Processing and Tracking Human Motions Using Optical, Inertial, and Depth Sensors

Thomas Helten

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, 22. Oktober 2013

Tag des Kolloquiums: 13. Dezember 2013

Dekan:
Prof. Dr. Mark Groves

Prüfungsausschuss

Vorsitzender:
Prof. Dr. Hans-Peter Seidel

Gutachter:
Prof. Dr. Meinard Müller
Prof. Dr. Christian Theobalt

Akademischer Mitarbeiter:
Dr. Levi Valgaerts

Summary

The processing of human motion data constitutes an important strand of research with many applications in computer animation, sport science and medicine. Currently, there exist various systems for recording human motion data that employ sensors of different modalities such as optical, inertial and depth sensors. Each of these sensor modalities have intrinsic advantages and disadvantages that make them suitable for capturing specific aspects of human motions as, for example, the overall course of a motion, the shape of the human body, or the kinematic properties of motions. In this thesis, we contribute with algorithms that exploit the respective strengths of these different modalities for comparing, classifying, and tracking human motion in various scenarios. First, we show how our proposed techniques can be employed, *e. g.*, for real-time motion reconstruction using efficient cross-modal retrieval techniques. Then, we discuss a practical application of inertial sensors-based features to the classification of trampoline motions. As a further contribution, we elaborate on estimating the human body shape from depth data with applications to personalized motion tracking. Finally, we introduce methods to stabilize a depth tracker in challenging situations such as in presence of occlusions. Here, we exploit the availability of complementary inertial-based sensor information.

Zusammenfassung

Die Verarbeitung menschlicher Bewegungsdaten stellt einen wichtigen Bereich der Forschung dar mit vielen Anwendungsmöglichkeiten in Computer-Animation, Sportwissenschaften und Medizin. Zurzeit existieren diverse Systeme für die Aufnahme von menschlichen Bewegungsdaten, welche unterschiedliche Sensor-Modalitäten, wie optische Sensoren, Trägheits- oder Tiefen-Sensoren, einsetzen. Alle diese Sensor-Modalitäten haben intrinsische Vor- und Nachteile, welche sie befähigen, spezifische Aspekte menschlicher Bewegungen, wie zum Beispiel den groben Verlauf von Bewegungen, die Form des menschlichen Körpers oder die kinetischen Eigenschaften von Bewegungen, einzufangen. In dieser Arbeit tragen wir mit Algorithmen bei, welche die jeweiligen Vorteile dieser verschiedenen Modalitäten ausnutzen, um menschliche Bewegungen in unterschiedlichen Szenarien zu vergleichen, zu klassifizieren und zu verfolgen. Zuerst zeigen wir, wie unsere vorgeschlagenen Techniken angewandt werden können, um z. B. in Echtzeit Bewegungen mit Hilfe von cross-modalem Suchen zu rekonstruieren. Dann diskutieren wir eine praktische Anwendung von Trägheitssensor-basierten Eigenschaften für die Klassifikation von Trampolinbewegungen. Als einen weiteren Beitrag gehen wir näher auf die Bestimmung der menschlichen Körperform aus Tiefen-Daten mit Anwendung in personalisierter Bewegungsverfolgung ein. Zuletzt führen wir Methoden ein, um einen Tiefen-Tracker in anspruchsvollen Situationen, wie z. B. in Anwesenheit von Verdeckungen, zu stabilisieren. Hier nutzen wir die Verfügbarkeit von komplementären, Trägheits-basierten Sensor-Informationen aus.

Acknowledgements

Firstly, I would like to thank my parents Ingrid and Hans-Klaus Helten for their help and continuous support of all the decisions that I made so far in my live. I hope that my future brings me physically closer again to my home place. In my heart, I never left.

I would like to apologize to my friends from my home village Walberberg for my scarce presence in the recent years. In this context, I want to explicitly thank Volker Susen and Andreas Schiebahn for continually organizing our Pentecostal tours, which gives us a regular opportunity to meet all the old friends who are less and less available for various reasons.

I would like to thank my colleagues from AG4, for their corporation, help, and friendship during my stay at MPI. In particular, I am grateful to my office-mates Andreas Baak and Srinath Sridhar who were giving me important advice and feedback when I needed it. In other times, they were a good counterpart for intensive and interesting discussions about research problems, aspects of modern programming languages, cultural and language related issues, and past and future development of mankind. Furthermore, I want to use the opportunity to thank my colleagues Helge Rhodin, Pablo Garrido, Chenglei Wu, and Srinath Sridhar for proofreading parts of this thesis.

Especially, I would like to give thanks to the ladies from the secretariat, Sabine Budde and Ellen Fries, who are giving us support booking our business trip, providing us with help fighting the bureaucracy of the university, being the source for important and interesting information, ensuring the operation of our coffee machine, and—last but not least—preparing the lunch after our CG-lunch event every week.

Of course, I would like to express my gratitude to my supervisors Meinard Müller and Christian Theobalt, for giving me the chance to explore the world of science, providing me guidance and support in the difficult times but also giving me plenty of freedom to develop my own ideas or peruse interests that might not have been directly useful in the short run.

Last but not least, I would like to thank Hans-Peter Seidel and the Max-Planck Gesellschaft for providing such a nice and open working environment, where one has various sources of inspiration and opportunities to interact and cooperate with so many researchers from different countries and fields of research. In the past five years, I have been given an place to work that I always enjoyed to come to and that I will definitely miss in the future.

Parts of this work were supported by the German Research Foundation with the research plan named “REKOBA: Rekonstruktion von Bewegungsabläufen aus niedrigdimensionalen Sensor- und Kontrolldaten” (DFG MU 2686/3-1). I would like to thank Meinard Müller and Andreas Weber for their hard work of writing the corresponding research proposal. I also thank my former colleagues, Björn Krüger and Jochen Tautges for the cooperation in this project. Furthermore, work in this thesis was supported by the Intel Visual Computing Institute. Finally, parts of this work were supported by the European Research Counsil (ERC) Grant “CapReal”. I would like to give thanks to Christian Theobalt for his effort writing the corresponding research proposal.

Contents

Chapter 1

Introduction

1.1 Motivation

Human motion data is used in many fields of research such as computer animation, sport sciences, and medicine. Furthermore, many practical applications such as *e. g.* movie and game productions, or medical rehabilitation scenarios, rely on algorithms that process human data.

In these contexts, many different systems have been developed that record motion data of various types and in different levels of expressiveness. In particular, these systems have been designed to fit the specific application intended. Examples of such systems are optical systems based on (color) cameras, inertial systems, or systems using depth sensing devices. All of these systems have intrinsic advantages and disadvantages as far as acquisition cost, setup complexity and quality of recorded data is concerned.

In scenarios related to computer animation, such as the production of feature films and high quality computer games, one typically uses marker-based optical *motion capture* (mocap) systems. These systems are based on a set of calibrated cameras to track the 3D-positions of 30–50 markers fixed to the body of an actor. From the movement of these markers over time, motion representations such as joint angles, which can easily be used to animate artificial human or non-human characters, can be computed. While these systems provide the highest quality of motion data obtainable, they are very expensive and difficult to set up. Also, because of the large setup overhead and costs, capture sessions need to be well-planned in advance. Furthermore, the usage of (infrared) cameras imposes constraints on the location such systems can be operated. Optimal tracking results are typically achieved in studios with controlled lighting conditions. As a consequence, high-quality optical mocap systems can only be afforded by a small number of people.

To overcome for some of the disadvantages of optical systems, other systems have emerged that use alternative types of sensors. One example are systems using inertial sensors that capture orientations with respect to a global coordinate system. Such inertial systems do not require extensive setup procedures and can be used in non-studio environments or even outside. Furthermore, inertial mocap systems are less expensive and less intrusive compared to marker-based optical mocap systems. As a consequence, they are available to a larger group of users and applicable in a wider range of scenarios such as sports training or medical rehabilitation. Also, they are found in many modern devices such as video game consoles or smartphones, where they serve as an additional

input modality. Unfortunately, inertial sensors do not provide as rich data as the optical systems mentioned above. Thus, about 20 inertial sensors are required to track the local configuration of the body, which renders them still too expensive to enable full-body motion tracking in home application scenarios.

Another alternative for tracking human motion are systems based on so-called depth cameras. Such devices capture the scene similar to a traditional color camera by observing it from one point of view. But instead of color they provide an image, where each pixel captures the distance of a point in the scene to the camera. Research on how to obtain human motion data from depth images has a long tradition. However, the price of the available sensors and the noisy characteristics of the their provided data made them unattractive for applications intended for a great number of people. This changed, when Microsoft launched their Kinect sensor that was an order of magnitude less expensive compared to previously available depth sensors. This paved the way for the application of full-body motion tracking to home user scenarios. Since then, intense research has been conducted on full-body motion estimation from depth images, where recent approaches show promising result. However, many challenges are yet unsolved. Firstly, model-based approaches require the creation of a model of the person to track. But, obtaining such a model is time consuming and requires expensive equipment such as full-body laser scanners or the help of an artist. Secondly, current tracking approaches are still prone to errors that stem from the limited information provided by depth data. Here, one example is estimating the rotation of certain body parts, such as arm and legs, which is difficult to deduce from depth images. Finally, occlusions, where parts of the body are not visible to the camera renders it impossible for a depth tracker to deduce any meaningful information of that portion of the body.

1.2 Contributions and Organization

In this thesis, we address some of the challenges that arise when dealing with human motion data originating from various sensors modalities. To better understand, why this challenges exist and why they are important to solve, we begin, in Chapter 2, by introducing the three sensors modalities that are used throughout this thesis. In particular, we will explain how the different sensor modalities—optical, inertial, and depth—work in principle and what kind of data they provide. Furthermore, we will discuss their specific advantages and disadvantages and elaborate on how this affects their applicability to scenarios such as motion comparison, motion classification, or motion reconstruction.

In Chapter 3, we will discuss various motion representations that originate from different sensor modalities and investigate their discriminative power in the context of motion identification and retrieval scenarios. As one main contribution, we introduce mid-level motion representations that allow for comparing motion data in a cross-modal fashion. In particular, we show that certain low-dimensional feature representations derived from inertial sensors are suited for specifying high-dimensional motion data. Our evaluation shows that features based on directional information outperform purely acceleration based features in the context of motion retrieval scenarios. This work was published in Helten *et al.* [2011b]. We conclude the chapter by presenting an application of the discussed techniques in the context of human motion reconstruction, which was published in Tautges *et al.* [2011].

In Chapter 4, we extend the methods introduced in Chapter 3 and apply them to a practical mo-

tion classification scenario. In particular, we consider the scenario of trampoline motions, where an athlete performs a routine consisting of a sequence of jumps that belong to predefined motion categories such as pike jumps or somersaults. As main contribution, we introduce a fully automated approach for capturing, segmenting, and classifying trampoline routines according to these categories. Since trampoline motions are highly dynamic and spacious, optical motion capturing is problematic. Instead, we reverted to a small number of inertial sensors attached to the athlete's body. To cope with measurement noise and performance differences, we introduce suitable feature and class representations that are robust to spatial and temporal variations while capturing the characteristics of each motion category. The experiments show that the approach reliably classifies trampoline jumps across different athletes even in the presence of significant style variations. This work has been published in Helten *et al.* [2011a].

Then, in Chapter 5, we will focus on reconstructing a three-dimensional representation of human motion in real-time from the input of a depth sensor. Previous tracking approaches often required a body model resembling the human to be tracked. Without such a personalization, the tracking accuracy degrades drastically. However, obtaining such a personalized model often involves expensive equipement such as full-body laser-scanners, which is prohibitive for home application scenarios. For this reason, we contribute with a robust algorithm for estimating a personalized human body model from just two sequentially captured depth images that is more accurate and runs an order of magnitude faster than the current state-of-the-art procedure. Then, we employ the estimated body model to track the pose in real-time from a stream of depth images using a tracking algorithm that combines local pose optimization and a stabilizing database look-up. Together, this enables accurate pose tracking that is more accurate than previous approaches. As a further contribution, we evaluate and compare our algorithm to previous work on a comprehensive benchmark dataset containing more than 15 minutes of challenging motions. This dataset comprises calibrated marker-based motion capture data, depth data, as well as ground truth tracking results. This work is published in Helten *et al.* [2013a].

Existing monocular full body trackers, as the tracker presented in Chapter 5, often fail to capture poses where a single camera provides insufficient data, such as non-frontal poses, and all other poses with body part occlusions. In Chapter 6, we present a novel sensor fusion approach for real-time full body tracking that succeeds in such difficult situations. It takes inspiration from previous tracking solutions, and combines a generative tracker and a discriminative tracker retrieving closest poses in a database. In contrast to previous work, both trackers employ data from a low number of inexpensive body-worn inertial sensors. These sensors provide reliable and complementary information when the monocular depth information alone is not sufficient. We also contribute by new algorithmic solutions to best fuse depth and inertial data in both trackers. One is a new visibility model to determine global body pose, occlusions and usable depth correspondences and to decide what data modality to use for discriminative tracking. We also contribute with a new inertial-based pose retrieval, and an adapted late fusion step to calculate the final body pose. The main ideas of this work are published in Helten *et al.* [2013d].

In Chapter 7, we conclude and give some outlook on future work.

1.3 Publications of the Author

[Helten *et al.* 2011b] **Thomas Helten**, Meinard Müller, Jochen Tautges, and Hans-Peter Seidel. Towards Cross-modal Comparison of Human Motion Data. In *Proceedings of the 33rd Annual Symposium of the German Association for Pattern Recognition (DAGM)*, 2011.

In this article, we consider the cross-model retrieval approach presented in Chapter 3. In particular, we focus on how to compare motion data that originates from optical mocap systems with motion data coming from systems that use inertial sensors.

[Helten *et al.* 2011a] **Thomas Helten**, Heike Brock, Meinard Müller, and Hans-Peter Seidel. Classification of Trampoline Jumps Using Inertial Sensors. In *Sports Engineering*, Volume 14, Issue 2, pages 155–164, 2011.

In this article, we show how trampoline motions can be classified using the techniques presented in Helten *et al.* [2011b]. Specifically, we describe the use of real-valued motion templates that were inspired by work of Müller and Röder [2006]. This publication consists of the main concepts introduced in Chapter 4.

[Helten *et al.* 2013a] **Thomas Helten**, Andreas Baak, Gaurav Bharaj, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Personalization and Evaluation of a Real-time Depth-based Full Body Tracker. In *Proccedings of the third joint 3DIM/3DPVT Conference (3DV)*, 2013.

Obtaining a personalized model for a model-based tracker is a challenging problem which is time consuming and requires expensive specialized equipement. In this article, we focus on obtaining a personalized model from only two sequentially shot depth images. Using an underlying parametric shape model and adaptive model-to-data correspondences, we achieve a shape reconstruction quality comparable to other state-of-the-art methods but in a fraction of the runtime and without user intervention. This publication covers the central ideas from Chapter 5.

[Helten *et al.* 2013d] **Thomas Helten**, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Real-time Body Tracking with One Depth Camera and Inertial Sensors. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2013.

Here, we show how an existing depth-based human motion tracker can be extended to better deal with challenging tracking scenarios that originate from occlusions. To this end, we fuse the information from the depth camera with complementary information from inertial sensors, see Chapter 6.

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, Bodo Rosenhahn. Multisensor-Fusion for 3D Full-Body Human Motion Capture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

In this publication, we show how inertial sensors can be used to stabilize the tracking results of an optical marker-less motion tracker. The main idea is that inertial sensors are not prone to occlusions and provide in form of orientations a complementary type of information. This

information can only hardly be obtained from optical data such as silhouette images that are used by many marker-less tracking approaches.

[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 describe how the orientations of inertial sensors can be used to reveal typical tracking errors that are common to optical markers-less trackers. Many of these errors stem from occlusions or from rotational ambiguities. The described algorithms make use of the fact, as mentioned above, that inertial sensors are not prone to occlusions and provide information that is complementary to the positional information provided by optical systems such as cameras.

[Tautges *et al.* 2011] Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, **Thomas Helten**, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. Motion Reconstruction Using Sparse Accelerometer Data. In *ACM Transactions on Graphics (TOG)*, Volume 30, Issue 3, May 2011.

In this contribution, we introduce an approach to reconstruct full-body human motions using sparse inertial sensor input. In particular, only four 3D accelerometers are used that are attached to the hands and feet of a person. The obtained sensor data is used in two ways. Firstly, it serves as query in a cross-modal retrieval context to find similar motions in a prerecorded database containing high-quality optical motion data. Secondly, the sensor readings control an motion synthesis step that fuses the retrieved motions, sensor readings and kinematic constraints in a unified optimization scheme. The main ideas, are briefly discussed in Section 3.5.

Chapter 2

Sensor Modalities

In this thesis, we focus on motion capture systems based on three different sensor modalities, optical, inertial, and depth sensors, which differ largely in acquisition cost, in the requirements on the recording conditions, and in the kind of data they provide. To this end, we summarize in this chapter some of the fundamental properties of such systems, introduce several motion representations and fix notations used throughout this thesis. In particular, in Section 2.1, we give an introduction to optical sensor systems which are often used in high-quality movie and game productions. Then, in Section 2.2, we focus on inertial sensor-based systems, which have been developed as a less expensive alternative to optical systems. Finally, in Section 2.3, we elaborate on depth sensor-based systems, which are suitable to be used in home user scenarios.

2.1 Optical Sensors

The highest quality of human motion data can be obtained from mocap systems that employ optical sensors. In particular, optical systems use a set of calibrated and synchronized cameras that are facing a so-called capture volume. Inside this volume, one or more actors are performing the motions to be recorded. The size of the capture volume is chosen in a way that every interior point is always seen by multiple cameras. By using multiple views of the same object, expressive 3D information can be deduced by triangulation. Depending on the underlying techniques, optical approaches can be classified into two different kinds: marker-based and marker-less approaches.

2.1.1 Marker-based Approaches

Optical marker-based approches (Figure 2.1 (a)), as used *e. g.* in the passive marker-based Vicon MX system[1] or the active marker-based PhaseSpace system[2], allow for recording human motions with high precision. This is achieved by tracking the positions of so-called markers that are attached to suits worn by the performing actors. The term “passive” or “active” refers to the kind of markers used. Passive markers are retro-reflective and are illuminated by light sources closely

[1] www.vicon.com
[2] www.phasespace.com

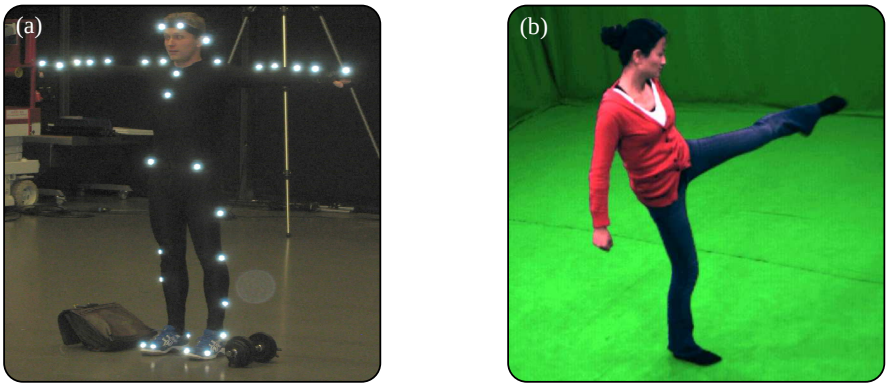

Figure 2.1. Typical optical motion capture approaches. **(a):** Marker-less motion capture system with actor in general apparel. The background is colored for easier foreground/background segmentation. **(b):** Marker-based system, where the actor wears a suite with retro-reflecting markers attached to it. Here, no background segmentation is required.

placed next to each camera, see also Figure 2.2 (a) and (b). In contrast, active systems use LEDs as markers that emit light without being illuminated externally. The idea behind using markers is, that they are easily detectable in the images recorded by the cameras in an robust and automatic manner. From synchronously recorded 2D positions of the markers, the system can then reconstruct 3D coordinates of marker positions using triangulation techniques, see also Figure 2.2 (c). These marker positions build the foundation for computing other useful motion data representations. The advantage of active marker-based systems over passive systems is that they can include an encoded labeling in the emitted light. Thus individual markers can be easily identified, which is—in practice—a non-trivial problem for passive systems.

2.1.2 Motion Data Representations

Marker positions. The simplest motion data representation obtainable from optical marker-based systems are the global 3D-positions of the markers. In our experiments, we use a set of about $M = 40$ markers which are attached to an actor's suit at well-defined locations following a fixed pattern. In this thesis, we model *marker positions* by the vector $\mathcal{P} := (\boldsymbol{p}_1, \ldots, \boldsymbol{p}_M)$, see also Figure 2.3 (a).

Pose parameters. The captured marker positions can then be used to deduce other motion data representations. One important example are parameters of so-called *kinematic chains*, which approximate the human skeleton as a set of rigid bodies, the bones, that are connected and constrained in their movement by a set of joints. We now give a mathematical introduction into kinematic chains and their parametrization. In this context, we use unit quaternions for representing rotations. Note that this is only one possible representation, alternatives are discussed, *e. g.*, in Murray *et al.* [1994]. From now on, if we mention quaternion, we always mean unit quaternion.

In the following, let $\mathbb{Q}$ be the space of quaternions, where one quaternion can be described using four scalar parameters $\mathsf{q} = (w, x, y, z) \in \mathbb{R}^4$. Alternatively, we refer to a quaternion using $\mathsf{q}(\phi, \boldsymbol{a}) \in \mathbb{R} \times \mathbb{R}^3$, which represents a rotation around an axis $\boldsymbol{a}$ by an angle ϕ. Furthermore, let $\mathsf{q}_1 \circ \mathsf{q}_2$ be the quaternion multiplication and $\mathsf{q}[\boldsymbol{v}]$ be the rotation of a vector $\boldsymbol{v} \in \mathbb{R}^3$ using the quaternion q.

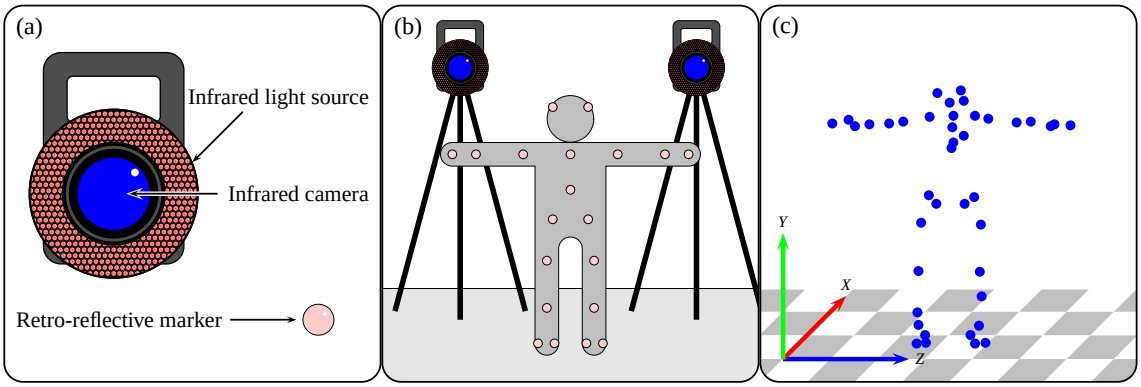

Figure 2.2. (a): Typical marker-based mocap equipment consisting of cameras, a light source mounted close to the camera, and a set of retro-reflecting markers. **(b):** The set-up mocap system consisting of multiple cameras surrounding a capture volume. One actor is standing inside, wearing a suit with markers attached to it. **(c):** The obtained 3D positions of the captured markers.

For further reading on quaternions, we refer to Shoemake [1985]. Let B be the number of bones in the kinematic chain, while J stands for the number of joints. We assume that for every bone $b \in \mathcal{B} = [1:B] := \{1, \dots, B\}$ there is a corresponding coordinate system F_b rigidly attached to it. This allows for a point $\boldsymbol{p} \in \mathbb{R}^3$ to defined relative to a bone.

Now, we describe a joint connecting two rigid bodies $b_1 \in \mathcal{B}$ and $b_2 \in \mathcal{B}$ as 2-tupel $j = (b_1, b_2) \in \mathbb{J} = \mathcal{B}^2$. For each joint, the spatial relationship between two bones is described by a transformation

$$T_j := (\mathsf{q}, \boldsymbol{r}) \in \mathbb{T} = \mathbb{Q} \times \mathbb{R}^3. \tag{2.1}$$

Here, q models a rotational offset between the two bones, while $\boldsymbol{r}$ stands for a transitional offset. In addition, we define the concatenation of two transformations T_1 and T_2 as

$$T_1 \cdot T_2 = (\mathsf{q}_1, \boldsymbol{r}_1) \cdot (\mathsf{q}_2, \boldsymbol{r}_2) := (\mathsf{q}_1 \circ \mathsf{q}_2, \mathsf{q}_1[\boldsymbol{r}_2] + \boldsymbol{r}_1). \tag{2.2}$$

Finally, transformations can be used to transform points relative to one rigid body b_1 to points relative to the other rigid body b_2. Let F_1 and F_2 be the coordinate systems of the two rigid bodies b_1 and b_2 that are connected by a joint $j = (b_1, b_2)$ with transformation T_j. The transformation of a point $\boldsymbol{v}_1 \in \mathbb{R}^3$ relative to F_1 to a point $\boldsymbol{v}_2 \in \mathbb{R}^3$ relative to F_2 is given by

$$T_j[\boldsymbol{v}_1] = (\mathsf{q}, \boldsymbol{r})[\boldsymbol{v}_1] := \mathsf{q}[\boldsymbol{v}_1] + \boldsymbol{r}. \tag{2.3}$$

In practice, we use two parametrized versions of this transformation. The first one is the *revolving joint* which models a joint that can rotate along an axis $\boldsymbol{a}_0 \in \mathbb{R}^3$. Its transformation is described as

$$T_j^{\text{rev}}(\chi) := (\mathsf{q}_j \circ \mathsf{q}(\chi, \boldsymbol{a}_j), \boldsymbol{r}_j). \tag{2.4}$$

Here, χ represents the angle the joint is rotated, while $\boldsymbol{r}_j$ is a constant translational offset and q_j is a constant rotational offset. Similarly, the *prismatic joint* describes a translation along an axis $\boldsymbol{a}_j \in \mathbb{R}^3$. Its transformation is defined as

$$T_j^{\text{pri}}(\chi) := (\mathsf{q}_j, \boldsymbol{r}_j + \chi \boldsymbol{a}_j), \tag{2.5}$$

where $\boldsymbol{a}_j$ represents the axis along which the joint is moved. The quantities $\boldsymbol{r}_j$,$\boldsymbol{a}_j$, and q_j are referred to as *joint properties*. Complex joints that can rotate about more than one axis can be modeled using two or three consecutive revolving joints.

Now, we can define a kinematic chain as $\mathcal{K} := (\mathcal{B}, \mathcal{J}, b_0)$, where $\mathcal{B} = [1 : B]$ are the bones and $\mathcal{J} \subset \mathbb{J}$ are the joints. Additionally, $b_0 \in \mathcal{B}$ marks one bone as so-called *root* of the kinematic chain. This bone is considered to be fixed *w.r.t.* some global coordinate system $\mathsf{F}_{\mathrm{GO}} = \mathsf{F}_{b_0}$. Note that the kinematic chain can be interpreted as a graph, with the bones as nodes and the joints as edges. In this thesis, all kinematic chains are trees that are directed graphs with a designated root node (b_0). For each joint (revolving or prismatic) a transformation T_j, $j \in \mathcal{J}$ is defined. Also, since we get one parameter χ_j for every transformation T_j, we denote a vector of all parameters by

$$\boldsymbol{\chi} := (\chi_1, \ldots, \chi_J)^T. \tag{2.6}$$

Since a kinematic chain is used to approximate the human skeleton with its bones and joints, we will refer to it as *kinematic skeleton* or simply *skeleton* in the rest of this thesis. Also, since the parameter vector $\boldsymbol{\chi}$ defines the pose of the skeleton it is called *pose parameters* or *pose*. A skeleton in a pose $\boldsymbol{\chi}$ is denoted by $\mathcal{K}_{\boldsymbol{\chi}}$.

Given a position $\boldsymbol{p} \in \mathbb{R}^3$ within the skeleton relative to some bone $b \in \mathcal{B}$, we can reconstruct its *global position* relative to F_{GO} for every given pose $\boldsymbol{\chi}$. Its global position is given as

$$\mathcal{K}_{\boldsymbol{\chi}}[\boldsymbol{p}] := \left(\prod_{j \in \mathcal{J}(p)} T_j \right) [\boldsymbol{p}], \tag{2.7}$$

where $\mathcal{J}(p)$ represents the unique ordered list of joints that connects the bone b to the root b_0. To model a global motion of the skeleton, the first tree joints are normally prismatic joints with their axes parallel to the cardinal axes X, Y, and Z of the global coordinate system F_{GO}. The process of obtaining global positions of points inside the skeleton—as for example joint positions—is called *forward kinematics*. For further reading, we refer to Murray *et al.* [1994].

Reversely, three steps are required, to obtain pose parameters $\boldsymbol{\chi}$ from a set of *captured marker* positions. Firstly, one must design a skeleton that resembles the actor whose motion data is to be transformed into pose parameters. This involves an accurate estimation of the joint properties $\boldsymbol{a}_0$, $\boldsymbol{r}_0$, and $\boldsymbol{q}_0$ for every joint of the skeleton. Secondly, one has to model the placement of the markers used with relation to the bones of the skeleton. The placement of this modeled *virtual markers* is then considered fixed for the actual conversion process. Finally, an optimization scheme is employed to find those pose parameters that induce a pose of the skeleton, where the positions of the virtual markers best explain the positions of the captured markers. This process is called *inverse kinematic*. For details, we refer to Bregler *et al.* [2004].

Surface mesh. Another important representation that is used in this thesis, are meshes $\mathcal{M}$, which represent the surface, *e. g.*, the skin and/or cloth of a virtual character in a movie or computer game, see Figure 2.3 (c). Mathematically, a mesh is given as graph, where its nodes are called vertices. Small groups of neighboring vertices now form faces. The most common form of a mesh is the triangle mesh, where each face consists of exactly three vertices. To reduce the number of parameters, meshes are often parametrized using *i. e.* the skeleton and joint angle concept mentioned

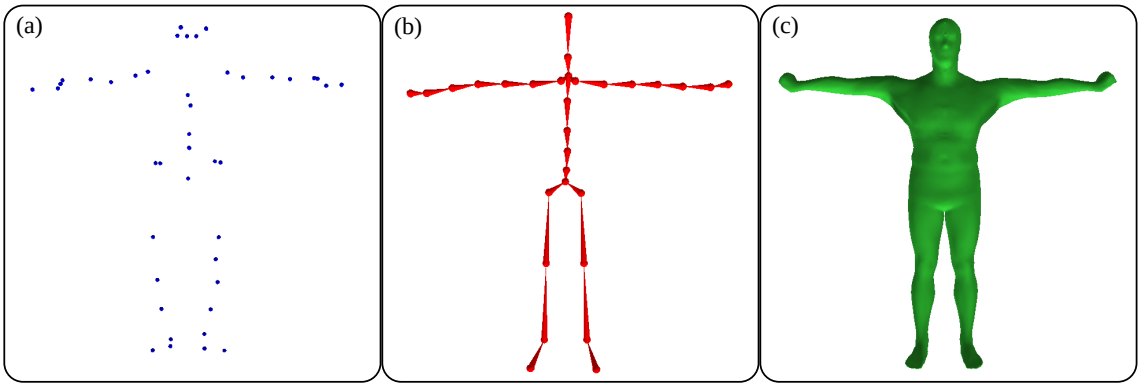

Figure 2.3. Different kinds of optical motion data representations for a person striking a so-called T-pose: **(a):** The marker positions $\mathcal{P}$. **(b):** The kinematic chain $\mathcal{K}_\chi$ with pose parameters χ that were obtained using the marker positions $\mathcal{P}$. **(c):** The triangle surface $\mathcal{M}_\chi$ defined by $\mathcal{K}_\chi$.

above. To this end, a process called *skinning* is used, which relates the position of each vertex in the mesh to a combination of joint positions. If now the skeleton is striking a pose χ, the vertex positions can be reconstructed from the joint positions that by itself have been reconstructed using forward kinematics. The resulting mesh is denoted by $\mathcal{M}_\chi$. For details on mesh skinning, we refer to James and Twigg [2005]. The acquisition of such a surface mesh for a give person is a non-trivial task and is in practice mostly done by manual modeling or by measurement using a laser scanner. Both processes are costly and time consuming. In Chapter 5, we contribute an approach that is easy and fast using only one inexpensive depth sensor, as introduced in Section 2.3.

In this thesis, the last two representations are also referred to as *body models*, since they mimic the overall appearance of the human body.

2.1.3 Marker-less Approaches

In contrast, marker-less approaches deduce full-body human motion data from multi-view images without requiring the actors to wear any special garment or markers, see also Figure 2.1 (b). This makes such systems easier to use and less intrusive than marker-based approaches. While eliminating some of the disadvantages of marker-based approaches, this generalization implies challenges in its own and is still subject to active research, see *e. g.* Bregler *et al.* [2004]; Deutscher and Reid [2005]; Bălan *et al.* [2007]; Pons-Moll *et al.* [2010, 2011]; Stoll *et al.* [2011]. The following overview over state-of-the-art approaches was published in Helten *et al.* [2013c].

Most marker-less approaches use some kind of underlying body model such as skeletons augmented by shape primitives like cylinders (Bregler *et al.* [2004]), surface meshes (Gall *et al.* [2009]; Pons-Moll *et al.* [2010]; Liu *et al.* [2011]) or probabilistic density representations attached to the human body Stoll *et al.* [2011]. Optimal skeletal pose parameters are often found by minimizing an error metric that assesses the similarity of the projected model to the multi-view image data using features. Local optimization approaches are widely used due to their high efficiency, but they are challenged by the highly multi-modal nature of the model-to-image similarity function Stoll *et al.* [2011]; Liu *et al.* [2011]. Global pose optimization methods can overcome some of these limitations, however at the price of needing much longer computation times, see *e. g.* Deutscher *et al.* [2000]; Gall *et al.* [2009]. Some approaches aim to combine the efficiency of local

methods with the reliability of global methods by adaptively switching between them (Gall *et al.* [2009]). Even though marker-less approaches succeed with a slightly simpler setup, many limitations remain: computation time often precludes real-time processing, recording is still limited to controlled settings, and people are expected to wear relatively tight clothing. Furthermore, marker-less motion capture methods deliver merely skeletal motion parameters.

In contrast, marker-less performance capture methods go one step further and reconstruct deforming surface geometry from multi-view video in addition to skeletal motion. Some methods estimate the dynamic scene geometry using variants of shape-from-silhouette methods or combinations of shape-from-silhouette and stereo, see *e. g.* Starck and Hilton [2005, 2007a,b]; Matusik *et al.* [2000]. But, in such approaches, establishing space-time coherence is difficult. Template-based methods deform a shape template to match the deformable surface in the real scene, which implicitly establishes temporal coherence (de Aguiar *et al.* [2008]; Vlasic *et al.* [2008]), also in scenes with ten persons. All the developments explained so far aim towards the goal of high-quality reconstruction, even if that necessitates complex and controlled indoor setup.

2.1.4 Advantages and Disadvantages

One particular strength of optical marker-based systems is that they provide positional motion data of high quality. In particular, the data can be used to compute several other motion representations that are of practical use in many fields. However, requiring an array of calibrated high-resolution cameras as well as special garment equipment, such systems are cost intensive in acquisition and maintenance. This drawback is partly removed by marker-less mocap systems, but overall the effort to setup and calibrate the system stays high. Furthermore, many of the available optical mocap systems are vulnerable to bright lighting conditions thus posing constraints on the recording environment (*e. g.* illumination, size of the capture volume, indoor).

