M}_I$ from the distance image I, to perform background subtraction, to filter out noise and to normalize according to the viewing direction. Let χ^*_{t-1} be the final pose estimate of the previous frame $t-1$. From χ^*_{t-1}, we obtain a pose hypothesis χ^{LocOpt}_t by refining χ^*_{t-1} with a local optimization procedure that takes the input depth data into account (Section 8.4.1). A second pose hypothesis is obtained as follows. We extract a 15-dimensional feature vector from $\mathcal{M}_I$, representing the 3D coordinates of the first five geodesic extrema (Section 8.4.2). Being a low-dimensional yet characteristic pose representation, the features permit rapid retrieval of similar full-body poses from a large pose database (Section 8.4.3). From the set of retrieved poses we choose a single pose hypothesis χ^{DB}_t using a distance function which takes the influence of the estimated pose of the previous frame χ^*_{t-1} into account. Based on a late-fusion voting scheme that combines two sparse distances measures, our algorithm decides between χ^{DB}_t and χ^{LocOpt}_t to find the final pose χ^*_t, see Section 8.4.4.

8.4.1 Local Optimization

In our local pose optimization, we follow a standard procedure as described in, *e. g.*, [Rosenhahn *et al.*, 2008a]. Here, the goal is to modify an initial pose χ such that the modified pose χ' fits to the point cloud $\mathcal{M}_I$ more accurately. To this end, we seek correspondences between vertices in $\mathcal{M}_\chi$ and points in $\mathcal{M}_I$.

Finding correspondences for all vertices $v \in \mathcal{M}_\chi$ is not meaningful for three reasons. Firstly, many vertices do not have semantically meaningful correspondences in $\mathcal{M}_I$, *e. g.*, the vertices in the back of the person. Secondly, the number of correspondences for the torso would be much higher than the number of correspondences in the limbs, which would disregard the importance of the limbs

for pose estimation. Thirdly, the computation time of local optimization increases with the number of correspondences.

Therefore, we use a predefined set $C_\chi \subseteq \mathcal{M}_\chi$ of mesh vertices as defined in Figure 8.6 (a). Here, we make sure that we select a couple of vertices for each body part. Using a local *kd*-tree built up in every frame, we efficiently obtain the ℓ nearest neighbors in $\mathcal{M}_I$ of each vertex $v \in C_\chi$ and claim correspondence of v to the median of its ℓ nearest neighbors to reduce the influence of noise. Using these correspondences, we obtain updated pose parameters χ' by applying an optimization framework similar to the one in [Rosenhahn *et al.*, 2008a].

8.4.2 Feature Computation

To obtain a sparse yet expressive feature representation for the input point cloud $\mathcal{M}_I$, we revert to the concept of geodesic extrema as introduced in [Plagemann *et al.*, 2010]. Such extrema often correspond to end effector positions, yielding characteristic features for many poses as indicated by Figure 8.10. Following [Plagemann *et al.*, 2010], we now summarize how one obtains such features. Furthermore, we introduce a novel variant of Dijkstra's algorithm that allows us to efficiently compute a large number of geodesic extrema. We model the tuple of the first n geodesic extremal points as

$$E_I^n := (e_I^1, \ldots, e_I^n) \in (\mathcal{M}_I)^n. \tag{8.2}$$

To compute E_I^n, the point cloud data is modeled as a weighted graph where each point in $\{p_1, \ldots, p_L\} := \mathcal{M}_I$ represents a node in the graph. We refer to a node by its index $\ell \in [1 : L]$. To efficiently build up the edge structure of the graph, we exploit the neighborhood structure in the pixel domain $\mathbb{Z}^2$ of the underlying depth image. Here, we consider the 8-neighborhood of each $p_\ell \in \mathcal{M}_I$ in the domain of the underlying image. For each such neighboring point $p_m \in \mathcal{M}_I$, we add an edge between m and ℓ of weight $w = \|p_m - p_\ell\|_2$ if w is less than a distance threshold τ. This way, we obtain a weighted edge structure in form of an adjacency list

$$\mathcal{E}(\ell) := \{(m, w) \in [1 : L] \times \mathbb{R}_+ \mid p_m \text{ and } p_\ell \text{ share an edge of weight } w\} \tag{8.3}$$

for $\ell \in [1 : L]$. Here, note that when building up the edge structure, the distance between any two points in $\mathcal{M}_I$ has to be evaluated only once.

In our approach, in contrast to the method in [Plagemann *et al.*, 2010], we need to obtain a fully connected graph with only one connected component in order to obtain meaningful geodesic extrema. In practice, however, the graph computed as described above is not fully connected if, for example, the depth sensor misses parts of the thin limbs, or due to occlusions, see Figure 8.7. To cope with such situations, we use an efficient union-find algorithm [Shapiro and Stockman, 2002] in order to compute the connected components. To reduce small artifacts and noise pixels, we discard all components that occupy a low number of nodes. Furthermore, we assume that the torso is the component with the largest number of nodes. All remaining components are then connected to the torso by adding an edge between the respective closest pair of pixels if the edge weight is less than 0.5 m, see the red dotted lines in Figure 8.7. This allows us to find meaningful geodesic extrema even if the initial graph splits into separate connected components, see Figure 8.10 (b) and (h) for the resulting geodesic extrema of the graphs shown in Figure 8.7.

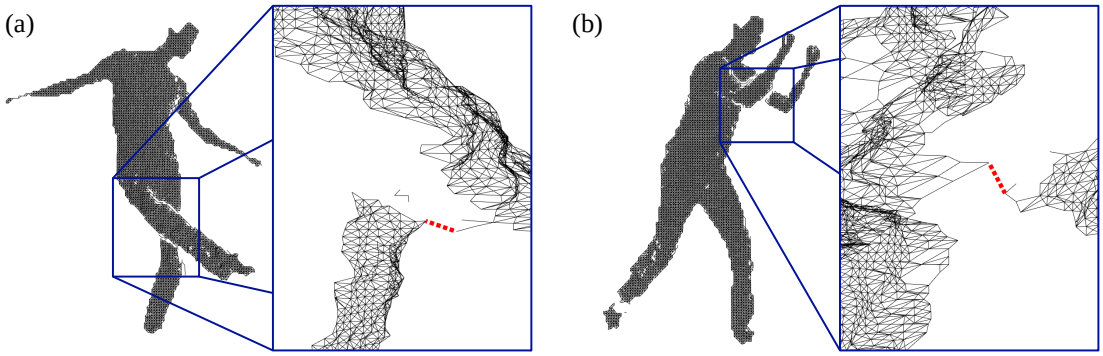


Figure 8.7. Graph obtained from the depth image (black lines) and a zoom-in from a more lateral viewpoint for two poses with self-occlusions. The initially disconnected graph is automatically connected using edges indicated by the red dashed lines, respectively.

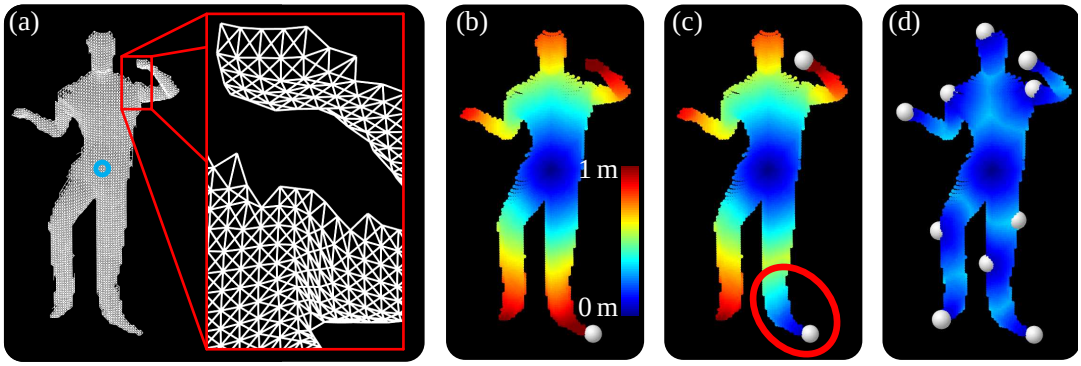


Figure 8.8. Computation of geodesic extrema using a variant of Dijstra's algorithm. **(a)**: Graph structure and source node (cyan circle). **(b)**: Geodesic distances and first geodesic extremum. **(c)**: Updated geodesic distances and second geodesic extremum. **(d)**: The first ten geodesic extrema.

We now show how a large number of extrema can be computed efficiently. Basically, we follow an iterative computation strategy. In each iteration, we use Dijkstra's algorithm [Cormen *et al.*, 2001] to compute the geodesic distances from a centroid node ℓ_0 (referred to as *source node*) to all other nodes in the graph. We then pick the node with the maximal distance as the corresponding extremal point. The efficiency of our algorithm is based on the observation that only in the first iteration of our algorithm, a full pass of Dijkstra has to be computed. In all remaining iterations one needs to consider only a small fraction of the nodes. As another observation, we only need to obtain the geodesic distances of each node and do not need to store the actual shortest path information which is usually saved in a predecessor array in Dijkstra's algorithm [Cormen *et al.*, 2001]. Therefore, we save additional time in each iteration by omitting the predecessor array.

As input to Algorithm 4, we use the graph structure with nodes, edges, and the designated source node ℓ_0, see Figure 8.8 (a). Additionally, we use a priority queue Q that stores tuples $(m, w) \in [1 : L] \times \mathbb{R}_+$ of nodes and weights sorted by increasing weight. The priority queue allows us to extract the tuple with the minimal weight by the Q.getMin() operation. To keep track of the distance values of each node, we use an auxiliary array Δ having L entries.

We start the algorithm by initializing Δ, see Lines 1-3. Then, we insert the source node into the previously empty priority queue Q in Line 5. We then iterate over all geodesic extrema to

Algorithm 4 Computation of Geodesic Extrema

Input: $\{p_1 \dots p_L\} := \mathcal{M}_I$: 3D point cloud with points $p_\ell \in \mathbb{R}^3$ and nodes $\ell \in [1:L]$
$\mathcal{E}(\ell) := \{(m,w) \in [1:L] \times \mathbb{R} | p_m$ and p_ℓ share an edge of weight $w\}$:
edge adjacency list defined on $\mathcal{M}_I$
$\ell_0 \in [1:L]$: index of the designated source node
Q: priority queue for elements $(m,w) \in [1:L] \times \mathbb{R}$
n: number of geodesic extrema to be computed
Output: $(e_I^1, \dots, e_I^n) \in (\mathcal{M}_I)^n$: the n first geodesic extrema of G

```
1:  for ℓ ← 1 to L do
2:      Δ[ℓ] ← ∞                                          ▷ Initialize distance array
3:  end for
4:  Δ[ℓ_0] ← 0                                            ▷ Distance to source
5:  Q.insert( (ℓ_0, 0) )
6:  for i ← 1 to n do                                     ▷ Compute the first n extrema
7:      while Q ≠ ∅ do
8:          ℓ ← Q.getMin()                                ▷ Get entry with minimal weight
9:          Q.removeMin()                                 ▷ Remove the entry from Q
10:         for each (m, w) ∈ E(ℓ) do                     ▷ For all nodes connected by an edge to p_ℓ
11:             if Δ[ℓ] + w < Δ[m] then
12:                 Δ[m] = Δ[ℓ] + w                       ▷ A shorter path has been found
13:                 Q.insert( (m, Δ[m]) )
14:             end if
15:         end for
16:     end while                                         ▷ Δ now contains the geodesic distances
17:     ℓ* ← arg max_{ℓ∈[1:L]} Δ[ℓ]   ▷ Note: the arg max must ignore nodes that were not reached
18:     e_I^i ← p_{ℓ*}                                    ▷ Store i^th extremum
19:     Δ[ℓ*] ← 0                                         ▷ Simulate edge insertion between p_{ℓ_0} and p_{ℓ*}
20:     Q.insert( (ℓ*, 0) )                               ▷ Let ℓ* act as new source
21: end for
```

be computed. The first pass of Dijkstra (Lines 7 to 16) stores the shortest geodesic distances from the source node to any other node in the graph in the array Δ, see Figure 8.8 (b). Then, the point corresponding to the node ℓ^* with the largest distance in Δ is taken as the first geodesic extremum e_I^1 (Lines 17 to 18). Note that if there are still nodes which are not reachable from the source node ℓ_0, they bear the same distance values ∞ as set in the initialization. Of course, such unreachable nodes should not be considered as geodesic extrema. Therefore, the arg max operator in Line 17 must ignore these nodes in order to recover the true geodesic extremum. In Figure 8.8 (b), the detected extremum e_I^1 is indicated by the gray sphere on the left foot. According to [Plagemann *et al.*, 2010], the next step is to add a zero-cost edge between ℓ_0 and ℓ^* and then to restart Dijkstra's algorithm to find e_I^2, and so on. This leads to a run time of $O(n \cdot D)$ for n extrema with D being the run time of Dijkstra's algorithm for the full graph. Note that the second run of Dijkstra's algorithm shows a high amount of redundancy: the entries in the array Δ corresponding to all nodes in the graph that are geodesically closer to ℓ_0 than to the node of e_I^1 will not change in the second run. For example, in Figure 8.8 (c), only the distance values of the nodes within the highlighted area have changed.

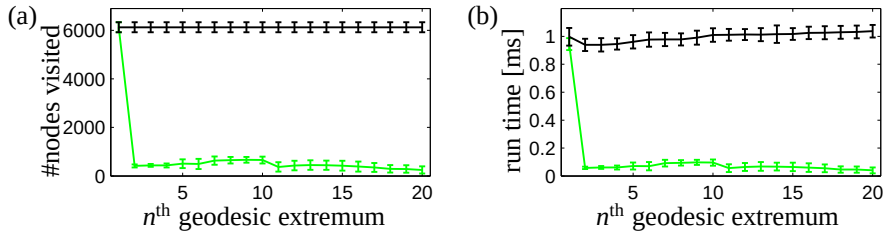


Figure 8.9. (a) Number of nodes visited and **(b)** run time in milliseconds to find the n^{th} geodesic extremal point for the baseline (black) and our optimized algorithm (green). Average values and standard deviation bars for a sequence of 400 frames from the dataset of [Ganapathi *et al.*, 2010] are reported.