2.2 Inertial Sensors

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 to maintain. Therefore, such sensors have become increasingly popular and are now widely used in many commercial products. However, inertial sensors do not provide positional data relative to a global coordinate system, which renders them difficult to use as a direct replacement for optical mocap systems.

2.2.1 Inertial Measurement Unit

The key-component of an inertial sensor-based mocap system is the so-called *inertial measurement unit* (IMU), which consists of two inertial sensor types, the accelerometer and the rate-of-turn sensor and one additional magnetic field sensor, see Figure 2.4 (a). All these sensor are nowadays put together into a small box that can easily be attached to an object or person. By fusing the information from all three sensor types, the IMU is able to tell its orientation $\mathbf{q}$ with respect to

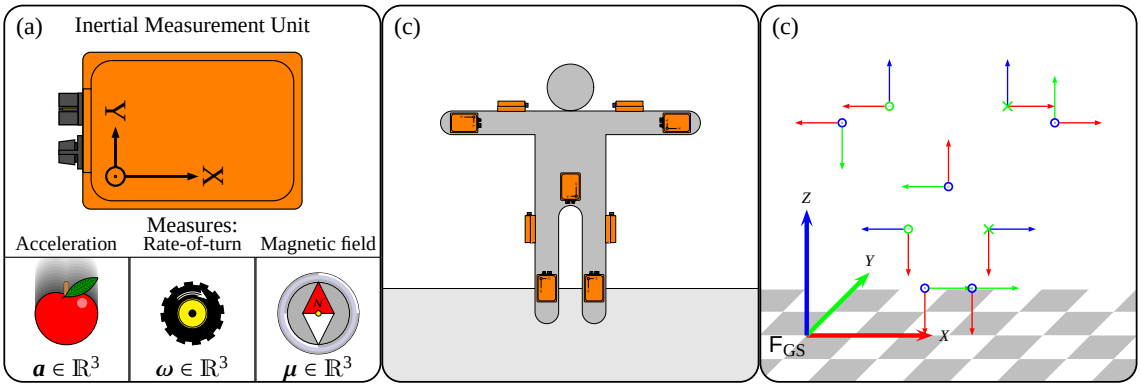

Figure 2.4. Working principle of inertial sensor-based mocap. **(a):** An inertial measurement unit (IMU) consists of an accelerometer, a rate-of-turn sensor, and a magnetic field sensor. By fusing all these information, an IMU can determine its orientation with respect to an global coordinate system F_{GS}. **(b):** To capture human motion data, several IMUs are attached to a person. **(c):** The resulting data are the orientations of all IMUs with respect to the common global coordinate system F_{GS}.

some global coordinate system F_{GS}. As mentioned above, inertial sensors cannot be used to infer meaningful positional information relative to a global coordinate system. This stems from the fact that positions have to be deduced from measured accelerations by twofold integration. Because of the measurement noise, this induces a large drift to the derived positions. Without compensating for that drift, the derived positions cannot be used practically. However, by attaching several IMUs to the limbs of an actor's body (Figure 2.4 (b)), one can obtain dense rotational information and deduce relative positional information about the actor's limb configuration, see Figure 2.4 (c).

The process of obtaining the orientation q involves several steps, which we will explain briefly in the following. The three sensors included in the IMU provide three basic measurements: the acceleration $\boldsymbol{a} \in \mathbb{R}^3$, the rate-of-turn or angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$, and the vector of the magnetic field $\boldsymbol{\mu} \in \mathbb{R}^3$. Note that the measured acceleration always contains, as one component, the acceleration caused by gravity. Therefore, the measured acceleration $\boldsymbol{a}$ can be thought of a superposition $\boldsymbol{a} = \bar{\mathsf{q}}[\boldsymbol{m} + \boldsymbol{g}]$ consisting of the gravity $\boldsymbol{g}$ and the actual acceleration $\boldsymbol{m}$ of the motion, see also Figure 2.5 (a). Here, the quantities $\boldsymbol{a}$, $\boldsymbol{\omega}$, and $\boldsymbol{\mu}$ are given in the sensors's local coordinate system F_{LS}, while $\boldsymbol{m}$ and $\boldsymbol{g}$ are given in the global coordinate system F_{GS}. The term $\bar{\mathsf{q}}[\cdot]$ represents the transformation from the global coordinate system to the sensor's local coordinate system (see also below).

If $\|\boldsymbol{m}\|$ is small with respect to $\|\boldsymbol{g}\|$, $\boldsymbol{a}$ can be used as approximation of $\boldsymbol{g}$. This fact is often exploited in many portable devices such as recent mobile phones to calculate the device's orientation with respect to the canonical direction of gravity (Lee and Ha [2001]). We use this fact, to define one axis Z of our global coordinate system F_{GS}:

$$\boldsymbol{g} \approx \boldsymbol{a}, \text{ if } \|\boldsymbol{m}\| \approx 0, \tag{2.8}$$

$$\Rightarrow \hat{\boldsymbol{g}} := \frac{\boldsymbol{g}}{\|\boldsymbol{g}\|} \approx \frac{\boldsymbol{a}}{\|\boldsymbol{a}\|}, \tag{2.9}$$

$$Z := \hat{\boldsymbol{g}}. \tag{2.10}$$

In order to obtain a valid global coordinate system, we need to define another axis. Most IMUs use the measurements of the magnetic field sensor to derive the canonical direction "north" $\hat{N}$.

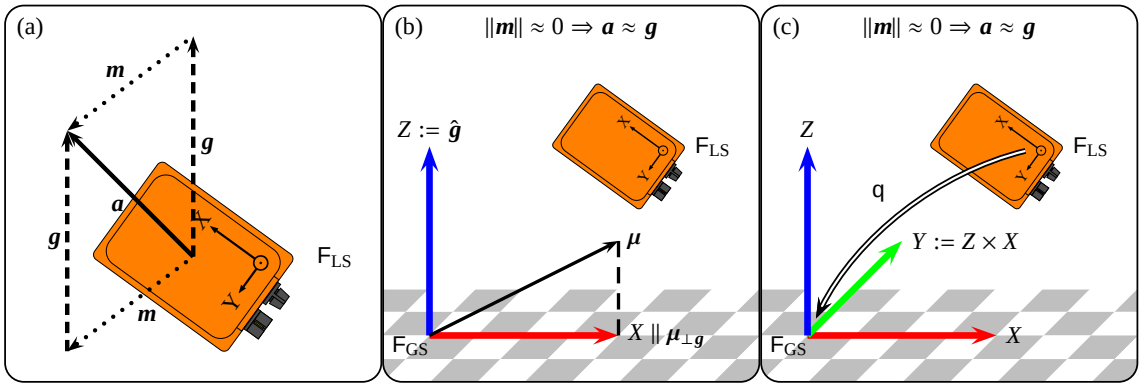

Figure 2.5. Measurement of the global coordinate system F_{GS}. **(a):** The measured acceleration $\boldsymbol{a}$ is a superposition of the acceleration induced by gravity $\boldsymbol{g}$ and the acceleration due to motion $\boldsymbol{m}$. **(b):** If $\boldsymbol{m}$ is negligible, the measured acceleration $\boldsymbol{a}$ can be considered a good approximation for $\boldsymbol{g}$. The direction of gravity $\hat{\boldsymbol{g}}$ defines the first axis of the global coordinate system F_{GS}. The second axis, X, is defined by the components of the magnetic field vector $\boldsymbol{\mu}$ that are perpendicular to Z. **(c):** The axis Y is defined to be perpendicular to both X and Z so that all three axes form a right handed coordinate system. The transformation from F_{LS} to F_{GS} is denoted by q.

This involves calculating the offsets inclination and declination between the direction to the north magnetic pole $\mathcal{N}$ and $\hat{N}$, which depend on the position on earth, where $\mathcal{N}$ is measured. For further reading on this topic we refer to Baak [2012]. In the following, we use $\boldsymbol{\mu}$ instead of $\hat{N}$ to define F_{GS}. By projecting $\boldsymbol{\mu}$ onto the horizontal plane defined by its normal direction Z, we obtain the direction of the X-axis of the global coordinate system F_{GS}. To be precise, we define

$$\mu_{\perp \boldsymbol{g}} := \boldsymbol{\mu} - \hat{\boldsymbol{g}} \langle \boldsymbol{\mu}, \hat{\boldsymbol{g}} \rangle \,, \text{and} \tag{2.11}$$

$$X := \frac{\mu_{\perp \boldsymbol{g}}}{\|\mu_{\perp \boldsymbol{g}}\|}, \tag{2.12}$$

see also Figure 2.5 (b). Here, $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. Furthermore, we define $Y := Z \times X$, where $\times$ is the cross-product of two vectors in $\mathbb{R}^3$. Finally, the orientation q is defined as the rotation to transform a vector from F_{LS} to F_{GS}. As in Section 2.1.2, the transformation itself is denoted by $\mathsf{q}[\cdot]$ and represented as unit quaternion, see Figure 2.5 (c). The inverse rotation is referred to by $\overline{\mathsf{q}}$.

2.2.2 Stabilization using Rate-of-Turn

The above definition of F_{GS} has one important flaw. It assumes that $\|\boldsymbol{m}\|$ is small. This might be true in some situations such as when the sensor is in rest or moves at constant speed. In general, however, this is not true. In particular, when capturing human motions which, involves complex muscle driven dynamics, the assumption does not hold. For this reason, the measurements from the rate-of-turn sensor are employed to stabilize the estimation of F_{GS}. To be precise, $\boldsymbol{\omega}$ represents the angular velocity of the inertial sensor or how the local coordinate system F_{LS} changes its orientation with respect to the global coordinate system F_{GS} over time, see Figure 2.6 (a). Equally one can say that $-\boldsymbol{\omega}$ represents how the global coordinate system F_{GS} changes with respect to F_{LS}, see Figure 2.6 (b). To compute the change over time $(\dot{X}, \dot{Y}, \dot{Z})$ of the axes (X, Y, Z) of the coordinate system F_{GS}, one has to convert the quantity $-\boldsymbol{\omega}$ from F_{LS} to F_{GS}. This can be done by using q,

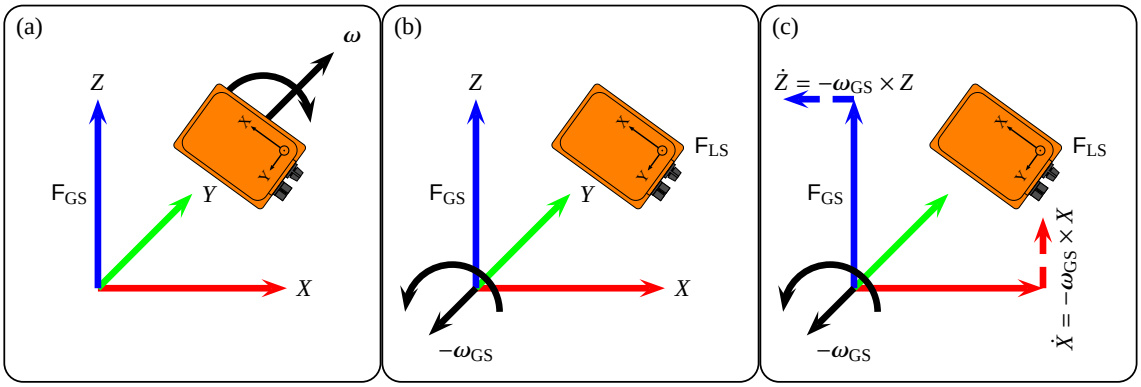

Figure 2.6. Prediction of how F_{GS} changes over time using $\boldsymbol{\omega}$. **(a):** $\boldsymbol{\omega}$ is measured by the sensor and describes how F_{LS} changes with respect to F_{GS}. Here, $\boldsymbol{\omega}$ is defined inside F_{LS}. **(b):** In contrast, $-\boldsymbol{\omega}_{GS}$ represents how F_{GS} changes with respect to F_{LS}. Note that here, $-\boldsymbol{\omega}_{GS} = \mathsf{q}[-\boldsymbol{\omega}]$ is defined inside F_{GS}. **(c):** The changes of the coordinate axes (X, Y, Z) of F_{GS} can now be expressed with: $(\dot{X}, \dot{Y}, \dot{Z}) = (-\boldsymbol{\omega}_{GS} \times X, -\boldsymbol{\omega}_{GS} \times Y, -\boldsymbol{\omega}_{GS} \times Z)$.

and is mathematically expressed by

$$-\boldsymbol{\omega}_{GS} = \mathsf{q}[-\boldsymbol{\omega}]. \tag{2.13}$$

Now, the change of the coordinate axes is defined as

$$\dot{X} = -\boldsymbol{\omega}_{GS} \times X, \tag{2.14}$$
$$\dot{Y} = -\boldsymbol{\omega}_{GS} \times Y, \text{and} \tag{2.15}$$
$$\dot{Z} = -\boldsymbol{\omega}_{GS} \times Z, \tag{2.16}$$

see also Figure 2.6 (c). For further reading, we refer to Murray *et al.* [1994]. With $(\dot{X}, \dot{Y}, \dot{Z})$ given at a point in time t and the axes $(X_{prev}, Y_{prev}, Z_{prev})$ of F_{GS} defined at some previous time t_{prev}, one can calculate a prediction for *e. g.* the X-axis of F_{GS} with

$$X_{pred} := X_{prev} + \int_{t_{prev}}^{t_{pred}} \dot{X}\, dt. \tag{2.17}$$

This holds analogously for Y_{pred}, and Z_{pred}.

To recapitulate, one can use $\boldsymbol{\omega}$ to predict the orientation of F_{GS} with respect to F_{LS} in situations $\|\boldsymbol{m}\|$ can not considered to be small. However, this prediction only works for a small amount of time, since $\boldsymbol{\omega}$ is subject to noise and integrating over a longer time will likely result in the prediction of F_{GS} drifting away from the definition of F_{GS} using $\boldsymbol{a}$ and $\boldsymbol{\mu}$, if $\|\boldsymbol{m}\| \approx 0$.

In practice, the computation of F_{GS} is often realized in a predictor/corrector scheme using a Kalman filter, which was presented in Kalman [1960]. Here, the angular velocity $\boldsymbol{\omega}$ serves in a predictor for F_{GS}. As corrector, the definition of F_{GS} using $\boldsymbol{a}$ and $\boldsymbol{\mu}$ is employed. This results in a drift-free definition of the global coordinate system F_{GS}, which is—to a great extend—independent of the individual IMU. This last fact is especially important in the context of human motion data acquisition, where the measurements of several IMUs is related to each other. For details and further reading, we refer to Lee and Ha [2001]; Kemp *et al.* [1998]; Luinge and Veltink [2005].

2.2.3 Motion Data Representations

Besides the directly measured quantities such as the acceleration $\boldsymbol{a}$, the angular velocity $\boldsymbol{\omega}$, the magnetic field $\boldsymbol{\mu}$, or the orientation q, inertial sensors can be used to derive many more interesting motion data representations that are used in practice. For example, when placing IMUs densely (in general one per limb) on a person to track, the orientations of the sensors can be used to derive a skeleton representation—including joint angles—which is similar to the one obtainable using optical sensor-based systems. This is for example used in the commercial solution provided by the Xsens MVN system[3]. However, the usage of a feasible number of IMUs is constrained by their cost. Furthermore, an estimation of the global position of the skeleton with respect to F_{GS} is not possible.

2.2.4 Advantages and Disadvantages

Inertial sensors in the context of human motion data acquisition have one important advantage: they do not need visual cues and work in almost every environment. This enables their application in places, where optical systems do not work reliably or where optical systems cannot be set up. In other words, inertial sensor-based systems can be used, where large recording volumes are required or where the lighting conditions can not be controlled. Furthermore, their reduced acquisition and setup costs make them available to a larger number of users. For these reasons, they are often used in low-cost movie productions or in sports training analysis. However, they have the drawback that they only measure their orientation and not their position with respect to some global coordinate system. For that reason, it is not possible to tell the global position of a person to be captured or the relative positions of several actors in the same scene. Also, the number of IMUs that are required for full body motion capture renders it still impractical to be used in home application scenarios.

2.2.5 Virtual Sensors

Local accelerations and directional information, as provided by inertial sensors, can also be generated from positional information that comes from an optical mocap system. This concept is called *virtual sensor*, since it simulates the output of a sensor, which does not exist in reality. In this context, we assume that a skeleton representation is present and its pose parameters can be obtained using the techniques described in Section 2.1.2. Now, a virtual sensor is considered to be rigidly attached to one bone of the skeleton. Given pose parameters $\boldsymbol{\chi}$, one can calculate the location and orientation of the sensor's local coordinate system F_{LS} with respect to the global coordinate system F_{GO}, which is defined by the optical marker-based tracker. Note that the global coordinate system F_{GO} is not the same as the global coordinate system F_{GS} defined earlier in this section. Nevertheless, since it is the same for all virtual sensors, it can be used to calculate a meaningful orientation q. Similarly, the position $\boldsymbol{p}$ of the sensor with respect to the global coordinate system F_{GO} can be computed. The global acceleration $\boldsymbol{m}$ is now obtained by double differentiation of $\boldsymbol{p}$. By adding the gravity vector $\boldsymbol{g}$ and transforming this quantity to the virtual sensor's local coordinate system F_{LS} using $\overline{\mathsf{q}}$, one finally gets the local acceleration $\boldsymbol{a} = \overline{\mathsf{q}}[\boldsymbol{m}+\boldsymbol{g}]$. In the same way, also

[3] http://www.xsens.com/en/general/mvn

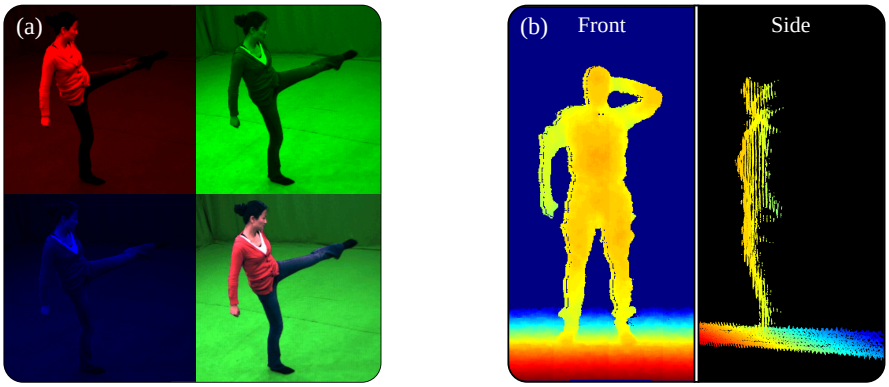

Figure 2.7. (a): Intensity images obtained from a traditional RGB-camera. **(top-left):** Red channel. **(top-right):** Green channel. **(bottom-left):** Blue channel. **(bottom-right):** Reconstructed color image. **(b):** Typical data obtained from a depth sensor. Red pixels are points close to the camera. Blue pixels are points far away from the camera. **(left):** Displayed from the front. **(right):** Displayed from the side.

suitable values for $\boldsymbol{\omega}$ and $\boldsymbol{\mu}$ could be computed. In this thesis, however, we will only use virtual sensors to obtain values for q and $\boldsymbol{a}$.

2.3 Depth Sensors

The third sensor modality covered in this thesis are so-called depth sensors. This sensor modality is related to optical sensors, described above, in the sense that they also use a camera to capture a scene from a given point of view. However, the provided data fundamentally differs. Traditional cameras, as used by optical mocap systems, provide a so-called intensity image. Here, each pixel of the image represents the intensity of light of a specific wavelength. In case of a standard RGB-camera, by combining *e. g.* the intensities of red, green, and blue light, a colored image covering a large portion of the color-space perceptible by humans can be reconstructed, see Figure 2.7 (a).

In contrast, depth cameras provide an image, where each pixel contains the distance of a point in the scene with respect to the camera. From such a so-called 2.5D depth map, a point cloud can be deduced, which gives an approximate 3D reconstruction of the scene. Since depth sensors are not much larger than intensity-based cameras, they provide an attractive way to obtain 3D geometry from a single viewpoint. However, since the scene is only captured from a single viewpoint, only surfaces are captured that are directly visible to the camera. An example can bee seen in Figure 2.7 (b).

2.3.1 Time of Flight Approaches

Currently, among depth sensors, there exist two different approaches using either *time-of-flight* (ToF) or *structured light*. Time-of-flight cameras measure the time Δt the light takes to fly the distance Δx between a point in the scene and the camera. Since the speed of light c is constant, this yields

$$\Delta x = c \cdot \Delta t. \tag{2.18}$$

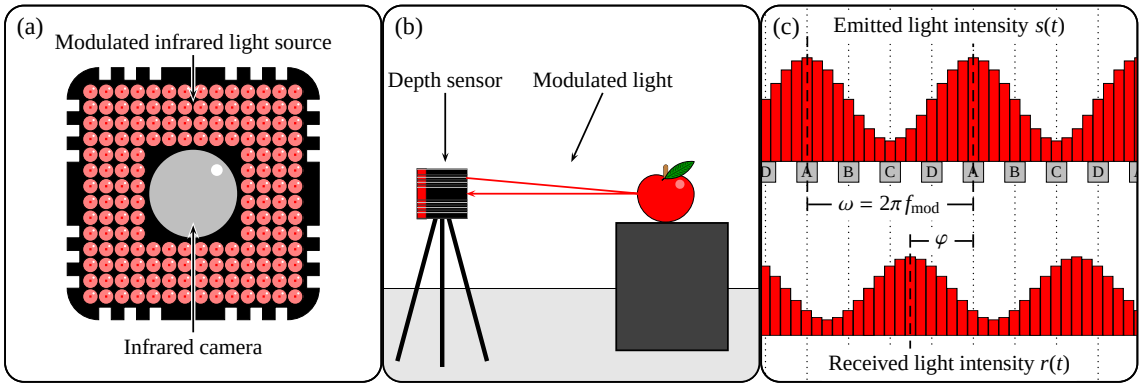

Figure 2.8. Working principle of a time-of-flight depth sensor. **(a):** The sensor seen from the front with the infrared light source surrounding an infrared camera. **(b):** The light source emits modulated light, which is reflected by the scene and captured by the camera. **(c):** The light modulation follows a sinusoidal pattern (top) with frequency f_{mod}, which is attenuated and phase shifted, when received from the scene (bottom). From the phase offset φ, measured by four samplings A,B,C, and D per full modulation cycle, the distance of a point in the scene can be deduced.

However, measuring Δt directly is not feasible, for that reason current ToF cameras use indirect techniques. Exemplarily, we will sketch the approach used by the SwissRanger camera by Mesa Imaging[4]. Other ToF-based depth sensors are constructed by PMD[5] or SoftKinetic[6]. Some of the images in this section are inspired by the manual for the SwissRanger SR4000 camera. The mathematical background is based on Kolb *et al.* [2009].

The main components of the SR4000 camera are same as for every other camera using the ToF approach: a controllable infrared light source and an infrared camera, see Figure 2.8 (a). In the case of the SR4000, the infrared light source emits modulated light, which is reflected by the scene and captured by the infrared camera, see Figure 2.8 (b). This modulation can be thought of as a sinusoidal change in the intensity of the emitted light and could be modeled by the function

$$s(t) := \cos(\omega t), \text{with} \tag{2.19}$$

$$\omega := 2\pi f_{\text{mod}}. \tag{2.20}$$

Here, f_{mod} is the modulation frequency of the light source. An example of such an intensity change is depicted in Figure 2.8 (c, top). Now, the received light in each pixel of the sensor of the camera is represented by the function

$$r(t) := b + a \cdot \cos(\omega t + \varphi). \tag{2.21}$$

Here, $a < 1$ represents the attenuation of the signal, b is some constant bias, and φ is the phase offset between the emitted signal s and the received signal r, see also Figure 2.8 (c, bottom). This phase offset φ originates in the time the light took to travel from the light source into the scene and back to the camera. As a consequence, Δx can be deduced by calculating φ. In practice, the

[4] www.mesa-imaging.ch
[5] www.pmd.com
[6] www.softkinetic.com

parameters a, b and φ, are obtained by sampling a so-called mixing function m, defined as

$$m(\tau) = s \otimes r \tag{2.22}$$

$$= \lim_{T\to\infty} \int_{-T/2}^{T/2} s(t) \cdot r(t+\tau)\, dt \tag{2.23}$$

$$= \frac{a}{2} \cos(\omega\tau + \varphi), \tag{2.24}$$

at different phase offsets $\tau_i = \frac{\pi}{2} i, i \in \{0, \ldots, 3\}$. The four resulting samples are called $A = m(\tau_0)$, $B = m(\tau_1)$, $C = m(\tau_2)$, and $D = m(\tau_3)$, see also Figure 2.8 (c). Now, we can compute

$$\varphi = \arctan2(D - B, A - C), \text{and} \tag{2.25}$$

$$\Delta x = c \cdot \Delta t = \frac{c}{2\omega}\varphi = \frac{c}{4\pi f_{\text{mod}}}\varphi. \tag{2.26}$$

This procedure is conducted for each pixel in the depth image independently.

Note that, using the above formulation, the effective measurable distance Δx of any point is bound to the interval $[0, \frac{c}{4\pi f_{\text{mod}}})$, which is dependent on the modulation frequency of the light. For example, if the modulation frequency is around 15 MHz, the interval is around $[0, 10)$ m. All distances outside this interval are implicitly mapped into this interval. For example, in case of $f_{\text{mod}} = 15$ MHz, an object at 12 m distance would appear to be at 2 m distance and so forth. In practice, the phase offset is determined using not only one set of samples but several, which are drawn over time. This is required to reduce the influence of noise to the measurement. Unfortunately, this also gives rise to systematic errors in situations, where the distance to be measured changes during the measurement, *e. g.*, when parts of the scene move. In this case, some of the measurements might stem from an object in the static background and some of the measurements origin from an object in the foreground. This also happens, in static scenes, close to corners of an object in the foreground. The resulting distance is some kind of average between the depth of the background and the foreground. As consequence, these depth pixels seem to fly, detached from geometry, in the scene. For this reason, this kind of error is called "flying pixels", see also Figure 2.10 (a).

Another typical error related to ToF-based Depth sensors is that originally sharp concave corners look rounded and smooth in the measured depth image. This error is related to the fact that light does not always take the shortest path from the light source to the camera. An example is shown in Figure 2.10 (b), where a part of the light is not directly reflected to the camera but bounces of the wall a second time. In this case, since the sensor averages over several measurements, the measured distance is higher than the real distance. This kind of error is called "multi path error". For further details on ToF imaging and its applications, we refer to Davis *et al.* [2013].

2.3.2 Structured Light Approaches

The other approach to obtain depth images is by means of structured light projection as, *e. g.*, employed by sensors using the design by Primesens[7] such as the first Microsoft Kinect[8] or the

[7] www.primesense.com

[8] www.microsoft.com/en-us/kinectforwindows

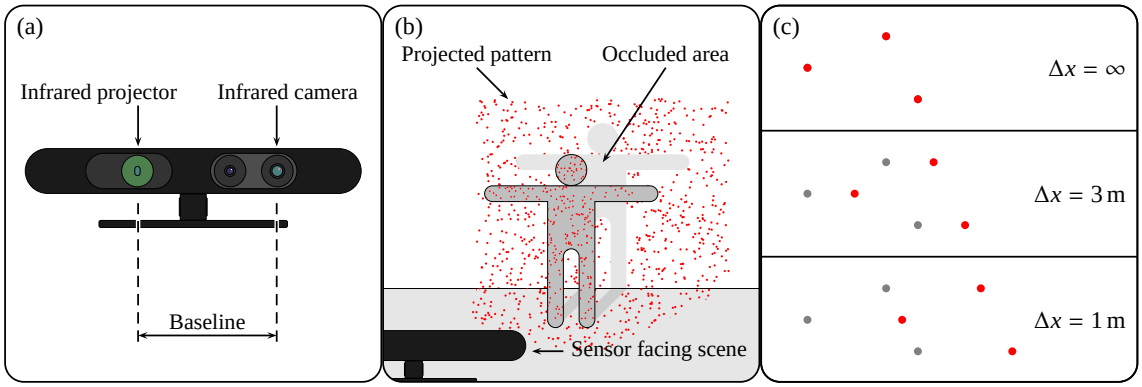

Figure 2.9. Working principle of depth cameras that use the structured light approach such as the Asus Xtion or the Microsoft Kinect. **(a):** Depth camera with locations of the infrared projector and the infrared camera. The distance between projector and camera is called baseline. **(b):** The projector projects a point pattern into the scene. **(c):** From the 2D-location of a point group in the pattern, seen from the camera (red), relative to the 2D-location of the same point group "seen" from the projector (gray), the distance of the point group to the sensor can be deduced.

Asus Xtion PRO LIVE[9]. The central components of such sensors are an infrared projector and an infrared camera that are separated from each other by a fixed offset, called the *baseline*, see also Figure 2.9 (a). Note that the middle "eye" depicted in the figure is a standard RGB intensity camera that is not used for obtaining 3D information and which is ignored in the following discussion.

The projector is used to project a fixed point pattern into the scene, see also Figure 2.9 (b). Depending on the distance and the baseline between the infrared camera and the projector, the observed pattern is distorted compared to the projected pattern. This effect is called depth dependent disparity. In the following, we consider the observed pattern and the projected pattern as two images of the same pattern. Algorithms which infer the original depth from two such images are called depth-from-stereo approaches or depth-from-disparity approaches. Here, for every point in the one image, the semantically same point is identified in the other image and their relative offset—their disparity—is calculated. Finally, the distance of the point can be deduced from the disparity, if the baseline of the cameras and their intrinsic parameters are known.

The naïve approach to identify semantically similar points would be to search for every point in the projector image the corresponding point in the camera image. However, the process of identifying semantically same points for general images is computationally complex and prone to errors, when the appearance or lighting differs too much in both images. For this reasons, practical implementations use different approaches for identifying corresponding parts of the images. Unfortunately, the exact technical details how this is done are not disclosed by the camera manufacturer[10].

The most probable approach to speed up the process of finding correspondences would be to design a pattern that by looking at an arbitrary group of points, the position with respect to the whole pattern can be deduced. This could be achieved by either encoding coordinates in the pattern or by making each point group of the pattern somehow unique. Independent of how exactly this is achieved, the design needs to be robust to strong distortions of the pattern and partial occlusions. Now, independent of what actual algorithm was chosen to identify correspondences, the depth is

[9] `www.asus.com/Multimedia/Xtion_PRO_LIVE`
[10] PrimeSense Patent WO 2007/043036 A1

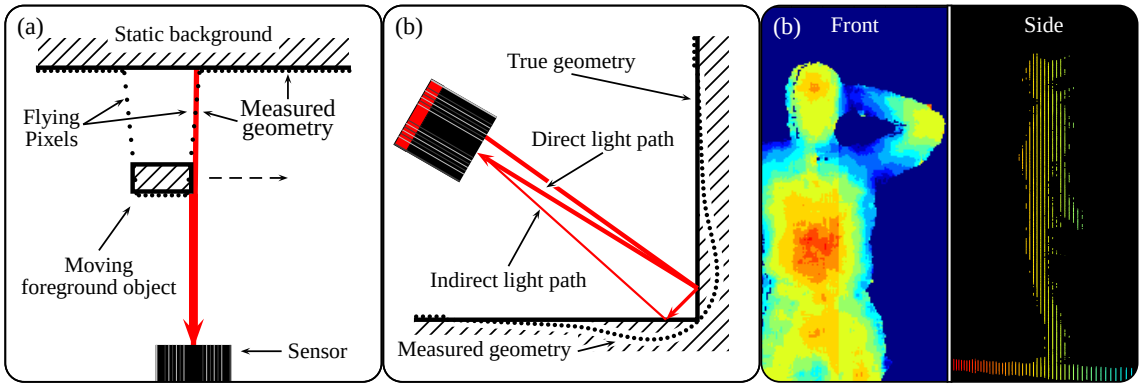

Figure 2.10. (a): "Flying pixels" effect that occurs if an object in the foreground is moving relative to a ToF-sensor. Light for the depth measurement of one pixel might stem from both, the foreground and the background. **(b):** The light for measuring the distance from one point in the scene might have taken not only the shortest path but multiple, possibly longer paths. This "multi-path" effect results *e. g.* in sharp concave corners observed by a ToF-sensor to appear rounded. **(c):** Artifacts of structured-light-based approaches. **(left):** Cloudy appearance of the depth data. **(right):** Quantized depth values.

calculated based on the disparity of two corresponding point groups, see Figure 2.9 (c).

Structured light-based approaches, in contrast to ToF approaches, only need a single image or measurement per time frame to obtain an depth estimate. This makes them robust to motion related artifacts such as flying pixels. Furthermore, multi-path-related problems also do not occur. However, the use of point groups for estimating the distance results in distance measures that are not point accurate but appear cloudy, see Figure 2.10 (c, left). Also, since the 2D-locations of the single points of the group are measured by a camera sensor with a finite spatial resolution, the resulting depth values are quantized, see Figure 2.10 (c, right) For further information on how structured light approaches work in detail, we refer to Zhang *et al.* [2003].

2.3.3 Motion Data Representations

Depth image. The most fundamental data representation that is obtained from a depth sensors is a so-called depth image $\mathcal{I}$, which is similar to a color image but encodes in each pixel the distance to a point in the scene. An example of a depth image is shown in Figure 2.7 (b, left) and Figure 2.10 (c, left).

Point Cloud. Using the intrinsic and, optionally, extrinsic parameters of the depth camera, one can deduce a point cloud of the scene from the depth image. Note that for each pixel i in the depth image maximal one point $\boldsymbol{p}_i \in \mathbb{R}^3$ in the scene can be reconstructed. This also implies that there is no information of points that are not exposed to the camera. An example of such a point cloud is shown in Figure 2.7 (b, right) and Figure 2.10 (c, right). Here, only the front half of the person is visible in the point cloud.

Joint angles and surface mesh. Determining pose parameter or mesh representations from depth images is an active field of research. In this thesis, we will contribute to this field tech-

niques and concepts that are presented in Chapter 5 and Chapter 6. An elaborated introduction into algorithms that deduce such high-level representations of human motion data from depth images is given in Section 5.1.

2.3.4 Advantages and Disadvantages

Depth camera-based systems present an easy way to obtain rich 3D geometry information from a scene. Additionally, the 3D data enables easier foreground/background segmentation compared to optical marker-less systems. Furthermore, even monocular depth data provides information rich enough for many full-body human motion capture approaches in controlled scenarios. in general, however, tracking from depth data is a challenging problem as depth data is subject to noise and systematic artifacts such as “flying pixels” or coarse quantization. Furthermore, monocular tracking approaches are susceptible to occlusion, where no information can be deduced. Here, one naïve idea would be the use of multiple depth sensors at the same time.

But, the use of several depth sensors simultaneously bears its own challenge since these cameras, in contrast to color cameras, interfere with each other’s measurement. In order to reduce the interference of multiple Kinects (structured light approach), Maimone and Fuchs [2012]; Butler *et al.* [2012] apply vibration patterns to each camera. These vibrations have the effect that the point pattern projected by one Kinect looks blurred when seen from a different Kinect. In contrast, the pattern does not look blurred for the Kinect it is projected from, since its projector is moved in the same way its camera is.

In case of ToF-based depth sensors, interference can be avoided by using different modulation frequencies for each camera. However, even when using multiple depth cameras, occlusions are difficult to prevent in many scenarios. Also, similar to optical systems, depths cameras cannot be used outside because the IR-light in the sun-light interferes with the IR-light emitted by the depth camera. Additionally, depth data, compared to color images, only reveals little information about the configuration of rotational symmetric parts of the body such as arms and legs. We will address some of these challenges in Chapter 6.

Chapter 3

Cross-modal Comparison and Reconstruction of Motions

The analysis and synthesis of human motion data plays an important role in various application fields ranging from computer animation (see *e. g.* Dontcheva *et al.* [2003] or Lee *et al.* [2002]) to sports sciences (see *e. g.* Boissy *et al.* [2007]) and medicine (see *e. g.* Liu *et al.* [2009]). For example, in movie animations, one key objective is to create naturally looking motion sequences (Arikan *et al.* [2003]). Here, a standard procedure is to use prerecorded human 3D motion capture data to animate virtual characters (see *e. g.* Chai and Hodgins [2005], Pullen and Bregler [2002], or Shiratori and Hodgins [2008]). In online scenarios, such as computer games, low-dimensional control signals are often used to generate a wide range of task-specific high-quality motion sequences (see *e. g.* Lee *et al.* [2002] or Shiratori and Hodgins [2008]). In medical care and rehabilitation scenarios, motion capturing techniques are employed for monitoring patients and for detecting abnormal motion patterns (Boissy *et al.* [2007]). In sport sciences, motion data is recorded and analyzed in order to better understand and optimize the motions performed by athletes (Liu *et al.* [2009]). In all of these application fields, the comparison of human motion sequences is of fundamental importance. Here, the notion of similarity used in the comparison does not only depend on the respective motion representation but also on the specific application (Müller [2007]). For example, in a rehabilitation scenario, one may be interested in only comparing selected parts of the human body with previously recorded motions of the same patient in order to measure the progress over the period of treatment (Boissy *et al.* [2007]). This may require a rather strict notion of similarity. In other applications such as data-driven computer animation, one objective is to retrieve full-body motions from a motion database allowing spatial and temporal variations in the comparison, which requires rather coarse notions of similarity (see *e. g.* Kovar and Gleicher [2004] or Müller *et al.* [2005]). Finally, the comparison of motion data obtained from different sensor modalities has gained in importance in applications such as data-driven computer animation (see *e. g.* Slyper and Hodgins [2008], Wang and Popovic [2009], or Tautges *et al.* [2011]).

In Tautges *et al.* [2011] a real-time animation system is described, which allows for presenting high-quality mocap sequences that were reconstructed from motions of a given database using low-cost accelerometers as input devices. Here, the central component is a cross-modal matching of continuously generated accelerometer readings against accelerations computed from existing mocap data. Using four 3-axis accelerometers fixed at the hands and the feet, the authors report on

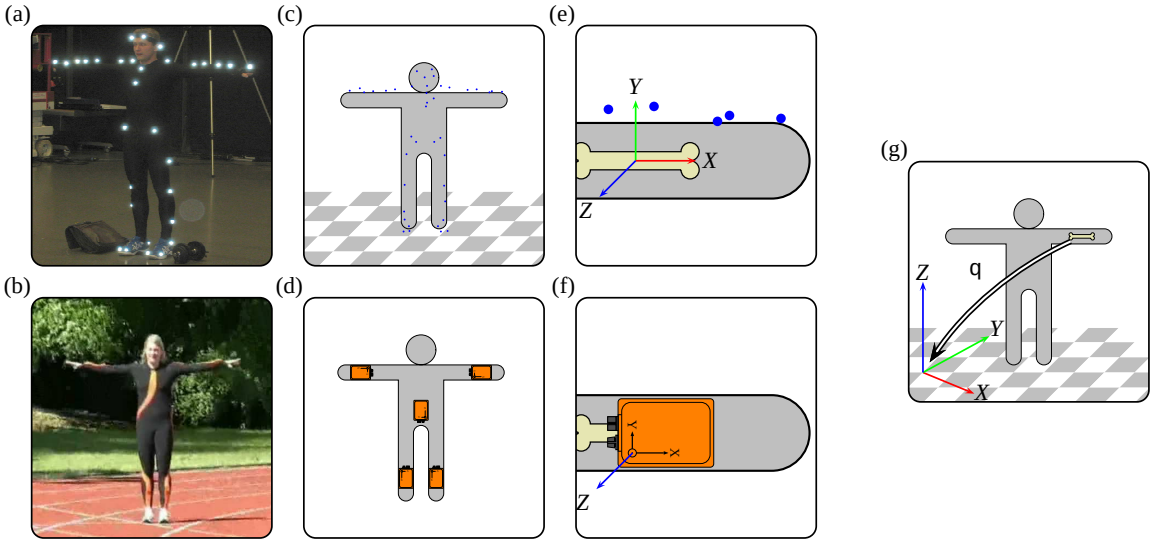

Figure 3.1. (a): Actor wearing a suit with 41 retro-reflective markers as used by an optical mocap system. **(b):** Actress wearing a suit with 5 Xsens MTx Sensors. **(c):** Positions of 41 markers provided by the optical system. **(d):** Locations of the sensors. **(e):** Limbs' positions and orientations defined by the positions of markers. **(f):** Inertial sensors measuring the orientation of the limb they are attached to. **(g):** Limb orientation expressed with respect to a global coordinate system.

promising reconstruction results. In the matching step, the authors use a mid-level representation based on accelerometer data. To enhance the descriptiveness of the mid-level representation and to reduce false positives, a so-called *lazy neighborhood graph* is employed. This run-time efficient data structure compares motions based on a time window to filter out acceleration trajectories that are not supported by motions in the database.

Contributions. In this chapter, we address the issue of cross-modal motion comparison while investigating the expressiveness of various motion representations in the context of general motion identification and retrieval scenarios. As one main contribution, we introduce various mid-level feature representations that facilitate cross-modal comparison of various motion types. Here, the main challenge consists of finding a good trade-off between robustness and expressiveness: on the one hand, a mid-level representation has to be robustly deducible from the data outputted by different mocap systems; on the other hand, the representation has to contain enough information to found the basis for discriminating motions within a certain application task. In particular, we show that certain low-dimensional orientation-based motion features are suited for accurately retrieving high-dimensional motion data as obtained from optical motion capturing. As a further main contribution of this chapter, we introduce a general framework for expressing separation and classification capability of different types of motion representations. These contributions have been published in Helten *et al.* [2011b]. For this reason, this chapter follows closely the explanations in Helten *et al.* [2011b], while adding some additional information.

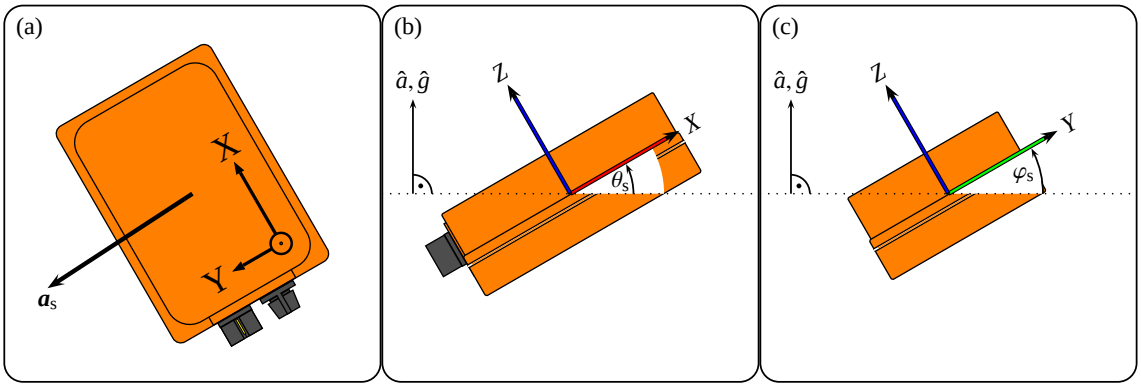

Figure 3.2. Illustration of the different feature values. **(a):** Measured acceleration $\boldsymbol{a}_s$ with respect to the sensors local coordinate system. **(b):** Pitch θ_s of a sensor with respect to the plane defined by $\hat{a}$ respectively $\hat{g}$. **(c):** Roll φ_s of a sensor with respect to the plane defined by $\hat{a}$ respectively $\hat{g}$.

Organization. The remainder of the chapter is organized as follows. In Section 3.2, we present our general framework for evaluating the discriminative power of feature representations. Then, Section 3.1, we introduce various mid-level feature representations that can be derived from the different sensor modalities. Our experiments using this framework are described in Section 3.3, where a special focus is put onto the investigation how the various feature representations behave under motion variations such as changes in the execution speed. In Section 3.4, extending this evaluation, we study the performance of different mid-level representations in the context of cross-modal motion retrieval. In Section 3.5, we present the application of cross-model motion comparison in the context of motion reconstruction. To this end, we give an introduction into the approach presented by Tautges *et al.* [2011] and explain how their approach employs techniques presented in this chapter. Finally, in Section 3.6 we conclude with an outlook on future work.

3.1 Features

In order to compare human motion data across different sensor modalities, one needs common mid-level representations that can be generated from the data outputted by the different sensors. On the one hand, such mid-level representations should be robustly computable from all modalities, and, on the other hand, they should contain sufficient information to realize the intended application. In the context of this chapter, our goal is to retrieve full-body motions from a database. The motion data inside the database was captured utilizing an optical, marker-based mocap system with 41 markers. The the query is given in form of a motion clip captured by five inertial sensors $s_1, \ldots, s_5$ that are placed at the hip next to the spine (s_1), both lower arms (left s_2, right s_3), and both lower legs (left s_4, right s_5), see Figure 3.1(a)–(d). Since all information supplied by the five inertial sensors can be simulated using the 41 marker position (as shown in Section 2.2.5), we use features close to the inertial data as common mid-level representation. Figure 3.1(e)–(g) shows an example of a common mid-level representation, where the direction of a limb is computed using both optical and inertial data. In the following subsections, we introduce three different feature representations based on local accelerations and directional information based on local and global coordinate systems.

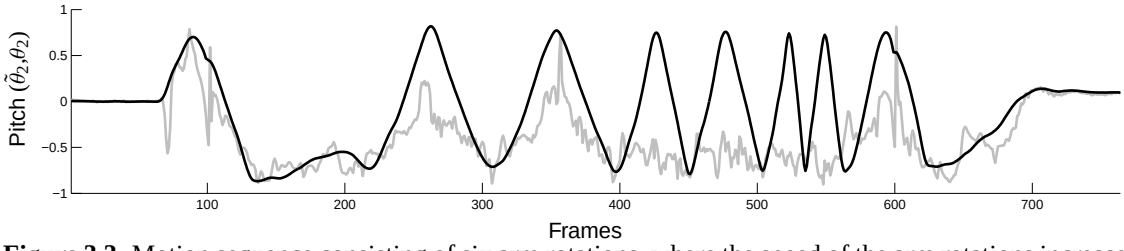

Figure 3.3. Motion sequence consisting of six arm rotations, where the speed of the arm rotations increases with each repetition. The pitch of the left forearm is shown, calculated by using $\tilde{\theta}_2$ (gray) and θ_2 (black).

3.1.1 Local Acclerations

As a first simple feature representation, we directly use the local accelerations as outputted by the accelerometers. We refer to Section 2.2 for an introduction into inertial sensors and the data they provide. Using five inertial sensor units $s_1, \ldots, s_5$, this results in five local accelerations $\boldsymbol{a}_s \in \mathbb{R}^3$ for $s \in [1:5]$. We then simply stack these five acceleration vectors to form a single vector

$$\boldsymbol{v}_{\boldsymbol{a}} = (\boldsymbol{a}_1^T, \ldots, \boldsymbol{a}_5^T)^T / C_{\boldsymbol{a}} \in \mathcal{F}_{\boldsymbol{a}}. \tag{3.1}$$

Here, $\mathcal{F}_{\boldsymbol{a}} := \mathbb{R}^{15}$ denotes the resulting feature space and $C_{\boldsymbol{a}}$ a constant used for normalization. In our experiments, $C_{\boldsymbol{a}} = 20$ turned out to be a suitable value. This normalization serves to make the distances functions as introduced in Section 3.2.1 comparable across the various features representations. Even though it is straightforward to derive local accelerations from inertial as well as from marker-based mocap data, this feature representation is not only prone to noise but also sensitive to motion variations as occurring when motions are performed by different actors. In particular, accelerations crucially depend on local and global differences in the speed a motion is executed, as will be discussed in Section 3.3.

3.1.2 Directions Relative to Acceleration

We now introduce a more robust motion representation which measures directions rather than magnitudes. To this end, we define a global up-direction using the direction of the gravity vector $\boldsymbol{g}$. By doing so, We are able to define a two degrees of freedom orientation of the sensor's local coordinate system relative to this global up-direction. Inspired by aviation, we call these two parameters *pitch* θ_s and *roll* φ_s, see Figure 3.2.

Recall from Section 2.2 that each measured acceleration is a superposition $\boldsymbol{a}_s = \overline{\mathsf{q}_s}[\boldsymbol{m}_s + \boldsymbol{g}]$ consisting of a component $\boldsymbol{m}_s$ that corresponds to the acceleration due to the movement of sensor s and a component $\boldsymbol{g}$ that corresponds to the gravity (which is independent of the respective sensor). Here q_s, $s \in [1:5]$, are the orientations of the five inertial sensors. In other words, an accelerometer always measures the acceleration caused by gravity, which is overlayed by the actual acceleration caused by the motion. If the sensor does not move ($\boldsymbol{m}_s = 0$) the measured acceleration $\boldsymbol{a}_s$ is equal to the gravity vector $\overline{\mathsf{q}_s}[\boldsymbol{g}]$.

We can use this fact to calculate an approximation of the sensor's pitch and roll using the direction of $\boldsymbol{a}_s$ as approximation for the global up-direction. The smaller the acceleration $\boldsymbol{m}_s$ is the more accurate this approximation becomes. These approximations denoted by $\tilde{\theta}_s$ and $\tilde{\varphi}_s$, are defined as

follows:

$$\hat{a}_s = \frac{\boldsymbol{a}_s}{\|\boldsymbol{a}_s\|}, \tag{3.2}$$

$$\tilde{\theta}_s = 1 - \frac{2}{\pi} \arccos\left\langle \hat{a}_s, (1,0,0)^T \right\rangle, \tag{3.3}$$

$$\tilde{\varphi}_s = 1 - \frac{2}{\pi} \arccos\left\langle \hat{a}_s, (0,1,0)^T \right\rangle. \tag{3.4}$$

Here, note that if the sensor's local Y-axis is perpendicular to the global up-direction, the pitch is determined by the rotation around the Y-axis. The resulting angle can be approximated by using an inner product between the X-axis and $\hat{a}_s$ approximating the up-direction, see Figure 3.2 (b). Similarly, the roll can be derived from the inner product between the Y-axis and the upward direction, see Figure 3.2 (c). In our definition, the resulting pitch and roll features, which we also refer to as *acceleration-based directional features*, are normalized to lie in the range between -1 and 1. Again, we stack these features for all five sensors $s_1, \ldots, s_5$ to form a single vector

$$\boldsymbol{v}_{\hat{a}} = (\tilde{\theta}_1, \tilde{\varphi}_1, \ldots, \tilde{\theta}_5, \tilde{\varphi}_5)^T \in \mathcal{F}_{\hat{a}}, \tag{3.5}$$

where $\mathcal{F}_{\hat{a}} := \mathbb{R}^{10}$ denotes the resulting feature space. Similar features are widely used in commercial products, as for example smartphones or game consoles. As noted before, such features are meaningful as long as the motion's acceleration component $\boldsymbol{m}_s$ is small. However, this assumption does not hold for dynamic motions, which exhibit significant accelerations that in many cases reach or even exceed the magnitude of gravity. In such cases, the measured acceleration $\boldsymbol{a}_s$ may significantly deviate from $\boldsymbol{g}$, which leads to corrupted pitch and roll values during dynamic motions, see Figure 3.3.

3.1.3 Directions Relative to Gravity

To address the above mentioned problem, one needs to approximate the global upward direction in a more robust way—in particular during dynamic phases, where $\boldsymbol{m}_s$ is not negligible. To achieve such an estimation, simple accelerometers do not suffice. We therefore use an inertial measurement unit (IMU) that outputs not only the local accelerations but also the sensor's orientation with respect to global coordinate system, see Section 2.2.1 and Section 2.2.2. Then, the direction $\hat{g}$ can be estimated by transforming the direction of the global Z-axis by means of the sensor's orientation $\mathfrak{q}_s$. More precisely, we define

$$\hat{g}_s = \overline{\mathfrak{q}_s}\left[(0,0,1)^T\right], \tag{3.6}$$

$$\theta_s = 1 - \frac{2}{\pi} \arccos\left\langle \hat{g}_s, (1,0,0)^T \right\rangle, \tag{3.7}$$

$$\varphi_s = 1 - \frac{2}{\pi} \arccos\left\langle \hat{g}_s, (0,1,0)^T \right\rangle. \tag{3.8}$$

Now, the values θ_s and φ_s exactly define (up to measurement errors of the IMU) pitch and roll as introduced in Section 3.1.2. The improvements in the case of highly dynamic motions are illustrated by Figure 3.3, which shows the values of $\tilde{\theta}_2$ and θ_2 over a motion sequence containing six arm rotations between (frames 210 and 575). Here, the arm rotations are performed at increasing speed, where the last rotation is performed almost three times faster than the first one. While θ_2

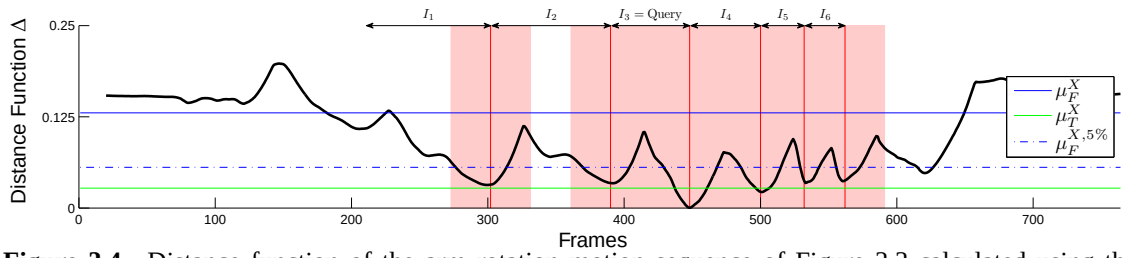

Figure 3.4. Distance function of the arm rotation motion sequence of Figure 3.3 calculated using the feature representation $\boldsymbol{v}_{\hat{g}}$. Indices corresponding to the six true matches are indicated by the six vertical red lines. The false alarm region consists of all indices outside the neighborhoods indicated by light red. The corresponding quality measures are indicated by horizontal lines.

clearly shows the periodic fluctuation of the pitch during the rotation, $\tilde{\theta}_2$ fails to display any meaningful information when the motion becomes faster. As before, we stack the pitch and roll features for all five sensors $s_1, \ldots, s_5$ to form a single vector

$$\boldsymbol{v}_{\hat{g}} = (\theta_1, \varphi_1, \ldots, \theta_5, \varphi_5)^T \in \mathcal{F}_{\hat{g}}, \tag{3.9}$$

where $\mathcal{F}_{\hat{g}} := \mathbb{R}^{10}$. The components are also referred to as *gravity-based directional features*.

The sensors we used here to determine the robust global upward direction provide the orientation q_s with all three *degrees of freedom* (DoF). But, since were are only transforming one direction (the global upwards direction) to the sensor's local coordinate system, we actually only need two DoF of the orientation q_s. Hence, one can also use combinations of inertial sensors which only consist of an accelerometer and a rate gyro, see also Luinge and Veltink [2005].

3.2 Evaluation Framework

In this section, we introduce a framework which is used to analyze the discriminative power of a given feature representation. A similar framework was used in Müller and Ewert [2009] for comparing audio representations. Let Q be a query motion clip and let D be a document of a database collection. The goal is to identify every sub-sequence of D which is similar to Q. Figure 3.3 shows an example were the document contains a motion sequence of roughly 15 seconds length captured at 50 Hz. The sequence contains six instances $(I_1, \ldots, I_6)$ of arm rotations of both arms, rotated in forward direction, beginning at frame 210 and ending at frame 575. The speed of the arm rotations increases over time. This example sequence is also used in Figure 3.4 and in Figure 3.5 (top). Considering I_3 as query, the task is now to identify the other arm rotations within the sequence.

3.2.1 Distance Function

The first step for the retrieval of those instances is the transformation of the query Q and the document D to suitable feature sequences $X = (X(1), \ldots, X(K))$ with $X(k) \in \mathcal{F}$ for $k \in [1:K]$ and $(Y(1), \ldots, Y(L))$ with $Y(\ell) \in \mathcal{F}$ for $\ell \in [1:L]$, respectively. Here $\mathcal{F}$ denotes the underlying feature space. For instance, if we consider the feature representation $\boldsymbol{v}_{\boldsymbol{a}}$, one has $\mathcal{F} = \mathcal{F}_{\boldsymbol{a}} = \mathbb{R}^{15}$. Furthermore, we define a *cost measure* $c : \mathcal{F} \times \mathcal{F} \to \mathbb{R}$. In the following, we simply use the L^2 distance as cost measure for the proposed feature representations. This is useful since our

features are normalized—so the L^2 is not mislead by strong outliers in the data—but other features representations may require other suited cost measures.

In order to identify a sequence X as sub-sequence inside Y we use *dynamic time warping* (DTW) to define a distance function Δ by

$$\Delta(\ell) := \frac{1}{K} \min_{a \in [1:\ell]} (\mathrm{DTW}(X, Y(a:\ell))) . \tag{3.10}$$

Here, $Y(a : \ell)$ denotes the subsequence of Y starting at frame a and ending at frame $\ell \in [1 : L]$. Furthermore, $\mathrm{DTW}(X, Y(a:\ell))$ denotes the DTW distance with respect to the cost measure c (see Müller [2007] for details). To avoid degenerations in the DTW alignment we use the modified step size condition with step sizes $(3, 1)$, $(1, 3)$, $(2, 1)$, $(1, 2)$, and $(1, 1)$ (instead of the classical step sizes $(1, 0)$, $(0, 1)$, and $(1, 1)$).

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 (3.10)) and ending at frame ℓ is similar to X. To determine the best match between Q and D, one can simply select the index $\ell_0 \in [1 : L]$ minimizing Δ. Then the best match is the motion sequence corresponding to the feature subsequence $(Y(a_{\ell_0}), \ldots, Y(\ell_0))$. The value $\Delta(\ell_0)$ is also referred to as the *cost* of the match. To look for the second best match, we exclude a *neighborhood* around the index ℓ_0 from further consideration to avoid large overlaps with the best match. In our case we exclude half the query length to the left and to the right by setting the corresponding values of the distance function Δ to ∞. To find subsequent matches, the above procedure is repeated until a certain number of hits have been retrieved or the costs of the matches are larger than a given threshold. Note that the retrieved matches can be naturally ranked according to their costs.

3.2.2 Quality Measures

In the context of motion retrieval and classification, the following two properties of Δ are of crucial importance. Firstly, the semantically correct matches (in the following referred to as the *true matches*) should correspond to local minima of Δ close to zero thus avoiding false negatives. Similar to Müller and Ewert [2009], we capture this property by defining μ_T^X to be the average of Δ over all indices that correspond to the local minima of the true matches for a given query X. Secondly, Δ should be well above zero outside a neighborhood of the desired local minima thus avoiding false positives. Recall from Section 3.2.1 that we use half the query length to the left and to the right to define such a neighborhood. The region outside these neighborhoods is referred to as *false alarm region*. We then define μ_F^X to be the average of Δ over all indices within the false alarm region. For our example shown in Figure 3.4, these values are indicated by suitable horizontal lines. In order to separate the true matches from spurious matches, it is clear that μ_T^X should be small whereas μ_F^X should be large. We express these two properties within a single number by defining the quotient $\alpha^X := \mu_T^X / \mu_F^X$. In view of a good separability, α^X should be close to zero. The quality measure α^X is rather soft, since unrelated regions with very large Δ-values may result in a small α^X-value. We therefore introduce a stricter quality measure by considering only the smallest Δ-values in the false alarm region. To this end we define the quantity $\mu_F^{X,5\%}$ which represents only the mean of all those values of Δ within the false alarm region which are smaller than the 5%

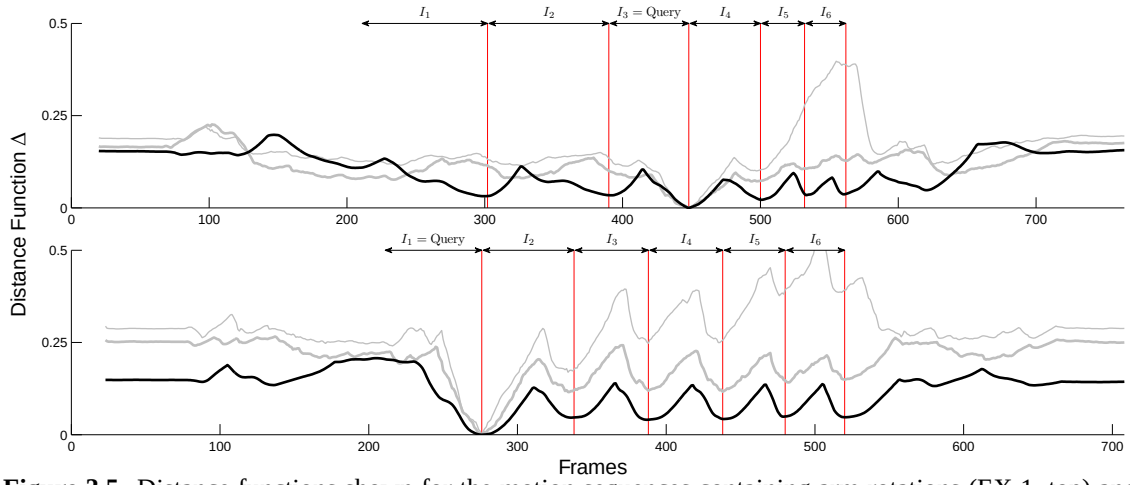

Figure 3.5. Distance functions shown for the motion sequences containing arm rotations (EX 1, top) and jumping jacks (EX 2, bottom) which are performed at increasing speed. The following feature representations were used: $\boldsymbol{v_a}$ (thin gray), $\boldsymbol{v}_{\hat{a}}$ (thick gray), and $\boldsymbol{v}_{\hat{g}}$ (thick black).

quantile of this region. The corresponding measure is referred to as $\beta^X := \mu_T^X / \mu_F^{X,5\%}$, which is stricter than α^X.

3.3 Feature Evaluation

In order to evaluate the presented feature representations, we use captured motion sequences using five Xsens MTx sensors. The sensors were placed on the lower arms, the lower legs and the hip of the body (see Figure 3.1). The relative orientations of the attached sensors with respect to the limbs were chosen such that the local X-axis of a sensor is parallel to the bone of the corresponding limb. As a consequence, pitch and roll of the sensor can be directly related to the pitch and roll of the corresponding limb.

3.3.1 Speed Dependence

In a fist experiment, we continue with our arm rotation example. Based on the quantitative measures α^X and β^X, we now study the discrimination capability of various feature representations. Using the same instance I_3 of the arm rotations as query Q as in Figure 3.4, all instances $I_1, \ldots, I_6$ are considered as true matches. The corresponding distance functions for all three feature representation introduced in Section 3.1 are shown in Figure 3.5 (top). It can be seen that only the distance function of the feature representation $\boldsymbol{v}_{\hat{g}}$ (thick black) has distinct local minima at every location of the true matches that are indicated by the vertical red lines. Besides the instance I_3 (which served as query), the other distance functions only show a local minimum at the end of instance I_4. This can be explained as follows. While instance I_4 was performed at almost the same speed as the query instance I_3, the other instances were performed at considerable different speeds. The instances I_1 and I_2 were performed slower and the instances I_5 and I_6 were performed faster. Although, the DTW based distance measure is able to compensate for the length differences imposed by performance variations, there are other variations among the motion instances beside

EX 1	μ_T^X	μ_F^X	α^X	μ_T^X	$\mu_F^{X,5\%}$	β^X
$\boldsymbol{v_a}$	0.173	0.161	1.071	0.173	0.119	1.452
$\boldsymbol{v}_{\hat{a}}$	0.087	0.135	0.647	0.087	0.080	1.096
$\boldsymbol{v}_{\hat{g}}$	0.027	0.131	0.209	0.027	0.056	0.489
EX 2	μ_T^X	μ_F^X	α^X	μ_T^X	$\mu_F^{X,5\%}$	β^X
$\boldsymbol{v_a}$	0.245	0.276	0.887	0.245	0.246	0.992
$\boldsymbol{v}_{\hat{a}}$	0.109	0.237	0.459	0.109	0.202	0.538
$\boldsymbol{v}_{\hat{g}}$	0.038	0.158	0.238	0.038	0.134	0.282

Table 3.1. Values of the quality measures for the different feature representations. The values belong to the two experiments described in Section 3.3.1.

simple length differences. Here, the large warping costs between the query and the database instances do not stem from temporal deformations but disagreeing feature values. In other words, speed variations do not only impose length differences but also varying feature values. We want to stress that this effect largely depends on the used feature set.

Such behavior can be explained by recalling the way the feature representations have been computed. The feature representations $\boldsymbol{v_a}$ and $\boldsymbol{v}_{\hat{a}}$ make use of the locally measured sensor acceleration $\boldsymbol{a}_{\mathrm{s}}$. As said before, the measured acceleration $\boldsymbol{a}_{\mathrm{s}}$ is a superposition of the acceleration due to movement $\boldsymbol{m}_{\mathrm{s}}$ and the acceleration due to gravity $\boldsymbol{g}_{\mathrm{s}}$. While $\boldsymbol{g}$ is always constant, $\boldsymbol{m}_{\mathrm{s}}$ largely depends on the execution speed of the motion. Let us consider two instances of the same motion performed at different speeds. If the motion is performed with doubled speed the acceleration due to movement will be four times larger. As a consequence the value of $\boldsymbol{a}_{\mathrm{s}}$ largely dependents on the speed of the motion and so the feature representations $\boldsymbol{v_a}$ and $\boldsymbol{v}_{\hat{a}}$ do as well. In contrast, $\boldsymbol{v}_{\hat{g}}$ does not make use of the measured acceleration $\boldsymbol{a}_{\mathrm{s}}$ and is therefor not affected by the variations of the performance speed.

Another example illustrating this effect is shown in Figure 3.5 (bottom), where six jumping jacks (frames 210–510) were performed with increasing speed. Here, the first repetition I_1 was taken as query. Although all three distance function clearly exhibit local minima at all six true match positions, the distance function with respect to the feature representation $\boldsymbol{v_a}$ rises continuously during the performance of the jumping, resulting in very high values at the true matching positions compared to the regions of the distance function where no jumping jacks were performed. This is also indicated by the values of the quality measures shown in Table 3.1, where EX 1 is the first experiment with arm rotations and EX 2 is the second experiment with jumping jacks. In case of EX 1 the value of β^X is 1.452 when using the feature representation $\boldsymbol{v_a}$ and 0.489 when using $\boldsymbol{v}_{\hat{g}}$. In case of EX 2, while all feature representations perform better, this relative improvement stays the same. To conclude, in both cases the feature representation $\boldsymbol{v}_{\hat{g}}$ outperforms the other two representations due to its immunity to the effects imposed by $\boldsymbol{m}_{\mathrm{s}}$.

3.3.2 Discriminative Power

In the following, we want to take a closer look on how the different feature representations perform for the task of discriminating different motion classes. To this end, we set up a database consisting

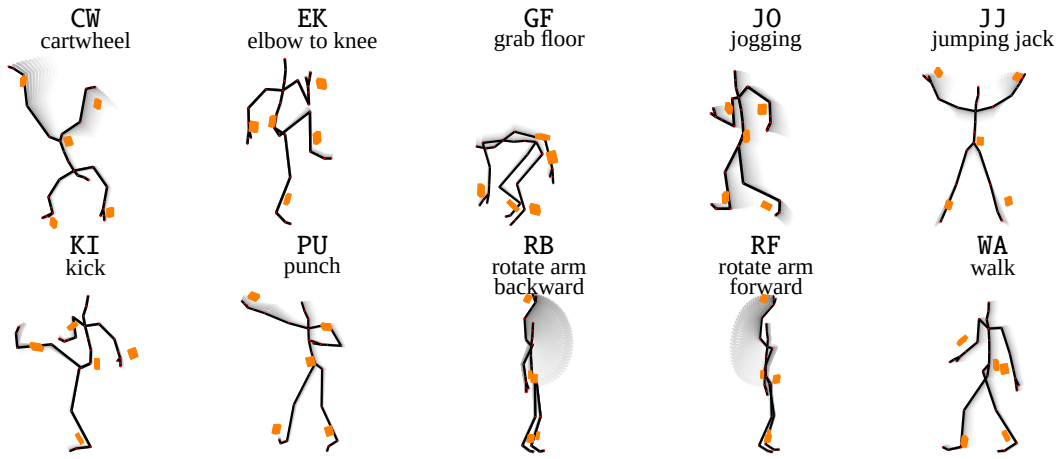

Figure 3.6. Motion classes used for the experiments in Section 3.3.2 and Section 3.4.

of ten motion classes with ten instances each. The motion classes used in the database are shown in Figure 3.6. The motions are performed by three different actors in different styles and speeds and recorded using five Xsens MTx devices. The resulting database consisting of 100 motion documents is denoted as $\mathrm{DB}_{\mathrm{xse}}$. For the evaluation of the discriminative power of the feature representations, we use sub-sequence retrieval instead of document based retrieval. In document based retrieval, the database consists of a set of pre-segmented motion documents. During the retrieval the query motion is compared to each of the motion documents in a global manner. The most similar motion documents are considered as the hits for a given query. Since such pre-segmentation is unlikely to occur in practical retrieval scenarios, we evaluate the performance of the proposed feature representations in a sub-sequence retrieval scenario. Here, a short query motion is located as a sub-sequence within one large continuous database document. To this end, we concatenate all 100 motion documents of the database $\mathrm{DB}_{\mathrm{xse}}$ to form one single database document D_{xse}. Concatenating the motions in the mentioned way leads to more confusion during the retrieval but it better resembles a realistic scenario.

We keep the knowledge which part of the document D_{xse} belongs to which of the original motion documents in a supplementary data structure. This knowledge is not used for retrieval but only for the automatic evaluation of retrieval results. Each of the previously mentioned 100 motion instances also serves as query to compute a total of 100 distance functions for every feature representation. For each of these distance functions the values of μ_T^X, μ_F^X, $\mu_F^{X,5\%}$, α^X and β^X are calculated. In order to get a quality measure for a given feature representation over a set of queries we average the values of μ_T^X, μ_F^X, $\mu_F^{X,5\%}$, α^X and β^X over all distance functions which were calculated using the same feature representation. We refer to the averaged quality measures as μ_T, μ_F, $\mu_F^{5\%}$, α and β. Table 3.2 (top) shows the results for this unimodal retrieval scenario. The rows contain the values of μ_T, μ_F, $\mu_F^{5\%}$, α and β for each of the feature representations $\boldsymbol{v_a}$, $\boldsymbol{v}_{\hat{a}}$ and $\boldsymbol{v}_{\hat{g}}$. It can bee seen that the feature representation $\boldsymbol{v}_{\hat{g}}$ ($\alpha = 0.429, \beta = 0.753$) outperforms the other two feature representations $\boldsymbol{v_a}$ ($\alpha = 0.537, \beta = 0.862$) and $\boldsymbol{v}_{\hat{a}}$ ($\alpha = 0.533, \beta = 0.839$). Compared to the two examples discussed in Section 3.3.1 the differences between the three feature representations are not that big. But here, the retrieval scenario is more complex since several motion class can be mixed up especially if they look similar under a given feature representation.

unimodal	μ_T	μ_F	α	μ_T	$\mu_F^{5\%}$	β
$\boldsymbol{v_a}$	0.132	0.234	0.537	0.132	0.160	0.862
$\boldsymbol{v}_{\hat{a}}$	0.120	0.222	0.533	0.120	0.150	0.839
$\boldsymbol{v}_{\hat{g}}$	0.088	0.205	0.429	0.088	0.125	0.753
cross-modal	μ_T	μ_F	α	μ_T	$\mu_F^{5\%}$	β
$\boldsymbol{v_a}$	0.194	0.233	0.822	0.194	0.166	1.275
$\boldsymbol{v}_{\hat{a}}$	0.160	0.211	0.752	0.160	0.151	1.093
$\boldsymbol{v}_{\hat{g}}$	0.129	0.206	0.618	0.129	0.135	0.963

Table 3.2. Averaged quality measures for the different feature representations belonging to **(top):** the unimodal scenario described in Section 3.3.2, and **(bottom):** then cross-modal scenario described in Section 3.4.