Therefore, to compute the 2nd pass, we keep the distance values of the 1st pass and let the node ℓ^* corresponding to e_I^1 act as the new source, see Lines 19 and 20. This way, the second iteration will be by an order of magnitude faster than the first iteration, as also confirmed by our experiments described in the subsequent paragraphs. Using ℓ^* as the new source, we update Δ with a pass of Dijkstra's algorithm and pick e_I^2 as the point with the maximal distance in the updated Δ, see Figure 8.8 (c). For the 3rd pass we let the node corresponding to e_I^2 act as the new source by setting the corresponding value in Δ to 0, and run Dijkstra again. This way, in the 3rd pass, only nodes in the graph that are closer to the node of e_I^2 than to all other previously used source nodes are touched. We proceed iteratively to compute the subsequent extremal points, see Figure 8.8 (d) for the resulting distance values and extrema after 10 iterations.

Our computational strategy leads to drastic improvements in the run time for each pass. To experimentally verify this, we evaluated the algorithm on a depth input sequence of 400 frames taken from the dataset of [Ganapathi *et al.*, 2010]. We computed the first 20 geodesic extrema for each of the 400 frames using both a baseline algorithm that runs a full Dijstra pass in each iteration and our optimized algorithm. We traced the number of nodes visited in each iteration as well as the actual run time for each iteration. Figure 8.9 shows that in the first iteration all reachable nodes in $\mathcal{M}_I$ (on average there were more than 6000 nodes in the graph) were visited. In the second iteration, only 413 ± 61 nodes (average $\pm$ standard deviation over all frames) were visited. This substantial reduction is also reflected by the run time of the algorithm, which drops from 1 millisecond in the first iteration to about 0.058 ± 0.0085 milliseconds in the second iteration, see Figure 8.9 (b). As a result, the overall run time for computing the first 20 geodesic extrema is only slightly higher than the run time of the original Dijkstra algorithm for computing the first geodesic extremum. Thus, the algorithm allows us to efficiently compute a large number of geodesic extrema.

The overall approach enables the detection of semantically meaningful end effector positions even in difficult scenarios. Figure 8.10 shows a number of challenging examples, where legs occlude each other (b)-(c), multiple body parts occlude each other (d)-(f), a fast punching motion with occlusions is performed (g)-(k), a leg is bent to the back (l), and the hands are outstretched to the camera (m). However, in poses where the end effectors are very close to other parts of the body, the topology of the graph may change and the detected extrema may differ from the actual set of end effectors, see Figure 8.11(a)–(c). In these poses, the left elbow, the left shoulder, and the left hip are selected as e_I^5, respectively. Also, flying mixed pixels can cause the topology of the graph to change, as depicted in Figure 8.11 (d), where we show a pose once from a frontal view and once from a side view. Note that although the left hand keeps a reasonable distance from the head, mixed pixels build a bridge in the graph from the hand to the head. Thus, the fifth extremum is located at the elbow. Figure 8.11(e) shows a similar situation in which the head is not detected due

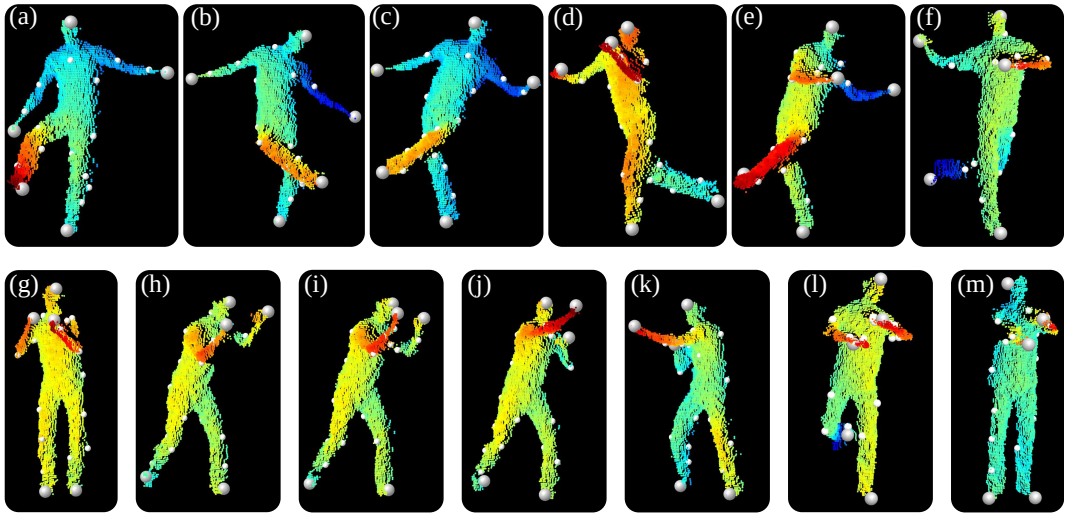


Figure 8.10. Example poses with the first five geodesic extrema drawn as big spheres, and extrema 6 to 20 drawn as smaller blobs. For many poses, the first five extrema correspond to the end effectors, even when self-occlusions are present.

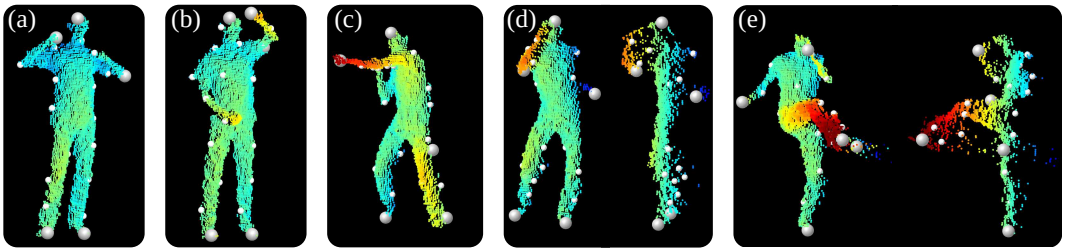


Figure 8.11. (a)-(c): Problematic example poses. In particular when the hands come close to the body, end effector detection becomes difficult. **(d)-(e)**: Flying mixed pixels lead to deviations in the end effector detection.

to mixed pixels. Instead, the fifth extremum is located at the hip.

In the subsequent section, we will explain the discriminative component of our framework, where pose candidates are obtained from the database by using the positions of the first five geodesic extrema as a query. If the end effectors are not revealed by these extrema, however, the obtained pose candidates are often meaningless. As will be explained later, the influence of such meaningless poses on the final pose estimates can be minimized with our combined generative and discriminative framework.

8.4.3 Database Lookup

As for the database lookup component, the goal is to identify a suitable full-body pose χ_t^{DB} from our pose database using the extracted geodesic extrema E_I^5 as the query input. However, as opposed to the database motions where the semantics of the end effector positions are known, the semantic labels of the extrema on the query side are not known. To partially solve for missing semantics, the method [Ganapathi *et al.*, 2010] uses a classifier trained on 'hand', 'head', and 'foot' patches of distance images. This process, however, is relatively expensive (taking 60 ms

per frame according to [Plagemann *et al.*, 2010]) and is thus not directly suitable in real-time scenarios. Also, when using depth data alone, misclassification of patches might occur because of missing color information, strong noise, and the low resolution of the measurements. In order to circumvent the classification problem, we propose a query strategy that does not rely on having a-priori semantic labels for the extracted geodesic extrema. Intuitively speaking, we use different queries that reflect different label assignments. As explained in the following, from the retrieved poses, we then choose a candidate pose that most likely corresponds to the correct labeling.

Let $\mathbb{S}_5$ be the symmetric group of all five-permutations. For a permutation σ and a five-tuple E, we denote the permuted tuple by σE. Now, let $\mathcal{S} \subseteq \mathbb{S}_5$ be a subset containing permutations σ such that the positions in σE_I^5 are close to the end effectors of the previous frame χ_{t-1}^*. More specifically, we define

$$\mathcal{S} := \{\sigma \in \mathbb{S}_5 | \forall n \in [1:5] : \| e_I^{\sigma(n)} - e_{\chi_{t-1}^*}^n \| < \mu\}. \tag{8.4}$$

In our experiments, we use a distance threshold of $\mu = 0.5$ meters to effectively and conservatively prune the search space while still allowing for large jumps in the end effector positions which may be present in fast motions. In frames with clear geodesic extrema, the number of considered permutations $|\mathcal{S}|$ typically drops to one. To further increase robustness to false estimations in the previous frame, we add additional permutations to $\mathcal{S}$ if we detect jumps in the positions of the geodesic extrema. To compute the additional permutations, we assume that the two lowest extrema *w.r.t.* the vertical axis, say e_I^1 and e_I^2, correspond to the feet. This leads to two possible label assignments where the label 'left foot' is assigned to either e_I^1 or e_I^2. For each of the two assignments, the remaining three extrema can receive $3! = 6$ different labelings. This leads to $2 \cdot 6 = 12$ additional permutations added to $\mathcal{S}$.

By querying the *kd*-tree of the pose database for K nearest neighbors for each permutation in $\mathcal{S}$, we obtain $K \cdot |\mathcal{S}|$ pose candidates $\chi_{k,\sigma}$ with $k \in [1:K]$ and $\sigma \in \mathcal{S}$. For each (k, σ), we define a distance value between the pose candidate $\chi_{k,\sigma}$ and the permuted E_I^5 by

$$\delta(\chi_{k,\sigma}, E_I^5) := 1/5 \cdot \| E_{\chi_{k,\sigma}} - \sigma E_I^5 \|_2. \tag{8.5}$$

Note that to compute the distance $\delta(\chi_{k,\sigma}, E_I^5)$, we stack the tuples $E_{\chi_{k,\sigma}}$ and σE_I^5 into 15-dimensional vectors, respectively. The result of the database lookup χ_{k^*,σ^*} for frame t is then chosen by also considering temporal consistency using

$$(k^*, \sigma^*) = \underset{(k,\sigma)}{\arg\min}\, \lambda \cdot \delta(\chi_{k,\sigma}, E_I^5) + (1 - \lambda) \cdot d_P(\chi_{k,\sigma}, \chi_{t-1}^*) \tag{8.6}$$

with a weighting factor λ that balances out the influence of d_P (defined in Equation (8.1)) and δ. In our experiments, we use $\lambda = 0.5$. Finally, we refine χ_{k^*,σ^*} to the hypothesis χ_t^{DB} using local optimization as described in Section 8.4.1.

8.4.4 Hypothesis Voting

At this stage, two alternative pose hypotheses have been derived, namely χ_t^{LocOpt} from the generative and χ_t^{DB} from the discriminative component. In the next step, we need to create a single, final pose χ_t^* taking both hypotheses into account. Recall that the pose hypothesis χ_t^{DB} might be inaccurate when the end effectors are not revealed. Therefore, it is not meaningful to take the

average pose of χ_t^{LocOpt} and χ_t^{DB} as final pose. Instead, for this late-fusion step, we propose a novel voting scheme that decides for either χ_t^{LocOpt} or χ_t^{DB} as the final pose χ_t^* based on an efficiently computable sparse and symmetric distance measure. With the proposed voting strategy, the local optimization and database lookup schemes benefit from each other. On the one hand, if the database lookup component fails, then the local optimization component can continue to track the motion. On the other hand, the local optimization might fail to track fast and abrupt motions. In such situations, the database lookup can reinitialize the tracking.

In the proposed voting scheme, we want to avoid a dominant influence of potential errors coming from the feature extraction or from the database lookup. Therefore, we use distance measures that revert to the original input point cloud $\mathcal{M}_I$ rather than to derived data. One possible distance measure could be defined by projecting $\mathcal{M}_\chi$ into a distance image and comparing it to I. In practice, however, because of the relatively low number of pixels in the thin limbs, such a distance measure is dominated by the torso. For this reason, we propose a novel distance metric that can be computed efficiently and that accounts for the importance of the limbs for pose estimation.

To this end, we combine two sparse distances measures. The first distance expresses how well the mesh is explained by the input data:

$$d_{\mathcal{M}_\chi \to \mathcal{M}_I} := \frac{1}{|C_\chi|} \sum_{v \in C_\chi} \min_{p \in \mathcal{M}_I} \|p - v\|_2. \tag{8.7}$$

Here, we revert to only the subset $C_\chi \subseteq \mathcal{M}_\chi$ of vertices as defined in Section 8.4.1, see also Figure 8.6 (a). Likewise, the second distance measure expresses how well $\mathcal{M}_I$ is explained by $\mathcal{M}_\chi$:

$$d_{\mathcal{M}_I \to \mathcal{M}_\chi} := \frac{1}{20} \sum_{n \in [1:20]} \min_{v \in \mathcal{M}_\chi} \|e_I^n - v\|_2. \tag{8.8}$$

To emphasize the importance of the limbs, we take only the first 20 geodesic extrema of the input depth data, which largely correspond to points on the limbs rather than the torso, see Figure 8.10. Since also for the mesh we take only a subset of vertices, see Figure 8.6 (a), the distance measures are sparse. Both distance measures can be computed efficiently because firstly, geodesic extrema can be extracted very efficiently (Section 8.4.2), and secondly, only a small number of points are taken into account. The final pose χ_t^* is then given through

$$\chi_t^* := \operatorname*{arg\,min}_{\chi \in \left\{\chi_t^{\text{DB}}, \chi_t^{\text{LocOpt}}\right\}} (d_{\mathcal{M}_\chi \to \mathcal{M}_I} + d_{\mathcal{M}_I \to \mathcal{M}_\chi}). \tag{8.9}$$

8.5 Experiments

We implemented the proposed hybrid tracking strategy in C++ and ran our experiments on a standard off-the-shelf desktop PC with a 2.6 GHz CPU. To numerically evaluate and to compare our hybrid strategy with previous work, we used the publicly available benchmark dataset of [Ganapathi *et al.*, 2010]. In this dataset, 28 sequences of ToF data (obtained from a Mesa Imaging SwissRanger SR 4000 ToF camera) aligned with ground truth marker positions (obtained from a marker-based motion capture system) are provided. This dataset comprises 7900 frames in total. In addition to numerically evaluating on this dataset, we demonstrate the effectiveness of the proposed algorithm in a real-time scenario with fast and complex motions captured from a PMD

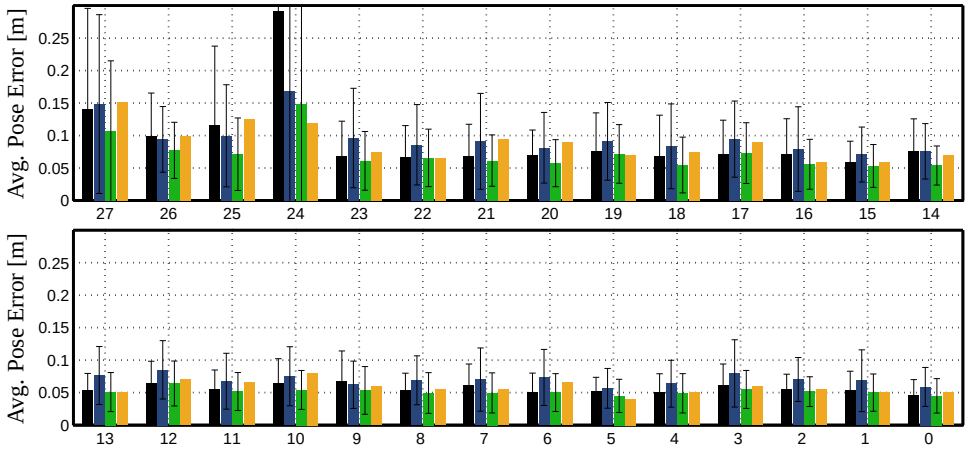


Figure 8.12. Average pose error and standard deviation of sequences 27 to 0 of the dataset of [Ganapathi *et al.*, 2010]. Bars left to right: Using only local optimization, using only the database lookup, results using the proposed fusion scheme, and values reported by [Ganapathi *et al.*, 2010] (without standard deviations).