3.4 Cross-modal Comparison

In Section 3.3.2 we evaluated the discriminative power of the feature representations in an unimodal scenario where both the queries and the database document consisted of measured inertial motion data. In this section, we evaluate the feature representations in the context of a cross-modal scenario, where the queries and the database contain different data modalities. In particular, we want to search in a database which comprises of high-dimensional 3D mocap using low-dimensional inertial sensors as query input. In the following, we use the documents in the database $\mathrm{DB}_{\mathrm{xse}}$ as queries. The database we want to search in consists of motion excerpts from the HDM05 database described in Müller *et al.* [2007]. This database consists of high quality motions recorded by a 12 camera Vicon optical mocap system. Here, we use the C3D data containing the marker positions to compute the virtual inertial sensors, see Section 2.2.5. These virtual sensors enable us to calculate the inertial-based feature vectors as described in Section 3.1 for the position-based data of a optical mocap system. Analogously to the $\mathrm{DB}_{\mathrm{xse}}$ database, we use ten instances from the ten motions classes shown in Figure 3.6. This again sums up to a total of 100 motion documents denoted as $\mathrm{DB}_{\mathrm{c3d}}$. To create a realistic retrieval scenario where we do not want to assume a pre-segmentation of the motion data, we again concatenate all documents in $\mathrm{DB}_{\mathrm{c3d}}$ to form one large continuous database document $\mathrm{D}_{\mathrm{c3d}}$. The frames of this document are annotated by the corresponding class labels, which are used as ground truth in the evaluation below.

3.4.1 Quality Measures

To evaluate the discriminative power in a cross-modal scenario, we calculate the distance functions on the database $\mathrm{D}_{\mathrm{c3d}}$ for every query taken from $\mathrm{DB}_{\mathrm{xse}}$ as well as for every feature representation. Analogously to Section 3.3.2, Table 3.2 (bottom) shows the averaged quality measures for the cross-modal scenario. Here, the feature representation $\boldsymbol{v}_{\hat{g}}$ ($\alpha = 0.618$, $\beta = 0.963$) performs best again. Both acceleration based feature representations $\boldsymbol{v}_{\hat{a}}$ ($\alpha = 0.752$, $\beta = 1.093$) and $\boldsymbol{v_a}$ ($\alpha = 0.822$, $\beta = 1.272$) perform considerably worse. Compared to the unimodal scenario described in Section 3.3.2 both measures α and β are worse for all feature representations. One reason for this general degradation is the fact that the inertial data origins from two different sources (virtual and real sensors). Another reason is that actors performing the motions are almost disjunct for both data sources; only one actor participated in both recording sessions.

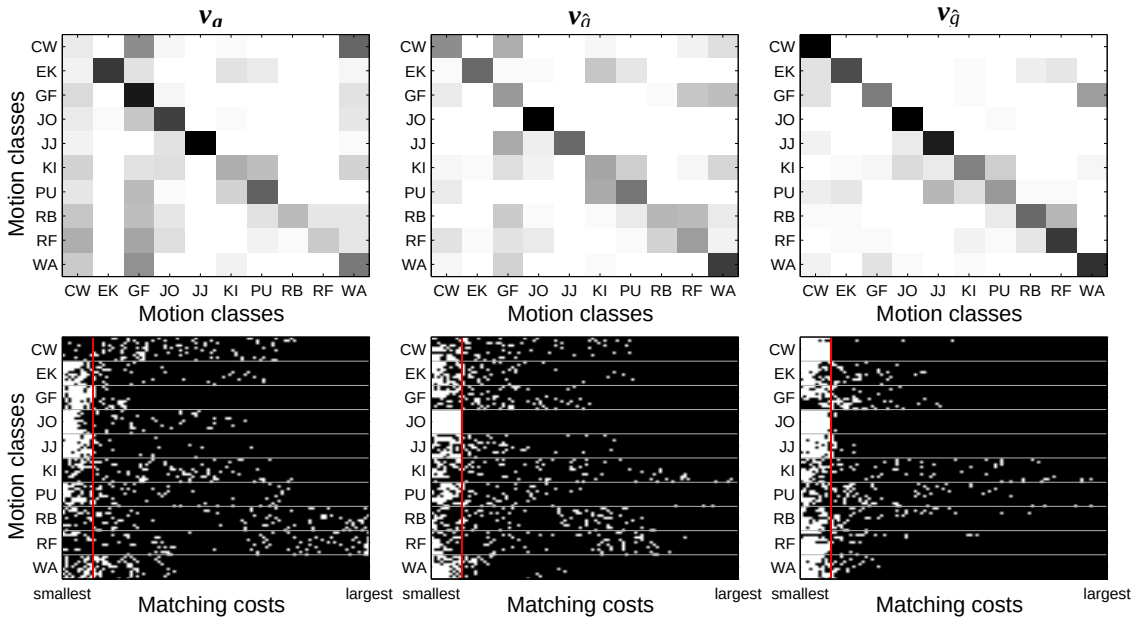

Figure 3.7. Confusion matrices **(top)** and true match distributions **(bottom)** of the three different feature representations.

3.4.2 Class Confusion

The above presented quality measures are well suited to compare the behavior of different feature representations in a quantitative manner. We now examine how motion classes are confused with regard to a given feature representation.

As described in Section 3.2.1, we obtain for each match a corresponding interval $[a_\ell : \ell]$ within the database. Counting the ground-truth class labels for the frames within $[a_\ell : \ell]$, we assign to the underlying match the class label with the largest count. When the class label of a match is equal to the motion class of the query, we call this a *true match* otherwise a *false match*. As there are ten instances of each motion class inside the database we get at most ten true matches. Since we can assign costs to each retrieved match based on the distance function Δ, we get a natural ranking of the retrieved matches. Considering the distribution of motion classes among the ten best matches—those with the lowest matching costs—one gets a good impression how the motion classes are mixed up under a given feature representation. A common means to visualize this are *confusion matrices*, which are shown for the three feature representations $\boldsymbol{v_a}$, $\boldsymbol{v_{\hat{a}}}$ and $\boldsymbol{v_{\hat{q}}}$ in Figure 3.7 (top row). The rows of a confusion matrix represent the motion classes of the query, whereas the columns represent the motion classes of the match. Dark entries indicate a large percentage of a motion class, whereas light colors indicate a low percentage. For example, the matrices show that most of the motion classes are confused with the motion class `CW` (first column) when using the feature representation $\boldsymbol{v_a}$. One reason is that most of the motion classes appear as short sub-sequences within the relatively long instances of the `CW` motion class, which are then confused when using a local, sub-sequence retrieval. Here, a global, document-based retrieval strategy may circumvent this problem, which, however, would require a suitable pre-segmentation. Another reason is that the motion class `CW` shows a lot of variance among the different motion instances even when performed by the same actor. In particular, the risk of confusion with the

	CW	EK	GF	JO	JJ	KI	PU	RB	RF	WA	Ø
$\boldsymbol{v_a}$	0.29	0.72	0.80	0.74	0.87	0.40	0.58	0.37	0.32	0.59	0.57
$\boldsymbol{v_{\hat{a}}}$	0.56	0.67	0.53	1.00	0.65	0.45	0.63	0.40	0.48	0.81	0.62
$\boldsymbol{v_{\hat{g}}}$	0.98	0.75	0.69	0.98	0.92	0.58	0.50	0.65	0.83	0.84	0.77

Table 3.3. Averaged maximal F-Measures for every feature representation and motion class. The last column shows for every feature representation the average over all motion classes.

motion class CW is high for short and dynamic motions classes such as KI, PU, RB, and RF. In contrast, using the directional feature representation $\boldsymbol{v_{\hat{g}}}$ the confusion is reduced significantly.

Another way of visualizing the matches retrieved for the queries of a given motion class is shown in Figure 3.7 (bottom column). These matrices visualize the distribution of the true matches among all retrieved motions. Every row of a matrix represents one query to the database. The columns indicate the rank of a match from least cost (at the left) to largest costs (at the right). Within one row the color indicates whether a match for a given query (row) and a given rank (column) is a true match (white) or a false match (black). The red line separates the ten matches with the highest ranks from the rest of the matches. This kind of visualization gives a good impression whether a given feature representation describes a given motion class well or not. For example, the motion class JO is well represented when using the feature vectors $\boldsymbol{v_{\hat{a}}}$ and $\boldsymbol{v_{\hat{g}}}$, whereas the motion class CW is only well represented using the feature vector $\boldsymbol{v_{\hat{g}}}$. Examples of motion classes which are poorly represented are RB and RF using the feature vector $\boldsymbol{v_a}$. This is due to the noise imposed by high velocity differences—and resulting acceleration differences—among the arm rotations (see also Section 3.3.1).

3.4.3 F-measure

To further quantify the retrieval results, we use another measure from the retrieval domain referred to as *maximum F-measure*. Let k, $k \in [1:K]$ be the rank of a given match, where K is the maximum rank (in our case $K = 100$). Now, for every k *precision* P_k and *recall* R_k are defined as $P_k := |T \cap M_k|/|M_k|$ and $R_k := |T \cap M_k|/|T|$. Here, M_k is the set of all matches up to rank k and T the set of all possible true matches (in our case $|T| = 10$). Combining precision and recall values for a given rank k yields the (standard) F-measure $F_k := 2 \cdot P_k \cdot R_k/(P_k + R_k)$. Now, the maximum F-measure is defined as $F := \max F_k, k \in [1:K]$. Table 3.3 shows the maximum F-measure for each motion class and every feature representation. The value was calculated by averaging the maximum F-measures over all queries of each motion class. Finally, the last column shows the average of all previous values over all motion classes. The better a given feature representation discriminates a motion class against all other motion classes the larger is the corresponding entry in the table. It can be seen that the feature representation $\boldsymbol{v_{\hat{a}}}$ is well suited to identify instances of motion class JO (1.00), whereas the feature representation $\boldsymbol{v_{\hat{g}}}$ performs particularly well for the motion classes CW (0.98), JO (0.98), and JJ (0.92). Furthermore, the identification of CW shows a drastic improvement under the feature representation $\boldsymbol{v_{\hat{g}}}$ (0.98) in comparison to $\boldsymbol{v_a}$ (0.29). Also, the arm rotations RB and RF perform much better under the feature representation $\boldsymbol{v_{\hat{g}}}$ (0.65 and 0.83) compared to the acceleration based feature representations $\boldsymbol{v_a}$ (0.40 and 0.48) and $\boldsymbol{v_{\hat{a}}}$ (0.37 and 0.32). Interestingly, there are some exceptions where $\boldsymbol{v_{\hat{g}}}$ does not outperform the other to feature representations, as with the motion classes GF and PU. For example, in case of motion

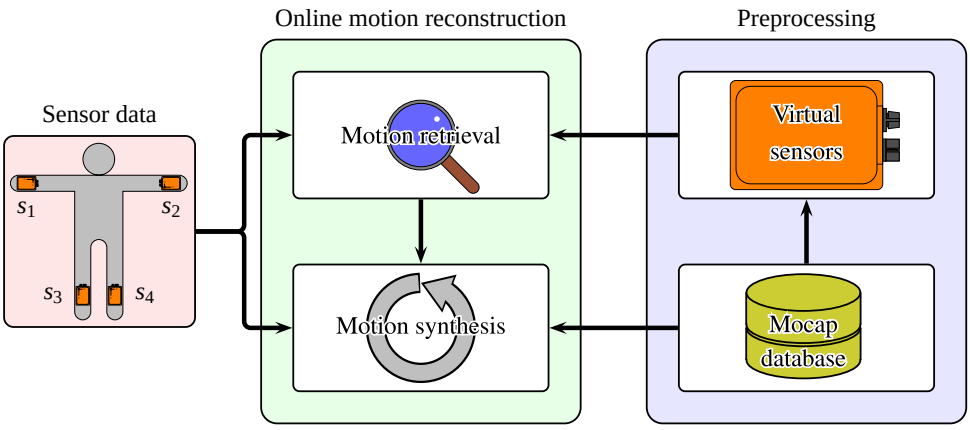

Figure 3.8. Overview of the motion reconstruction approach presented by Tautges *et al.* [2011].

class PU, $\boldsymbol{v}_{\hat{g}}$ (0.50) is worse compared to $\boldsymbol{v}_{\hat{a}}$ (0.63) and even $\boldsymbol{v}_{\boldsymbol{a}}$ (0.58). Here, on the one hand, the orientations of both arms—including roll and pitch—shows large variations among the actors, While, on the other hand, all punching motion exhibit characteristic peaks in the acceleration data. But, in general, again $\boldsymbol{v}_{\hat{g}}$ is much better suited to identify most motion classes than the feature representations $\boldsymbol{v}_{\boldsymbol{a}}$ and $\boldsymbol{v}_{\hat{a}}$.

3.5 Applications

in this section, we discuss an application of cross-modal motion retrieval. To this end, we take a look on the approach presented in Tautges *et al.* [2011], where the authors use techniques similar to the ones presented in this chapter to facilitate real-time full-body reconstruction of motions from sparse inertial input. In particular, they use the sensor data of four accelerometers $s_1, \ldots, s_4$, placed at the wrists and ankles of a person, to control the reconstruction. An overview of the employed framework can be seen in Figure 3.8.

In a preprocessing step, a database containing high-dimensional mocap data, which has been recorded using a traditional marker-based optical mocap system, is set up. As next step, the authors employ a virtual sensor concept similar to the one presented in Section 2.2.5 to simulate accelerometer readings of the four sensors mentioned above. These accelerations are then used to compute a mid-level representation, which consists of the stacked accelerations of the four sensors. Similar to Section 3.1.1 this a feature $\boldsymbol{v}_{\boldsymbol{a}}$ defined by

$$\boldsymbol{v}_{\boldsymbol{a}} := (\boldsymbol{a}_1^T, \ldots, \boldsymbol{a}_4^T)^T \in \mathbb{R}^{12}. \tag{3.11}$$

Now, during the online motion reconstruction, the readings from the four accelerometers serve as input for retrieving a motion from the database that is similar to the performed motion. The retrieved motion is then used as basis for a motion synthesis step, which combines the motion reconstructed so far, the retrieved motion and the sensor readings in a unifying optimization scheme. This scheme ensures temporal coherence, similarity to the retrieved motion and similarity of the accelerations induced by the reconstructed motion to the sensor readings.

Remember from the previous sections that the feature $\boldsymbol{v}_{\boldsymbol{a}}$ has performed worst compared to the other features $\boldsymbol{v}_{\hat{a}}$ or $\boldsymbol{v}_{\hat{g}}$. The authors in Tautges *et al.* [2011] use pure accelerations as features to

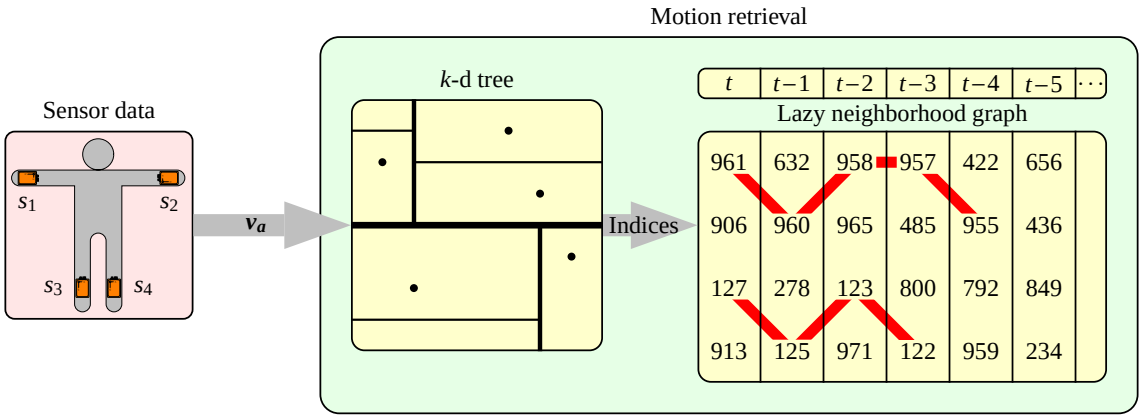

Figure 3.9. Schematic of the motion retrieval approach used by Tautges *et al.* [2011]. **(left):** The sensor readings a time t are converted into a mid-level representation $\boldsymbol{v_a}$. **(middle):** Using a k-d tree the K (here, $K = 4$) most similar frames in the database are identified. **(right):** The indices of those K frames are added to the lazy neighborhood graph. The indices of consecutive frames are connected by an edge (red) if their offset is 1 or 2. The longer the sequence the higher the probability that the matching motion sequence resembles the query motion.

show what kind of motion reconstruction accuracy is obtainable.

To compensate for low descriptive power of the feature $\boldsymbol{v_a}$ and to facilitate real-time motion reconstruction speed, they employ two key components: an online motion retrieval data structure for motion retrieval and a combined optimization scheme for motion synthesis.

Motion retrieval. One important difference of the approach by Tautges *et al.* [2011], compared to the techniques explained earlier in this chapter, is the search algorithm to identify a motion in the database. Because of the requirements of a real-time algorithm, the approach mentioned in Section 3.2.1 cannot be used since it is too slow when the database becomes larger. Furthermore, the approach introduced in Section 3.2.1 requires that the complete query is known. This might not be possible in on online reconstruction scenario, where the sensor readings that serve as query are obtained continuously.

For these reasons, the authors employ a different approach, where a k-d tree is used to index the features $\boldsymbol{v_a}$ for every frame in the database. Now every time a new sensor reading arrives, its feature representation is used to retrieve the K closest neighbors based on the k-d tree. Since $\boldsymbol{v_a}$ has little expressiveness the retrieved frames might stem from various motions that are semantically not similar to the motion to be reconstructed. Therefore, the authors use a so-called *lazy neighborhood graph* to filter out unwanted results. A similar approach has been presented in Krüger *et al.* [2010]. Here, the central idea is that if the database contains continuous motion segments, a continuous query stream should result in a stream of retrieved indices that stem from similar database locations.

The lazy-neighborhood graph keeps track of a history of the retrieved indices from the k-d tree. Depending on the offset between an index at frame t and an index at time $t - 1$, the two indices are connected with an edge. These possible offsets are similar to the step sizes as used in the DTW-based retrieval strategy described in Section 3.2.1. A sequence of connected indices ending at the current time defines a motion segment in the database. The authors now assume that the

longer the sequence the higher the probability that the motion represented by this sequence is semantically similar to the performed motion. The longest sequences serve then as basis for the motion synthesis. For further details on the lazy neighborhood graph we refer to Tautges [2012] and Krüger *et al.* [2010].

Motion synthesis. As said before, the motion synthesis step presented in Tautges *et al.* [2011] employs an optimization scheme that incorporates three priors. The first prior ensures that the synthesized motion results in a pose explainable by the motions obtained in the motion retrieval step. Here, not only the spatial properties, such as the joint positions are of interest but also the kinematic behavior represented by the velocities and accelerations of the joints. The second prior forces that the accelerations implied by the synthesized motion explain the accelerations measured by the four sensors placed at the extremities of the body. Since this prior implements direct control of the synthesis by the sensors measurements, it is referred to as control prior. Finally, the third prior induces that the noise of the sensors does not result in an unstable synthesized motion. To this end, it limits the possible accelerations between two consecutive frames. For further details, we refer to Tautges [2012].

Discussion. The presented approach by Tautges *et al.* [2011] shows how the concepts of cross-modal motion retrieval can be used to facilitate real-time motion reconstruction using sparse inertial sensors as input. However, there is still room for improvement. One possible direction of further research is to include a more stable and expressive mid-level representation as for example $\boldsymbol{v}_{\hat{a}}$ or $\boldsymbol{v}_{\hat{g}}$ instead of $\boldsymbol{v}_{\boldsymbol{a}}$. Also the inclusion of the information obtained by other sensors such as optical or depth sensors might be helpful. In Chapter 6, we will present a motion reconstruction approach that fuses information obtained from inertial sensors with information from a monocular depth sensor.

3.6 Conclusions

The analysis of human motions using various types of motion capture has become a major strand of research in areas such as sports, medicine, human computer interaction and computer animation. In particular, because of low cost and easy set-up, inertial-based mocap systems are becoming more and more popular, even though these sensors provide less expressive mocap data compared to optical systems. In this chapter, we have presented a systematic analysis of various feature representations that can be derived from customary inertial sensors. As one main result, we showed that directional features relating the sensor to the direction of gravity outperform purely acceleration-based features within various retrieval scenarios. In particular, it turns out that rate-of-turn data is necessary to enhance the roll and pitch estimates in the case of dynamic, fast changing motions. As further contribution, we introduced a general separation measure based on a local variant of dynamic time warping, which allows for assessing the discriminative power of different features representations. We demonstrated how our feature representations can be used within a cross-modal retrieval scenario, where inertial-based query motions are used to retrieve high-quality optical mocap data.

Because of the increasing relevance of motion sensors for monitoring and entertainment purposes,

the fusion of various sensor modalities as well as cross-domain motion analysis and synthesis will further gain in importance. We showed an example were the cross-modal comparison was used in the context of motion reconstruction. In particular, sparse accelerometer readings where used to identify high-quality 3D human motions in a database which was recorded using an optical mocap system. Such a reconstruction of high-quality 3D human motions using database knowledge has become a major principle used in computer animation and the gaming industry. Here, our analysis results and methods constitute a suitable foundation for estimating the performance of the various motion representations. We will also use and extend techniques presented in this chapter in Chapter 6, where we use orientations obtained from inertial sensors to identify motions in a database consisting of high-dimensional optical mocap data. These retrieved motions are then used in a combined depth/inertial tracking framework to robustly estimate human pose even in challenging scenarios such as when occlusions occur. Here, the specific advantages of different sensors modalities are combined, to obtain better results compared to using one sensor modality alone.

Chapter 4

Classification of Trampoline Jumps Using Inertial Sensors

In this chapter, we apply techniques from the previous chapters with the objective to automatically classify trampoline motion sequences. This constitutes a challenging application scenario because of the high complexity in terms of dynamics and recording volume. In trampolining, an athlete performs a routine that consists of a sequences of trampoline jumps that belong to predefined motion categories such as pike jump or a somersault. The classification problem then consists in automatically segmenting an unknown trampoline routine into its individual jumps and to classify these jumps according to the given motion categories. Here, further challenges arise from the fact that there is a wide spectrum on how a jump from a specific category may be actually performed by an athlete.

As introduced in Chapter 2, there exist many ways for recording human motion sequences, including optical, inertial and depth-based (mocap) systems. For recapitulation, optical motion capture systems, which are widely used in movie and game productions, provide very rich and easy to interpret data. On the downside, such systems impose strong restrictions concerning the size of the capture volume and lighting conditions. This makes them difficult to use in our trampolining scenario. The main disadvantage of depth sensor is the limited recording volume of a single sensor. Using multiple depth sensors, however, would increase the setup effort and the simultaneous use of depth sensors is not trivial, see Section 2.3.4. Avoiding such restriction, inertial-based sensors have become a low-cost alternative, which is increasingly used in entertainment, monitoring and sports applications Boissy *et al.* [2007]; Harding *et al.* [2008]; Ohgi *et al.* [2002]; Sabatini *et al.* [2005]. The drawback of such systems is that the provided data—accelerations and angular velocities—are difficult to handle and prone to noise. Here, additional sensor information has been used to derive more robust global orientation data Kemp *et al.* [1998].

Contributions. We introduce a motion classification pipeline for automatically classifying trampoline routines based on inertial sensor input, see Figure 4.1 for an overview. As one contribution, we discuss how to transform the inertial raw data into meaningful and robust feature representations underlying our classification scheme. As for the predefined motion categories, we use suitable training data to learn class representations that described the characteristics of a specific

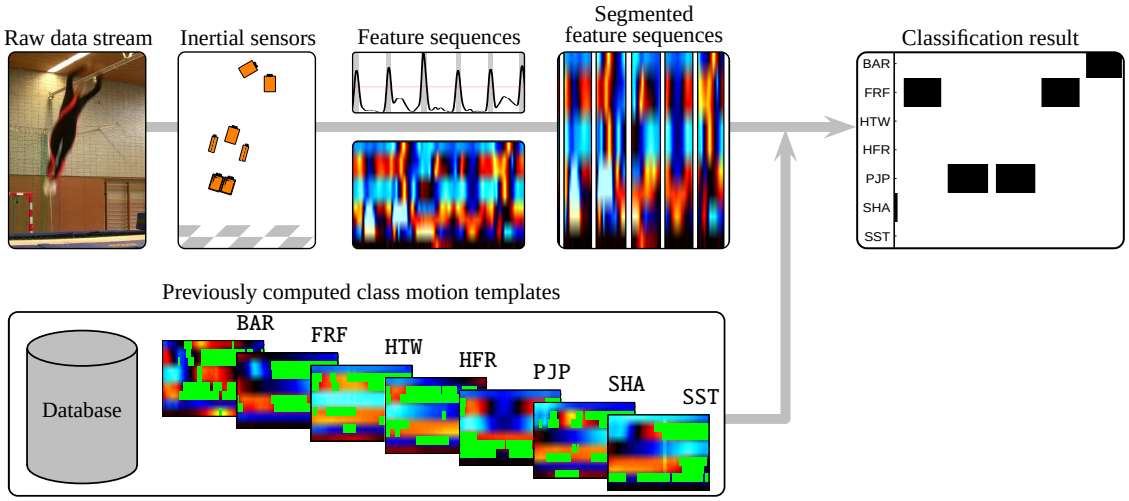

Figure 4.1. Classification pipeline used in this chapter. **(bottom):** Class representations are computed for each of the motion categories in a preprocessing step. **(top):** An unknown trampoline routine is converted into a feature sequences which is then segmented into single jump. Finally the segmented jumps are compared to the class templates and labeled with the name of the most similar class.

trampoline jump. Here, as a further contribution, we extend the concept of boolean motion templates Müller and Röder [2006] to the real-valued case. In particular, we introduce the notion of variance templates that allow for blending out performance variations to be left unconsidered in the classification stage. In our classification pipeline, an athlete, being equipped with a small number of inertial sensors, performs a trampolining routine. The resulting motion stream is first segmented into individual jumps, which are then classified by comparing the segments with the previously learned class representations using a suitable similarity measure. To prove the practicability of our approach, we have recorded trampoline motions consisting of 750 individual jumps that comprise 13 different classes performed by four different athletes. We report on various experiments which show that our procedure yields a high classification accuracy even in the presence of significant style variations across the different athletes. This chapter closely follows Helten *et al.* [2011a], where the concepts presented here have been published.

Organization. The remainder of this chapter is organized as follows. We start by discussing some basics on trampolining (Section 4.1) as well as what kind of sensor data we use (Section 4.2). Then, we describe our segmentation procedure (Section 4.3), discuss various feature representations (Section 4.4), and introduce the class representations in form of real-valued motion templates (Section 4.5). Subsequently, the actual classification procedure is described and evaluated demonstrating the practicability of our approach (Section 4.6). Finally, we close this chapter with an outlook on future work (Section 4.7).

4.1 Trampoline Motions

In this section, we describe some characteristics of trampoline motions, which can be exploited for segmentation and classification tasks. Trampolining is closely related to gymnastics where

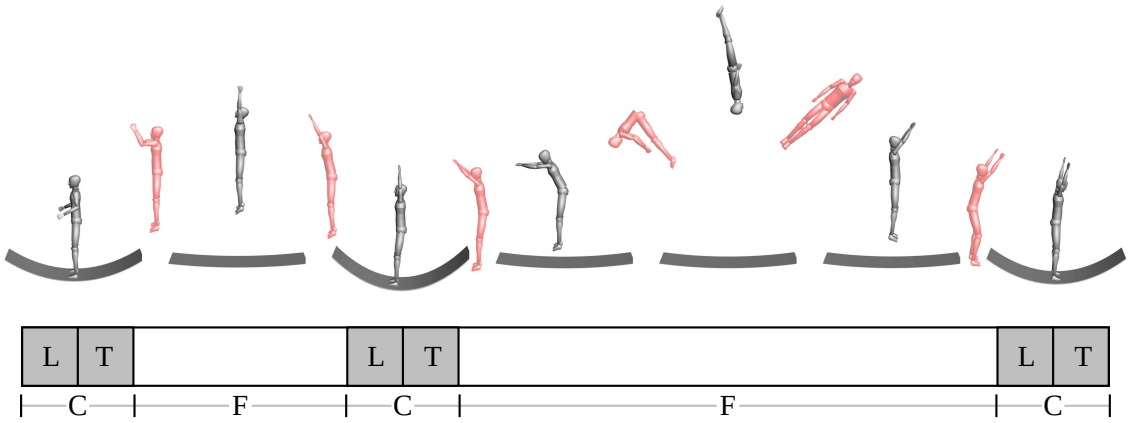

Figure 4.2. Phases of a trampoline jump comprising a contact phase (C), a landing phase (L), a takeoff phase (T), and flight phase (F).

athletes perform a sequence of acrobatic moves. During a trampoline performance there are two alternating phases. Firstly, there is a *flight phase* in which the actual moves are performed and, secondly, there is a *contact phase* in which the athlete gains momentum for her/his next move, see Figure 4.2. Furthermore, a contact phase can be separated into two sub-phases, a *landing phase* and a *takeoff phase*. In the following, a trampoline *jump* is defined to be the concatenation of one takeoff phase at the beginning, one flight phase in the middle, and one landing phase at the end.

During these three phases, the athlete assumes and executes different poses and rotations, see Figure 4.3. The first three subfigures (Figure 4.3(a)–(c)) show different body poses assumed during the contact phase of a jump. Since these poses are determined during the landing phase of a jump, they are referred to as *landing poses*. During the flight phase the athlete assumes certain body poses (Figure 4.3(d)–(f)) and/or executes rotations (Figure 4.3(g)–(i)) around the body's lateral and/or longitudinal axis. A given combination of a landing pose in the takeoff phase, poses and rotations during the flight phase and a landing pose in the landing phase of a jump completely characterize a given jump. In the following, all jumps which contain the same sequence of poses and rotations are considered to belong to the same *jump class*. Table 4.1 shows thirteen jump classes of low and intermediate difficulty along with a short description. For example, the class "tuck jump" (`TJP`) starts with the pose "on feet" (`Fe`) during the takeoff phase, it continues with the pose "tucked" (`Tu`), and finishes with the landing pose "on feet" (`Fe`). Another example is the jump class `BAR`, also known as Barani, consisting of the landing pose "on feet" (`Fe`) in the beginning, a 360 degree somersault forwards (`F360`) combined with a 180 degree twist (`T180`) and ending on the feet (`Fe`). In trampolining, the most basic jump class is the straight jump (`STR`) which only consists of the pose "on feet" at the beginning and at the end of the jump. During competitions athletes have to perform so called *routines* which are a sequences of jumps. Here, a routine starts with a number of straight jumps to gain momentum. After this preparation the athlete has to perform a sequence of ten jumps from a set of predefined jump classes. Then, in our classification scenario, the task is to segment the routine and to determine the classes of the performed jumps.

In total, we recorded 109 routines with difficulty scores ranging from 0.4 to 3.1 comprising a total of 750 jumps. Out of these 109 routines, we chose 13 routines to form a routine database $\mathcal{D}_R$. From the remaining 96 routines, we manually assembled for each of the 13 jump classes

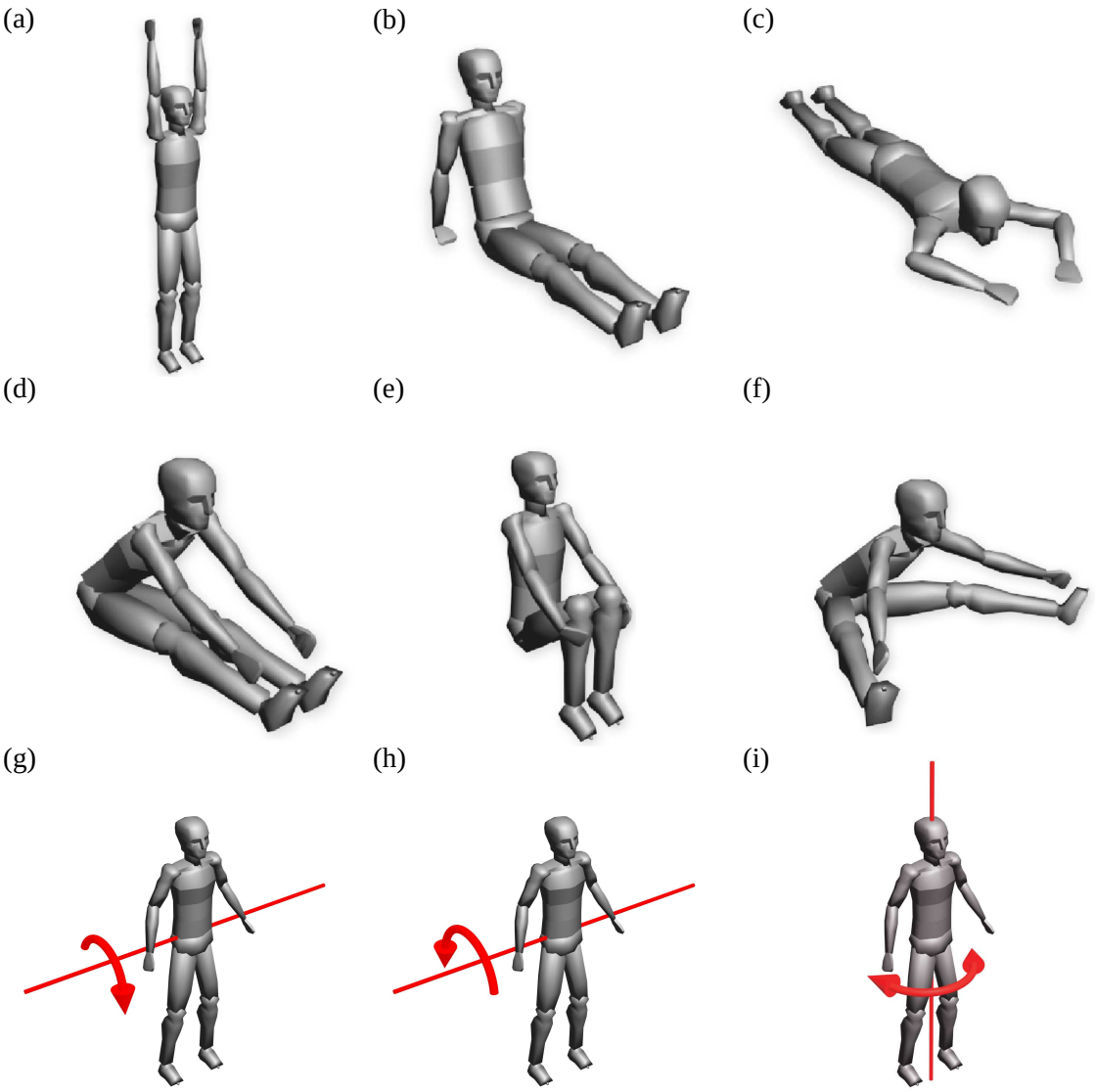

Figure 4.3. (a)–(c): Landing poses during the contact phase: on feet (`Fe`), seated (`Se`) and on the front (`Fr`). **(d)–(f):** Different body poses during the flight phase: piked (`Pi`), tucked (`Tu`) and straddled (`St`). **(g)–(i):** Rotations around main body axes during flight phase: lateral forwards (`F*`), lateral backwards (`B*`) and twists around longitudinal axis (`T*`).

16 instances—four instances for each of the four actors. The resulting dataset, containing 208 jumps, is denoted as cut database $\mathcal{D}_C$. We then partitioned $\mathcal{D}_C$ into two databases $\mathcal{D}'_C$ and $\mathcal{D}''_C$ each containing two jumps per actor from all 13 jump classes, amounting to 104 jumps.

4.2 Sensors

As stated before, there are many ways to record human motion data using, *e. g.* optical, inertial or depth-based mocap systems. A general overview of current optical mocap techniques can be

ID	Description	Poses and rotations during phases		
		Takeoff	Flight	Landing
BAR	Barani	Fe	T180, F360	Fe
FRF	Front to feet	Fr		Fe
HTW	Half twist	Fe	T180	Fe
HFR	Half twist to front	Fe	T180, F90	Fr
PJP	Pike jump	Fe	Pi	Fe
SHA	Seat half twist to feet	Se	T180	Fe
SST	Seat to feet	Se		Fe
BWB	Somersault backwards piked	Fe	Pi, B360	Fe
BWS	Somersault backwards to seat	Fe	Tu, B360	Se
BWC	Somersault backwards tucked	Fe	Tu, B360	Fe
SJP	Straddle jump	Fe	St	Fe
STR	Straight jump	Fe		Fe
TJP	Tuck jump	Fe	Tu	Fe

Table 4.1. Low and intermediate level jumps used for classification. The table shows how the jumps are composed of the poses and rotations displayed in Figure 4.3.

found in Chapter 2.

The most widely used motion capture systems for analyzing sport motions are optical marker-based systems as introduced in Section 2.1. Here, a set of calibrated cameras is used to record 2D images of an actor wearing a suit with retro-reflective or active markers, see Figure 4.4 (a). The advantage of such systems is clearly their precision. However, there are also some drawbacks, as illustrated by Figure 4.4 (a). For example, the lighting during the recording must be dim so that the markers can be distinguished from the background. Furthermore, the setup of the systems is cumbersome as many cameras need to be carefully placed, aligned, and calibrated in order to cover the large capture volume as needed for trampoline motions, see Figure 4.4 (b).

For these reasons, in many sports applications, human motion is often recorded using much cheaper devices such as single high-speed cameras or even standard consumer camcorders. Here, the recorded video stream has to be manually annotated using specialized software tools, from which various motion parameters such as joint positions or joint angles are derived. Obviously, the quality of the used cameras highly influences the accuracy of the deduced motion data. For example, if the camera has a low temporal resolution, motion blur as shown in Figure 4.4 (c) renders the correct positioning of annotations impossible. Furthermore, as the main drawback of such video based methods, the manual annotation process makes large-scale experiments with a high data throughput infeasible.

In this chapter, we use an inertial sensor-based mocap system consisting of seven Xsens MTx[1] sensor units denoted by $s_1, \ldots, s_7$. The sensors are placed inside a suit (see Figure 4.4 (d)) together with a wireless transmission system which sends the measured data directly to a computer. For this reason, inertial sensors do not pose any restrictions on the lighting requirements and can be used in many locations, even outdoor. Figure 4.4 (e) shows the placement of the seven sensors in our setup fixed at the trunk, the forearms, the upper legs and the lower legs of the athlete. Furthermore, as indicated by Figure 4.4 (f), the sensors are carefully attached in such a way that their local X-axes are aligned parallel to the limbs while pointing away from the body's center. In general, inertial

[1] http://www.xsens.com

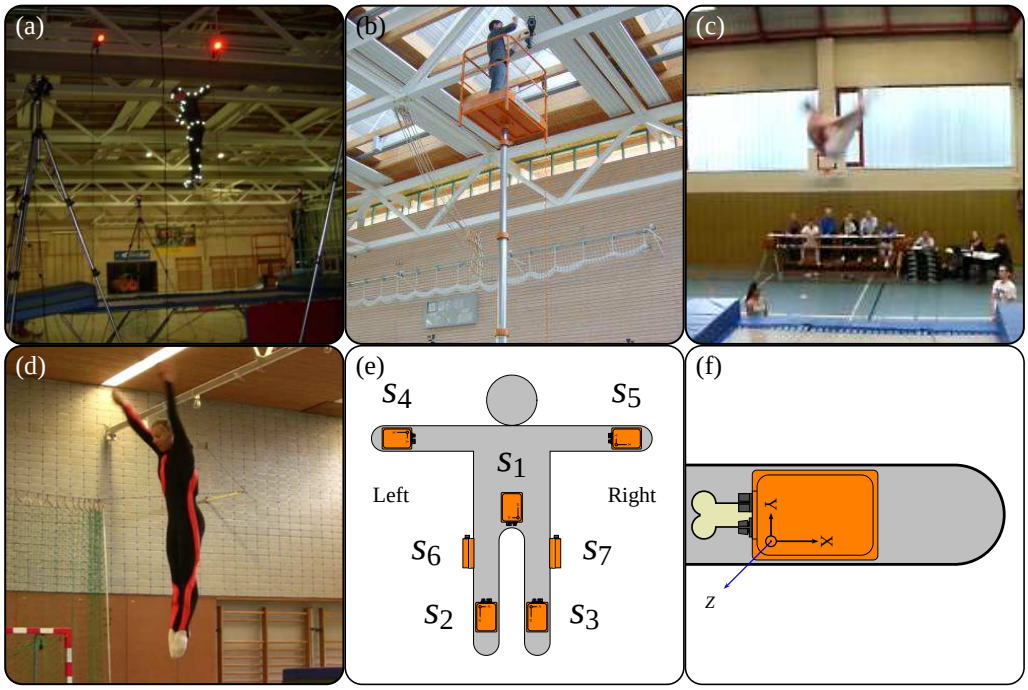

Figure 4.4. **(a):** Recordings using optical systems require controlled lighting conditions. **(b):** Cumbersome setup of an optical mocap system. **(c):** Optical recordings suffer from motion blur in case of fast motion. **(d):** Actor wearing a suit containing inertial sensors. **(e):** Locations of the seven motion sensors attached to the human body as used in this paper. **(f):** Inertial sensors are attached in direction of the body's limb and can measure the limb's orientation.

sensors only provide accelerations $\boldsymbol{a}$ and angular velocities $\boldsymbol{\omega}$ which are rather unintuitive quantities prone to noise. By combining inertial sensors with other sensor types Kemp *et al.* [1998]; Luinge and Veltink [2005], as done in the Xsens MTx units, it is possible to calculate full 3 degree of freedom global orientations denoted by q, see also Section 2.2.1. We now fix some further notations used in the rest of the chapter.

A *sensor data stream* is modeled as a sequence $D = (S_1, S_2, \ldots, S_K)$ of sensor readings $S_k \in \mathcal{S}$ for $k \in [1:K] := \{1, 2, \ldots, K\}$ (*w.r.t.* a fixed sampling rate, in our case 100 Hz). Here, $\mathcal{S}$ denotes the space of sensor readings and K denotes the number of frames. Each sensor reading S_k consists of the orientations, accelerations and angular velocities measured by the seven sensors:

$$S_k := \left(\mathsf{q}^k_{\mathsf{s}_1}, \ldots, \mathsf{q}^k_{\mathsf{s}_7}, \boldsymbol{a}^k_{\mathsf{s}_1}, \ldots, \boldsymbol{a}^k_{\mathsf{s}_7}, \boldsymbol{\omega}^k_{\mathsf{s}_1}, \ldots, \boldsymbol{\omega}^k_{\mathsf{s}_7}\right), \quad k \in [1:K], \tag{4.1}$$

where $\mathsf{q}^k_{\mathsf{s}} \in \mathbb{R}^4$, $\boldsymbol{a}^k_{\mathsf{s}} \in \mathbb{R}^3$, and $\boldsymbol{\omega}^k_{\mathsf{s}} \in \mathbb{R}^3$ for all $\mathsf{s} \in \{\mathsf{s}_1, \ldots, \mathsf{s}_7\}$ and all $k \in [1:K]$.

4.3 Segmentation

The first step of our proposed classification pipeline is the segmentation of an unknown trampoline motion sequence into separate jumps. Here, we make use of the two phases, the contact phase and the flight phase, which segment jumps in a natural way, see Figure 4.5 (a). While the actual jump is performed during the flight phase, the athlete gains momentum during the contact phase,

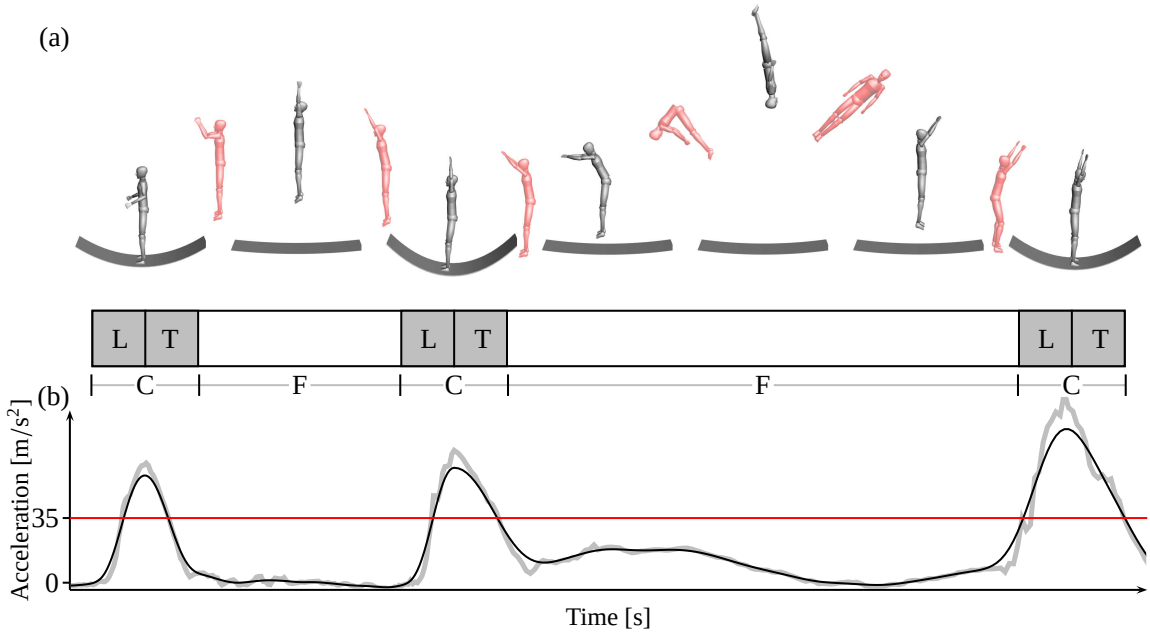

Figure 4.5. (a): Phases of a trampoline jump comprising a contact phase (C), a landing phase (L), a takeoff phase (T), and flight phase (F). **(b):** Absolute acceleration measured by sensor s_1 (light gray), as well as low pass filtered acceleration (black) and threshold (red) as used for the automatic segmentation described in Section 4.3.

which is always related to a large acceleration of the whole body. This acceleration can be measured using the sensor s_1 which is located at the athlete's trunk. As one can see in Figure 4.5 (b), the measurement of $\|\boldsymbol{a}_{s_1}\|_2$ is rather noisy. For this reason, we apply a low pass filter L of width corresponding to 0.1 seconds to the measured accelerations to obtain $\boldsymbol{a} := L(\|\boldsymbol{a}_{s_1}\|_2)$. Then, we label those frames k that satisfy the heuristic $\boldsymbol{a}^k > \tau$ to be a contact phase frame, where τ is a suitably chosen threshold. In practice, the value $\tau = 35\,\mathrm{m/s^2}$ turned out to be reasonable. We conducted an experiment to get a quantitative impression how well this simple segmentation algorithm works. To this end, we automatically segmented the 13 routines from the routine database $\mathcal{D}_R$ and compared the results with the manually generated ground-truth segmentations. Here, we considered a jump to be segmented correctly when the computed interval ends only differed from the ground-truth interval ends by a maximum of 15 frames (0.15 s). The experiment showed that, in total, 94% of the jumps were segmented correctly. Here, the wrongly segmented jumps were exclusively at the very beginning or the end of the trampoline routines, where the athletes were still in the preparatory phase and the accelerations were comparatively low. Actually, all of the important jumps during the routine were segmented correctly.

4.4 Feature Representation

As for the classification step, the raw sensor input is much too noisy and inconsistent to yield good motion representations. This is partly due to the noise introduced by the measurements itself. Even more problematic is the fact that different performances of the same jump may reveal significant spatial, dynamical, and temporal differences. In particular, there are many actor-specific

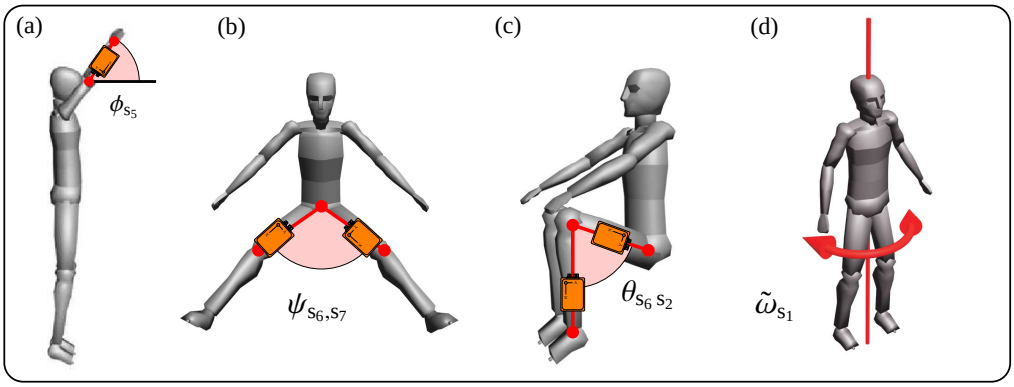

Figure 4.6. Illustration of examples for the various feature types. **(a):** Inclination of a limb. **(b):** Enclosed angle between two limbs belonging to different extremities. **(c):** Enclosed angle between two limbs belonging to the same extremity. **(d):** Angular velocity along the vertical axis of the body.

ID	Type	Description
F_1	ϕ_{s_1}	Inclination of lower spine
F_2	ϕ_{s_2}	Inclination of left lower leg
F_3	ϕ_{s_3}	Inclination of right lower leg
F_4	ϕ_{s_4}	Inclination of left forearm
F_5	ϕ_{s_5}	Inclination of right forearm
F_6	θ_{s_6,s_2}	Angle between left lower and upper leg
F_7	θ_{s_7,s_3}	Angle between right lower and upper leg
F_8	ψ_{s_6,s_7}	Angle between left upper and right upper leg
F_9	$\tilde{\omega}_{s_1}$	Absolute angular velocity around the body's longitudinal axis

Table 4.2. Description of the used features with feature ID and type.

performance variations within a jump class. Therefore, instead of working on the raw data itself, we derive from the inertial data suitable feature representations that encode important and intuitive properties of the athlete's body configuration while being invariant under global variations such as the actor's facing direction. In Section 4.5, we describe how to deal with local performance variations by introducing suitable class representations. We now introduce three different *feature types*. The first feature type ϕ_s measures the angle between the X-axis of a sensor s and the horizontal plane. If the sensor is aligned as shown in Figure 4.4 (f), this angle is the same as the angle between the limb and the horizontal plane, see Figure 4.6 (a). In other words, the feature ϕ_s measures the inclination of a limb with respect to the ground plane. The second feature type $\theta_{s,t}/\psi_{s,t}$ measures the enclosed angle between two limbs. Here, the only difference between $\theta_{s,t}$ and $\psi_{s,t}$ is the way the feature is computed. The feature $\psi_{s,t}$ measures the angle between limbs belonging to different extremities (Figure 4.6 (b)), while the feature $\theta_{s,t}$ measure the angle between limbs belonging to the same extremity (Figure 4.6 (c)). Finally, the third type of feature $\tilde{\omega}_s$ captures the angular velocity of the sensors X-axis. In other words, this feature type measures the velocity a limb rotates around its longitudinal axis. The exact formulas used to compute the introduced feature types are given in the appendix of this paper.

In the following, we will show how the features $\phi_s, \theta_{s,t}, \psi_{s,t}$ and $\tilde{\omega}_s$ can be computed. To this end, we assume the sensor data stream is defined as shown in Section 4.2. The rotations inside the

sensor data stream must be given in a suitable rotation representation, for instance, unit quaternions (see Shoemake [1985]). Furthermore, if q is a rotation in a given representation, then let $\mathsf{q}[\boldsymbol{v}]$ be the 3-dimensional vector $\boldsymbol{v}$ rotated by q. The features $\phi_{\mathrm{s}}, \theta_{\mathrm{s,t}}, \psi_{\mathrm{s,t}}$ and $\tilde{\omega}_{\mathrm{s}}$ are now defined as

$$\phi_{\mathrm{s}} = 1 - \frac{2}{\pi} \arccos\left\langle (0,0,1)^T, \mathsf{q}_{\mathrm{s}}\left[(1,0,0)^T\right]\right\rangle, \tag{4.2}$$

$$\theta_{s,t} = 1 - \frac{2}{\pi} \arccos\left\langle \mathsf{q}_{\mathrm{s}}\left[(1,0,0)^T\right], \mathsf{q}_{\mathrm{t}}\left[(1,0,0)^T\right]\right\rangle, \tag{4.3}$$

$$\psi_{s,t} = 1 - \frac{2}{\pi} \arccos\left\langle \mathsf{q}_{\mathrm{s}}\left[(-1,0,0)^T\right], \mathsf{q}_{\mathrm{t}}\left[(1,0,0)^T\right]\right\rangle, \text{and} \tag{4.4}$$

$$\tilde{\omega}_{\mathrm{s}} = \frac{2}{3\pi} \left|(\boldsymbol{\omega}_{\mathrm{s}})_x\right|. \tag{4.5}$$

Here, $\langle \cdot, \cdot \rangle$ denotes the scalar product of two vectors, while $(\cdot)_x$ is the x-component of a vector. Please note that the features are normalized to vary roughly in the range of $[-1, 1]$. This fact will be important for the class representation introduced in Section 4.5.

Based on these three feature types, we define in total nine *features*, as shown in Table 4.2. Mathematically, a feature is a function $F : \mathcal{S} \to \mathbb{R}$. By forming a vector of f features for some $f \geq 1$, one obtains a combined feature $F : \mathcal{S} \to \mathbb{R}^f$ referred to as a *feature set*. In this paper, F is equal to one of the following feature sets

$$F_{\texttt{I5A3W}} := (F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8, F_9)^T, \tag{4.6}$$

$$F_{\texttt{A3W}} := (F_6, F_7, F_8, F_9)^T, \tag{4.7}$$

$$F_{\texttt{I5W}} := (F_1, F_2, F_3, F_4, F_5, F_9)^T, \text{ or} \tag{4.8}$$

$$F_{\texttt{I5A3}} := (F_1, F_2, F_3, F_4, F_5, F_6, F_7, F_8)^T, \tag{4.9}$$

where the index (*e. g.* `I5A3W`) gives a hint on what features are included in the feature set. The part `I5` stands for the five inclination type features F_1, F_2, F_3, F_4, F_5, `A3` represents the three angular type features F_6, F_7, F_8, and `W` corresponds to the one angular velocity type feature F_9. This naming convention becomes important in Section 4.6, where we discuss the importance of the different feature types for the proposed classification scenario. Figure 4.7 shows how a feature set $F = F_{\texttt{I5A3W}}$ is applied to a sensor data stream D. The result is represented by a *feature matrix* $F(D) = (F(S_1), \ldots, F(S_K))$ with f rows and K columns, where in this case $f =$ and $K = 132$. Each row in such a feature matrix represents one feature, while each column represents the feature values $F(S_k)$ for a frame $k \in [1:K]$.

4.5 Class Representation

Based on feature matrices, we now describe a representation that captures characteristic properties of an entire motion class. To this end, we adapt the concept of *motion templates* (MTs), which was previously introduced in Müller and Röder [2006]. Here, given a class $C = \{D_1, \ldots, D_N\}$ consisting of N example motions D_n, $n \in [1 : N]$, one first converts all motions into features matrices X_n. Then, the idea is to compute a kind of average matrix. However, note that the N motions generally have a different length. Therefore, dynamic time warping is applied to temporally align the motions and to warp all feature matrices to yield the same length. The average matrix X_C over the warped feature matrices is then referred to as class motion template. In Müller and Röder

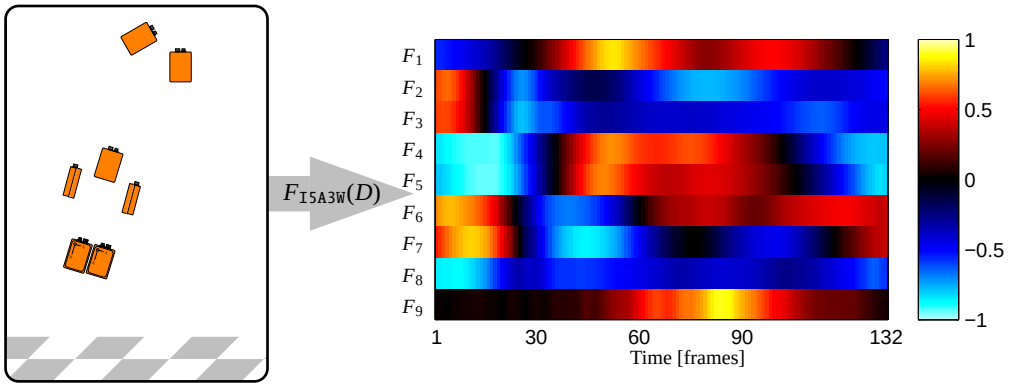

Figure 4.7. Feature representation of a jump of class $C = \texttt{BAR}$ using feature set $F_{\texttt{I5A3W}}$.

[2006], this concept is applied to boolean-valued features matrices yielding boolean feature matrices. As a consequence, regions in the class MT with the values zero/one indicate periods in time (horizontal axis) where certain features (vertical axis) consistently assume the same values zero/one in all training motions, respectively. By contrast, regions with values between zero and one indicate inconsistencies mainly resulting from variations in the training motions (and partly from inappropriate alignments). This property of MTs can then be used to automatically mask out the variable aspects of a motion class when being compared with an unknown motion data stream. This makes motion classification very robust even in the presence of significant performances differences, see Müller and Röder [2006] for details.

We now apply the concept of motion templates to our trampoline classification scenario. Let $\mathbb{C} = \{\texttt{BAR}, \ldots, \texttt{TJP}\}$ be the set of all considered jump categories and let $C \in \mathbb{C}$ be one of the motion classes. By using a feature set F, we convert all example motions contained in C into feature matrices. Opposed to Müller and Röder [2006], however, our features are real-valued, so that we need some modifications in the MT computation. To balance out the importance of the various features contained in F, we first normalize all features to approximately have the same range $[-1, 1]$. As an example, Figure 4.8(a)–(c) shows the resulting feature matrices of three example jumps from the class $C = \texttt{BAR}$. Then, as in Müller and Röder [2006], we temporally warp the normalized feature matrices and compute an average matrix X_C, see Figure 4.8 (d).

Now, starting with real-valued instead of boolean-valued feature matrices, the inconsistencies are not revealed as described in Müller and Röder [2006]. Instead, we compute a *variance template* V_C, which encodes the entry-wise variance of the N warped feature matrices, see Figure 4.8 (e). Here, the idea is that inconsistent regions in the real-valued feature matrices induce larger variances than consistent regions. Now the variance template can be used to mask out inconsistencies in X_C. In our setting, we mask out those regions of X_C, where the value in V_C is larger than the 75% quantile of all values of V_C. In other words, the 25% most variant values are ignored, see Figure 4.8 (f). Here, the percentage value of 25% has been determined experimentally, yielding a good trade-off between preserving sufficient motion characteristics while suppressing unwanted motion variations. The remaining 12 masked class templates are shown in Figure 4.9.

Mathematically, we model the masking as a separate *mask matrix* $M_C \in \mathbb{R}^{f \times K}$, where a value of 0

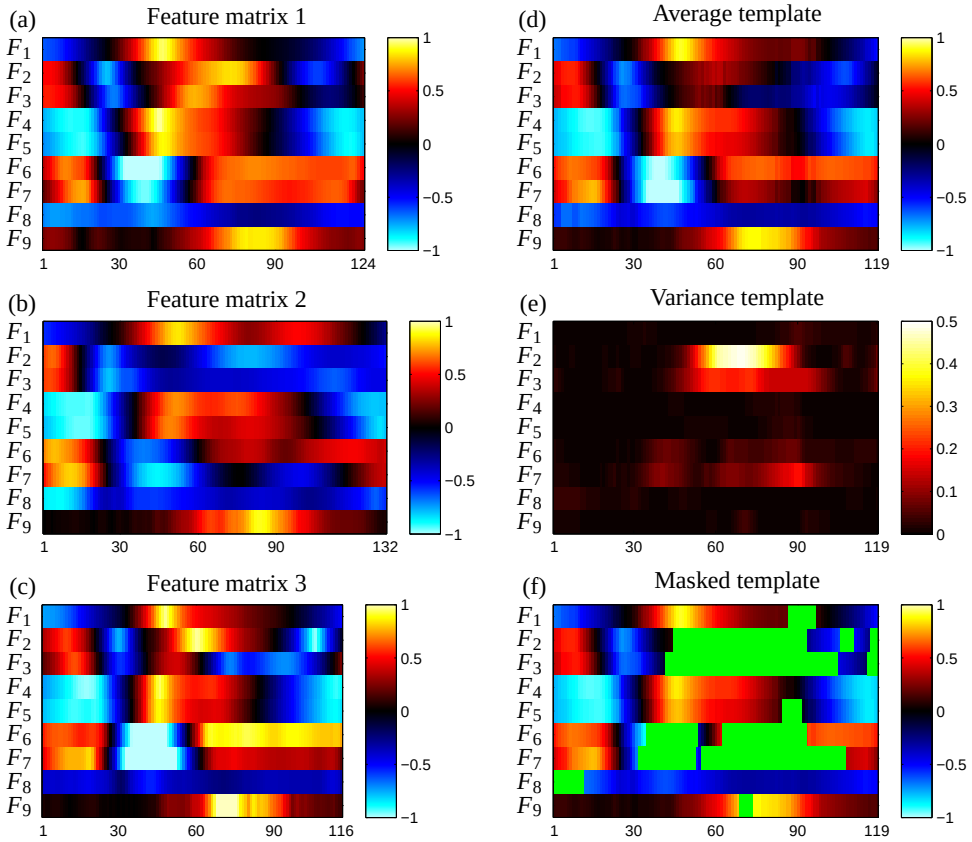

Figure 4.8. Template computation: **(a)–(c):** Feature matrices for three different jumps from the class BAR. **(d):** Average of aligned feature matrices (average template). **(e):** Variances of aligned feature matrices (variance template). **(f):** Template, where regions with 25% highest variances are masked out (masked template).

means that the value is masked out. The entries of M_C can be computed in the following way:

$$M_C(i,j) := \begin{cases} 1: & V_C(i,j) \leq Q_{75\%}(V_C) \\ 0: & \text{else} \end{cases} \tag{4.10}$$

for $i \in [1:f]$ and $j \in [1:K]$. Here, $Q_{75\%}(V_C)$ is the 75% quantile of V_C. Later in this paper, we will introduce a scenario where we seek to amplify the influence of certain feature functions. This can be modeled by allowing other values beside 0 and 1 inside the mask matrix.

4.6 Classification and Experiments

For the classification we locally compare an unknown jump with all class MTs X_C for $C \in \mathbb{C}$ and then label the jump according to the class MT having the smallest distance to the jump. In the following, let $Y \in \mathbb{R}^{f \times L}$ be the feature matrix of an unknown jump to be classified, where L is the length of the jump and f is the number of features. We use as distance measure a variant of dynamic time warping (DTW) as described in Müller and Röder [2006]. Especially, we adjust the local cost measure c in order to be compatible with our masking. Let $m(k) := \sum_{i=1}^{f} M_C(i,k)$, then

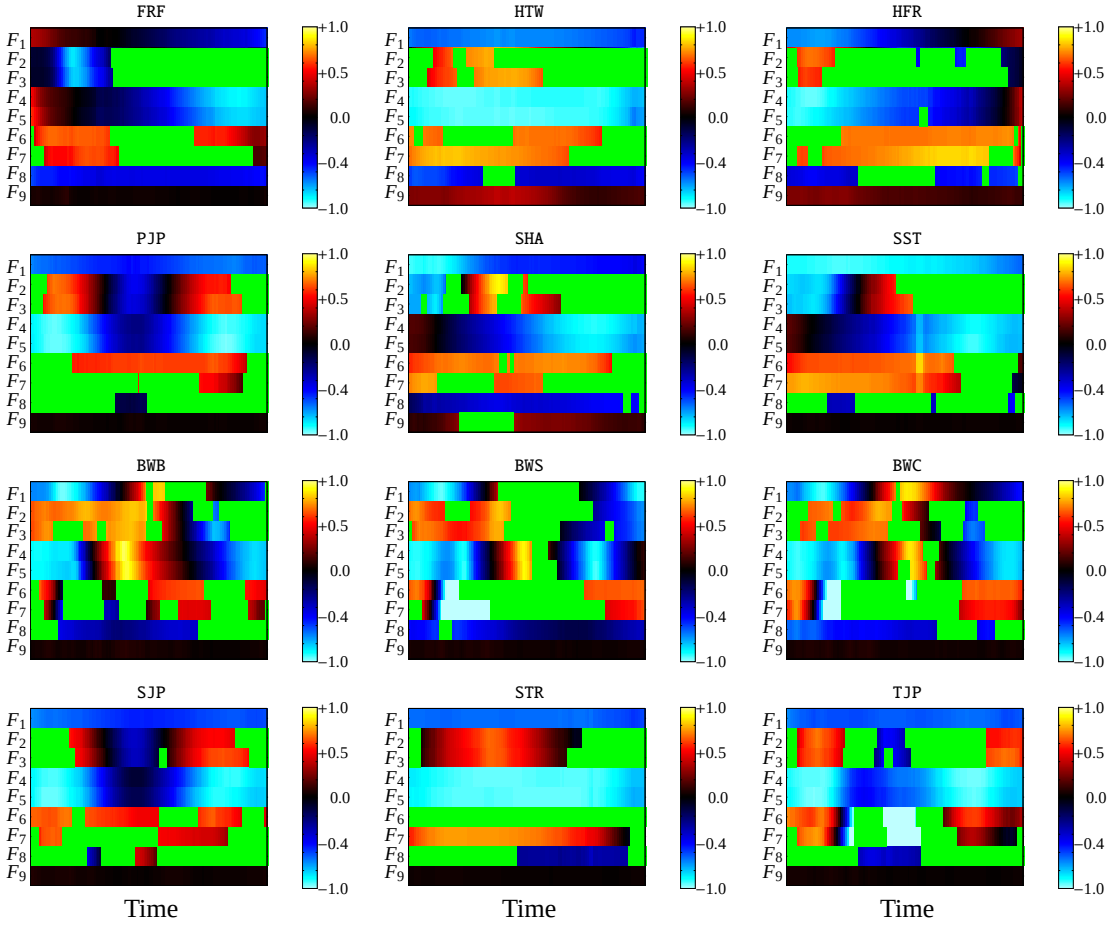

Figure 4.9. Depiction of the masked templates for twelve out of the thirteen jump classes. The template of the missing class BAR is shown in Figure 4.8 (f).

we define the masked local cost measure

$$c(k,\ell) := \left(\frac{1}{m(k)} \sum_{i=1}^{f} M_C(i,k)\ (X_C(i,k) - Y(i,\ell))^2 \right)^{\frac{1}{2}}, \tag{4.11}$$

for $m(k) \neq 0$ and $c(k,l) = 0$ for $m(k) = 0$, where $k \in [1:K]$ and $\ell \in [1:L]$. Now, the *distance* Δ_C between a class C with MT X_C and mask M_C and a feature matrix Y is defined as

$$\Delta_C(Y) := \frac{1}{K} \mathrm{DTW}\,(X_C, Y)\,, \tag{4.12}$$

where DTW denotes the DTW-distance between the sequences of columns defined by X_C and Y using the local cost measure c. Finally, the classification problem for an unknown jump with feature matrix Y can be solved by identifying the class $C \in \mathbb{C}$ which has the smallest distance $\Delta_C(Y)$.

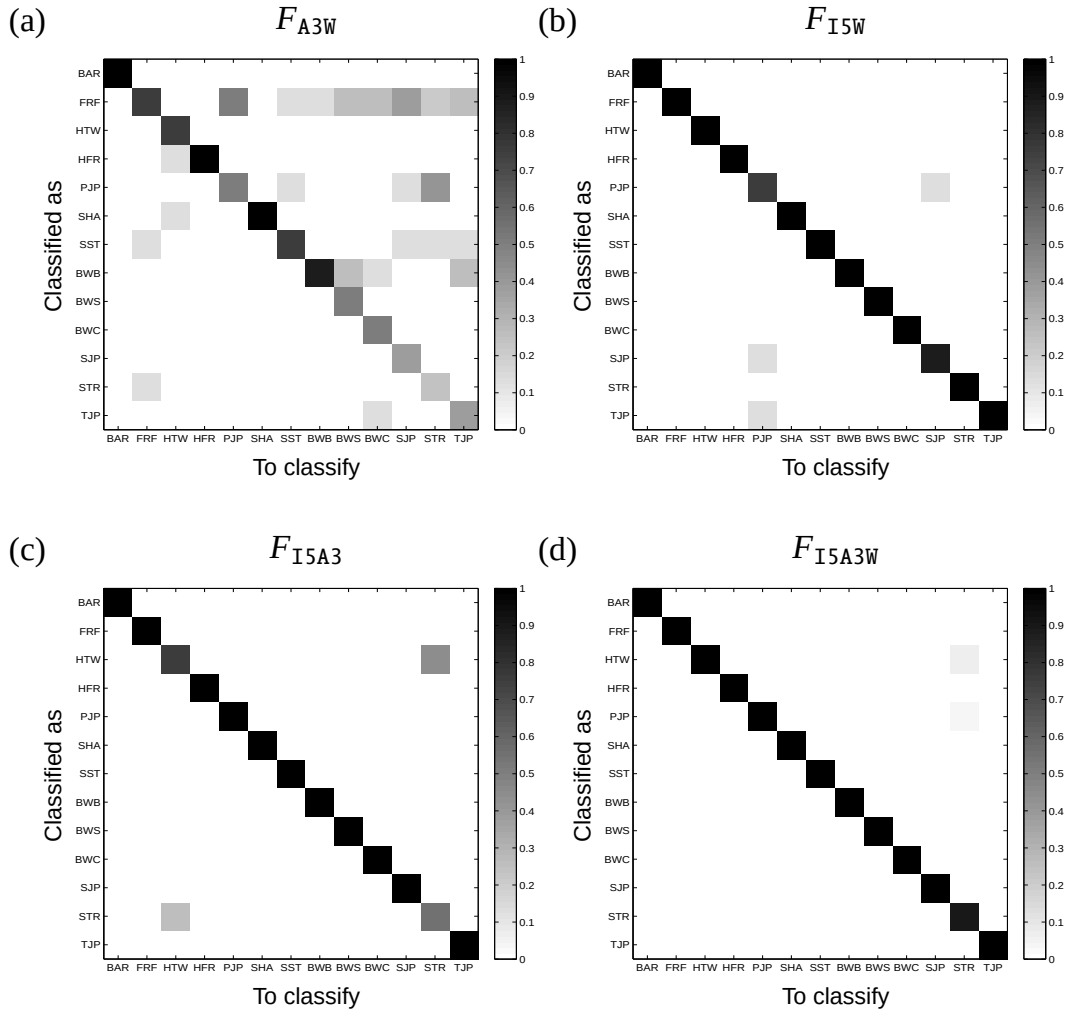

Figure 4.10. Confusion matrices showing the influence of the different feature types. The learning database is $\mathcal{D}'_C$ while the evaluation database is $\mathcal{D}''_C$. In all four cases the quantile mask introduced in Section 4.5 is used.