Camcube 2 in a natural and unconstrained environment, see Figure 8.13 and Figure 8.14. In the accompanying video [Baak *et al.*, 2011a], we show that the same framework also works with the Microsoft Kinect depth sensor without any further adjustments.

8.5.1 Feature extraction

First, we evaluate the effectiveness of the proposed feature extractor on the benchmark dataset. Not all ground truth markers in all frames are visible, thus, for this evaluation, we use only the 3992 frames in which all five end effector markers are visible. A good recognition performance of the feature extractor is needed for a successful subsequent database lookup. In 86.1% of the 3992 frames, each of the found five geodesic extrema E_I^5 is less than 0.2 meters away from its corresponding ground truth marker position. This shows that we can effectively detect the end effector positions for most motions contained in the test dataset.

8.5.2 Quantitative evaluation

We run our motion reconstruction algorithm on the benchmark dataset. Since the surface mesh of the actor is not part of that dataset, we scale the input point cloud data so that it roughly fits the proportions of our actor. We manually fix correspondences between each motion capture marker and a mesh vertex. For a test sequence with T frames, let M_t be the number of visible motion capture markers in frame t, let $m_{t,i}$ be the 3D position of the i^{th} visible marker in frame t and $\tilde{m}_{t,i}$ the position of the corresponding mesh vertex of the reconstructed pose. As also used in [Ganapathi *et al.*, 2010], the average pose error for a sequence is computed as

$$\bar{\epsilon}_{\text{avg}} := \frac{1}{\sum_{t=1}^{T} M_t} \sum_{t=1}^{T} \sum_{i=1}^{M_t} \|m_{t,i} - \tilde{m}_{t,i}\|_2. \tag{8.10}$$

	Total	Preparation	Local Optim.	E_I^{20}	Lookup	Voting
Full resolution	16.6 ms	1.2 ms	5.7 ms	6.2 ms	1.2 ms	0.9 ms
	100%	7%	34%	37%	7%	5%
Half resolution	10.0 ms	1.1 ms	4.6 ms	1.5 ms	1.2 ms	0.9 ms
	100%	11%	46%	15%	12%	9%

Table 8.1. Average run times in milliseconds over all frames of the benchmark dataset.

Whereas the overall accuracy of the tracking algorithm is expressed by means of Equation (8.10), potential local tracking errors can be averaged out. Therefore, we use this evaluation measure to show tendencies in the accuracy by comparing different pose estimation strategies for all benchmark sequences, see Figure 8.12. To this end, we report how the local optimization component (Section 8.4.1) and the database lookup component (Section 8.4.3) perform individually, without being combined with the late-fusion hypothesis voting. When using only local optimization (1st bar) the method often gets stuck in local minima and loses track. When using only a database lookup (2nd bar), poses where the end effectors are not revealed by the first five geodesic extrema may cause a false lookup result. Thus, in terms of the average pose error, both methods, when run separately, do not perform well on all sequences. The 3rd bar shows the result of the proposed hybrid strategy which leads to substantial improvements. Also in comparison to [Ganapathi *et al.*, 2010] (last bar, std. dev. values were not available), we achieve comparable results for basic motions and perform significantly better in the more complex sequences 20 to 27. Only for sequence 24, the method [Ganapathi *et al.*, 2010] performs better than our approach. The reason for this is that this sequence contains a 360° rotation around the vertical axis, which cannot be handled by our framework. However, our system can cope with rotations in the range of ±45° since we normalize the input data based on the estimated viewing direction. For the benchmark dataset, the hypothesis voting component decided in 22.5 % of the frames for the retrieval component, and in 77.5 % for the local optimization from the previous frame. With our hypothesis voting, we significantly reduced the average pose error of the final pose estimate in comparison to either method ran individually.

8.5.3 Run time

In Table 8.1, we report the average run times of our motion reconstruction framework in milliseconds per frame. In [Ganapathi *et al.*, 2010], the authors report a performance of 4 FPS on downsampled input data. By contrast, with our proposed algorithm, we achieve 60.4 FPS (16.6 ms per frame) on average on the full resolution input data, and 100 FPS (10.0 ms per frame) with half of the resolution which we track with nearly the same accuracy. The run times are comparable to or even better than other state-of-the art approaches like [Shotton *et al.*, 2011] where the authors report "at least 10×" speedup with respect to [Ganapathi *et al.*, 2010]. As for a more detailed analysis, we also give the run time of each algorithmic component, namely the data preparation phase (Section 8.3), the local optimization component (Section 8.4.1), the feature extraction (Section 8.4.2), the database lookup (Section 8.4.3), and the voting (Section 8.4.4). Note that our efficient algorithms lead to run times that are well distributed among the different components, such that no clear bottleneck is present. For the full resolution, the run time of local optimization and the feature extraction are approximately the same. The latter benefits most from downsampling the data.

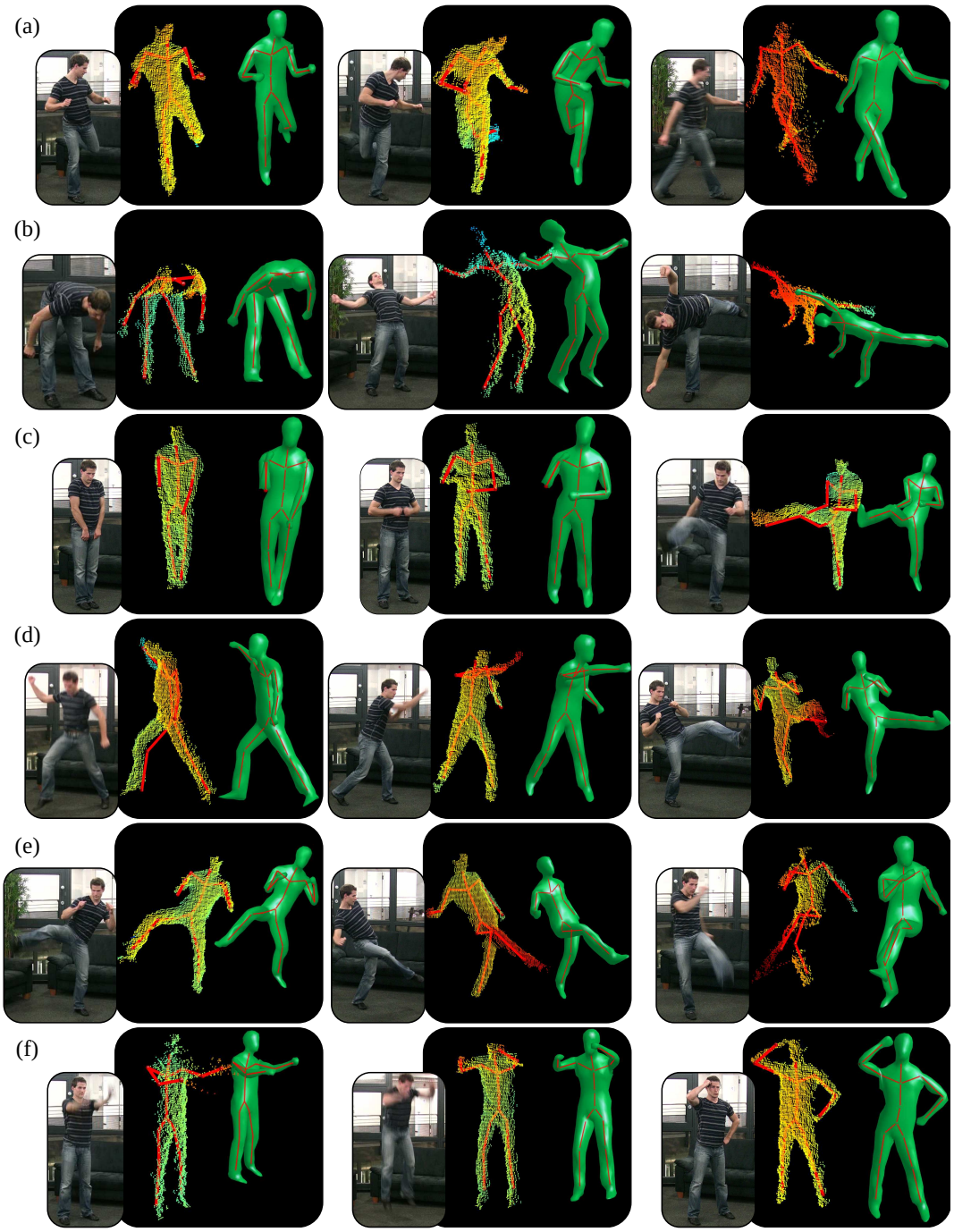


Figure 8.13. Snapshots of our results on fast and complex motions on data captured with a PMD camera. For every motion we show a video frame of the actor (not used for tracking), ToF data overlaid with the reconstructed skeleton, and a rendering of the corresponding mesh.

8.5.4 Qualitative evaluation

In Figure 8.13, we show example results of fast and complex motions captured in an unconstrained environment. The considered motions are much faster and contain more challenging poses than the

ones used in [Ganapathi *et al.*, 2010]. The leftmost image in each subfigure shows a video frame of the motion captured from a separate video camera not used for the motion reconstruction. In the room where we recorded the data, the video camera was standing to the left of the depth camera. The middle image shows the depth data overlaid with the estimated skeleton of the pose χ_t^*. The rightmost image depicts a rendering of the surface mesh in the corresponding pose.

The first row (Figure 8.13 (a)) shows some frames of a successfully tracked motion sequence. Even though the left foot is bent to the back and is nearly hidden from the depth camera, the 3D geometry of the legs has been recovered correctly. Figure 8.13(b) depicts difficult bending motions. Despite of the fact that such poses were not part of our pose database, the motions were tracked successfully. For such motions, the result χ_t^{DB} of the lookup step does not reflect the true pose of the actor. Thus, our voting scheme decided in each frame correctly for the local optimization component which successfully tracked the motions. Figure 8.13(c) contains typical failure cases. The first two images show poses with severe self-occlusion which are still a challenge for motion reconstruction. Nonetheless, the overall pose is reliably captured and arm tracking quickly recovered once the occlusion was resolved. The rightmost image shows a case where the right arm was not visible in the depth input data. Since the proposed method assumes that at least parts of all limbs are still visible in the depth data, the pose of the right arm is not correctly recovered. Figure 8.13(d) shows examples of fast jumping, punching, and kicking motions where the first two motions are additionally rotated to more than $\pm 45^\circ$ around the vertical axis with respect to the frontal viewing direction. The poses in this row are roughly recovered. However, small misalignments of some limbs might occur as visible in the right leg and the right arm, respectively. Also note the inaccuracy in the left leg (third pose). Such minor inaccuracies can locally occur and are typically resolved after a few frames. Figure 8.13(e) shows some poses of a successful reconstruction of a sequence with fast and complex kicking motions. Note that in the second pose of Figure 8.13 (e) it is difficult to distinguish the left leg from the right leg when having only the depth data of a single frame. However, since the local optimization and the database lookup components use temporal continuity priors, the legs can be tracked successfully. Finally, Figure 8.13 (f) contains a very fast arm rotation motion in a pose where the arms are close to being outstretched to the camera (first image), and a jumping motion in a similar pose (second image). Although only a small part of the arm is visible to the depth camera due to self-occlusions, the 3D geometry of the arm is successfully recovered. The last image shows a pose where the hands touch different parts of the body. Despite of the fact that in such poses not all geodesic extrema in E_I^5 correspond to the end effectors, the motion has been tracked successfully since the voting scheme decided for the local optimization component. In the accompanying video [Baak *et al.*, 2011a] we show the performance of our prototype implementation also with the Microsoft Kinect depth camera.

First experiments showed that actors with different body proportions can be tracked if they are not too different from our body model. Therefore, we scaled the input data to roughly match the proportions of the model, see Figure 8.14 and the accompanying video for examples.

8.5.5 Limitations

In the proposed framework, we rely on certain model assumptions in several stages of the framework. For example, we use a rigged surface mesh that is assumed to fit the respective actor to be tracked. Therefore, we cannot directly track persons with substantially different body proportions than the ones reflected by the surface mesh. However, as shown in [Weiss *et al.*, 2011], the depth

Figure 8.14. Experimental results with a different person (ToF data from a PMD camera).

input can be used to estimate the body shape of the actual person. First experiments have shown that one can use the same motion reconstruction pipeline after applying a preprocessing step where the pose database is retargeted to correspond to the estimated body shape.

A second limitation arises in situations where only parts of the actor are visible in the field of view of the depth camera. Two assumptions within our framework lead to false pose estimates in these situations. Firstly, in the local optimization component, correspondences between the mesh and the depth data for all body parts of the mesh are established. If some limbs are not visible in the depth data, then the correspondences will inherently be semantically incorrect. Secondly, the geodesic extrema will not correspond to the limbs anymore and retrieved database poses can no longer stabilize the pose estimation. Therefore, the full body of the actor should always be visible in the depth data.

Another problematic situation can occur when the end effectors are not revealed for a longer period of time. Although we run two pose estimation components in parallel, each component in isolation does not give satisfying pose estimates as shown in the accompanying video—it is the combination that facilitates stable and accurate results. Therefore, if one of the components fails for an extended period of time, the results might become unstable. For example, if the end effectors are not revealed by the geodesic extrema for many successive frames, our algorithm continues to track using only local optimization. Then, fast motions lead to unstable pose estimation results, which are resolved as soon as the end effectors are detected again. To overcome this limitation, additional techniques for detecting end effectors could be employed. For example, a fast method for detecting body parts similar to [Shotton *et al.*, 2011] could be used to identify the end effectors and supplement the geodesic extrema detections.

Finally, we expect the user to face the camera and rotate the body only within a typical range for interaction ($\pm 45^\circ$). We make use of this assumption in the normalization step in Section 8.3.4. By normalizing the depth data with respect to the estimated viewing direction, we can use a comparatively small pose database that contains each pose only in one normalized orientation. However, with our proposed method, the estimates for the viewing direction become unstable once the user leaves the admissible range of rotations. Since the database lookup component relies on a correct normalization of the input data, the retrieved pose hypotheses will not reflect the true pose in such

cases. Another problem with strong rotations of the body is that then limbs are more likely hidden behind the body. To meliorate pose estimates in these situations, one could employ a dynamic model for simulating hidden limbs.

8.6 Conclusions

In this chapter, we introduced a combined generative and discriminative framework that facilitates robust as well as efficient full-body motion reconstruction from noisy depth image streams. As one main ingredient, we described an efficient algorithm for computing robust and characteristic features based on geodesic extrema. These extracted geodesic extrema are used as query to retrieve semantically meaningful pose candidates from a 3D pose database, where no a-priori semantic labels of the extrema are necessary. Finally, a stable fusion of local optimization and global database lookup is achieved with a novel sparse distance measure that also accounts for the importance of the limbs. For all components of the pipeline, we have described efficient algorithms that facilitate real-time performance of the whole framework. In our experiments we improved on the results of previous work, both in terms of efficiency and robustness of the algorithm, as well as complexity of the tracked motions.

As for future work, we plan to integrate a dynamic model for achieving stable motion reconstruction also for 360°-rotations and for occluded limbs. Furthermore, the fast run time of our method is one main building block that could spur further research for capturing several interacting people. Finally, we aim to integrate a method for automatically estimating the surface mesh of the person from depth data only.

Chapter 9

Conclusions

Analyzing, tracking, and reconstructing human motions constitute important topics in computer vision and computer graphics. In the analysis, one main challenge comes from the fact that semantically similar motions may exhibit significant spatial and temporal variations. Although warping and alignment techniques can effectively cope with temporal variations, they may be prohibitive when dealing with large collections of mocap data due to complex computations and high memory requirements. Further challenges can be found in the tracking and reconstruction tasks, where a 3D representation of the human pose from camera- or other sensor data is to be estimated. In particular when using a small number of cameras, the task is considerably under-constrained. Moreover, noise in the underlying data, fast motions, or self-occlusions render the problem even more difficult.

In this thesis, we introduced novel data-driven algorithms to effectively tackle challenges in motion analysis, tracking, and reconstruction. As one main contribution, we developed a robust framework for content-based annotation of unstructured collections of mocap data. In particular, we introduced a novel keyframe-based search algorithm which can cope with significant temporal variations in the motions. By applying this algorithm in the annotation framework, we were able to efficiently handle large collections of mocap data. As a further contribution, we developed novel data-driven approaches for marker-less tracking and reconstruction. As common underlying methodology, we dynamically extracted and integrated prior knowledge at runtime from a database of mocap data using content-based retrieval techniques. In this way, we significantly increased the accuracy of the tracking and effectively prevented tracking errors. The application of novel sensor types in motion analysis and reconstructions represents another main contribution of this thesis. Here, we showed how inertial sensors can be used for effectively detecting tracking errors. Moreover, we introduced a novel approach for data-driven motion reconstruction from a single depth camera which greatly improved accuracy, stability, and efficiency in comparison to previous state-of-the-art algorithms. Efficiency issues played a major role in all parts of the thesis. We developed and implemented efficient content-based techniques for retrieval and annotation of large mocap collections. Furthermore, in the last part of the thesis, we introduced efficient algorithmic components for data-driven motion reconstruction that lead to an overall run time of about 16 milliseconds per frame, which exceeds the requirements for real-time capability.

There are several interesting directions of research for future work indicated by this thesis. Challenging open problems for mocap annotation can be found for the analysis of gestures. Here, the

automated temporal segmentation into the basic movement phases, which are represented by the succession preparation—hold—stroke—hold—retraction, is still an open problem. Also, detecting and automatically labeling different types of gestures is a difficult research topic. The main challenges come from the high inter- and intra-person variability for the same type of gesture. Also, the extreme differences in the temporal extend of different gestures increases the difficulty of the annotation task, where some gestures last for several seconds (for example, a 'calm' gesture with an extended 'hold'-phases), and other gestures might only last for a fraction of a second (for example, a 'pointing' gesture).

Invariance to different motion style represents one main challenge of robust mocap annotation procedures. As a complementary goal, strategies for identifying a specific actor in mocap data could be explored in future work. Here, the extraction of motion nuances plays an important role. Such automated analyses may be applied in areas such as medical rehabilitation, where one goal is to detect anomalies in motion style. Moreover, the amount of deviation from a normal gait could be measured.

As for the data-driven stabilization of tracking, we explored a method for integrating retrieved motion priors from a mocap database. In this context, techniques for automatically enhancing the database knowledge in an online learning manner could be subject to future research. In one possible scenario, motions which are correctly tracked with a high confidence could be integrated into the database. Therefore, a suitable confidence measure would have to be developed that correctly predicts whether a motion contains tracking errors. Here, special care has to be taken in order to avoid that false predictions lead to an inclusion of motions with tracking errors into the database.

Our approach for real-time tracking from a depth camera may open numerous applications in the areas of human computer interaction, virtual reality, medical rehabilitation, or sports sciences. However, some applications require even more robustness and stability than achieved with current state-of-the-art algorithms. For example, in our current approach, limbs that are not visible in the depth data (*e. g.*, because of self-occlusions), might be mapped to the nearest visible body part. One possible approach for improving tracking in such situations might include a bottom-up detection of limb visibility with an approach similar to [Shotton *et al.*, 2011]. Then, limbs could be included or excluded from the motion reconstruction based on the detected visibility. As a further improvement, approaches that are also capable of reconstructing 360°-rotations could be explored. At least two problems would have to be solved for this task. Firstly, rotations around the longitudinal body axis inevitably lead to strong occlusion of the limbs. Secondly, the front/back ambiguity would have to be solved: for many body shapes, the depth data for a pose facing the camera is similar to the depth data for a corresponding pose facing away from the camera. One possible solution for this problem could exploit temporal continuity assumptions. However, human motions can be abrupt and unpredictable, where continuity assumptions might fail. Therefore, the integration of knowledge derived from additional sensors could assist this task. For example, a color camera could be used to estimate whether the front or the back of the person is visible based on a detection of skin color in the region of the head. Another solution could be found by combining motion reconstruction based on a depth camera with an additional inertial sensor. When placed on the back of the person, the sensor could be used to resolve the front/back ambiguity. Moreover, orientation data from the inertial sensor may lead to a stabilized detection of the viewing direction, which could lead to improved pose reconstructions also in challenging body orientations.

A further extension is motivated by body-controlled computer games. In order to control a game, a full 3D pose reconstruction of the player might not always be required. However, the raw depth input data might not carry enough information to control the game. Instead, the progression of the game might be controlled by a mid-level representation comprising semantic motion classes. For example, the input of a simple rafting game could consist of the motion classes “walk to the left”, “walk to the right”, or “jump”, which could be detected from the player’s motion. One advantage of such a strategy is that local tracking errors which still permit a correct classification of the motion would not influence the gameplay. In the game, the motions of the virtual character could be played back based on the mid-level motion class input using previously recorded mocap data. In particular, the displayed motion of the virtual character would only resemble the rough order of motion events instead of exactly copying the poses of the player. Motivated by such a scenario, classification methods similar to the ones in Part I could be extended and combined with methods for real-time motion reconstructions as described in Part III in order to build a real-time input controller for computer gaming applications.

Appendix A

Inertial Measurement Unit

An inertial measurement unit (IMU) consist of two basic sensors, namely a 3D accelerometer and a 3D angular rate sensor. Often, such an IMU is enhanced by an additional magnetic field sensor. In this appendix, we describe the different sensors contained in an IMU (Sections A.1–A.3) and briefly motivate how orientation data is commonly derived from the measured data (Section A.4).

As a prerequisite, we first define a global, fixed coordinate system of the inertial world F^{GI}, see Figure A.1. In this coordinate system, the X-axis points towards the north pole, the Z-axis is the vertical axis pointing upwards, and the Y-axis is chosen to yield a right-handed orthonormal coordinate system. The coordinate system is defined in this way since an inertial sensor is able to measure the direction towards north using a magnetometer as well as the upwards direction using the accelerometer. The orientation of the sensor is defined as a rotation q^{I} that rotates the basis vectors of F^{LI} to the basis vectors of F^{GI}.

A.1 Magnetic field sensor

In the process of estimating the orientation of an IMU, the magnetic field sensor plays a crucial role. In the following, we will explain that the magnetic field sensor allows an IMU to estimate

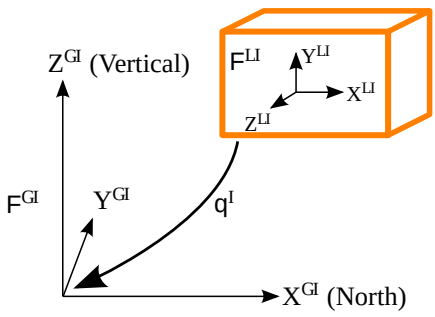


Figure A.1. Relation between the global inertial coordinate system F^{GI} and the local sensor coordinate system F^{LI}

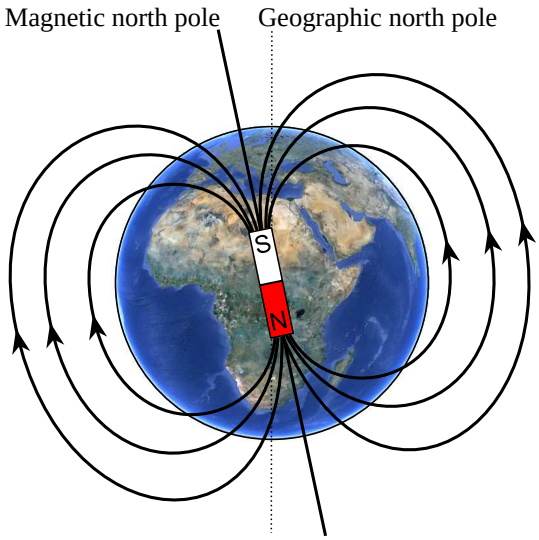


Figure A.2. Sketch of the earth magnetic field. Underlying image of the earth obtained with Google Earth. Copyright note: Google earth, ©2012, TerraMetrics, ©2012, Cnes/Spot Image.

a fixed reference direction. As sketched in Figure A.2, the magnetic field vectors on the surface of the earth are observed as horizontal vectors parallel to the ground near the equator, and are perpendicular to the surface near the magnetic poles. The magnetic field sensor is a 3-dimensional compass that measures the direction of the magnetic field. For estimating a stable orientation of the sensor, we need to estimate the direction towards the magnetic north pole from the measured magnetic field. Note that this is only possible when being sufficiently far away from the magnetic poles.

The term “magnetic north pole” is confusing in the sense that it actually describes the south pole end of a bar magnet approximation of the earth magnetic field, see Figure A.2. The reason for this is historic: magnets are used in a compass to determine the direction towards north. More specifically, the needle of a compass itself consists of a bar magnet. The north pole end of this needle is labeled with “N”. Since the north pole end of one magnet attracts the south pole end of another magnets, the “N” on the compass needle is attracted by magnetic south pole of the earth. Thus, the north pole on earth should more correctly be named “north-seeking” pole. In the following, we will choose the conventional naming of the north pole although it is opposed to its physical interpretation.

The direction and magnitude of the magnetic field vectors mostly depend on the latitudinal coordinates on earth. For example, on the equator, the direction of the magnetic field vectors would be approximately parallel to the ground, pointing towards the magnetic north pole. The magnetic field vector $\boldsymbol{m}$ is commonly described by the inclination and declination angles (φ and θ) as well as the magnitude $\|\boldsymbol{m}\|_2$ in Tesla units. The inclination φ depicts the angle with respect to the horizontal plane, where positive values represent the downwards direction, and negative values represent the upwards direction. The declination θ is defined as the angle of difference between true geographic north and the magnetic north. The meaning of a positive declination is that the true geographic north is located west of the measured magnetic north. All three quantities change slowly in time and can be looked up in the database of the National Oceanic and Atmospheric Administration (NOAA) [2012] for a given location and time. For Saarbrücken in 2012, the corresponding values

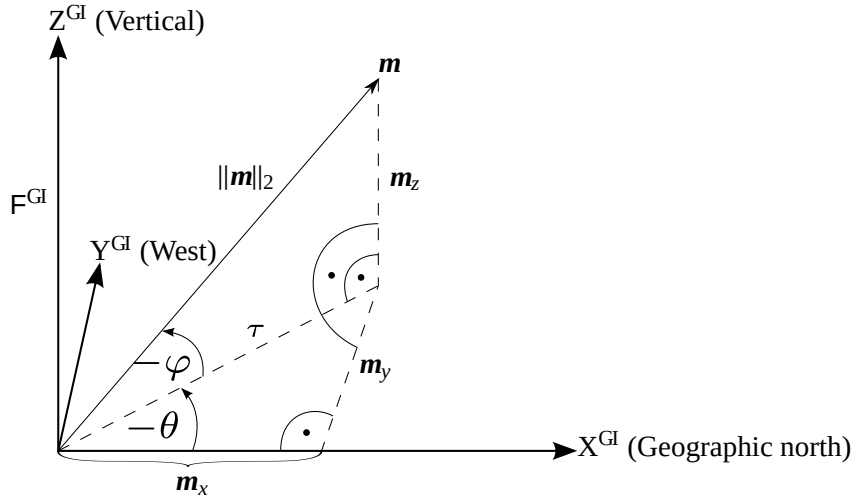


Figure A.3. Computation of the magnetic field vector $\boldsymbol{m}$ in the global inertial coordinate system F^{GI} using the inclination and declination angles.