4.6.1 Influence of Feature Types

We first report on an experiment for investigating how the quality of the classification depends on the used feature types. To this end, we use confusion matrices, which give a qualitative impression which jump classes are classified correctly, and which jump classes are confused among each other. Such confusion matrices display the ratio of how many motions from a given class (abscissa) were classified as a certain class (ordinate), where dark entries represent a high percentage of motions. If the used feature types discriminate jump classes well, this would result in a dark diagonal leading from the top left of the matrix to the bottom right. In this experiment, we use the jumps from database $\mathcal{D}'_C$ to learn the motion templates and use $\mathcal{D}''_C$ for evaluation.

Figure 4.10 shows the confusion matrices for the four different feature sets defined in Section 4.4, where the feature set $F_{\texttt{I5A3W}}$ includes all feature types, while the feature sets $F_{\texttt{A3W}}$, $F_{\texttt{I5W}}$, and $F_{\texttt{I5A3}}$

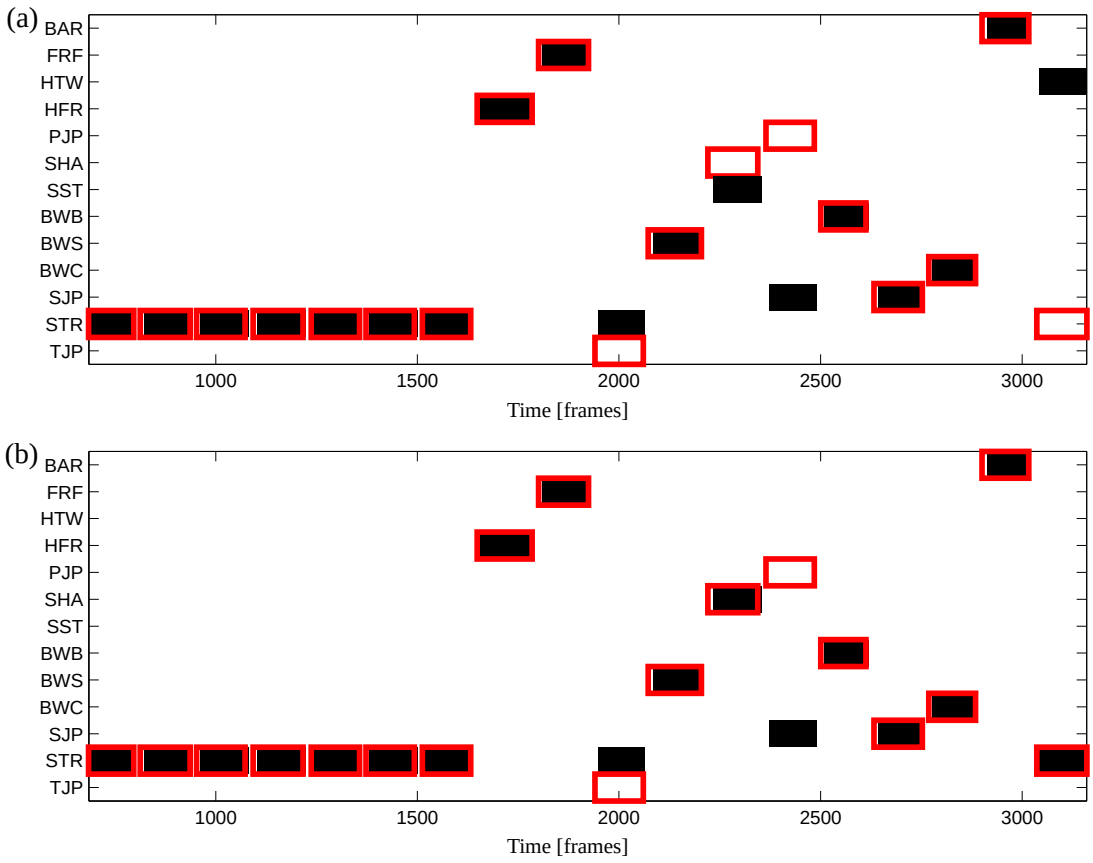

Figure 4.11. Classification results for routine scenario (red: manual annotation, black: automatic classification). The class representations where learned using database $\mathcal{D}'_C$, while the classified routines are taken from database $\mathcal{D}_R$. **(a):** Classification result for an example routine when using quantile masks. **(b):** Classification result for the same routine when using weighted masks. **(c):** Classification accuracy for the 13 learned jump classes using different masking techniques.

lack one of the feature types. In Figure 4.10 (a) one can see that the feature set $F_{\texttt{A3W}}$, which omits the inclination aspect, performs worst. This is expressed by the many high-valued off-diagonal entries which are an indication for massive miss-classifications. This shows that the feature set $F_{\texttt{A3W}}$ is too sparse for distinguishing different jump categories. Figure 4.10 (b) shows the results for the feature set $F_{\texttt{I5W}}$. Here, while most of the jumps were classified correctly, the jump classes `PJP`, `SJP`, and `TJP` are mixed up among each other. This is due to the fact that these jump classes only differ in the configuration of the legs during the flight phase. For example, in both jump classes `PJP` and `SJP` the legs are straight to the front during flight. The only difference is that in the jump class `SJP` the legs are additionally straddled. If the feature set contains inclination and angle feature types, as shown in Figure 4.10 (c), the classification works better for the jump classes `PJP`, `SJP`, and `TJP`, but now other jump classes as `STR` and `HTW` get mixed up. Here, these two jump classes only differ in a rotation around the bodies longitudinal axis. For this reason, the feature that measures the angular velocity is needed to capture the difference between the two jump classes. Finally, Figure 4.10 (d) shows, that the proposed feature set $F_{\texttt{I5A3W}}$ almost perfectly separates all jump classes from each other.

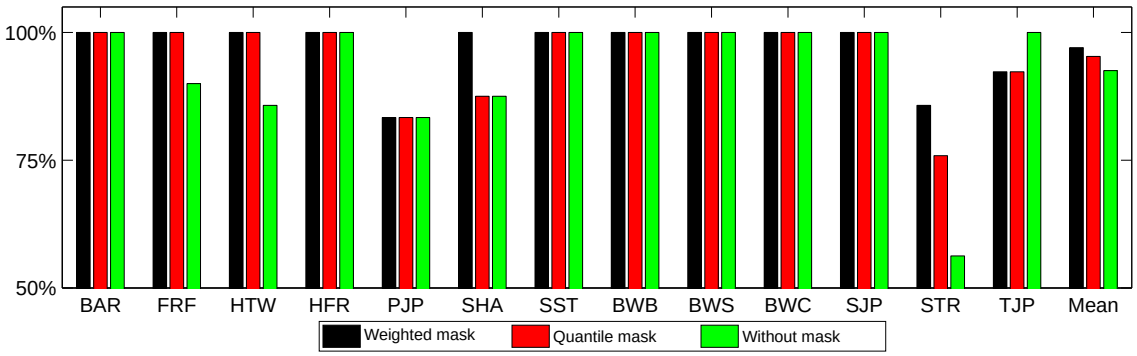

Figure 4.12. Overall classification accuracy for the 13 learned jump classes using different masking techniques.

4.6.2 Routine Classification

As main experiment, we combine the automatic segmentation from Section 4.3 with the classification introduced above. Here, our task is to evaluate how well the overall pipeline performs in a realistic trampolining scenario. Furthermore, we discuss how the masking proposed in Section 4.5 affects the retrieval results. For this evaluation, we use the thirteen routines from the database $\mathcal{D}_R$ for evaluation, while the motion templates are again learned from the databases $\mathcal{D}'_C$. Furthermore, we use quantile masks as defined in Equation (4.10). Figure 4.11 (a) displays a classified routine, where the black regions represent the automatic classification result and the red rectangles indicate the manual ground-truth annotations. It can be seen that for this example 14 out of 18 jumps were classified correctly. Here, for example, the misclassification of the jump `SHA` (frames 2200–2350) with the class `SST` is due to the fact that the feature F_9 is the only feature which is actually able to capture the difference between this two classes. Similarly, on can explain the confusion between `STR` (frames 3050–3200) and `HTW`. In such cases, the influence of the feature F_9 on the local cost measure c is not large enough (its only one ninth compared to the features $F_1, \ldots, F_8$). In order to better separate the confused jump classes from each other, one can increase the influence of the feature F_9 by replacing all ones in the quantile mask matrices of the class representations belonging to feature F_9 with some value larger than one (five in our case). The effect of such so called *weighted mask matrices* can be seen in Figure 4.11 (c), where the previously misclassified jumps `SHA` and `STR` are now classified correctly. The misclassifications between the jump `TJP` (frames 1950–2075) with `STR` and the jump `PJP` (frames 2380–2500) may be explained as follows. First note that the performance variations between jumps that belong to the same class are often significant—even within the jumps of the same athlete. Such variations are actually masked out by our local cost measure. Now, the differences between two jump classes such as `TJP` and `STR` or `PJP` and `SJP` are often subtle and only refer to a single motion aspect. It may happen that such aspects are actually masked out by our masking concept, which in turn leads to unwanted confusion. These examples indicate the trade-off between robustness on the one hand and discrimination capability on the other hand.

In addition to this qualitative analysis, we performed a quantitative analysis to measure the classification accuracy for each jump class. We say that an automatically segmented jump has been classified correctly if its segment boundaries lie in the neighborhood of an annotated jump (using a tolerance of 0.15 sec) and if the computed class label coincides with the annotated label. In

our experiments, we consider three different masking strategies: binary masking (quantile mask), weighted masking, and no masking at all. Figure 4.11 (c) reveals that the classification results are very good for most classes regardless the masking strategy used. This again shows that our proposed features are capable to capture relevant motion characteristics. When using the weighted mask matrix the classification results are generally better than when using the binary mask. Especially the jump classes SHA and STR, as in the previous paragraph, benefit from the use of weighted masking. A good example how masking in general improves the classification results are the jump classes FRF, FTW, and STR. Here the variances, within the jump classes are very high among actors and result in misclassified jumps, whenever the masking is not used. On the contrary the jump class TJP does not benefit from masking out variant regions, since, in this case, these regions also contain the only information that is able to discriminate this jump class from other jump classes.

4.7 Conclusions

In this chapter, we introduced a pipeline for the automatic segmentation and classification of trampoline routines based on inertial sensor input. Here, our motivation for using inertial sensors was that such sensors deal with dynamic motions better and do not impose constraints as far as the recording volume or lighting conditions are concerned. As our main contribution, we discussed suitable feature representations that are invariant to spatial variations and robust to measurement noise. Based on this feature representations, we introduced real-valued motion templates that grasp the characteristics of an entire jump class. To handle significant performance variations, we introduced a masking scheme based on variance templates. Furthermore, we presented a weighting strategy to enhance the influence of certain features. For future work we want to apply these techniques in an online scenario, where we assess the performance of an athlete and directly give feedback for performance improvement. A possible means of such feedback might be the sonification of certain motion parameters with respect to a learned reference performances.

Chapter 5

Human Shape Estimation Using Depth Sensors

Tracking 3D human motion data constitutes an important strand of research with many applications to computer animation, medicine or human-computer-interaction. In recent years, the introduction of inexpensive depth cameras like Time-of-Flight cameras or the Microsoft Kinect has boosted the research on monocular tracking since they constitute comparably cheap to obtain so-called 2.5 dimensional depth maps, see also Section 2.3. Tracking from such depth input is especially appealing in home consumer scenarios, where a user controls an application only by using his own body as an input device and where complex hardware setups are not feasible.

While depth data facilitates background subtraction compared to pure image based approaches, tracking still remains challenging because of the high dimensionality of the pose space and noise in the depth data. Currently, there exist three different strategies to harness depth data for tracking human motions. Discriminative approaches detect body parts or joint-positions directly from the depth images. Such approaches often neglect the underlying skeletal topology of the human which may lead to improbable joint locations and jitter in the extracted motion. Generative approaches fit a parametric model to the depth data using an optimization scheme. Here, the accuracy of the final tracking result is dependent on the degree to which the body model matches the true body shape of the person. In practice, such models are often obtained in a preprocessing step, *e. g.*, using laser scanners which are not available in home consumer scenarios. Finally, hybrid approach combine the the advantages of discriminative and generative approaches and show good results for fast motions in real-time scenarios.

Recently, first attempts have been made to obtain the shape of a person by fitting a parametric model to a set of depth images of a strictly defined calibration pose. However, the runtime in the orders of one hour as well as the requirement of a fixed calibration pose limit the applicability in a practical scenario.

Contributions. We contribute with algorithmic solutions that improve the performance of model-based depth trackers, by providing a personalized shape of the tracked person that is calculated from only two sequentially taken depth images. In particular, we present a new shape estimation method that makes model fitting an order of magnitude faster compared to previous ap-

Figure 5.1. (From left to right): Actor standing in the front of a single Kinect camera. Color coded depth data (red is near, blue is far) as obtained from the Kinect. Automatically estimated body shape of the actor. Two complex poses reliably tracked with our algorithm (left: input depth, right: estimated pose).

proaches Weiss *et al.* [2011] at no loss of quality. Secondly, we extend an existing tracking algorithm by Baak *et al.* [2011] to obtain a personalized version that works with arbitrary body shapes. As another contribution, we deployed an extensive dataset of 15 minutes of calibrated depth and marker-based motion capture (mocap) data which was used to evaluate our proposed tracker and which is publicly available to the research community. We also contribute with suitable error metrics to make different trackers comparable on our data set. The contributions presented in this chapter have been published in Helten *et al.* [2013a]. This chapter closely follows that publication. Additionally, the discussion of related work in Section 5.1, was presented in Helten *et al.* [2013c].

Organization. The remainder of the chapter is organized as follows. After discussing related work and introducing some of the challenges current approaches face (Section 5.1), we present our novel shape estimation method in Section 5.2. Then, in Section 5.3, we describe our personalized tracker and evaluate it with respect to previous approaches. Finally, we conclude in Section 5.4 with a discussion of limitations and an outlook to future work.

5.1 Full-body Depth-Trackers

Depth-based tracking of full-body human motion focuses on using inexpensive recording equipment that is easy to setup and to use in home user applications. As a consequence, depth based approaches have to deal with various challenges that marker-less tracking approaches do not face. Commercial systems that make use of this kind of motion tracking can be found *e. g.* in the Microsoft Kinect for XBox[1], the SoftKinetic IISU Middleware[2] for pose and gesture recognition, as well as the SilverFit[3] system for rehabilitation support. So far, several depth-based tracking methods have been published that can be classified into three basic types: Generative approaches, discriminative approaches and hybrid approaches. Key parts of this section have been published in and closely follow Helten *et al.* [2013c].

[1] http://www.xbox.com/Kinect
[2] http://www.softkinetic.com
[3] http://www.silverfit.nl/en.html

5.1.1 Generative Approaches

Generative approaches use parametrized body models that are fit into the depth data using optimization schemes. In particular, the optimization process maximizes a model-to-image consistency measure. This measure is hard to optimize due to the inherent ambiguity in the model-to-data projection. In particular, when using monocular video cameras, this ambiguity precludes efficient and reliable inference of a usable range of 3D body poses. Depth data reduce this ambiguity problem but it is still one of the main algorithmic challenges to make generative methods succeed.

A first approach for obtaining pose and surface of articulated rigid objects from ToF depth images was presented in Pekelny and Gotsman [2008]. Under the assumption that the movement of the tracked object is small *w.r.t.* the capture speed of the depth camera, the authors track individual bones from a manually pre-labeled depth image using an iterative closest point (ICP) approach. In each frame, previously unlabeled depth pixels are assigned to the bone that best explains the unlabeled depth pixel. However, this approach was not real-time capable, running at around 0.5 frames per second (FPS). Another approach Knoop *et al.* [2009] that is specialized on human motion, generates point correspondences for an ICP based optimization from both 3D and 2D input. An example for 2D input could be a body part or feature detector working on 2D color images. All 3D points that could be projected onto the 2D feature point now define a ray in 3D space. The closest point of this ray to the model is used to generate a traditional 3D point constraint. The authors report a performance of 25 fps with this method, but the approach is limited to simple non-occluded poses since otherwise the tracker would converge to an erroneous pose minimum from which it cannot recover. Another early approach for real time capable depth-based motion tracking from monocular views was presented in Bleiweiss *et al.* [2009]. Here, the authors describe a general pipeline for obtaining pose parameters of humans from a stream of depth images that are then used to drive the motion of a virtual character in *e. g.* video games. To further increase the performance of generative approaches Friborg *et al.* [2010] proposed porting the computational intense local optimization to the graphics processor. However, all these approaches tend to fail irrecoverably when the optimization is stuck in a local minimum. This problem also exists in other vision-based approaches and was *e. g.* discussed in Demirdjian *et al.* [2005]. In general, these tracking errors occur due to the ambiguous model-to-data mapping in many poses, as well as fast scene motion. While the latter problem can be remedied by increasing the frame rate, the former was addressed by more elaborated formulations of the energy function. One option was lately presented in Ganapathi *et al.* [2012], where the authors proposed a modified energy function that incorporates empty space information, as well as inter-penetration constraints. A completely different approach was shown in Ye *et al.* [2012]. Here, multiple depth cameras were used for pose estimation which reduces the occlusion problem and enabled capturing the motion of multiple person using high resolution body models. The approach is not real-time capable, though. With all these depth-based methods, real-time pose estimation is still a challenge, tracking may drift, and with exception to Ye *et al.* [2012], the employed shape models are rather coarse which impairs pose estimation accuracy.

5.1.2 Discriminative Approaches

Discriminative approaches focus on detecting certain features in the depth data—such as joint locations—and later combine these independent cues to form a body pose hypothesis. These fea-

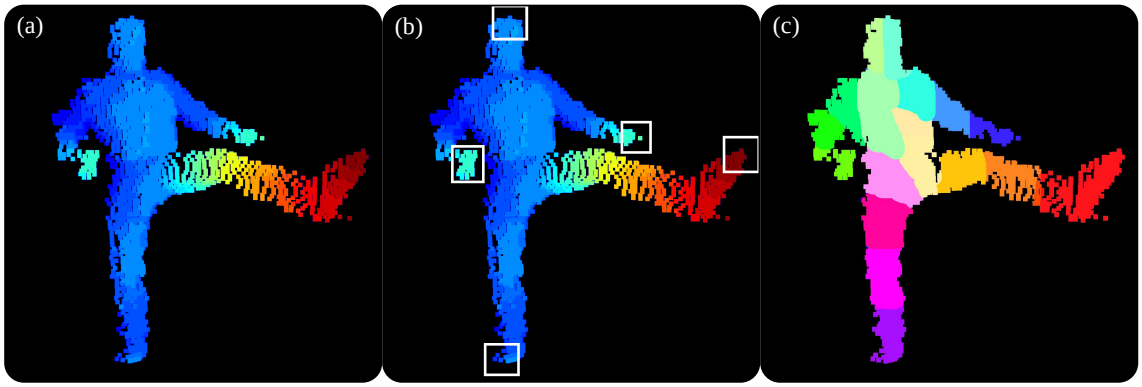

Figure 5.2. Typical intermediate results of discriminative depth tracking approaches. **(a):** Input depth image. **(b):** Detected geodesic extrema positions as proposed by Plagemann *et al.* [2010]. **(c):** Detected body parts as presented in Shotton *et al.* [2011].

ture are often learned for a pre-defined set of poses. For this reason, discriminative methods are not dependent on a numerical optimization procedure, and can infer pose also without temporal context and continuity. One algorithm for detecting human body parts in depth images was presented in Plagemann *et al.* [2010]. Here, the authors use so-called geodesic extrema calculated by iteratively using Dijkstra's algorithm on a graph deduced by connecting all depth pixels in the 2.5D depth data into a map. The assumption here is that geodesic extrema generally align with salient points of the human body, such as the head, the hands, or the feet, see also Figure 5.2 (b). To label the retrieved geodesic extrema according to the corresponding body part, the authors employ local shape descriptors on normalized depth image patches centered at the geodesic extrema's positions. Another body part detection approached is pursued in Zhu *et al.* [2010], where the authors deduce landmark positions from the depth image and include regularizing information from previous frames. These positions are then used in a kinematic self retargeting framework to estimate the pose parameters of the person. In contrast, the approach described in Shotton *et al.* [2011] uses regression forest learned on simple pair-wise depth features to do a pixel-wise classification of the input depth image into body parts, see also Figure 5.2 (c). To obtain a working regression forest for joint classification that works under a large range of poses, though, the authors had to train the classifier on approx. 500 000 synthetically generated and labeled depth images. For each body part, joint positions are then inferred by applying a mean shift-based mode finding approach on the pixels assigned to that body part. Using also regression forests for body part detection, Girshick *et al.* [2011] determine the joint positions by letting each depth pixel vote for the joint positions of several joints. After excluding votes from too distant depth pixels and applying a density estimator on the remaining votes, even the probable positions of non-visible joints can be estimated. Finally, Taylor *et al.* [2012] generate correspondences between body parts and a pose and size parametrized human model, which they also achieve by using depth features and regression forests. The parameters of this model are then found using a one shot optimization scheme, *i. e.* without iteratively recomputing the established correspondences. Discriminative approaches show impressive tracking results, where some discriminative methods even succeed in detecting joint information also in non-frontal occluded poses. However, since they often detect features in every depth frame independently, discriminative approaches tend to yield temporally unstable pose estimations results. Furthermore, for many learning-based methods, the effort to train classifiers can be significant.

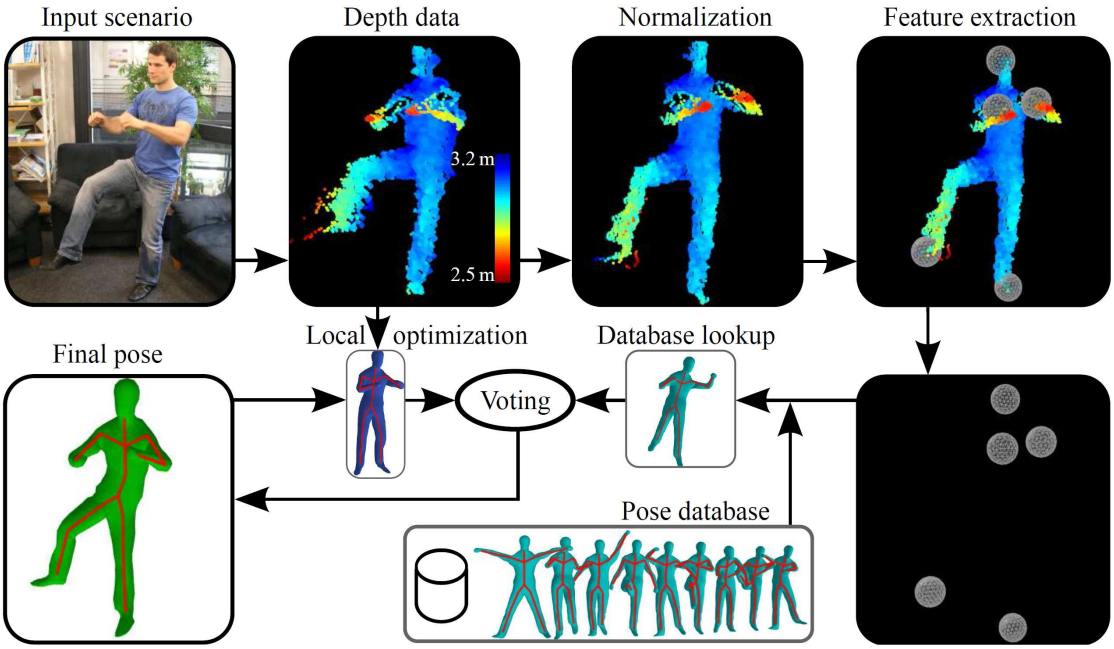

Figure 5.3. Overview of the hybrid depth tracker presented by Baak *et al.* [2011]. This figure was taken by courtesy of Andreas Baak from his thesis (Baak [2012]).

5.1.3 Hybrid Approaches

Combining the ideas of generative and discriminative approaches, hybrid approaches try to harness the advantages from both tracker types. On the one hand, hybrid trackers inherit the stability and temporal coherence of pose estimation results common to generative trackers. On the other hand, they show the robustness of pose inference even in partly occluded poses that characterizes discriminative approaches. A first method, in the domain of 3D surface reconstruction, was presented in Salzmann and Urtasun [2010]. Here, the discriminative tracker is used for initializing the surface model, while the generative tracker enforces the observance of distance constraints. The authors also sketched, how their approach can be applied to human pose reconstruction. At the same time, the first method with specialization to human pose estimation was presented in Ganapathi *et al.* [2010]. This work combines the geodesic extrema-based body part recognition presented in Plagemann *et al.* [2010] with a generative pose optimization scheme based on articulated ICP. Furthermore, the authors introduce a dataset comprising of calibrated ToF depth images and ground-truth marker positions that serves as common benchmark for future work in that field. The works by Baak *et al.* [2011] and by Ye *et al.* [2011] also use a discriminative tracker to initialize a generative pose estimation algorithm. In detail, the approach presented in Ye *et al.* [2011] uses a database consisting of 19 300 poses. For each of these poses, four synthesized depth images were rendered from different views. Using a principal axis based normalization, the point clouds are indexed using their coefficients in a PCA subspace. Here, the normalization of the point cloud in combination with the rendering from four different views is used to retrieve poses from the database independent from the orientation *w.r.t.* the depth camera. Note that by storing four different views in the database, the index size is increased to 77 200, while still only 19 300 poses are contained in the database. During tracking, the input point cloud is normalized in the same way, its PCA-coefficients are calculated and used for retrieving a similar point cloud

in the database. Finally, they refine the retrieved pose using the Coherent Drift Point algorithm presented in Liao *et al.* [2009]. This approach shows good pose estimation results on the benchmark dataset introduced in Ganapathi *et al.* [2010]. However, their approach does not run in real time—inferring the pose in one frame takes between 60 s and 150 s.

In contrast, the approach showcased in Baak *et al.* [2011] uses a modified iterated version of Dijkstra's algorithm to calculate geodesic extrema similar to the approach in Plagemann *et al.* [2010]. The stacked positions of the first five geodesic extrema, which often co-align with the head, hands and feet, serve as index into a pose database consisting of 50 000 poses. The suitability of such an approach has been previously discussed in Krüger *et al.* [2010], where the authors used the stacked positions of the body's extremities (head, hands, and feet) to index a database containing high dimensional motion data. As index structure the authors employed a *k*-d tree facilitating fast nearest neighbor searches. To be invariant to certain orientation variations of the person, Baak *et al.* normalize the query and the database poses based on information deduced from the depth point cloud. The incorporated generative tracker is a standard ICP approach that builds correspondences between preselected points from the parametrized human model and points in the depth point cloud. In each frame, they conduct two local optimizations, one initialized using the pose from the previous frame and one using the retrieved pose from the pose database. Using a late fusion step they decide based on a sparse Hausdorff-like distance function which pose obtained from the two local optimizations best describes the observed depth image. This pose is then used as final pose hypothesis, see Figure 5.3 for an overview of their approach. While not showing as good results as the approach presented in Ye *et al.* [2011], their tracker runs much faster at around 50–60 frames per second, enabling very responsive tracking. Another real-time approach was recently proposed by Wei *et al.* [2012]. Here, the authors use a discriminative body-part detector similar to Shotton *et al.* [2011] to augment a generative tracker. In particular, they use the pose obtained from the discriminative tracker only for initialization at the beginning of the tracking and for reinitializing the generative tracker in cases of tracking errors. For detecting wrongly tracked frames, they measure how well their body model with the current pose parameters explains the observed point cloud. Hybrid approaches, harnessing the advantages of both tracking worlds, are able to show superior performance compared too purely discriminative or generative approaches. However, even the current state-of-the-art hybrid trackers still have limitations, which we will elaborate on in the following.

5.1.4 Challenges

While providing good overall tracking results, hybrid approaches still suffer from the noisy character and the sparsity of the depth data and are prone to ambiguities originating from occlusions. In this section, we will focus on challenges that are related to the accuracy of the used body model. For a discussion of other challenges such as occlusions, we refer to Chapter 6.

Most trackers use an underlying model of the human body. Such models vary drastically ranging from simple representations as graphs (Pekelny and Gotsman [2008]; Zhu *et al.* [2010]; Shotton *et al.* [2011]; Girshick *et al.* [2011]; Taylor *et al.* [2012]; Salzmann and Urtasun [2010]; Ye *et al.* [2011]), over articulated rigid bodies (Knoop *et al.* [2009]; Friborg *et al.* [2010]; Ganapathi *et al.* [2012]; Wei *et al.* [2012]) to complex triangle meshes driven by underlying skeletons using skinning approaches (Baak *et al.* [2011]; Ye *et al.* [2012]; Ganapathi *et al.* [2010]). Here, the complexity of the model mainly depends on the intended application. While some approaches are only in-

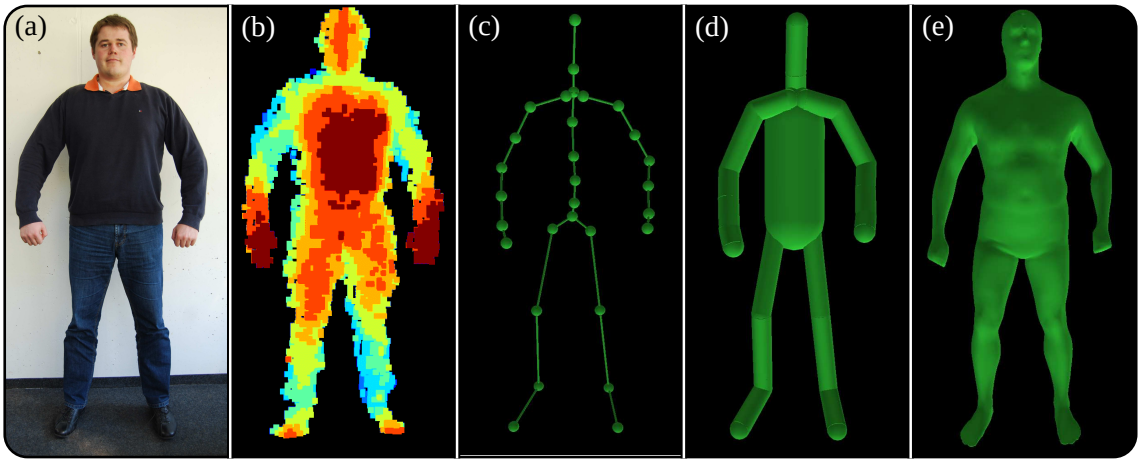

Figure 5.4. (a): Body shape of a person to be tracked. **(b):** Depth image of shape. **(c):** Graph model. **(d):** Model based on articulated cylinders and spheres . **(e):** High resolution surface model.

terested in tracking specific feature points of the body such as the positions of the extremities (Plagemann *et al.* [2010]) or joint positions (Shotton *et al.* [2011]), other approaches try to capture pose parameters such as joint angles (Baak *et al.* [2011]; Ganapathi *et al.* [2012]; Taylor *et al.* [2012]; Ganapathi *et al.* [2010]; Ye *et al.* [2011]; Wei *et al.* [2012]), or even the complete surface of the person including cloth wrinkles and folds (Ye *et al.* [2012]). Another requirement for a detailed surface model is the energy function used in generative or hybrid approaches. In particular, ICP-based trackers benefit from an accurate surface model to build meaningful correspondences between the model and the point cloud during optimization. In order to circumvent the problem of obtaining an accurate model of each individual person, some approaches use a fixed body model and scale the input data instead Baak *et al.* [2011]. However, this approach fails for persons with very different body proportions.

In general, the model of the tracked person is often assumed to be created in a preprocessing step using manual modeling or special equipment as full-body laser scanners. However, this is time consuming and involves expensive equipment, which renders it unfeasible in home application scenarios. To this end, most algorithms applied in these scenarios, such as Shotton *et al.* [2011], use a different approach. In a preprocessing step the authors use a large number of body models of different sizes and proportions to learn a decision-forest-based classifier that is able to label depth pixels according to the body part they belong to. As a consequence, this classifier becomes invariant to the size of the person and its proportions. During the actual tracking, the learned classifier can be used without obtaining an actual body model of the tracked person. Based on the labeled depth pixel the authors employ a heuristic to deduce the most probable joint position. This approach runs in real-time and works for many tracking applications.

However, for some augmented reality applications the reconstruction quality obtained from simple graphical body models may not be sufficient enough. A popular example is virtual try-on, where the person can wear a piece of virtual apparel that plausibly interacts with the person's body motion. Here, an accurate reconstruction of the person's body surface is beneficial in order to ensure believable visual quality or to give good indication whether the cloth actually fits. Also, model-based trackers that use high-detailed surface models benefit, if the shape of the model closely resembles the shape of the tracked person. Here, one can see that the tracking results

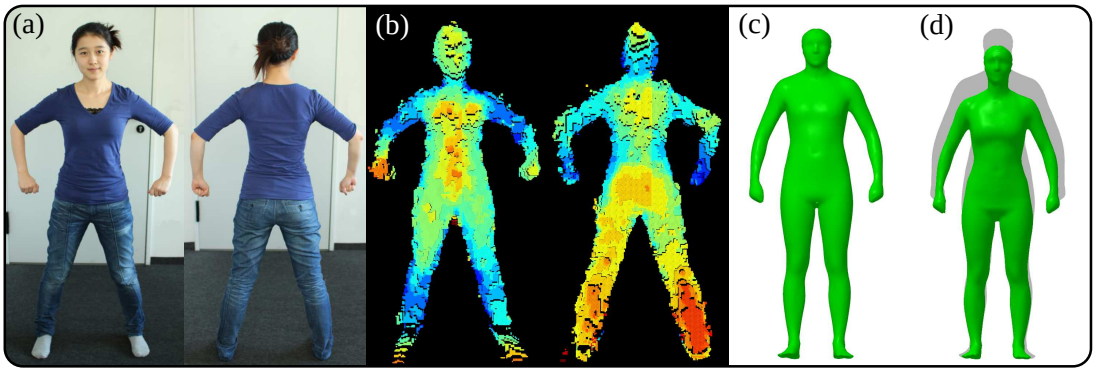

Figure 5.5. Shape estimation. **(a):** Calibration poses. **(b):** Depth input of poses. **(c):** Initial shape. **(d):** Estimated shape.

improve the better the body model matches. One possible approach would be to infer a high resolution body model from depth data in a preprocessing step and then use this model for tracking, visualization or physical simulations of objects in the augmented scene. Recently, one approach Weiss *et al.* [2011] has addressed this issue. Here, the authors fit a pose and shape parametrized model into the depth point clouds using an ICP-based approach. The point clouds were obtained from four sequentially captured depth images showing the person from the front, the back and two sides. However, the fact that the person had to reproduce the same pose in all four images and the optimization's runtime of about one hour makes this approach not applicable in home user scenarios.

5.2 Personalized Body Shape Estimation

In this section, we introduce a novel procedure for estimating the body shape from a single depth camera using only two different calibration poses and within only a minute of fitting time, see Figure 5.5 for an overview. In addition, even if the user only roughly matches the required calibration poses, our shape estimation algorithm achieves accurate results. We propose two innovations to achieve high speed and high accuracy. Firstly, our optimization scheme works purely in the 3D domain and does not revert to 2D data representations as silhouettes or contours as used in Weiss *et al.* [2011]. However, note that the richer 3D contour is implicitly represented in the 3D-domain. Using 3D cues instead of 2D cues typically results in fewer ambiguities and occlusion problems such as an arm in front of the observed body, which would be invisible in the observed contour. Secondly, in our optimization scheme we use a cost function that is not only based on distances of corresponding points, but also considers normal-based distances between points and planes. As a result, the optimization is less likely to get stuck in local minima and the speed of convergence is increased significantly.