are

$$\varphi = 64^\circ\ 53' = -66.883^\circ \tag{A.1}$$

$$\theta = 1^\circ\ 8' = 1.133^\circ \tag{A.2}$$

$$\|\boldsymbol{m}\|_2 = 48.255\,\mu\mathrm{T} \tag{A.3}$$

For an algorithm with the goal of estimating the global coordinate system F^{GI} by using the magnetic field vector $\boldsymbol{m}$ it is crucial to know how to mathematically express the measured vector $\boldsymbol{m}$ in F^{GI}. In the following, we show how $\boldsymbol{m}$ given in the global coordinate system F^{GI} can be easily computed from the quantities inclination, declination, and magnitude. In Figure A.3, we draw a sketch of the geometric relations of the quantities with respect to F^{GI}. The inclination is depicted as $-\varphi$ since positive values are defined as pointing downwards in the definition of [NOAA, 2012], and we want it to point upwards in the sketch. Similarly, the declination is depicted as $-\theta$ since we draw it in the direction opposite to its definition. From elementary geometry, we can determine the components of the magnetic field vector as follows:

$$\boldsymbol{m}_z = \|\boldsymbol{m}\|_2 \sin -\varphi = -\|\boldsymbol{m}\|_2 \sin\varphi \tag{A.4}$$

$$\tau = \|\boldsymbol{m}\|_2 \cos -\varphi = \|\boldsymbol{m}\|_2 \cos\varphi \tag{A.5}$$

$$\boldsymbol{m}_x = \tau \cos -\theta = \|\boldsymbol{m}\|_2 \cos\varphi \cos\theta \tag{A.6}$$

$$\boldsymbol{m}_y = \tau \sin -\theta = \|\boldsymbol{m}\|_2 \cos\varphi \sin -\theta = -\|\boldsymbol{m}\|_2 \cos\varphi \sin\theta. \tag{A.7}$$

Using these computations, the ground truth vector of the earth magnetic field expressed in the coordinate system F^{GI} can be computed. The magnetic field sensor can estimate this direction expressed in its local coordinate system F^{LI}. By claiming correspondence of the ground truth direction expressed in F^{GI} and the estimated direction expressed in F^{LI}, the directional information can be used as a constraint for estimating the rotational offset q^{I}. However, a directional constraint alone does not carry enough information to fully determine q^{I}. Also, the magnetic field sensor reacts comparatively slow on changes of the IMU's orientation since the strength of the earth magnetic field is low. Moreover, the earth magnetic field is often disturbed by ferromagnetic

materials or electronic devices which emit an electromagnetic field. Additionally, environments can even be completely shielded against the influence of the earth magnetic field. Therefore, in practice, special care has to be taken when incorporating these measurements for estimating the orientation of the device. For example, if the measured magnitude of the magnetic field vector deviates from the ground truth magnitude, then it is likely that the earth magnetic field is either shielded or superimposed by disturbing fields. Then, the measurements should not have a strong influence on the overall estimated orientation of the device. In an IMU, additional sensors as presented in the next sections also have an influence on the estimated orientation q^{I}.

A.2 Angular rate sensor

The angular rate sensor measures the change of the orientation q^{I} over time, *i. e.*, the first temporal derivative of q^{I}. Thus, given an initial orientation, the measurements can be used to update the orientation of the sensor over time. Although theoretically this sensor alone yields enough information to update the orientation estimates over time, noise in the measurements quickly leads to drift in the orientation estimates.

In the following, we show how one can perform computations with rotational velocities. Therefore, we derive an algorithm that can be used to compute rotational velocities (as measured by an inertial sensor) from a given stream of orientations q^{I}—similar computations can be applied for the opposite task of updating orientations given a stream of rotational velocities. The vector $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^T$ represents the 3-dimensional angular velocity, *i. e.*, the sensor's angular velocity around its X, Y, and Z axis. By $\hat{\boldsymbol{\omega}} = (0, \omega_x, \omega_y, \omega_z)^T$, we embed the angular velocity into a quaternion with a zero real part. In physics books, the definition of the angular velocity in the time-continuous case is noted as:

$$\dot{\mathsf{q}} = \frac{d\mathsf{q}(t)}{dt} = \frac{1}{2}\hat{\boldsymbol{\omega}}(t)\mathsf{q}(t) \qquad \text{with} \tag{A.8}$$

$$\hat{\boldsymbol{\omega}}(t) = 2\frac{d\mathsf{q}(t)}{dt}\overline{\mathsf{q}(t)} \tag{A.9}$$

Let us transform this equation into the discrete time case using a sampling rate of $1/\Delta t$. Let $\mathsf{q}(t - \Delta t)$ and $\mathsf{q}(t)$ be the orientations of the previous and the current sampled instances of time, respectively. Then, the derivative discretizes to

$$\frac{d\mathsf{q}(t)}{dt} \approx \frac{\mathsf{q}(t) - \mathsf{q}(t - \Delta t)}{\Delta t}. \tag{A.10}$$

Now, in order to extract the rotational velocity (a 3-dimensional vector) from the 4-dimensional quaternion, we extract the imaginary components using the $\mathrm{Im}(\cdot)$ operator:

$$\boldsymbol{\omega}(t) = 2\,\mathrm{Im}\left(\frac{\mathsf{q}(t) - \mathsf{q}(t - \Delta t)}{\Delta t} \circ \overline{\mathsf{q}(t - \Delta t)}\right) \tag{A.11}$$

Note that the difference in the numerator is implemented as a pointwise difference of the scalar

Algorithm 5 Computation of rotational velocity

Input: $\mathsf{q}(t)$: Current orientation, mapping from F^{LI} to F^{GI}.
$\mathsf{q}(t-\Delta t)$: Previous orientation, mapping from F^{LI} to F^{GI}.
Δt: Time that has passed between the current and the previous orientation.
Output: ω: Rotation that maps from the previous time to the current time, given in F^{LI}.

1: **if** $\langle \mathsf{q}(t), \mathsf{q}(t-\Delta t)\rangle < 0$ **then**
2: $\quad \mathsf{q}(t) \leftarrow (-\mathsf{q}(t))$
3: **end if**
4: $\omega \leftarrow \frac{2}{\Delta t} \operatorname{Im}\left(\mathsf{q}(t) \circ \overline{\mathsf{q}(t-\Delta t)}\right)$

components of the quaternions. Equation (A.11) simplifies to

$$\omega(t) = \frac{2}{\Delta t} \operatorname{Im}\left(\mathsf{q}(t) \circ \overline{\mathsf{q}(t-\Delta t)} - \mathsf{q}(t-\Delta t) \circ \overline{\mathsf{q}(t-\Delta t)}\right) \tag{A.12}$$

$$= \frac{2}{\Delta t} \operatorname{Im}\left(\mathsf{q}(t) \circ \overline{\mathsf{q}(t-\Delta t)} - (1,0,0,0)\right) \tag{A.13}$$

$$= \frac{2}{\Delta t} \operatorname{Im}\left(\mathsf{q}(t) \circ \overline{\mathsf{q}(t-\Delta t)}\right) \tag{A.14}$$

The last simplifications is possible since the subtraction of the quaternion identity does not change imaginary components extracted by the $\operatorname{Im}(\cdot)$ operator.

Care has to be taken when the quaternion difference $\mathsf{q}(t) \circ \overline{\mathsf{q}(t-\Delta t)}$ is implemented. The reason is that there are two quaternions q and $-\mathsf{q}$ that represent the same rotation in the sense that when q or $-\mathsf{q}$ are used to rotate a vector, they yield the same rotated vector. However, they represent different rotation paths in the sense that they rotate about a different axis with a different angle. Assuming that q represents a rotation about the axis $\boldsymbol{r}$ with an angle α, then $-\mathsf{q}$ represents a rotation about $-\boldsymbol{r}$ with an angle $2\pi - \alpha$ (or $-\alpha$):

$$\mathsf{q} = (\cos\frac{\alpha}{2}, \boldsymbol{r}\sin\frac{\alpha}{2}) \tag{A.15}$$

$$-\mathsf{q} = (-\cos\frac{\alpha}{2}, -\boldsymbol{r}\sin\frac{\alpha}{2}) \tag{A.16}$$

$$= (\cos\frac{2\pi-\alpha}{2}, (-\boldsymbol{r})\sin\frac{\alpha}{2}). \tag{A.17}$$

This shows its effect when computing the mentioned quaternion difference. Only if the dot product between both quaternions is larger than 0, the quaternion difference reflects the shortest possible rotation. With a sufficiently small temporal quantization Δt, the shortest possible rotation will always reflect the correct representation of the rotational velocity. The correct way to compute the rotational velocity in local sensor coordinates is summarized in Algorithm 5.

A.3 Accelerometer

The linear acceleration of a rigid body is the second temporal derivative of the positional trajectory of its center of mass, given in the continuous time case as

$$\widetilde{\boldsymbol{a}} = \dot{\boldsymbol{v}} = \ddot{\boldsymbol{x}} \tag{A.18}$$

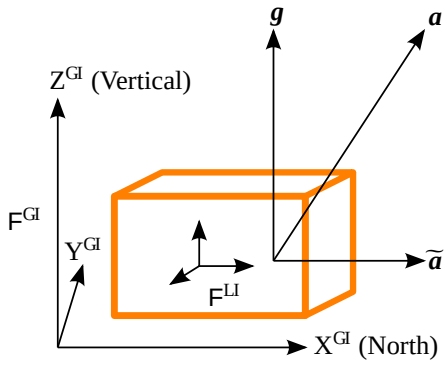


Figure A.4. The measured acceleration $\boldsymbol{a}$ is an overlay of the acceleration due to the motion $\tilde{\boldsymbol{a}}$ and the acceleration due to gravity $\boldsymbol{g}$.

In the following, we show how the gravity affects the acceleration readings of an acceleration sensor in order to motivate how one can make use of the gravity for the task of estimating the orientation of the IMU. To this end, we derive equations in the discrete time case for simulating sensor readings given the trajectory of its center of mass in coordinates of F^{GI}.

First, Equation (A.18) has to be discretized. The derivative can be discretized in different ways, in the simplest case by a one-sided difference. Let the position of the center of mass of the sensor $\boldsymbol{p}(t)$ be given in coordinates of F^{GI}. In the following, we use the superscript $^{\mathrm{GI}}$ to indicate that the computed velocity and acceleration are given in F^{GI} as well. We derive the formula for computing the acceleration of the motion $\tilde{\boldsymbol{a}}^{\mathrm{GI}}$:

$$\boldsymbol{v}(t)^{\mathrm{GI}} = \frac{\boldsymbol{p}(t) - \boldsymbol{p}(t-\Delta t)}{\Delta t} \tag{A.19}$$

$$\boldsymbol{v}(t-\Delta t)^{\mathrm{GI}} = \frac{\boldsymbol{p}(t-\Delta t) - \boldsymbol{p}(t-2\Delta t)}{\Delta t} \tag{A.20}$$

$$\tilde{\boldsymbol{a}}_t^{\mathrm{GI}} = \frac{\boldsymbol{v}(t)^{\mathrm{GI}} - \boldsymbol{v}(t-\Delta t)^{\mathrm{GI}}}{\Delta t} \tag{A.21}$$

$$= \frac{\boldsymbol{p}(t-2\Delta t) - 2\boldsymbol{p}(t-\Delta t) + \boldsymbol{p}(t)}{\Delta t^2}. \tag{A.22}$$

An accelerometer measures an overlay of the acceleration $\tilde{\boldsymbol{a}}^{\mathrm{GI}}$ due to motion and the acceleration due to gravity $\boldsymbol{g}^{\mathrm{GI}} \approx (0, 0, 9.81)^T \ \mathrm{m/s^2}$, see Figure A.4.

$$\boldsymbol{a}^{\mathrm{GI}}(t) = \tilde{\boldsymbol{a}}^{\mathrm{GI}}(t) + \boldsymbol{g}^{\mathrm{GI}} \tag{A.23}$$

Note that at first sight the measurement $\boldsymbol{g}^{\mathrm{GI}} \approx (0, 0, 9.81)^T \ \mathrm{m/s^2}$ seems to be unintuitive since the measured vector points upwards although the gravitational field on earth points towards the center of the earth (downwards). This can be explained as follows. The measured component of the gravity is zero if, *e. g.*, the sensor is far away from any gravitational influence in space. Also, if the sensor within the gravitational field of the earth falls down in a perfect vacuum, the measured gravity component will be zero. However, in a non-vacuum, the air friction slows down falling objects (resulting in a measured upwards acceleration) and as soon as the terminal velocity of a falling object is reached, the full gravitational component $\boldsymbol{g}^{\mathrm{GI}}$ (in upwards direction) will be measured. Exactly the same situation is present when the sensor lies on a static surface—the upwards acceleration that prevents the sensor from falling down will be measured.

As shown in [Boynton, 2001], the magnitude of the gravity on earth is not constant but varies mainly with latitude and altitude. Therefore, in order to simulate effects of the gravity on the sensor, the exact magnitude of the vector should be looked up. The acceleration in global coordinates $\boldsymbol{a}^{\mathrm{GI}}$ has to be transformed into the local coordinate system of the sensor using the current orientation of the sensor $\mathsf{q}(t)$ to yield the sensor reading $\boldsymbol{a}^{\mathrm{LI}}(t)$. Let $\mathsf{q}[\boldsymbol{v}]$ be the vector $\boldsymbol{v}$ rotated with q. Then,

$$\boldsymbol{a}^{\mathrm{LI}}(t) = \overline{\mathsf{q}(t)}[\boldsymbol{a}^{\mathrm{GI}}(t)]. \tag{A.24}$$

Now, suppose that the acceleration due to motion $\widetilde{\boldsymbol{a}}$ is zero. In this case, only the gravitational component $\boldsymbol{g}^{\mathrm{GI}}$ is measured by the sensor. Then, we know that for the sought orientation $\mathsf{q}(t)$ of the device the property

$$\boldsymbol{a}^{\mathrm{LI}}(t) = \overline{\mathsf{q}(t)}[\boldsymbol{g}^{\mathrm{GI}}] \tag{A.25}$$

has to hold. Since $\boldsymbol{a}^{\mathrm{LI}}(t)$ is measured by the sensor and $\boldsymbol{g}^{\mathrm{GI}}$ is known, Equation (A.25) can give a hint about the orientation of the IMU.

A.4 Orientation Data

For obtaining orientation data, all available measurements should be taken into account. As John L. Crassidis has shown in his survey [2007], there is a multitude of methods for solving this task described in a large body of literature. However, as he points out, the Kalman filter framework "remains the method of choice for the great majority of of applications" [John L. Crassidis, 2007].