5.2.1 Shape Model

Mathematically, our shape model is given as a mesh consisting of vertices and triangular faces. Let P be the number of vertices and, as explained below, let $\boldsymbol{\varphi}$ be a vector of shape parameters. Henceforth, we assume that the mesh is rigged with a skeleton which is driven by a pose parameter

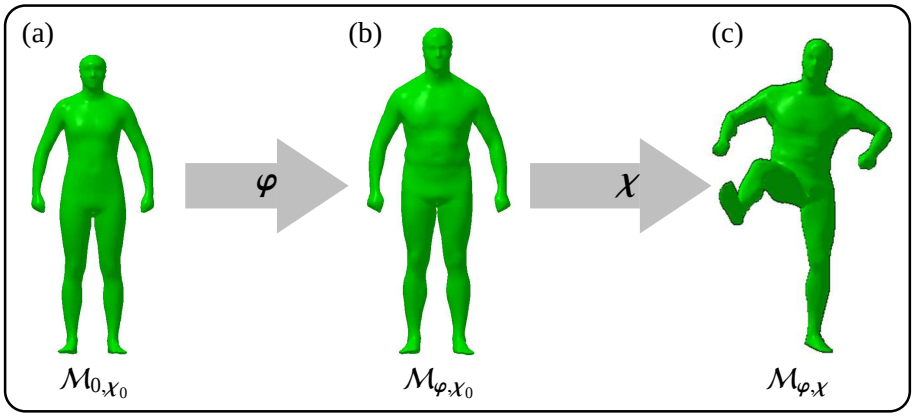

Figure 5.6. (a): Average mesh $\mathcal{M}_{0,\chi_0}$ in standard pose χ_0. **(b):** Personalized mesh $\mathcal{M}_{\varphi,\chi_0}$ in standard pose χ_0 given a shape parameter vector φ. **(c):** Personalized mesh $\mathcal{M}_{\varphi,\chi}$ given in a pose χ.

vector χ using linear blend skinning, see also Section 2.1.2. Hence, the 3D coordinates of the mesh depend on both φ and χ and can be represented as the stacked vector $\mathcal{M}_{\varphi,\chi} \in \mathbb{R}^{3\cdot P}$. Furthermore, let $\mathcal{M}_{\varphi,\chi}(p)$ denote the 3D coordinate of the p^{th} vertex, $p \in [1:P] := \{1, 2, \ldots, P\}$. Finally, from the triangulation one can derive a normal vector $\mathcal{N}_{\varphi,\chi}(p) \in \mathbb{R}^3$ for each vertex.

Our body model is a statistical model of human pose and body shape similar to Jain *et al.* [2010]. The statistical model is a simplified SCAPE model (Anguelov *et al.* [2005]), where we omit the terms responsible for modeling muscle bulging in order to speed up computations. Our model is generated from scans of $S = 127$ young male and female persons (Hasler *et al.* [2009]). This certainly limits the expressiveness of the model to a certain extent. However, as our experiments will show, even with a model generated from a relatively small number of scans we achieve better accuracy than Weiss *et al.* [2011] where 2 500 scans were used to construct the statistical model.

5.2.2 Model Construction

We follow the approach presented in Hasler *et al.* [2009]. Here, the authors register a template mesh with $P = 6\,449$ vertices into a point cloud using global and local mesh deformations. Given the S laser-scans, let $\mathcal{M}_s \in \mathbb{R}^{3P}$, $s \in [1:S]$ be the stacked vertex positions of the fitted meshes. Now, the *average mesh* is defined as

$$\mathcal{M}_{0,\chi_0} = \frac{1}{S}\sum_{s=1}^{S} \mathcal{M}_s, \tag{5.1}$$

see also Figure 5.6 (a). Note that to this end, all meshes $\mathcal{M}_s$ have to assume the same pose which is called the *standard pose* and is denoted by the index χ_0. Then, we compute the auto correlation matrix

$$C = \frac{1}{S}\sum_{s=1}^{S} (\mathcal{M}_s - \mathcal{M}_{0,\chi_0})(\mathcal{M}_s - \mathcal{M}_{0,\chi_0})^T. \tag{5.2}$$

Let Φ_s, $s \in [1:S]$ be the eigenvectors of C, sorted from most significant to least significant. A suitable eigenvector-matrix is now defined as

$$\Phi = [\Phi_1 \cdots \Phi_R], \tag{5.3}$$

with $R \leq S$. The corresponding vector $\boldsymbol{\varphi} \in \mathbb{R}^R$ is called the shape parameter vector or short *shape parameters*.

Using $\mathcal{M}_{0,\boldsymbol{\chi}_0}$, Φ, and $\boldsymbol{\varphi}$ we obtain a family of different body shapes in the following way:

$$\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0} := \mathcal{M}_{0,\boldsymbol{\chi}_0} + \Phi \cdot \boldsymbol{\varphi} \tag{5.4}$$

In Hasler *et al.* [2009] it was shown that by using dimensionality reduction techniques, one obtains already a wide range of naturally looking shapes of different people for a low-dimensional $\boldsymbol{\varphi}$. The mesh $\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}$ is called the *personalized mesh*, see also Figure 5.6 (b). In our experiments, we use the $R = 13$ most significant Eigenvectors. The shape space that is spanned by these vectors covers the overall body size, gender specific differences, muscularity and other coarse features. It does not cover fine details as facial features or wrinkles and fold or asymmetric body properties. However, as the following experiments show, it still enables us to reconstruct the overall appearance of a person with a better accuracy than previous approaches that use much complexer body models.

As for the underlying skeleton, we use a model containing 51 joints similar to Stoll *et al.* [2011]. Not all joints possess a full degree of freedom (DoF). For example, the spine is represented by several coupled joints that are parametrized by only 3 DoFs, which results in a smooth bending of the whole spine. In our experiments, we represent the pose of a person with 31 DoFs (3 translational and 28 rotational) encoded by the pose parameter vector $\boldsymbol{\chi}$. The skeleton was once manually fitted to the average shape corresponding to the parameter vector $\boldsymbol{\varphi} = 0$ in the pose $\boldsymbol{\chi}_0$. To be able to transfer the skeleton to other shapes, we represent the position of each joint as a linear combination of its surrounding vertices. Note that, using this kind of formulation, our model has two independent sets of parameters: Shape parameters $\boldsymbol{\varphi}$ and pose parameters $\boldsymbol{\chi}$. As a consequence, identical shape parameters always induce and identical shape and the same pose parameters always result in the same pose. This property is important for the shape optimization process described below.

5.2.3 Fitting Model to Data

Our shape estimation problem can be formalized as follows. First, we assume a target point cloud is given T consisting of points $T(q) \in \mathbb{R}^3$ for $q \in [1 : Q]$, where Q denotes the number of points. In our setting we assume that T is a depth image as supplied by a Kinect camera, but point clouds from other sources could also be used. The goal is to jointly optimize the shape and pose parameters of our shape model to best explain the given target point cloud.

Firstly, the shape and pose parameter vectors are initialized by $\boldsymbol{\varphi} = \boldsymbol{\varphi}_{\text{init}}$ and $\boldsymbol{\chi} = \boldsymbol{\chi}_{\text{init}}$. In our scenarios, we set $\boldsymbol{\varphi}_{\text{init}} = 0$ and $\boldsymbol{\chi}_{\text{init}}$ to the standard pose parameter $\boldsymbol{\chi}_0$ translated to the mean center of the point cloud T. In order to make the shape model compatible with the target point cloud T, we transform the shape model surface into a mesh point cloud. To this end, we basically consider the 3D coordinates $M(p) := \mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p)$, $[1 : P]$, of the mesh vertices. Since in our setting the target point cloud T comes from a depth image and hence only shows one side of the actor, we also restrict the mesh point cloud to the points that are visible from the depth camera's perspective (the rough orientation of the body is assumed to be known in the calibration phase). To simplify notation, we still index the restricted point cloud by the set $[1 : P]$.

We establish correspondences between the target point cloud and the mesh point cloud based on closest points. For each point $M(p)$, we define the corresponding point $T(q_p)$ to be the point that

minimizes the Euclidean distance between $M(p)$ and the point cloud T. Similarly, for each point $T(q)$ the point $M(p_q)$ is defined.

Based on these correspondences, we now introduce our optimization scheme. It is well known from the literature that one obtains faster convergence rates in rigid shape registration based on iterative closest points (ICP) when using point-plane constraints instead of point-point constraints, see Chen and Medioni [1992] and references therein. Furthermore, such constraints are more robust to noise from depth sensors leading to a more stable convergence. On the other hand, point-to-plane constraints are problematic when correspondences are far apart. Therefore, we design an energy functional that incorporates both point-point as well as point-plane constraints. First, for a pair $(p, q) \in [1:P] \times [1:Q]$ let

$$d_{\text{point}}(p, q) = \|M(p) - T(q)\|_2 \tag{5.5}$$

denote the Euclidean distance between the points $M(p)$ and $T(q)$. Next, we use the normal information supplied by the mesh to define a point-plane constraint. Let $N(p) = \mathcal{N}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p)$, $p \in [1:P]$, be the normal vector at the p^{th} vertex. Then, the distance between the point $T(q)$ and the plane defined by the normal $N(p)$ that is anchored at the point $M(p)$ is given by

$$d_{\text{normal}}(p, q) = \langle M(p) - T(q), N(p) \rangle. \tag{5.6}$$

Next, we fix a suitable threshold τ (in our experiments $\tau = 15\,\text{mm}$) to decide which of the distances should be considered depending on how far the two corresponding points are apart and we define

$$d_\tau(p, q) := \begin{cases} d_{\text{point}}(p, q), & \text{if } \|M(p) - T(q)\|_2 > \tau, \\ d_{\text{normal}}(p, q), & \text{otherwise.} \end{cases} \tag{5.7}$$

Finally, in the definition of the energy functional $E(\boldsymbol{\varphi}, \boldsymbol{\chi}|T)$ we consider all correspondences from the mesh point cloud to the target point cloud and vice versa:

$$E(\boldsymbol{\varphi}, \boldsymbol{\chi}|T) := \sum_{p \in [1:P]} d_\tau(p, q_p) + \sum_{q \in [1:Q]} d_\tau(p_q, q). \tag{5.8}$$

To minimize Equation (5.8), we use a conditioned gradient descent solver as described in Stoll *et al.* [2011]. To this end, we compute the analytic partial derivatives of $E(\boldsymbol{\varphi}, \boldsymbol{\chi}|T)$ with respect to the shape parameters $\boldsymbol{\varphi}$ and the pose parameters $\boldsymbol{\chi}$.

Let $\Phi_{\boldsymbol{\chi}_0}(p, i) \in \mathbb{R}^{3\times 1}$ be the sub-matrix of Φ that influences the p-th vertex of $\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}}$ and is multiplied with the i-th shape parameter $\boldsymbol{\varphi}_i$ in $\boldsymbol{\varphi}$. Now, we define $\Theta_{\boldsymbol{\chi}}(p)[\cdot]$ to be the linear blend skinning transformation of vertex p, so that

$$\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p) = \Theta_{\boldsymbol{\chi}}(p)[\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}(p)], \tag{5.9}$$

$$\mathcal{N}_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p) = \Theta_{\boldsymbol{\chi}}(p)[\mathcal{N}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}(p)], \text{ and} \tag{5.10}$$

$$\Phi_{\boldsymbol{\chi}}(p, i) = \Theta_{\boldsymbol{\chi}}(p)[\Phi_{\boldsymbol{\chi}_0}(p, i)], \tag{5.11}$$

with $p \in [1:P]$ and $i \in [1:|\boldsymbol{\varphi}|]$. Note that $\Theta_{\boldsymbol{\chi}}(p)[\cdot]$ does not apply a translational offset to directional vectors such as $\mathcal{N}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}(p)$ or displacement vectors such as $\Phi_{\boldsymbol{\chi}_0}(p, i)$ but only to positional vectors such as $\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}(p)$.

The partial derivatives of the distance functions d_{point} and d_{normal} with respect to the i-th shape parameter $\boldsymbol{\varphi}_i$ are defined as

$$\frac{\partial d_{\text{point}}(p,q)}{\partial \boldsymbol{\varphi}_i} = 2\langle M(p) - T(q), \Phi_{\boldsymbol{\chi}}(p,i)\rangle, \text{ and} \tag{5.12}$$
$$\frac{\partial d_{\text{normal}}(p,q)}{\partial \boldsymbol{\varphi}_i} = 2\langle M(p) - T(q), N(p)\rangle \cdot \langle \Phi_{\boldsymbol{\chi}}(p,i), N(p)\rangle.$$

Analogously, the partial derivative with respect to $\boldsymbol{\chi}$ are

$$\frac{\partial d_{\text{point}}(p,q)}{\partial \boldsymbol{\chi}} = 2\langle M(p) - T(q), \mathcal{M}'_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p)\rangle, \text{ and} \tag{5.13}$$
$$\frac{\partial d_{\text{normal}}(p,q)}{\partial \boldsymbol{\chi}} = 2\langle M(p) - T(q), N(p)\rangle \cdot \langle \mathcal{M}'_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p), N(p)\rangle, \text{ with}$$
$$\mathcal{M}'_{\boldsymbol{\varphi},\boldsymbol{\chi}}(p) = \begin{cases} \boldsymbol{a}_{j(p)} \times \mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}(p), & j(p) \text{ is a revolute joint;} \\ \left\langle \boldsymbol{a}_{j(p)}, \mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}(p) \right\rangle \hat{\boldsymbol{a}}_{j(p)}, & \text{else} \end{cases} \quad \text{and} \tag{5.14}$$
$$\hat{\boldsymbol{a}}_{j(p)} = \frac{\boldsymbol{a}_{j(p)}}{\|\boldsymbol{a}_{j(p)}\|_2}. \tag{5.15}$$

Here, $\boldsymbol{a}_{j(p)}$ is the axis of the joint $j(p)$ that is influenced by the element in $\partial\boldsymbol{\chi}$ used for differentiation. However, if that joint does not directly influence $\mathcal{M}_{\boldsymbol{\varphi},\boldsymbol{\chi}_0}$, $\boldsymbol{a}_{j(p)}$ is 0. For details, we refer to Section 2.1.2.

Note that in contrast to numeric differentiation, analytic derivatives enable faster and more stable convergence. We repeat the process in an ICP fashion, where between two iterations, the correspondences are updated using the newly estimated parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\chi}$. We further speed up the overall optimization procedure by using a multi-scale approach, where we start with only a small number of correspondences and successively increase the number of correspondences until we use one correspondence for every point in T and for every vertex in M.

Finally, we want to note that our optimization procedure can be easily extended to consider several target point clouds to be jointly optimized against. More precisely, having K target point clouds $T_1, \ldots, T_K$, the objective is to estimate K pose parameter vectors $\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_K$, but one joint shape parameter vector $\boldsymbol{\varphi}$. In the optimization, the energy functional is defined as the sum $\sum_{k\in[1:K]} E(\boldsymbol{\varphi}, \boldsymbol{\chi}_k | T_k)$, see Equation (5.8). Our experiments show that using only $K = 2$ different depth images (one from the front of the body and one from the back) are already sufficient to obtain an accurate shape estimate, see Figure 5.5. This easy extension to multiple target point clouds is possible because our model has independent pose and shape parameters, see Section 5.2.1.

5.2.4 Evaluation

To evaluate the accuracy of our proposed method and to compare it with previous methods, we conducted similar experiments as reported in Weiss *et al.* [2011]. As for the test data, we considered the body shapes of six different persons of different size and gender (three males, three females), see also Figure 5.7. For each person, we recorded two depth images, one showing the front and the other the back of the body, see Figure 5.5. Furthermore, using a full-body laser scanner, we generated for each person a surface point cloud with a resolution of about 350 000 vertices. These scans serve as ground-truth (GT).

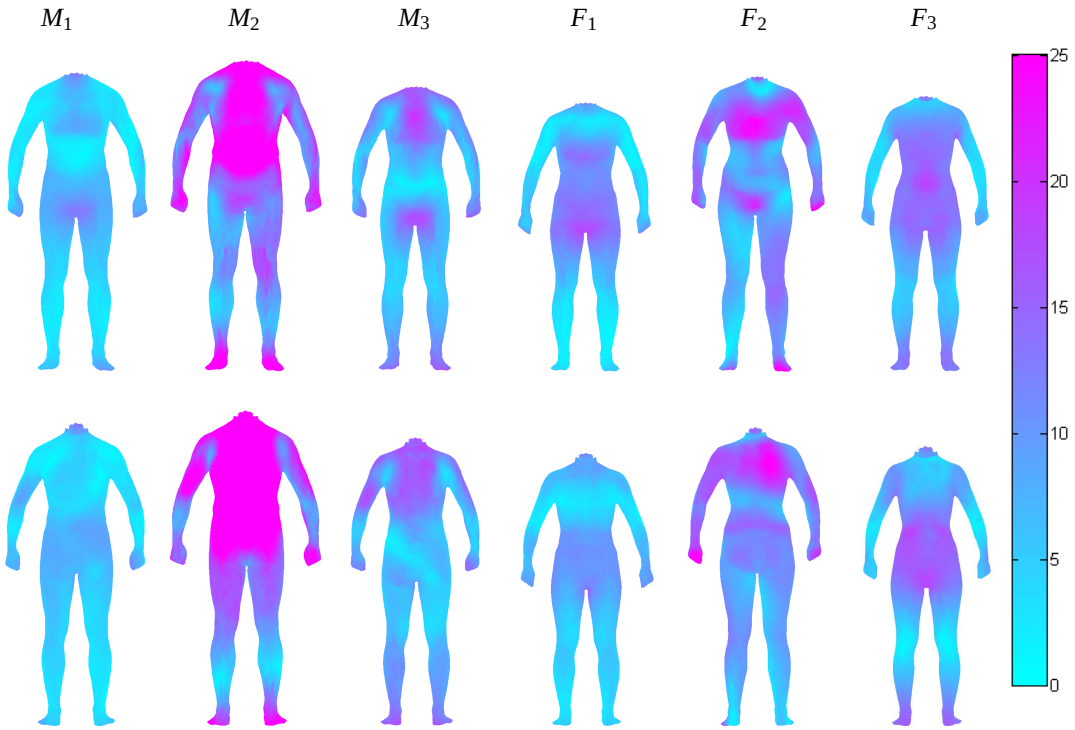

Figure 5.7. Vertex-to-vertex distances given in millimeters for three male (M_1–M_3) and three female (F_1–F_3) subjects. **(top):** Shown from the front and **(bottom):** from the back.

Now, let $\boldsymbol{\varphi}^*$ be the optimized shape parameter vector obtained by our algorithm when using the two depth images as target point clouds (the pose parameter vectors $\boldsymbol{\chi}_1$ and $\boldsymbol{\chi}_2$ are not used in the evaluation). Furthermore, to obtain a ground-truth shape, we use the same algorithm as before, however, this time using the laser scanner point cloud as target. Let $\boldsymbol{\varphi}^{\mathrm{GT}}$ denote the resulting optimized shape parameter vector. To compare the shapes resulting from $\boldsymbol{\varphi}^*$ and $\boldsymbol{\varphi}^{\mathrm{GT}}$, one needs to generate the corresponding meshes. However, to this end, one also requires pose parameters, and simply taking the standard pose parameter vector $\boldsymbol{\chi}_0$ is usually not the right choice, since the different shape parameters may also have a substantial influence on the assumed pose. Therefore, we compensate for this effect by taking the standard pose for the laser scan shape and by suitably adjusting the pose parameters for the estimated shape. To this end, we again apply our optimization algorithm using $\mathcal{M}_{\boldsymbol{\varphi}^{\mathrm{GT}},\boldsymbol{\chi}_0}$ as target point cloud and only optimize over the pose parameter vector $\boldsymbol{\chi}$ leaving $\boldsymbol{\varphi} = \boldsymbol{\varphi}^*$ fixed. Let $\boldsymbol{\chi}^*$ denote the result. As for the final evaluation, we then compare the mesh $\mathcal{M}_{\boldsymbol{\varphi}^*,\boldsymbol{\chi}^*}$ (representing our shape estimation result) with $\mathcal{M}_{\boldsymbol{\varphi}^{\mathrm{GT}},\boldsymbol{\chi}_0}$ (representing the ground truth shape). Since vertex correspondences of these two meshes are trivial (based on the same index set $[1:P]$), one can directly compute the vertex-to-vertex Euclidean distances in the same way as Weiss *et al.* [2011].

The vertex-to-vertex distances are indicated in Figure 5.7, which also shows the mean, variance and maximum over these distances. For example, for the first male actor M_1, the mean average is 5.1 mm and the maximal distance is 14.1 mm. Overall, the achieved accuracies (in average 10.1 mm) are good and comparable to (in average 10.17 mm) reported in Weiss *et al.* [2011]. There are various reasons for inaccuracies. In particular, using only 13 of the most significant Eigenvec-

	M_1	M_2	M_3	F_1	F_2	F_3	∅
μ	5.1	18.7	9.1	6.8	11.4	9.2	10.1
σ	2.5	9.5	4.0	3.7	4.9	4.4	4.8
max	14.1	46.3	20.5	18.7	30.1	19.4	24.9

Table 5.1. Mean μ, standard deviation σ, and maximum max in millimeters over all vertices. The heads where removed from the error calculation because of their bad representation in the shape model.

tors in Equation (5.4) does not allow us to capture all shape nuances which may lead to higher errors, such as for the actors M_2 and F_2. In these cases, either similar shapes might be not spanned by the training data of the shape model or the 13-dimensional approximation of shape variations might be too coarse. Furthermore, note that the depth image resolution (which is roughly 20 mm at the used distance of 2.6 m) as well as the mesh resolution (where neighboring vertices often have a distance of 20 mm) puts limits on the achievable accuracy. Nonetheless, overall good accuracy is achieved with a compact model.

Besides its accuracy, our approach has two further main benefits: efficiency and robustness. It only requires 50–60 seconds to estimate the shape parameter vector (and the two pose parameter vectors) from two target depth point clouds. This is substantially faster than the 3 900 seconds (65 minutes) reported by Weiss *et al.* [2011]. The running times were measured using a C++ implementation of our algorithm executed on an Intel Xeon CPU @ 3.10 GHz. Furthermore, jointly optimizing for shape and pose introduces a high degree of robustness and allows us to use only two different depth images to obtain accurate shape estimates. Actually, an additional experiment, where we used four target point clouds (using two additional depth images) only slightly improved the overall accuracies (from 10.1 mm when using two poses to 8.4 mm when using four poses). Besides implementation issues, these substantial improvements in running time and robustness are the result of using a relatively small number of optimization parameters, reverting to reliable 3D correspondences, using a more effective parametrization of the body model, and combining point and plane constraints.

5.3 Personalized Depth Tracker

As discussed in Section 5.1.3, the tracker presented in Baak *et al.* [2011] combines a generative with a discriminative approach. The discriminative tracker finds closest poses in a database, but that database is specific to an actor of a certain body shape. If the shape of the tracked person does not match the shape of the actor used to generate the database, the retrieved poses might not match. Also, the generative tracker employed by Baak *et al.*, uses a fixed body model that is not adapted to the tracked person. If now a person with a different shape has to be tracked, the local optimization might not find an optimal solution to fit the model into the point cloud of the depth image. In particular, this becomes evident if the the person is smaller than the model used by the generative tracker. Here, the tracker tries to squeeze the large model into the small point cloud, which results in strong pose errors.

To overcome some of these limitations, Baak *et al.* propose a scaling of the input point cloud along the axes of the depth image. While this works for actors with similar body proportions, such an approach fails if the proportions differ. Here, an example are the actors F_1 and F_2, where the arms

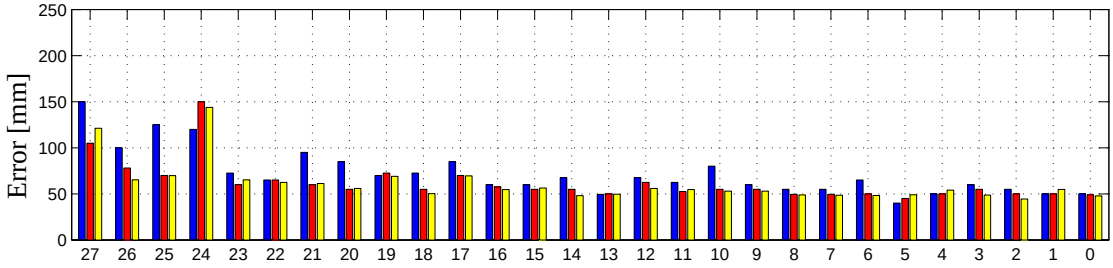

Figure 5.8. Average tracking error of sequences 0 to 27 of the dataset provided by Ganapathi *et al.* [2010]. The sequences were tracked using the tracker proposed by Ganapathi *et al.* [2010] **(blue)**, by Baak *et al.* [2011] **(red)**, and our proposed tracker **(yellow)**.

of F_1 are longer than the ones of F_2 when compared to the overall body size.

In our approach we suggest a different strategy by recomputing the entire set of poses in the database using the estimated personalized mesh. The database needs to be computed only once for each actor, which takes around 12 minutes for 50 000 poses using unoptimized code. An efficient GPU implementation would yield further speedups. Furthermore, we also replace the model in the generative tracker. The resulting personalized depth tracker captures even fast and complex body poses (jumping jack, sitting down) reliably and in real-time, see Figure 5.1 and also the accompanying video of Helten *et al.* [2013a] for some qualitative results. In the following, we will give some quantitative results with comparison to other approaches.

5.3.1 Evaluation on the Standford Dataset

In a first experiment, we compare our personalized tracker to previous approaches based on the dataset and error metrics described in Ganapathi *et al.* [2010]. The results of this evaluation are depicted in Figure 5.8. One can see that our tracker gives at least comparable results to the previous approaches presented by Ganapathi *et al.* [2010] and by Baak *et al.* [2011] and exceeds the results of the previous approaches in many cases. Please note that for this evaluation marker positions of markers attached to the actor's body are predicted and compared to ground truth marker positions obtained with an optical marker based mocap system. We think that this way of evaluating the tracking accuracy is not well suited for the specific requirements in home consumer scenarios. For example, in some reconstruction scenarios one is only interested in reconstructing the joint positions of the user, as it is done for example in many Kinect applications. On the other hand, when it comes to augmented reality scenarios, such as virtual try-on applications, one is rather interested in tightly approximating the depth image of the user to get a well fitting overlay of simulated objects such as cloths. In order to address these two evaluation aspects, we recorded a dataset with ground truth tracking results.

5.3.2 Our Evaluation Dataset

For our evaluation, we recorded a dataset Helten *et al.* [2013b] using both a Microsoft Kinect as well as a Phasespace active marker-based mocap system simultaneously. It comprises various kinds of motion performed by five actors (three male: M_1, M_2, and M_3 and two female: F_1 and

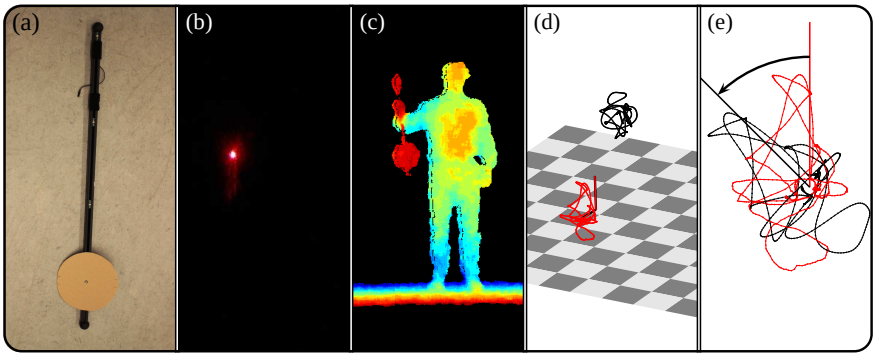

Figure 5.9. (a): Modified calibration wand with a cardboard disc around one marker. **(b):** Illuminated marker shown in an image from the RGB-camera of the Kinect. **(c):** Cardboard disk is clearly visible in the Kinect's depth image. **(d):** Reconstructed marker trajectories from Kinect **(red)** and optical mocap system **(black)**. **(e):** Estimation of the rotational offset between both trajectories after centering at their mean.

Difficulty	Description
D_1	Slow arm rotations, leg rotations, bending of upper body
D_2	Simple arm and leg motions, and grabbing
D_3	Punching, kicking, fast arm motion, and jumping
D_4	Sitting on the floor, rotating on the spot, and walking in circles

Table 5.2. Description of the four difficulties D_1–D_4 from the evaluation dataset.

F_2). The body models for each actor were estimated with the method from Section 5.2. We defined four groups of motions of different difficulties D. They range from easy to track motion sequences (D_1), simple arm and leg motions (D_2), fast movements such a kicking and jumping (D_3), to very hard to track motions such as sitting down, walking in circles, or rotating in place (D_4). An overview over the four difficulties is shown in Table 5.2. In total we recorded a set of 40 sequences, 2 takes from every of the 4 difficulties performed by each of the 5 actors. We used half of the recorded motions to build the pose database of the tracker, which contains a total of 50 000 poses. The other half of the sequences is used for evaluation and is referred to as *evaluation dataset*. We use the notation *<actor><difficulty>* to refer to a specific sequence from the evaluation dataset, *e. g.* M_2D_4 refers to the sequence of difficulty D_4 performed by actor M_2.

Calibration. In order to make the tracking results from the depth trackers comparable to the ground truth data we need to calibrate the Kinect with respect to the marker-based system. Since the location of the Kinect camera is unknown a priori and the frame capturing of the Kinect cannot be externally synchronized, such a calibration consists of two parts, a temporal calibration and a spatial calibration. While the spatial calibration only needs to be done once, the temporal calibration must be done for every captured sequence. We perform the temporal calibration by calculating a space invariant but time varying feature for two corresponding trajectories from both the marker-based and the Kinect recording. The temporal offset is then determined by identifying the lag that maximizes the cross correlation of both features. In our case, it turned out that the absolute velocities of the trajectories are a robust feature for temporal calibration even under the presence of tracking errors. A suitable trajectory could, for instance, be the position of a joint or another well defined point over a period of time.

For spatial calibration of both the Kinect and the marker-based system, we use a calibration wand with a single active LED (see Figure 5.9 (a)). Here, the idea is to determine the trajectory of the marker using both recording devices, and to register the trajectories to each other. While the marker-based system provides the marker's trajectory in a straight forward way, we need some additional processing to obtain the trajectory from the Kinect. The Kinect records depth and video simultaneously, see Figure 5.9 (b) and (c), and both streams are calibrated relative to each other. We can thus get the LED trajectory from the Kinect by recording in a dark room, thresholding the intensity image to identify the pixel position of the LED, and extracting corresponding depth information from the depth channel. Using the intrinsic parameters of the Kinect, we calculate the 3D position of the marker from the 2D position and depth value. Figure 5.9 (d) shows a reconstructed marker trajectory (red) from Kinect footage. Now, we temporally align the trajectories with the method described above. The resulting trajectories are then aligned spatially by determining a rigid transform for point correspondences (Figure 5.9 (e)).

Joint Tracking Error. In a first experiment, we want to evaluate how accurate the various depth-based trackers capture the joint positions of an actor. To this end, we used the marker data from the phase space system to animate a kinematic skeleton using inverse kinematics. We consider the resulting joints positions as ground truth data for the evaluation. In the following we assume that the sequences of the trackers and the ground-truth data have been temporally and spatially aligned using the procedure described above.

Since all trackers use a slightly different set of joints, we select for each tracker a subset of 20 joints that are close to semantic positions in the body such as the lower back, the middle of the back, the upper back, the head, the shoulders, the elbows, the wrists, the hands, the hips, the knees, the ankles, and the feet. We now measure for every frame the distance between the tracked joints and the ground truth joints. Since the corresponding joints from the different trackers do not lie at the exact same positions, *i. e.* even in a reference pose, we need to normalize for an offset. Therefore, we calculate the average local displacement of the joint relative to the corresponding ground-truth joint, and subtract this offset from the position of the tracked joint. Here, local displacement means that we consider the 3D displacement vector within the local coordinate frame of the ground-truth joint.

The average errors—over all joints and frames of one sequence—for the various actors and sequences are shown in Figure 5.10. One can see that the tracker of the Kinect SDK performs worst with an average error of 95.8 millimeters over all sequences. The tracker presented by Baak *et al.* [2011] shows an average error of 82.6 millimeters over all sequences, while our tracker performs best with an error of 73.8 millimeters.

Surface Tracking Error. In a second experiment, we assess the quality of the tracker by quantifying how well the tracked mesh at each frame approximates the point cloud recorded by the Kinect, referred to as *surface tracking error*. To this end, we first calculate a so-called *distance map* for every frame of a tracked sequence, by determining for every foreground point in the depth image of the Kinect the distance to the closest point on the mesh. Now, the straightforward way to compute a suitable surface tracking error would be to take the maximum distance from each distance map. Unfortunately, it turns out that the maximum is very unstable due to noise in the depth image and inaccuracies of the background subtraction. Here, a quantile value is better suited since

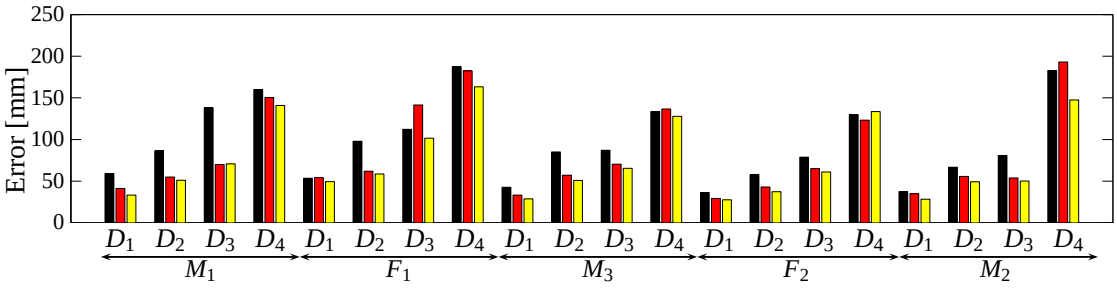

Figure 5.10. Average joint tracking error in millimeters for each sequence from the evaluation dataset that were tracked by the tracker of the Kinect SDK **(black)**, Baak *et al.* **(red)**, and our tracker **(yellow)**.

it filters out influences of noise. We tested several quantiles and it turned out that a 97%-quantile is a good compromise between robustness to outliers and responsiveness to tracking errors. Please note that since the Kinect SDK does not provide a tracked mesh, we cannot calculate this error for the tracker of the Kinect SDK.

Figure 5.11 (top) shows the surface tracking error over sequence F_1D_1. The red curve represents the error of the tracker by Baak *et al.* [2011] while the yellow curve is the result of our personalized tracker. The black vertical line at 22.7 seconds indicates a point in time where the surface tracking error of Baak *et al.* is significantly higher than that of our tracker. Figure 5.11(b)–(f) shows that this corresponds to a notable tracking error. In the middle, Figure 5.11 (b) displays the depth image recorded by the Kinect. In the distance map, cyan colors depict small distances around 0 millimeters while magenta colors represent high distance values of 25 millimeters and up. On the right, Figure 5.11 (c) and (d) shows the distance map (left) and the tracked mesh of their tracker, Figure 5.11 (e) and (f) depicts the distance map and the tracked mesh of our tracker. Our tracker tracks the right arm of actor F_1 correctly while it was merged with the upper body by the tracker of Baak *et al.* .

Table 5.3 lists the average surface tracking errors of the different sequences, actors and trackers. Our tracker performs significantly better than the tracker of Baak *et al.* [2011]. Especially sequence M_2D_4—which is one of the hardest sequences—is tracked considerably better by our tracker (average error of 110 mm) than by the tracker by Baak *et al.* (average error of 153 mm) Of course our tracker also has limitations, *e. g.*, when the actor does not face the camera (as in sequences of difficulty D_4) or when parts of the body are occluded or outside of the recording volume of the Kinect—which occasionally happens during all sequences. While we cannot do anything about the later source of errors, in Chapter 6, we will present an approach to deal with occlusions and the difficult to track non-frontal poses.