In the following, we will give a high-level idea about the Kalman filter and we refer to Welch and Bishop [1995] and references therein for a more in-depth introduction. The Kalman filter is a type of recursive predictor-corrector framework. It is used to estimate the current state vector which can be thought of as being a vector containing the current orientation of the sensor, the current acceleration, the rate of turn and the magnetic field vector. In the first step, the current state is predicted one timestep into the future by means of a user-supplied linear model. This model could in the simplest case assume that the orientation is updated with the current rate of turn data and the remaining state variables stay constant. In the second step, the predicted state is corrected by integrating all available measurements. The corrected state is the output of the current time step and it is used as the starting point for the next time step.

What makes the Kalman filter interesting is the thorough modeling of noise in all steps of the framework. For example, in addition to the available measurements, their uncertainties can be taken into account. For example, if a disturbed magnetic field is detected from the magnetic field sensor, its estimate of the north direction should be integrated with only a very high uncertainty. As a result of modeling the noise, not only the current state vector, but also its covariance is estimated. Therefore, the certainty of the estimated orientation is supplied as an output as well. In fact, it can be shown that a Kalman filter is optimal in the sense that it minimizes the estimated error covariance—when some presumed conditions are met [Welch and Bishop, 2001].

Bibliography

[Agarwal and Triggs, 2004] Ankur Agarwal and Bill Triggs. Learning to track 3D human motion from silhouettes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2–9. ACM, 2004.

[Agarwal and Triggs, 2006] Ankur Agarwal and Bill Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 28(1):44–58, 2006.

[Arikan *et al.*, 2003] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics (TOG)*, 22(3):402–408, 2003.

[Arikan *et al.*, 2005] Okan Arikan, David A. Forsyth, and James F. O'Brien. Pushing people around. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 59–66. ACM, 2005.

[Assa *et al.*, 2005] Jackie Assa, Yaron Caspi, and Daniel Cohen-Or. Action synopsis: Pose selection and illustration. *ACM Transactions on Graphics (TOG)*, 24(3):667–676, 2005.

[Azad *et al.*, 2008] Pedram Azad, Tamim Asfour, and Rüdiger Dillmann. Robust real-time stereo-based markerless human motion capture. In *IEEE/RAS International Conference on Humanoid Robots*, pages 700–707, 2008.

[Baak *et al.*, 2008] Andreas Baak, Meinard Müller, and Hans-Peter Seidel. An efficient algorithm for keyframe-based motion retrieval in the presence of temporal deformations. In *Proceedings of the 1st ACM SIGMM International Conference on Multimedia Information Retrieval (ACM MIR)*, pages 451–458, 2008.

[Baak *et al.*, 2009] Andreas Baak, Bodo Rosenhahn, Meinard Müller, and Hans-Peter Seidel. Stabilizing motion tracking using retrieved motion priors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1428–1435, 2009.

[Baak *et al.*, 2010] Andreas Baak, Thomas Helten, Meinard Müller, Gerard Pons-Moll, Bodo Rosenhahn, and Hans-Peter Seidel. Analyzing and evaluating markerless motion tracking using inertial sensors. In *Proceedings of the 3rd International Workshop on Human Motion. In Conjunction with ECCV.*, volume 6553 of *Lecture Notes of Computer Science (LNCS)*, pages 137–150. Springer-Verlag, 2010.

[Baak *et al.*, 2011a] Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. Accompanied video to [Baak *et al.*, 2011b]. `http://www.youtube.com/watch?v=QWNn01FWUkk`, 2011. [Online; accessed 13-March-2012].

[Baak *et al.*, 2011b] 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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1092–1099, 2011.

[Baak *et al.*, 2013] 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 Andrea Fossati, Juergen Gall, Helmut Grabner, Xiaofeng Ren, and Kurt Konolige, editors, *Consumer*

Depth Cameras for Computer Vision: Research Topics and Applications. Springer-Verlag London (to appear), 2013.

[Baker, 1987] James E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, pages 14–21. L. Erlbaum Associates Inc., 1987.

[Barbič *et al.*, 2004] Jernej Barbič, Alla Safonova, Jia-Yu Pan, Christos Faloutsos, Jessica K. Hodgins, and Nancy S. Pollard. Segmenting motion capture data into distinct behaviors. In *Proceedings of Graphics Interface (GI)*, pages 185–194. Canadian Human-Computer Communications Society, 2004.

[Beaudoin *et al.*, 2007] Philippe Beaudoin, Michiel van de Panne, and Pierre Poulin. Automatic construction of compact motion graphs. Technical report, Université de Montréal, 2007.

[Beaudoin *et al.*, 2008] Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. Motion-motif graphs. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 117–126, 2008.

[Besl and McKay, 1992] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transaction on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[Bleiweiss *et al.*, 2009] Amit Bleiweiss, Eran Kutliroff, and Gershom Eilat. Markerless motion capture using a single depth sensor. In *SIGGRAPH ASIA Sketches*, 2009.

[Bo and Sminchisescu, 2010] Liefeng Bo and Cristian Sminchisescu. Twin gaussian processes for structured prediction. *International Journal of Computer Vision (IJCV)*, 87(1-2):28–52, 2010.

[Boynton, 2001] Richard Boynton. Precise measurement of mass. In *Proceedings of the 60th Annual Conference of the Society of Allied Weight Engineers*, pages 1–28. Society of Allied Weight Engineers, Inc., 2001.

[Brand and Hertzmann, 2000] Matthew Brand and Aaron Hertzmann. Style machines. In *Proceedings of ACM SIGGRAPH*, Computer Graphics Proceedings, pages 183–192. ACM Press, 2000.

[Bray *et al.*, 2006] Matthieu Bray, Pushmeet Kohli, and Philip H. Torr. Posecut: Simultaneous segmentation and 3D pose estimation of humans using dynamic graph-cuts. In *Proceedings of the 9th European Conference on Computer Vision (ECCV)*, volume 3952 of *Lecture Notes in Computer Science*, pages 642–655. Springer-Verlag, 2006.

[Bregler *et al.*, 2004] Christoph Bregler, Jitendra Malik, and Katherine Pullen. Twist based acquisition and tracking of animal and human kinematics. *International Journal of Computer Vision (IJCV)*, 56(3):179–194, 2004.

[Brodie *et al.*, 2008] Matthew Brodie, Alan Walmsley, and Wyatt Page. Fusion motion capture: a prototype system using inertial measurement units and GPS for the biomechanical analysis of ski racing. *Sports Technology*, 1(1):17–28, 2008.

[Brox *et al.*, 2006] Thomas Brox, Bodo Rosenhahn, Daniel Cremers, and Hans-Peter Seidel. Nonparametric density estimation with adaptive anisotropic kernels for human motion tracking. In *Proceedings 2nd Workshop on Human Motion*, volume 4814 of *Lecture Notes in Computer Science*, pages 152–165. Springer-Verlag, 2006.

[Brubaker *et al.*, 2007] Marcus Brubaker, David J. Fleet, and Aaron Hertzmann. Physics-based person tracking using simplified lower-body dynamics. In *IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE Computer Society Press, 2007.

[Brubaker *et al.*, 2010] Marcus A. Brubaker, David J. Fleet, and Aaron Hertzmann. Physics-based person tracking using the anthropomorphic walker. *International Journal of Computer Vision (IJCV)*, 87:140–155, 2010.

[Bălan *et al.*, 2005] Alexandru O. Bălan, Leoid Sigal, and Michael J. Black. A quantitative evaluation of video-based 3D person tracking. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS)*, pages 349–356, 2005.

[Bălan *et al.*, 2007a] Alexandru O. Bălan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed human shape and pose from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[Bălan *et al.*, 2007b] Alexandru O. Bălan, Leonid Sigal, Michael J. Black, and Horst W. Haussecker. Shining a light on human pose: On shadows, shading and the estimation of pose and shape. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

[Chai and Hodgins, 2005] Jinxiang Chai and Jessica K. Hodgins. Performance animation from low-dimensional control signals. *ACM Transactions on Graphics (TOG)*, 24(3):686–696, 2005.

[Chen *et al.*, 2009] Jixu Chen, Minyoung Kim, Yu Wang, and Qiang Ji. Switching gaussian process dynamic models for simultaneous composite motion tracking and recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2655–2662, 2009.

[Chiu *et al.*, 2004] Chih-Yi Chiu, Shih-Pin Chao, Ming-Yang Wu, Shi-Nine Yang, and Hsin-Chih Lin. Content-based retrieval for human motion data. *Journal of Visual Communication and Image Representation (JVCI)*, 15(3):446–466, 2004.

[CMU, 2003] CMU. Carnegie-mellon mocap database. `http://mocap.cs.cmu.edu`, 2003.

[Cooper *et al.*, 2007] Seth Cooper, Aaron Hertzmann, and Zoran Popović. Active learning for real-time motion controllers. *ACM Transactions on Graphics (TOG)*, 26(3):5, 2007.

[Cormen *et al.*, 2001] Thomas H. Cormen, Clifford Stein, Charles E. Leiserson, and Robert L. Rivest. *Introduction to Algorithms*. MIT Press, 2001.

[Dambreville *et al.*, 2008] Samuel Dambreville, Romeil Sandhu, Anthony Yezzi, and Allen Tannenbaum. Robust 3D pose estimation and efficient 2d region-based segmentation from a 3D shape prior. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 5303 of *Lecture Notes in Computer Science*, pages 169–182. Springer-Verlag, 2008.

[Daniilidis, 1999] Konstantinos Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research (IJRR)*, 18(3):286–298, 1999.

[de la Torre *et al.*, 2008] Fernando de la Torre, Jessica Hodgins, Adam Bargteil, Xavier Martin, Justin Macey, Alex Collado, and Pep Beltran. Carnegie mellon university multimodal activity (CMU-MMAC) database. Technical report, Robotics Institute, 2008. [Online; accessed 13-March-2012].

[Dejnabadi *et al.*, 2006] Hooman Dejnabadi, Brigitte M. Jolles, Emilio Casanova, Pascal Fua, and Kamiar Aminian. Estimation and visualization of sagittal kinematics of lower limbs orientation using body-fixed sensors. *IEEE Transactions on Biomedical Engineering*, 53(7):1385–1393, 2006.

[Demirdjian *et al.*, 2005] David Demirdjian, Leonid Taycher, Gregory Shakhnarovich, Kristen Graumanand, and Trevor Darrell. Avoiding the streetlight effect: tracking by exploring likelihood modes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 357–364, 2005.

[Demuth *et al.*, 2006] Bastian Demuth, Tido Röder, Meinard Müller, and Bernhard Eberhardt. An information retrieval system for motion capture data. In Mounia Lalmas, Andy MacFarlane, Stefan Rger, Anastasios Tombros, Theodora Tsikrika, and Alexei Yavlinsky, editors, *Advances in Information Retrieval*, volume 3936 of *Lecture Notes in Computer Science*, pages 373–384. Springer-Verlag, 2006.

[Deng *et al.*, 2009] Zhigang Deng, Qin Gu, and Qing Li. Perceptually consistent example-based human motion retrieval. In *Proceedings of the symposium on Interactive 3D graphics and games (I3D)*, pages 191–198. ACM, 2009.

[Deng *et al.*, 2011] Liqun Deng, Howard Leung, Naijie Gu, and Yang Yang. Real-time mocap dance recognition for an interactive dancing game. *Computer Animation and Virtual Worlds (CAVW)*, 22:229–237, 2011.

[Deutscher and Reid, 2005] Jonathan Deutscher and Ian Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision (IJCV)*, 61(2):185–205, 2005.

[Elson, 1994] Matt Elson. Character animation systems. In *ACM SIGGRAPH Courses*, 1994.

[Eyes, JAPAN Co. Ltd., 2012] Eyes, JAPAN Co. Ltd. The world biggest state of the art motion capture library on earth. `http://www.mocapdata.com`, 2012. [Online; accessed 13-March-2012].

[Forbes and Fiume, 2005] Kevin Forbes and Eugene Fiume. An efficient search algorithm for motion data using weighted PCA. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 67–76. ACM Press, 2005.

[Fossati *et al.*, 2010] Andrea Fossati, Miodrag Dimitrijevic, Vincent Lepetit, and Pascal Fua. From canonical poses to 3D motion capture using a single camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(7):1165–1181, 2010.

[Foxlin, 2005] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer Graphics and Applications (CG&A)*, 25(6):38–46, 2005.

[Friborg *et al.*, 2010] Rune Friborg, Søren Hauberg, and Kenny Erleben. GPU accelerated likelihoods for stereo-based articulated tracking. In *ECCV 2010 Workshop on Computer Vision on GPUs (CVGPU)*, 2010.

[Gall *et al.*, 2009] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1753. IEEE, 2009.

[Gall *et al.*, 2010a] Juergen Gall, Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision (IJCV)*, 87(1-2):75–92, 2010.

[Gall *et al.*, 2010b] Juergen Gall, Angela Yao, and Luc Van Gool. 2D action recognition serves 3D human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 425–438. Springer-Verlag, 2010.

[Gall *et al.*, 2011] Juergen Gall, Andrea Fossati, and Luc van Gool. Functional categorization of objects using real-time markerless motion capture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1969–1976, 2011.

[Ganapathi *et al.*, 2010] Varun Ganapathi, Christian Plagemann, Sebastian Thrun, and Daphne Koller. Real time motion capture using a single time-of-flight camera. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[Gao *et al.*, 2006a] Yan Gao, Lizhuang Ma, Yiqiang Chen, and Junfa Liu. Content-based human motion retrieval with automatic transition. In Tomoyuki Nishita, Qunsheng Peng, and Hans-Peter Seidel, editors, *Advances in Computer Graphics*, volume 4035 of *Lecture Notes in Computer Science*, pages 360–371. Springer-Verlag, 2006.

[Gao *et al.*, 2006b] Yan Gao, Lizhuang Ma, Junfa Liu, Xiaomao Wu, and Zhihua Chen. An efficient algorithm for content-based human motion retrieval. In Zhigeng Pan, Ruth Aylett, Holger Diener, Xiaogang Jin, Stefan Gbel, and Li Li, editors, *Technologies for E-Learning and Digital Entertainment*, volume 3942 of *Lecture Notes in Computer Science*, pages 970–979. Springer-Verlag, 2006.

[Giese and Poggio, 2000] Martin A. Giese and Tomaso Poggio. Morphable models for the analysis and synthesis of complex motion patterns. *International Journal of Computer Vision (IJCV)*, 38:59–73, 2000.

[Girshick *et al.*, 2011] Ross B. Girshick, Andie Shotton, Pushmeet Kohli, Antonio Criminisi, and Andrew Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *IEEE International Conference on Computer Vision (ICCV)*, pages 415–422, 2011.

[Gleicher, 2000] Michael Gleicher. Animation from observation: Motion capture and motion editing. *ACM SIGGRAPH Computer Graphics*, 33(4):51–54, 2000.

[Grassia, 1998] F. Sebastian Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics, GPU, and Game Tools*, 3(3):29–48, 1998.

[Grest *et al.*, 2007] Daniel Grest, Volker Krüger, and Reinhard Koch. Single view motion tracking by depth and silhouette information. In *Proceedings of the Scandinavian conference on Image analysis (SCIA)*, pages 719–729. Springer-Verlag, 2007.

[Guan *et al.*, 2009] Peng Guan, Alexander Weiss, Alexandru O. Bălan, and Michael J. Black. Estimating human shape and pose from a single image. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1381–1388, 2009.

[Guo *et al.*, 2011] Xiaocui Guo, Qiang Zhang, Rui Liu, Dongsheng Zhou, and Jing Dong. 3D human motion retrieval based on ISOMAP dimension reduction. In *Proceedings of the international conference on Artificial Intelligence and Computational Intelligence (AICI)*, pages 159–169. Springer-Verlag, 2011.

[Harada *et al.*, 2007] Tatsuya Harada, Taketoshi Mori, and Tomomasa Sato. Development of a tiny orientation estimation device to operate under motion and magnetic disturbance. *The International Journal of Robotics Research (IJRR)*, 26(6):547–559, 2007.

[Hasler *et al.*, 2010] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1823–1830, 2010.

[Heck and Gleicher, 2007] Rachel Heck and Michael Gleicher. Parametric motion graphs. In *Proceedings of the symposium on Interactive 3D graphics and games (I3D)*, pages 129–136. ACM, 2007.

[Heikkila and Silven, 1997] Janne Heikkila and Olli Silven. A four-step camera calibration procedure with implicit image correction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1106–1112. IEEE Computer Society, 1997.

[Helten *et al.*, 2011] Thomas Helten, Heike Brock, Meinard Müller, and Hans-Peter Seidel. Classification of trampoline jumps using inertial sensors. *Sports Engineering*, 14:155–164, 2011. 10.1007/s12283-011-0081-4.

[Herda *et al.*, 2004] Loma Herda, Raquel Urtasun, and Pascal Fua. Implicit surface joint limits to constrain video-based motion capture. In *Proceedings of the 8th European Conference on Computer Vision (ECCV)*, volume 3022 of *Lecture Notes in Computer Science*, pages 405–418. Springer-Verlag, 2004.

[Hol *et al.*, 2008] Jeroen D. Hol, Thomas B. Schön, and Fredrik Gustafsson. Relative pose calibration of a spherical camera and an IMU. In *IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 21–24, 2008.

[Hsu *et al.*, 2005] Eugene Hsu, Kari Pulli, and Jovan Popović. Style translation for human motion. *ACM Transactions on Graphics (TOG)*, 24(3):1082–1089, 2005.

[Huynh, 2009] Du Q. Huynh. Metrics for 3D rotations: Comparison and analysis. *Journal of Mathematical Imaging and Vision*, 35(2):155–164, 2009.

[John L. Crassidis, 2007] Yang Cheng John L. Crassidis, F. Landis Markley. Survey of nonlinear attitude estimation methods. *Journal of Guidance Control and Dynamics*, 30(1):12–28, 2007.

[Keogh *et al.*, 2004] Eamonn J. Keogh, Themis Palpanas, Victor Brian Zordan, Dimitrios Gunopulos, and Marc Cardle. Indexing large human-motion databases. In *Proceedings of the international conference on Very large data bases (VLDP)*, pages 780–791, 2004.

[Kirk *et al.*, 2005] Adam G. Kirk, James F. O'Brien, and David A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 782–788, 2005.

[Knoop *et al.*, 2009] Steffen Knoop, Stefan Vacek, and Rüdiger Dillmann. Fusion of 2D and 3D sensor data for articulated body tracking. *Robotics and Autonomous Systems*, 57(3):321–329, 2009.

[Kolb *et al.*, 2010] Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-flight sensors in computer graphics. *Computer Graphics Forum (CGF)*, 29(1):141–159, 2010.

[Kovar and Gleicher, 2004] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (TOG)*, 23(3):559–568, 2004.

[Kovar *et al.*, 2002] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics (TOG)*, 21(3):473–482, 2002.

[Kovar *et al.*, 2008] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. In *ACM SIGGRAPH 2008 classes*, pages 51:1–51:10. ACM, 2008.

[Krüger *et al.*, 2010] Björn Krüger, Jochen Tautges, Andreas Weber, and Arno Zinke. Fast local and global similarity searches in large motion capture databases. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 1–10. Eurographics Association, 2010.

[Kunze and Lukowicz, 2008] Kai Kunze and Paul Lukowicz. Dealing with sensor displacement in motion-based onbody activity recognition systems. In *Proceedings of the International Conference on Ubiquitous Computing (UbiComp)*, pages 20–29. ACM, 2008.

[Lau *et al.*, 2009] Manfred Lau, Ziv Bar-Joseph, and James Kuffner. Modeling spatial and temporal variation in motion data. *ACM Transaction on Graphics (TOG)*, 28(5):171:1–171:10, 2009.

[Lee and Lee, 2006] Jehee Lee and Kang Hoon Lee. Precomputing avatar behavior from human motion data. *Graphical Models*, 68(2):158–174, 2006.

[Lee and Nevatia, 2009] Mun Wai Lee and Ramakant Nevatia. Human pose tracking in monocular sequence using multilevel structured models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 31(1):27–38, 2009.

[Lee *et al.*, 2002] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics (TOG)*, 21(3):491–500, 2002.

[Lee *et al.*, 2009] Yongjoon Lee, Seong Jae Lee, and Zoran Popović. Compact character controllers. *ACM Transaction on Graphics*, 28(5):169:1–169:8, 2009.

[Lee *et al.*, 2010] Yoonsang Lee, Sungeun Kim, and Jehee Lee. Data-driven biped control. *ACM Transaction on Graphics (TOG)*, 29(4):129:1–129:8, 2010.

[Lewis *et al.*, 2000] John Paul Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Proceedings of the annual conference on Computer graphics and interactive techniques (SIGGRAPH)*, pages 165–172. ACM Press/Addison-Wesley Publishing Co., 2000.

[Li *et al.*, 2007] Chuanjun Li, Punit R. Kulkarni, and Balakrishnan Prabhakaran. Segmentation and recognition of motion capture data stream by classification. *Multimedia Tools and Applications*, 35:55–70, 2007.

[Li *et al.*, 2010] Rui Li, Tai-Peng Tian, Stan Sclaroff, and Ming-Hsuan Yang. 3D human motion tracking with a coordinated mixture of factor analyzers. *International Journal of Computer Vision (IJCV)*, 87(1-2):170–190, 2010.

[Lindner *et al.*, 2010] Marvin Lindner, Ingo Schiller, Andreas Kolb, and Reinhard Koch. Time-of-flight sensor calibration for accurate range sensing. *Computer Vision and Image Understanding (CVIU)*, 114(12):1318–1328, 2010. Special issue on Time-of-Flight Camera Based Computer Vision.

[Liu and Shah, 2008] Jingen Liu and Mubarak Shah. Learning human actions via information maximization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[Liu *et al.*, 2003] Feng Liu, Yueting Zhuang, Fei Wu, and Yunhe Pan. 3D motion retrieval with motion index tree. *Computer Vision and Image Understanding (CVIU)*, 92(2-3):265–284, 2003.

[Liu *et al.*, 2005] Guodong Liu, Jingdan Zhang, Wei Wang, and Leonard McMillan. A system for analyzing and indexing human-motion databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 924–926. ACM Press, 2005.

[Liu *et al.*, 2006] Guodong Liu, Jingdan Zhang, Wei Wang, and Leonard McMillan. Human motion estimation from a reduced marker set. In *Proceedings of the symposium on Interactive 3D graphics and games (I3D)*, pages 35–42. ACM, 2006.

[Liu *et al.*, 2011] Yebin Liu, Carsten Stoll, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Markerless motion capture of interacting characters using multi-view image segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1249–1256, 2011.

[López-Méndez *et al.*, 2011] Adolfo López-Méndez, Marcel Alcoverro, Montse Pardàs, and Josep R. Casas. Real-time upper body tracking with online initialization using a range sensor. In *IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 391–398, 2011.

[Lowe, 1980] David G. Lowe. Solving for the parameters of object models from image descriptions. In *Image Understanding Workshop*, pages 121–127, 1980.

[Matlab, 2012] MATLAB camera calibration toolbox. `http://www.vision.caltech.edu/bouguetj/calib_doc`, 2012. [Online; accessed 13-March-2012].

[Metoyer *et al.*, 2008] Ronald Metoyer, Victor Zordan, Benjamin Hermens, Chun-Chi Wu, and Marc Soriano. Psychologically inspired anticipation and dynamic response for impacts to the head and upper body. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):173–185, 2008.

[Microsoft, 2012] Microsoft. Kinect SDK beta. `http://www.microsoft.com/en-us/kinectforwindows`, 2012. [Online; accessed 13-March-2012].

[Min *et al.*, 2010] Jianyuan Min, Huajun Liu, and Jinxiang Chai. Synthesis and editing of personalized stylistic human motion. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games (I3D)*, pages 39–46. ACM, 2010.

[Moeslund and Granum, 2001] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding (CVIU)*, 81(3):231–268, 2001.

[Moeslund *et al.*, 2006] 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 (CVIU)*, 104(2):90–126, 2006.

[Moeslund *et al.*, 2011] Thomas B. Moeslund, Adrian Hilton, Volker Krüger, and Leonid Sigal, editors. *Visual Analysis of Humans - Looking at People*. Springer-Verlag, 2011.

[Mukai and Kuriyama, 2005] Tomohiko Mukai and Shigeru Kuriyama. Geostatistical motion interpolation. *ACM Transaction on Graphics (TOG)*, 24(3):1062–1070, 2005.

[Mukai and Kuriyama, 2009] Tomohiko Mukai and Shigeru Kuriyama. Pose-timeline for propagating motion edits. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 113–122. ACM, 2009.

[Müller and Röder, 2006] Meinard Müller and Tido Röder. Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 137–146. ACM Press, 2006.

[Müller and Röder, 2007] Meinard Müller and Tido Röder. A relational approach to content-based analysis of motion capture data. In Bodo Rosenhahn, Reinhard Klette, and Dimitri Metaxas, editors, *Human Motion - Understanding, Modeling, Capture, and Animation*, pages 477–506. Springer-Verlag, 2007.

[Müller *et al.*, 2005] Meinard Müller, Tido Röder, and Michael Clausen. Efficient content-based retrieval of motion capture data. *ACM Transactions on Graphics (TOG)*, 24(3):677–685, 2005.

[Müller *et al.*, 2007] Meinard Müller, Tido Röder, Michael Clausen, Bernhard Eberhardt, Björn Krüger, and Andreas Weber. Documentation: Mocap Database HDM05. Computer Graphics Technical Report CG-2007-2, Universität Bonn, 2007. `http://www.mpi-inf.mpg.de/resources/HDM05`.

[Müller *et al.*, 2008] Meinard Müller, Bastian Demuth, and Bodo Rosenhahn. An evolutionary approach for learning motion class patterns. In *Pattern Recognition*, volume 5096 of *Lecture Notes in Computer Science*, pages 365–374. DAGM, Springer-Verlag, 2008.

[Müller *et al.*, 2009] Meinard Müller, Andreas Baak, and Hans-Peter Seidel. Efficient and robust annotation of motion capture data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 17–26, 2009.

[Müller, 2007] Meinard Müller. *Information Retrieval for Music and Motion*. Springer-Verlag, 2007.

[Murray *et al.*, 1994] Richard M. Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994.

[Neff *et al.*, 2008] Michael Neff, Michael Kipp, Irene Albrecht, and Hans-Peter Seidel. Gesture modeling and animation based on a probabilistic re-creation of speaker style. *ACM Transactions on Graphics*, 27(1), 2008.

[Nintendo, 2012] Nintendo. Nintendo homepage. `http://www.nintendo.com`, 2012. [Online; accessed 13-March-2012].

[NOAA, 2012] NOAA. Magnetic field calculator of the national oceanic and atmospheric administration. `http://www.ngdc.noaa.gov/geomag-web/`, 2012. [Online; accessed 02-May-2012].

[Numaguchi *et al.*, 2011] Naoki Numaguchi, Atsushi Nakazawa, Takaaki Shiratori, and Jessica K. Hodgins. A puppet interface for retrieval of motion capture data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 157–166. ACM, 2011.

[Ohio State University, 2012] Ohio State University. Open motion data project. `http://accad.osu.edu/research/mocap/mocap_data.htm`, 2012. [Online; accessed 13-March-2012].

[Okada and Soatto, 2008] Ryuzo Okada and Stefano Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 434–445. Springer-Verlag, 2008.

[Okada and Stenger, 2008] Ryuzo Okada and Björn Stenger. A single camera motion capture system for human-computer interaction. *IEICE Transactions on Information and Systems*, E91-D:1855–1862, 2008.

[Park and Martin, 1994] Frank Chongwoo Park and B. J. Martin. Robot sensor calibration: solving $AX = XB$ on the euclidean group. *IEEE Transactions on Robotics and Automation*, 10(5):717–721, 1994.

[Pejsa and Pandzic, 2010] Tomislav Pejsa and Igor S. Pandzic. State of the art in example-based motion synthesis for virtual characters in interactive applications. *Computer Graphics Forum (CFG)*, 29(1):202–226, 2010.

[Pekelny and Gotsman, 2008] Yuri Pekelny and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum (CGF)*, 27(2):399–408, 2008.

[PhaseSpace, 2012] PhaseSpace. PhaseSpace motion capture. `http://www.phasespace.com`, 2012. [Online; accessed 13-March-2012].

[Plagemann *et al.*, 2010] Christian Plagemann, Varun Ganapathi, Daphne Koller, and Sebastian Thrun. Realtime identification and localization of body parts from depth images. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[Pohlheim, 1999] Hartmut Pohlheim. *Evolutionäre Algorithmen: Verfahren, Operatoren und Hinweise*. Springer-Verlag, 1999.

[Pons-Moll *et al.*, 2008] Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, , and Bodo Rosenhahn. Multimodal human motion database MPI08. `http://www.tnt.uni-hannover.de/project/MPI08_Database/`, 2008. [Online; accessed 13-March-2012].

[Pons-Moll *et al.*, 2010] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 663–670, 2010.

[Pons-Moll *et al.*, 2011] 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 *IEEE International Conference on Computer Vision (ICCV)*, pages 1243–1250, 2011.

[Poppe, 2010] Ronald Poppe. A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[Pradhan and Prabhakaran, 2009] Gaurav N. Pradhan and Balakrishnan Prabhakaran. Indexing 3-D human motion repositories for content-based retrieval. *IEEE Transactions on Information Technology in Biomedicine*, 13(5):802–809, 2009.

[Primesense, 2012] Primesense. Primesense NITE middleware. http://www.primesense.com, 2012. [Online; accessed 13-March-2012].

[Pullen and Bregler, 2002] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: texturing and synthesis. *ACM Transactions on Graphics (TOG)*, 21(3):501–508, 2002.

[Raptis *et al.*, 2011] Michalis Raptis, Darko Kirovski, and Hugues Hoppe. Real-time classification of dance gestures from skeleton animation. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 147–156. ACM, 2011.

[Ren *et al.*, 2011] Cheng Ren, Xiaoyong Lei, and Guofeng Zhang. Motion data retrieval from very large motion databases. In *Proceedings of the International Conference on Virtual Reality and Visualization (ICVRV)*, pages 70–77, 2011.

[Roetenberg, 2006] Daniel Roetenberg. *Inertial and magnetic sensing of human motion*. PhD thesis, University of Twente, 2006.

[Romero *et al.*, 2010] Javier Romero, Hedvig Kjellström, and Danica Kragic. Hands in action: real-time 3D reconstruction of hands in interaction with objects. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 458–463, 2010.

[Rosales and Sclaroff, 2000] Rómer Rosales and Stan Sclaroff. Inferring body pose without tracking body parts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 721–727, 2000.

[Rosales and Sclaroff, 2006] Rómer Rosales and Stan Sclaroff. Combining generative and discriminative models in a framework for articulated pose estimation. *International Journal of Computer Vision (IJCV)*, 67:251–276, 2006.

[Rose *et al.*, 1998] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: multidimensional motion interpolation. *IEEE Computer Graphics and Applications (CG&A)*, 18(5):32–40, 1998.

[Rosenhahn *et al.*, 2007a] Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. Scaled motion dynamics for markerless motion capture. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1203–1210. IEEE, 2007.

[Rosenhahn *et al.*, 2007b] Bodo Rosenhahn, Thomas Brox, and Joachim Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision (IJCV)*, 73:243–262, 2007.

[Rosenhahn *et al.*, 2007c] Bodo Rosenhahn, Reinhard Klette, and Dimitris Metaxas, editors. *Human Motion: Understanding, Modeling, Capture, and Animation*. Springer-Verlag, 2007.

[Rosenhahn *et al.*, 2008a] Bodo Rosenhahn, Christian Schmaltz, Thomas Brox, Joachim Weickert, Daniel Cremers, and Hans-Peter Seidel. Markerless motion capture of man-machine interaction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE Computer Society Press, 2008.

[Rosenhahn *et al.*, 2008b] Bodo Rosenhahn, Christian Schmaltz, Thomas Brox, Joachim Weickert, and Hans-Peter Seidel. Staying well grounded in markerless motion capture. In *Proceedings of the 30th DAGM symposium on Pattern Recognition*, pages 385–395. Springer-Verlag, 2008.

[Safonova and Hodgins, 2007] Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transaction on Graphics (TOG)*, 26(3), 2007.

[Sakamoto *et al.*, 2004] Yasuhiko Sakamoto, Shigeru Kuriyama, and Toyohisa Kaneko. Motion map: image-based retrieval and segmentation of motion data. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 259–266. ACM Press, 2004.

[Salzmann and Urtasun, 2010] Mathieu Salzmann and Raquel Urtasun. Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[Schmaltz *et al.*, 2007] Christian Schmaltz, Bodo Rosenhahn, Thomas Brox, Daniel Cremers, Joachim Weickert, Lennart Wietzke, and Gerald Sommer. Region-based pose tracking. In *Pattern Recognition and Image Analysis*, volume 4478 of *Lecture Notes in Computer Science*, pages 56–63. Springer-Verlag, 2007.

[Schwarz *et al.*, 2010] Loren Arthur Schwarz, Diana Mateus, Victor Castañeda, and Nassir Navab. Manifold learning for ToF-based human body tracking and activity recognition. In *Proceedings on the British Machine Vision Conference (BMVC)*, pages 1–11, 2010.

[Schwarz *et al.*, 2011] Loren Schwarz, Artashes Mkhytaryan, Diana Mateus, and Nassir Navab. Estimating human 3D pose from time-of-flight images based on geodesic distances and optical flow. *IEEE Conference on Automatic Face and Gesture Recognition (FG)*, 2011.

[Seo *et al.*, 2009] Yongduek Seo, Young-Ju Choi, and Sang Wook Lee. A branch-and-bound algorithm for globally optimal calibration of a camera-and-rotation-sensor system. In *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[Shakhnarovich *et al.*, 2003] Gregory Shakhnarovich, Paul Viola, and Trevor Darrell. Fast pose estimation with parameter-sensitive hashing. In *IEEE International Conference on Computer Vision (ICCV)*, pages 750–757. IEEE Computer Society, 2003.

[Shapiro and Stockman, 2002] Linda G. Shapiro and George C. Stockman. *Computer Vision*. Prentice Hall, 2002.

[Shin and Oh, 2006] Hyun Joon Shin and Hyun Seok Oh. Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pages 291–298. Eurographics Association, 2006.

[Shiratori and Hodgins, 2008] Takaaki Shiratori and Jessica K. Hodgins. Accelerometer-based user interfaces for the control of a physically simulated character. In *ACM SIGGRAPH Asia*, pages 1–9. ACM, 2008.

[Shiu and Ahmad, 1989] Yiu Cheung Shiu and Shaheen Ahmad. Calibration of wrist-mounted robotic sensors by solving homogeneous transform equations of the form $AX = XB$. *IEEE Transactions on Robotics and Automation*, 5(1):16–29, 1989.

[Shoemake, 1985] Ken Shoemake. Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*, 19(3):245–254, 1985.

[Shotton *et al.*, 2011] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[Siddiqui and Medioni, 2010] Matheen Siddiqui and Gérard Medioni. Human pose estimation from a single view point, real-time range sensor. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, 2010.

[Sidenbladh *et al.*, 2002] Hedvig Sidenbladh, Michael J. Black, and Leonid Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 2353, pages 784–800. Springer-Verlag, 2002.

[Sigal *et al.*, 2008] Leonid Sigal, Alexandru O. Bălan, and Michael J. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. In *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1337–1344. MIT Press, 2008.

[Sigal *et al.*, 2010] 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 (IJCV)*, 87:4–27, 2010.

[Slyper and Hodgins, 2008] Ronit Slyper and Jessica Hodgins. Action capture with accelerometers. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2008.

[Sminchisescu and Jepson, 2004] Cristian Sminchisescu and Allan Jepson. Generative modeling for continuous non-linearly embedded visual inference. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 759–766, 2004.

[Sminchisescu and Triggs, 2003] Christian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research (IJRR)*, 22(6):371–391, 2003.

[Software, 2012] SmithMicro Software. Poser. http://http://poser.smithmicro.com/, 2012. [Online; accessed 13-March-2012].

[Stoll *et al.*, 2011] Carsten Stoll, Nils Hasler, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Fast articulated motion tracking using a sums of gaussians body model. In *IEEE International Conference on Computer Vision (ICCV)*, pages 951–958, 2011.

[Strobl and Hirzinger, 2006] Klaus H. Strobl and Gerd Hirzinger. Optimal hand-eye calibration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4647–4653, 2006.

[Tao *et al.*, 2007] Yaqin Tao, Huosheng Hu, and Huiyu Zhou. Integration of vision and inertial sensors for 3D arm motion tracking in home-based rehabilitation. *International Journal of Robotics Research (IJRR)*, 26(6):607–624, 2007.

[Tenorth *et al.*, 2009] Moritz Tenorth, Jan Bandouch, and Michael Beetz. The TUM Kitchen Data Set of Everyday Manipulation Activities for Motion Tracking and Action Recognition. In *IEEE International Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS), in conjunction with ICCV*, pages 1089–1096, 2009.

[Thong *et al.*, 2004] Y. Thong, Malcolm Woolfson, John Crowe, Barrie Hayes-Gill, and D. Jones. Numerical double integration of acceleration measurements in noise. *Measurement*, 36(1):73–92, 2004.

[Tran and Sorokin, 2008] Du Tran and Alexander Sorokin. Human activity recognition with metric learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 548–561. Springer-Verlag, 2008.

[Tsai and Lenz, 1988] Roger Y. Tsai and Reimar K. Lenz. Real time versatile robotics hand/eye calibration using 3D machine vision. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 1, pages 554–561, 1988.

[Urtasun and Darrell, 2008] Raquel Urtasun and Trevor Darrell. Sparse probabilistic regression for activity-independent human pose inference. In *IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 1–8. IEEE Computer Society Press, 2008.

[Urtasun *et al.*, 2006] Raquel Urtasun, David J. Fleet, and Pascal Fua. 3D people tracking with Gaussian process dynamical models. In *IEEE Conference of Computer Vision and Pattern Recognition (CVPR)*, pages 238–245. IEEE Computer Society Press, 2006.

[Vicon, 2012] Vicon. Vicon motion capture system. `http://www.vicon.com/`, 2012. [Online; accessed 13-March-2012].

[Vlasic *et al.*, 2007] 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*, 26(3):35, 2007.

[Vlasic *et al.*, 2008] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3):1–9, 2008.

[Vondrak *et al.*, 2008] Marek Vondrak, Leonid Sigal, and Odest Chadwicke Jenkins. Physical simulation for probabilistic motion tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008.

[Wang and Lee, 2009] Jung-Ying Wang and Hahn-Ming Lee. Recognition of human actions using motion capture data and support vector machine. In *Proceedings of the WRI World Congress on Software Engineering (WCSE)*, pages 234–238. IEEE Computer Society, 2009.

[Wang and Popovic, 2009] Robert Y. Wang and Jovan Popovic. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics (TOG)*, 28(3), 2009.

[Wei and Chai, 2010] Xiaolin Wei and Jinxiang Chai. Videomocap: modeling physically realistic human motion from monocular video sequences. *ACM Transactions on Graphics (TOG)*, 29(4):42:1–42:10, 2010.

[Wei and Chai, 2011] Xiaolin K. Wei and Jinxiang Chai. Intuitive interactive human character posing with millions of example poses. *IEEE Computer Graphics and Applications (CG&A)*, 31(4):78–88, 2011.

[Weiss *et al.*, 2011] Alexander Weiss, David Hirshberg, and Michael J. Black. Home 3D body scans from noisy image and range data. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1951–1958, 2011.

[Welch and Bishop, 1995] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

[Welch and Bishop, 2001] Greg Welch and Gary Bishop. An introduction to the kalman filter. In *ACM SIGGRAPH courses*. ACM, 2001.

[Wikipedia, 2012] Wikipedia. Motion capture — wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Motion_capture&oldid=479354716`, 2012. [Online; accessed 13-March-2012].

[Witten *et al.*, 1999] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes*. Morgan Kaufmann Publishers, 1999.

[Wu *et al.*, 2003] Ming-Yang Wu, Chih-Yi Chiu, Shih-Pin Chao, Shi-Nine Yang, and Hsin-Chih Lin. Content-based retrieval for human motion data. In *The IPPR Conference on Computer Vision, Graphics, and Image Processing*, pages 605–612, 2003.

[Wu *et al.*, 2009a] Shuangyuan Wu, Zhaoqi Wang, and Shihong Xia. Indexing and retrieval of human motion data by a hierarchical tree. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology (VRST)*, pages 207–214. ACM, 2009.

[Wu *et al.*, 2009b] Shuangyuan Wu, Shihong Xia, Zhaoqi Wang, and Chunpeng Li. Efficient motion data indexing and retrieval with local similarity measure of motion strings. *The Visual Computer: International Journal of Computer Graphics*, 25(5-7):499–508, 2009.

[Xiang and Zhu, 2007] Jian Xiang and Hongli Zhu. Motion retrieval based on an efficient index method for large-scale mocap database. In Vincent Duffy, editor, *Digital Human Modeling*, volume 4561 of *Lecture Notes in Computer Science*, pages 234–242. Springer-Verlag, 2007.

[Xiang, 2007] Jian Xiang. Motion retrieval with temporal-spatial features based on ensemble learning. In *Proceedings of the International Conference on Intelligent Computing (ICIC)*, pages 300–308. Springer-Verlag, 2007.

[Xsens, 2012] Xsens. 3D motion tracking. `http://www.xsens.com`, 2012. [Online; accessed 13-March-2012].

[Ye *et al.*, 2011] Mao Ye, Xianwang Wang, Ruigang Yang, Liu Ren, and Marc Pollefeys. Accurate 3D pose estimation from a single depth image. In *IEEE International Conference on Computer Vision (ICCV)*, pages 731–738, 2011.

[Zhao *et al.*, 2009] Liming Zhao, Aline Normoyle, Sanjeev Khanna, and Alla Safonova. Automatic construction of a minimum size motion graph. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 27–35. ACM, 2009.

[Zhu *et al.*, 2008] Youding Zhu, B. Dariush, and K. Fujimura. Controlled human pose estimation from depth image streams. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, 2008.

[Zhu *et al.*, 2010] Youding Zhu, Behzad Dariush, and Kikuo Fujimura. Kinematic self retargeting: A framework for human pose estimation. *Computer Vision and Image Understanding (CVIU)*, 114(12):1362–1375, 2010. Special issue on Time-of-Flight Camera Based Computer Vision.

[Zordan *et al.*, 2005] Victor Brian Zordan, Anna Majkowska, Bill Chiu, and Matthew Fast. Dynamic response for motion capture animation. *ACM Transactions on Graphics (TOG)*, 24(3):697–701, 2005.

[Zordan *et al.*, 2007] Victor Zordan, Adriano Macchietto, Jose Medin, Marc Soriano, Chun-Chih Wu, Ronald Metoyer, and Robert Rose. Anticipation from example. In *Proceedings of the ACM symposium on Virtual reality software and technology (VRST)*, pages 81–84. ACM, 2007.