5.4 Conclusions

In this chapter, we presented a personalized real-time tracker of human body poses from single depth images that is more accurate than related approaches from the literature. Key to its success is personalization. We developed a new approach to estimate the personalized shape of an actor based on a parametric body model, which is much faster and more accurate than previous methods. We also presented a new real-time pose tracker that exploits this model and automatically adjusts to every actor. In conjunction, these two contributions allow us to track both skeletal joint loca-

	D_1	D_2	D_3	D_4	∅
M_1	61 (66)	81 (84)	116 (139)	102 (138)	90 (106)
M_2	56 (54)	77 (84)	75 (71)	110 (153)	80 (91)
M_3	56 (59)	76 (88)	89 (104)	93 (108)	79 (90)
F_1	64 (74)	84 (102)	115 (172)	97 (129)	90 (119)
F_2	46 (49)	62 (66)	80 (82)	105 (117)	73 (79)

Table 5.3. Averaged surface tracking errors in millimeters for each sequence of the evaluation dataset that were tracked by our tracker. For comparison the error using the tracker proposed by Baak *et al.* is shown in parenthesis.

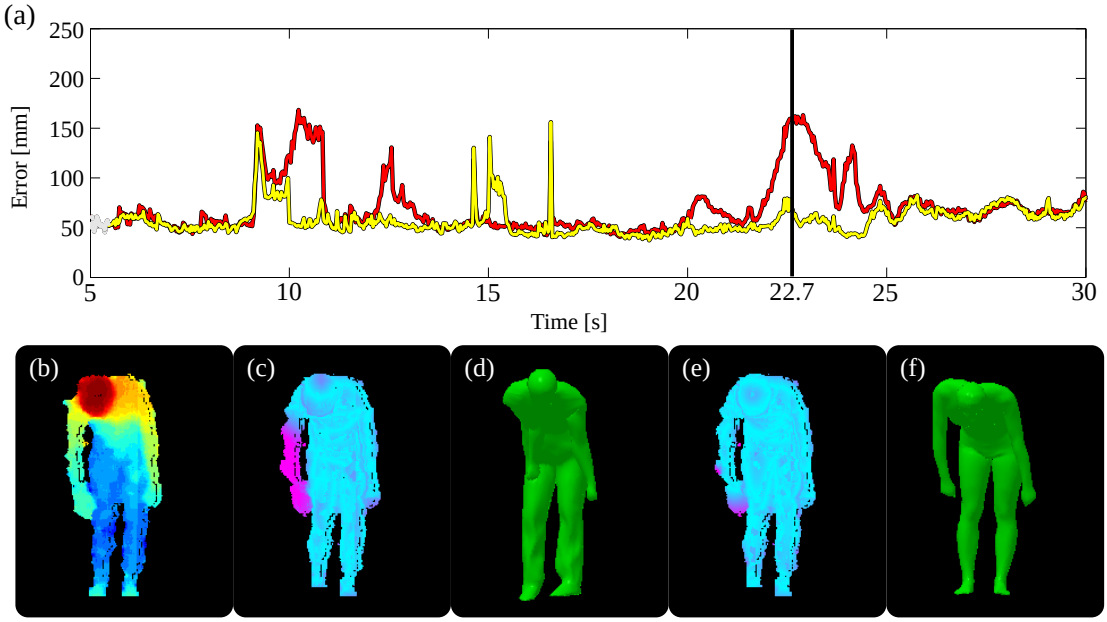

Figure 5.11. (a): Surface tracking error in millimeters for sequence F_1D_1 tracked by of Baak *et al.* **(red)** and our tracker **(yellow)**. **(b)–(f):** Status at 22.7 seconds. **(b):** Depth image at (red front, blue back). **(c):** Distance map of tracker of Baak *et al.*. **(d):** Tracked mesh for tracker of Baak *et al.*. **(e):** Distance map for our tracker. **(f):** Tracked mesh for our tracker.

tions as well as the shape of the body more accurately than with previous methods. We confirm this through extensive evaluations against ground truth on a comprehensive test dataset which is publicly available.

While our proposed approach shows significant improvements, it still fails in some challenging tracking situations such as when the person is not facing the camera or if parts of the body are occluded. These drawbacks are common for most depth-tracking approaches and are related to the limited information that monocular depth data provides. To this end, we will include additional sensor information that stabilizes the tracking. Here, inertial sensors become an interesting choice because there are not prone to occlusions and provide with orientations complementary information that only hardly can be obtained from depth images. In Chapter 6, we will further discuss this topic and present one possible solution to this issue.

Chapter 6

Real-time Motion Tracking by Fusing Sensor Modalities

As showed in Chapter 5, the tracking of full-body human motion constitutes an important strand of research in computer vision with many applications, *e. g.* in computer animation, sports, HCI or rehabilitation. Most of the trackers introduced so far can be classified into three families—discriminative approaches, generative approaches, and approaches combining both strategies. While discriminative trackers detect cues in the depth image and derive a pose hypothesis from them using a retrieval strategy, generative trackers optimize for the parameters of a human model to best explain the observed depth image. Combining discriminative and generative approaches, hybrid trackers have shown good results for fast motions in real-time scenarios, where tracked actors face the camera more or less frontally. However, noise in the depth data, and the ambiguous representation of human poses in depth images are still a challenge and often lead to tracking errors, even if all body parts are actually exposed to the camera. In addition, if large parts of the body are occluded from view, tracking of the full pose is not possible. Using multiple depth cameras can partially remedy the problem (see *e. g.* Ye *et al.* [2012]), but does not eradicate occlusion problems, and is not always practical in home user scenarios. Depth data alone may thus not be sufficient to capture poses accurately in such challenging scenarios.

In this chapter, we show that fusing a depth tracker with an additional sensor modality, which provides information complementary to the 2.5D depth video, can overcome these limitations. In particular, we use the orientation data obtained from a sparse set of inexpensive inertial measurement devices fixed to the arms, legs, the trunk, and the head of the tracked person. Inertial sensor units can nowadays be mass produced at low cost and can be found in almost any mobile device. We include this additional information as stabilizing evidence in a hybrid tracker that combines generative and discriminative pose computation. Our approach enables us to track fast and dynamic motions, including non-frontal poses and poses with significant self-occlusions, accurately and in real-time.

Contributions. Our method is the first to adaptively fuse inertial and depth information in a combined generative and discriminative monocular pose estimation framework. To enable this, we contribute with a novel visibility model for determining which parts of the body are visible

to the depth camera. This model tells what data modality is reliable and can be used to infer the pose, and enables us to more robustly infer global body orientation even in challenging poses. Our second contribution is a generative tracker that fuses depth and inertial cues depending on body part visibility, and finds pose parameters via optimization. As a third contribution, we introduce two separate retrieval schemes for handling depth and inertial cues for retrieving database poses during discriminative tracking. The final pose is found in a late fusion step which uses the results of both trackers mentioned above. We evaluate our proposed tracker on an extensive dataset including calibrated depth images, inertial sensor data, as well as ground-truth data obtained with a traditional marker-based mocap system. We also show qualitatively and quantitatively that it accurately captures poses even under stark occlusion where other trackers fail. The contributions discussed in this chapter have been published in Helten *et al.* [2013d]. For this reason we closely follow the explanation therein.

Organization. We start with discussing typical challenges that stem from the limited information provided by monocular depth images in Section 6.1. Then, in Section 6.2, we introduce the visibility model which provides important information to the other parts of our tracking framework. In Section 6.3, we describe our contributions to the generative tracker, while, in Section 6.4, we elaborate on the changes made to the discriminative tracker. How the information of the different components are fused into a final pose hypothesis is described in Section 6.5. The evaluation of our tracker with respect to previous approaches is described in Section 6.6. Finally, we conclude and give an outlook in Section 6.7.

6.1 Expressiveness of Depth Data

In Chapter 5, we addressed one of the challenges for current depth tracking approaches that stem from the level of accuracy of the underlying model that is used. We showed that accurate approximations of the person to track can be achieved using only two depth images as input. Here, we want to discuss two additional challenges to current depth tracking approaches that stem from lack of expressiveness of depth data: rotational ambiguities and occlusions. For an introduction into state-of-the-art depth tracking approaches we refer to Section 5.1.

6.1.1 Rotational Ambiguities

Depth data contains rich information about the relative location of objects which enables easy background subtraction compared to vision based approaches on intensity images. However, depth images reveal only little information about the surface structure and no color information at all. This makes it hard to determine the correct orientation of rotational symmetric objects, such as the body extremities. Since most depth trackers only depend on very simplistic underlying body models with isotropic extremities (Knoop *et al.* [2009]; Friborg *et al.* [2010]; Ganapathi *et al.* [2012]; Wei *et al.* [2012]) or even graphs (Pekelny and Gotsman [2008]; Salzmann and Urtasun [2010]; Zhu *et al.* [2010]; Girshick *et al.* [2011]; Shotton *et al.* [2011]; Ye *et al.* [2011]; Taylor *et al.* [2012]) that do not have any volume at all, they can simply ignore the aforementioned problem. However, these trackers also do not provide any pose information about the twist of the arms or the legs. In contrast, trackers that use complex triangle meshes for defining the surface of the body (Baak *et al.*

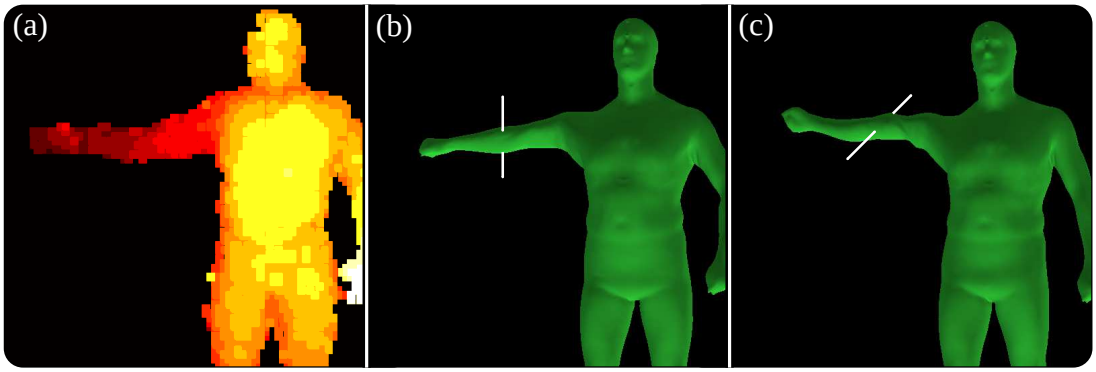

Figure 6.1. Rotational ambiguities of depth data. **(a):** Input depth image. **(b):** One typical output from a generative pose estimation procedure. Note that the axis of the elbow joint is vertical. **(c):** Another possible output, the axis of the elbow joint is now horizontal.

[2011]; Ye *et al.* [2012]; Ganapathi *et al.* [2010]) should not ignore rotational ambiguities. In particular, for these approaches the used generative tracker might converge to different results depending on its initialization.

An example can be seen in Figure 6.1. Here, the depth image shown in Figure 6.1 (a) reveals only little information on how the arm is oriented. Two possible solutions of a generative tracker are depicted in in Figure 6.1 (b) and (c). The difference between both solutions lies in the twist of the arm. While in Figure 6.1 (b) the axis of the right elbow joint is oriented vertically, it is oriented horizontally in Figure 6.1 (c). In this example, the latter would semantically be the correct pose estimation result. At first glance this might not have huge impact on the overall performance of the tracker. However, a erroneously tracked pose might serve as initialization for the next frame. Lets consider the scenario that the tracked person bends her arm with the forearm pointing upwards. While this is a straight-forward task for the generative tracker initialized with the pose shown in Figure 6.1 (c), a local optimization starting with the pose shown in Figure 6.1 (b) is more likely to get stuck in a local minimum. Unfortunately, none of the presented trackers employs methods to prevent this. While pure generative trackers are likely to fail in such situations and may not be able to proceed, discriminative trackers completely avoid this issue by tracking each frame independently and not relying on local optimization. In contrast, hybrid approaches, such as presented in Baak *et al.* [2011]; Wei *et al.* [2012], detect the failure of their generative tracker and reinitialize it using pose estimations of their discriminative tracker.

Similar challenges are also faced in other tracking fields as *e. g.* marker-less motion capture. Here, so called silhouette-based trackers that estimate the pose of the person from multiple, binary (foreground vs. background) images, suffer from the same challenge being unable to determine the correct orientation of the extremities of the person. One approach to tackle this was presented in Pons-Moll *et al.* [2010], where the authors included information from another sensor modality to correctly detect the orientation of the extremities independent from ambiguous depth information. In particular, their approach relies on orientation data obtained from five inertial sensors attached to the lower legs, forearms and the trunk of the person. By including the measured orientations into the energy function of their generative approach, tracking errors in rotationally symmetric limbs could be avoided. These ideas could be directly integrated into the energy function of a depth-based tracker. However, in case of tracking from monocular depth images another problem related to the lack of expressiveness is even more challenging: occlusions.

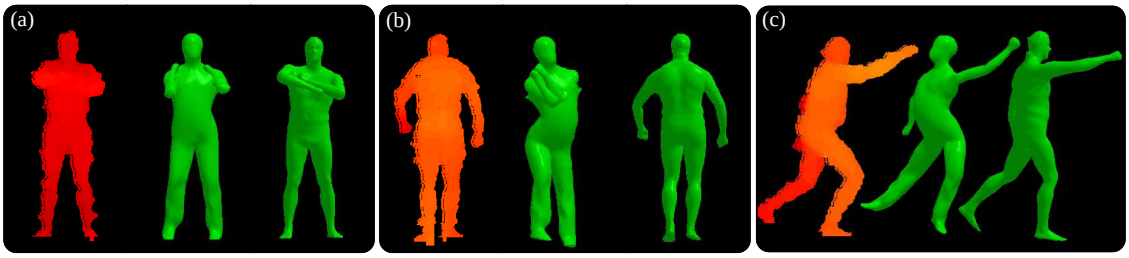

Figure 6.2. Three typical failure cases of a current real-time tracker combining generative and discriminative pose estimation (Baak *et al.* [2011]) (left: input depth image; middle: recovered pose of body model with catastrophic pose errors; right: significantly better result using our approach): **(a):** Occluded body parts, **(b):** non-frontal poses, **(c):** and both at the same time.

6.1.2 Occlusions

The third and by far greatest challenge for today's depth trackers are occlusions. Occlusions stem from the fundamental principle how depth images (and other optical data) are obtained. Light is reflected by some object and detected by some light sensitive sensor inside the camera. If light from an object, *e. g.* a body part, cannot reach the sensor of the camera because another object in between, the object is occluded. As a consequence, one cannot obtain any usable information about the occluded object. Present depth trackers deal with occlusions in various ways. Some trackers simply avoid this by requiring the tracked person to strike only poses where all body parts are clearly visible to the depth camera Baak *et al.* [2011]; Ganapathi *et al.* [2010]; Wei *et al.* [2012]. Such trackers often show undefined behavior if the requirements are not met, see Figure 6.2 for some representative failure cases. Some discriminative trackers allow for non frontal poses but do not give any pose hypothesis for non-visible parts (Zhu *et al.* [2010]; Shotton *et al.* [2011]; Taylor *et al.* [2012]; Wei *et al.* [2012]). In contrast, the approach presented in Girshick *et al.* [2011] uses a regression forest-based approach to learn the relative joint positions for a depth pixel based on depth values in its neighborhood. Calculating the density mean on a set of votes yields a hypothesis even for occluded joints. As most learning based approaches, this approach shows good results on poses close to the one used for learning and vice versa. In a pure generative setting, the approach proposed in Ganapathi *et al.* [2012] includes two additional constraints into the energy function to produce plausible results for occluded body parts. The first constraint prevents body parts from entering empty space, *i. e.* parts in the depth image where no foreground pixels were detected. The second constraint prevents body parts from inter-penetrating. However, without an actual measurement it is impossible to deduce the correct pose for occluded body parts.

We see two ways that could help tracking in difficult scenes. Firstly, occlusions could be reduced by dynamically moving the cameras during the recording of the scene. Secondly, occlusions could be handled by adding another input modality that does not depend on visual cues. As for the first approach, the authors in Ye *et al.* [2012] make use of three Kinect depth cameras that are carried by operators around a scene. At a given frame, the depth input of the three Kinects is then fused into one point cloud representation of the whole scene. Using a generative tracking approach, the poses of the persons are tracked by fitting a rigged surface mesh into the point cloud. While this approach shows good results even for multiple persons in close contact, the runtime of the approach is not real-time and the use of multiple Kinect cameras is not feasible in home user scenarios. Even when using multiple depth cameras, occlusions are difficult to prevent in many tracking scenarios.

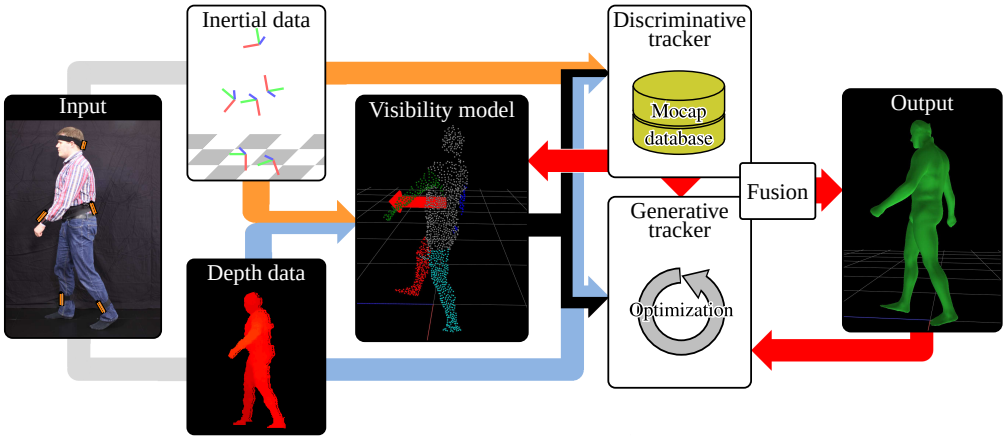

Figure 6.3. Overview of the components of our proposed tracker. The arrows indicate the data-flow between the components: inertial data **(orange)**, depth data **(blue)**, visibility data **(black)**, and pose data **(red)**.

As for the second approach, the fusion of different sensor modalities has become a successful approach for dealing with challenging tasks. An approach combining two complementary sensor types for full body human tracking in large areas was presented in Ziegler *et al.* [2011]. Here, densely placed inertial sensors, one placed on every limb of the body, provide an occlusion independent estimation of the persons body configuration using measured global orientations. Since inertial sensors cannot measure their position, this information is provided by a depth sensing laser system mounted to a robot accompanying the tracked person. Unfortunately, their approach does not include the rich depth information for supporting the tracking of the person's body configuration. Their approach rather solves two independent sub task, determining the local body configuration and estimating the global position of the person.

At this point, we want to take a second look at the approach presented in Pons-Moll *et al.* [2010], which we also discussed in Section 6.1.1. In this approach, the main intention of using inertial sensors in a classical marker-less tracking framework was to prevent erroneous tracking that stems from the ambiguous representation of body extremities in silhouette images. Another interesting side-effect is that the inertial sensors provide information about the limb orientations even in situations when the limbs are not visible to the camera. While in the presented scenario this effect was not important because multiple cameras enabled an almost occlusion free observation of the tracked person, this effect might be very important in monocular tracking approaches. In particular, many current depth-based trackers would benefit from additional information that does not depend on visual cues.

In the following, we take the state-of-the-art depth tracker proposed in Chapter 5, which is based on the tracker presented in Baak *et al.* [2011] as example. This tracker uses discriminative features detected in the depth data, so-called *geodesic extrema* $E_{\mathcal{I}}$, to query a database containing pre-recorded full-body poses. These poses are then used to initialize a generative tracker that optimizes skeletal pose parameters χ of a mesh-based human body model $\mathcal{M}_{\chi} \subseteq \mathbb{R}^3$ to best explain the 3D point cloud $\mathcal{M}_{\mathcal{I}} \subseteq \mathbb{R}^3$ of the observed depth image $\mathcal{I}$. In a late fusion step, the tracker decides between two pose hypotheses: one obtained using the database pose as initialization or one obtained that used the previously tracked poses as initialization. This tracker makes two assumptions: The person to be tracked is facing the depth camera and all body parts are visible

to the depth camera, which means it fails in difficult poses mentioned earlier (see Figure 6.2 for some examples). We overcome its limitations by modifying every step in the original algorithm to benefit from depth and inertial data together. In particular, we introduce a *visibility model* to decide what data modality is best used in each pose estimation step, and develop a discriminative tracker combining both data. We also empower generative tracking to use both data for reliable pose inference, and develop a new late fusion step using both modalities. See Figure 6.3 for an overview of our proposed tracker.

6.2 Visibility Model

Body Model Similar to Chapter 5, we use a body model comprising a surface mesh $\mathcal{M}_\mathcal{X}$ of 6 449 vertices, whose deformation is controlled by an embedded skeleton of 62 joints and 42 degrees of freedom via surface skinning, see also Section 2.1.2. The model is adapted to the actor utilizing the method described in Chapter 5 using a laser scan as target point cloud. However, also two depth images could be used, which makes it applicable in case of a home user scenario. Furthermore, let $\mathcal{B}_{\text{all}} := \{\text{larm}, \text{rarm}, \text{lleg}, \text{rleg}, \text{body}\}$ be a set of *body parts* representing the left and right arm, left and right leg and the rest of the body. Now, we define five disjoint subsets $\mathcal{M}_\mathcal{X}^b, b \in \mathcal{B}_{\text{all}}$ containing all vertices from $\mathcal{M}_\mathcal{X}$ belonging to body part b.

Sensors As depth camera we use a Microsoft Kinect running at 30 fps, but in Section 6.6 we also show that our approach works with time-of-flight camera data. As additional sensors, we use *inertial measurement units* (IMUs), which are able to determine their relative orientation with respect to a global coordinate system, irrespective of visibility from a camera. IMUs are nowadays manufactured cheaply and compactly, and integrated into many hand-held devices, such as smart phones and game consoles. In this chapter, we use six Xsens MTx IMUs, attached to the trunk (s_{root}), the forearms ($s_{\text{larm}}, s_{\text{rarm}}$), the lower legs ($s_{\text{lleg}}, s_{\text{rleg}}$), and the head ($s_{\text{head}}$), see Figure 6.6 (a). The sensor s_{root} gives us information about the global body orientation, while the sensors on arms and feet give cues about the configuration of the extremities. Finally, the head sensor is important to resolve some of the ambiguities in sparse inertial features. For instance, it helps us to discriminate upright from crouched full body poses. The sensors' orientations are described as the transformations from the sensors' local coordinate systems to a global coordinate system and are denoted by q_{root}, q_{larm}, q_{rarm}, q_{lleg}, q_{rleg}, and q_{head}. In our implementation, we use unit quaternions for representing these transformations, as they best suit our processing steps. In this chapter, we also use the virtual sensor concept introduced in Section 2.2.5. For clarity, we add $\mathcal{X}$ or $\mathcal{S}$ to the index, *e. g.* $q_{\mathcal{S},\text{root}}$ denotes the measured orientation of the real sensor attached to the trunk, while $q_{\mathcal{X},\text{root}}$ represents the readings of the virtual sensor for a given pose $\mathcal{X}$. Note, while the exact placement of the sensors relative to the bones is not so important, it needs to be roughly the same for corresponding real and virtual sensors. Furthermore, an orientation of a sensor at time t is denoted as $q_{\text{root}}(t)$. For further reading on the used sensors we refer to Chapter 2.

Our visibility model enables us to reliably detect global body pose and the visibility of body parts in the depth camera. This information is then used to establish reliable correspondences between the depth image and body model during generative tracking, even under occlusion. Furthermore, it enables us to decide whether inertial or depth data are most reliable for pose retrieval.

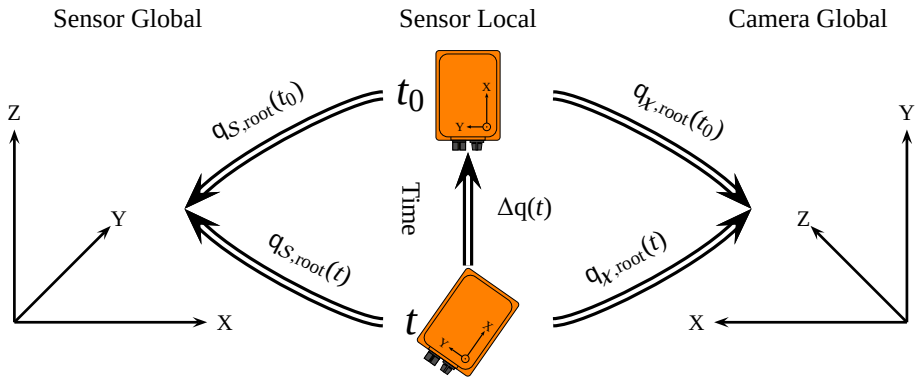

Figure 6.4. Relationship between the different IMU coordinate systems and orientations.

Global body position and orientation. In Baak *et al.* [2011], the authors use plane fitting to a heuristically chosen subset of depth data to compute body orientation, and compute translation from the depth centroid. Their approach fails if the person is not roughly facing the camera or body parts are occluding the torso. Inertial sensors are able to measure their orientation in space independent of occlusions and lack of data in the depth channel. We thus use the orientation of the sensor s_{root} to get a good estimate of the front direction $\boldsymbol{f}$ of the body within the global coordinate system of the camera, even in difficult non-frontal poses, as shown in Figure 6.5 (b). However, inertial sensors measure their orientation with respect to some global sensor coordinate system that in general is not identical to the global coordinate system of the camera, see also Figure 6.4. For that reason, we calculate the transformation $q_{\chi,root}(t)$ in a similar fashion as described in Pons-Moll *et al.* [2010] using relative transformations $\Delta q(t) := \overline{q_{\mathcal{S},root}(t_0)} \circ q_{\mathcal{S},root}(t)$ with respect to an initial orientation at time t_0. Here, $\bar{q}$ denotes the inverse transformation of q, while $q_2 \circ q_1$ expresses that transformation q_2 is executed after transformation q_1. The transformations $\overline{q_{\mathcal{S},root}(t_0)}$ and $q_{\mathcal{S},root}(t)$ can be directly obtained from the measurement of the sensor. The desired transformation from the coordinate system of the sensor to the global coordinate system of the camera at time t is now $q_{\chi,root}(t) = q_{\chi,root}(t_0) \circ \Delta q(t)$. Note that $q_{\chi,root}(t_0)$ cannot be measured. Instead, we calculate it using virtual sensors and an initial pose $\chi(t_0)$ at time t_0. For this first frame, we determine the front direction $\boldsymbol{f}(t_0)$ as described in Baak *et al.* [2011] and then use our tracker to compute $\chi(t_0)$. In all other frames, the front facing direction is defined as

$$\boldsymbol{f}(t) := q_{\chi,root}(t) \circ \overline{q_{\chi,root}(t_0)}[\boldsymbol{f}(t_0)]. \tag{6.1}$$

Here, q[$\boldsymbol{v}$] means that the transformation q is applied to the vector $\boldsymbol{v}$, Figure 6.5 (b).

Body part visibility. The second important information supplied by our visibility model is which parts of the model are visible from the depth camera. To infer body part visibility, we compute all vertices $\mathcal{V}_\chi \subseteq \mathcal{M}_\chi$ of the body mesh that the depth camera sees in pose χ. To this end we resort to rendering of the model and fast OpenGL visibility testing. Now, the *visibility* of a body part b is defined as

$$V_b := \frac{|\mathcal{M}_\chi^b \cap \mathcal{V}_\chi|}{|\mathcal{M}_\chi^b|}. \tag{6.2}$$

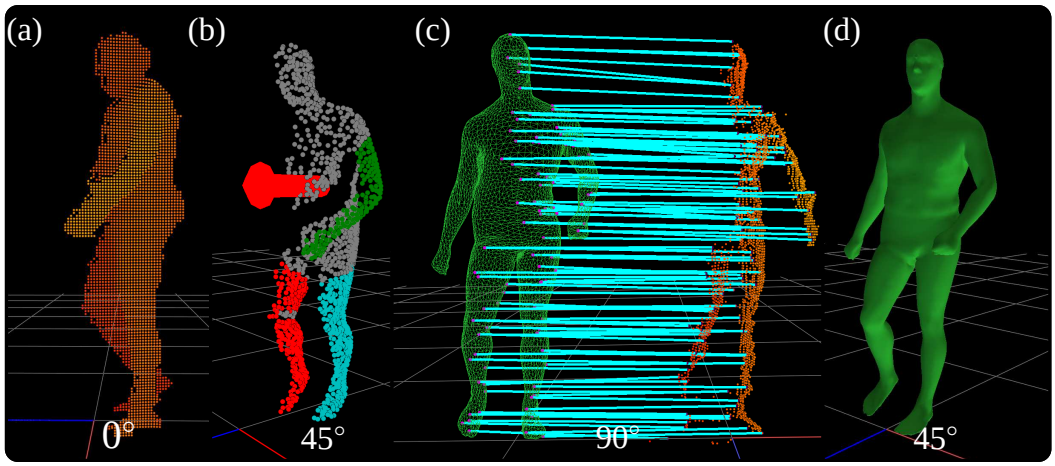

Figure 6.5. Tracking of frame at 5.0 s of sequence D_6 from our evaluation dataset. The views are rotated around the tracked person, where offset *w.r.t.* the depth camera is depicted at the bottom of each subfigure. **(a):** Input depth data. **(b):** Output of the visibility model. Note: the right arm is not visible. **(c):** Correspondences used by the generative tracker. Note: no correspondences with right arm. The pose parametrized mesh was moved to the left for better visibility. **(d):** Final fused pose.

The set of *visible body parts* is denoted as $\mathcal{B}_{\text{vis}} := \{b \in \mathcal{B}_{\text{all}} : V_b > \tau_3\}$. Note, that the accuracy of $\mathcal{B}_{\text{vis}}$ depends on $\mathcal{M}_{\mathcal{X}}$ resembling the actual pose assumed by the person in the depth image as closely as possible which is not known before pose estimation. For this reason, we choose the pose $\mathcal{X} = \mathcal{X}^{\text{DB}}$, obtained by the discriminative tracker which yields better results than using the pose $\mathcal{X}(t-1)$ from the previous step, (see Section 6.4). To account for its possible deviation from the "real" pose and to avoid false positives in the set $\mathcal{B}_{\text{vis}}$, we introduce the threshold $\tau_3 > 0$. In the tested scenarios, values of τ_3 up to 10% have shown a good trade-off between rejecting false positives and not rejecting too many body parts, that are actually visible.

In the rendering process also a *virtual* depth image $\mathcal{I}_{\mathcal{X}}$ is created, from which we calculate the first $M = 50$ geodesic extrema in the same way as for the real depth image $\mathcal{I}$, see Baak *et al.* [2011]. Finally, we denote the vertices that generated the depth points of the extrema with $\mathcal{V}_{\mathcal{X}}^{M}$.

6.3 Generative Pose Estimation

Similar to Baak *et al.* [2011], generative tracking optimizes skeletal pose parameters by minimizing the distance between corresponding points on the model and in the depth data. Baak *et al.* fix $\mathcal{V}_{\mathcal{X}}$ manually, and never update it during tracking. For every point in $\mathcal{V}_{\mathcal{X}}$ they find the closest point in the depth point cloud $\mathcal{M}_{\mathcal{I}}$, and minimize the sum of distances between model and data points by local optimization in the joint angles. Obviously, this leads to wrong correspondences if the person strikes a pose in which large parts of the body are occluded.

In our approach, we also use a local optimization scheme to find a pose $\mathcal{X}$ that best aligns the model $\mathcal{M}_{\mathcal{X}}$ to the point cloud $\mathcal{M}_{\mathcal{I}}$. In contrast to prior work, it also considers which parts of the body are visible and can actually contribute to explaining a good alignment on the depth image. Furthermore, we define subsets $\mathcal{X}_b, b \in \mathcal{B}_{\text{all}}$ of all pose parameters in $\mathcal{X}$ that affect the corresponding point sets $\mathcal{M}_{\mathcal{X}}^{b}$. We define the set of *active pose parameters* $\mathcal{X}_{\text{act}} := \bigcup_{b \in \mathcal{B}_{\text{vis}}} \mathcal{X}_b$.

Finally, the energy function is given as

$$d(\mathcal{M}_\mathcal{X}, \mathcal{M}_\mathcal{I}) := d_{\mathcal{M}_\mathcal{X} \to \mathcal{M}_\mathcal{I}} + d_{\mathcal{M}_\mathcal{I} \to \mathcal{M}_\mathcal{X}} \tag{6.3}$$

$$d_{\mathcal{M}_\mathcal{X} \to \mathcal{M}_\mathcal{I}} := \frac{1}{M} \sum_{v \in \mathcal{V}_\mathcal{X}^M} \min_{p \in \mathcal{M}_\mathcal{I}} \|p - v\|_2 \tag{6.4}$$

$$d_{\mathcal{M}_\mathcal{I} \to \mathcal{M}_\mathcal{X}} := \frac{1}{N} \sum_{e \in E_\mathcal{I}^N} \min_{v \in \mathcal{M}_\mathcal{X}} \|e - v\|_2. \tag{6.5}$$

Here, $E_\mathcal{I}^N$ represents the first $N = 50$ geodesic extrema in $\mathcal{I}$, while $\mathcal{V}_\mathcal{X}^M$ is a subset of $\mathcal{V}_\mathcal{X}$ containing $M = 50$ visible vertices, see Section 6.2 for details. A visualization for the resulting correspondences can be seen in Figure 6.5 (c). As opposed to Baak *et al.*, we minimize $d(\mathcal{M}_\mathcal{X}, \mathcal{M}_\mathcal{I})$ using a gradient descent solver similar to the one used in Stoll *et al.* [2011] and employ analytic derivatives.

6.4 Discriminative Pose Estimation

In hybrid tracking, discriminative tracking complements generative tracking by continuous reinitialization of pose optimization when generative tracking converges to an erroneous pose optimum (see also Section 6.5). We present a new discriminative pose estimation approach that retrieves poses from a database with 50 000 poses obtained from motion sequences recorded using a marker-based mocap system. It adaptively relies on depth features for pose look-up, and new inertial features, depending on visibility and thus reliability of each sensor type. In combination, this enables tracking of poses with strong occlusions, and it stabilizes pose estimation in front-facing poses.

Depth-based database lookup. In order to retrieve a pose $\boldsymbol{\chi}_\mathcal{I}^{\mathrm{DB}}$ matching the one in the depth image from the database, Baak *et al.* [2011] use geodesic extrema computed on the depth map as index. In their original work, they expect that the first five geodesic extrema $E_\mathcal{I}^5$ from the depth image $\mathcal{I}$ are roughly co-located with the positions of the body extrema (head, hands and feet). The geodesic extrema also need to be correctly labeled. Further on, the poses in their database are normalized *w.r.t.* to global body orientation which reduces the database size. As a consequence, also queries into the database need to be pose normalized. We use Baak *et al.*'s geodesic extrema for depth-based lookup, but use our more robust way for estimating $\boldsymbol{f}(t)$ for normalization, see Section 6.2. Our method thus fairs better even in poses where all geodesic extrema are found, but the pose is lateral to the camera.

Inertial-based database lookup. In poses where not all body extrema are visible, or where they are too close to the torso, the geodesic extrema become unreliable for database lookup. In such cases, we revert to IMU data, in particular their orientations relative to the coordinate system of the sensor $\mathrm{s}_{\mathrm{root}}$, see Figure 6.6 (a). Similar to the depth features based on geodesic extrema, these normalized orientations $\hat{\mathrm{q}}_b(t) := \overline{\mathrm{q}_{\mathrm{root}}(t)} \circ \mathrm{q}_b(t)$, $b \in \mathcal{B} = \{\mathrm{larm}, \mathrm{rarm}, \mathrm{lleg}, \mathrm{rleg}, \mathrm{head}\}$ are invariant to the tracked global orientation of the person but capture the relative orientation of various parts of the person's body. However, using these normalized orientations directly as index has one

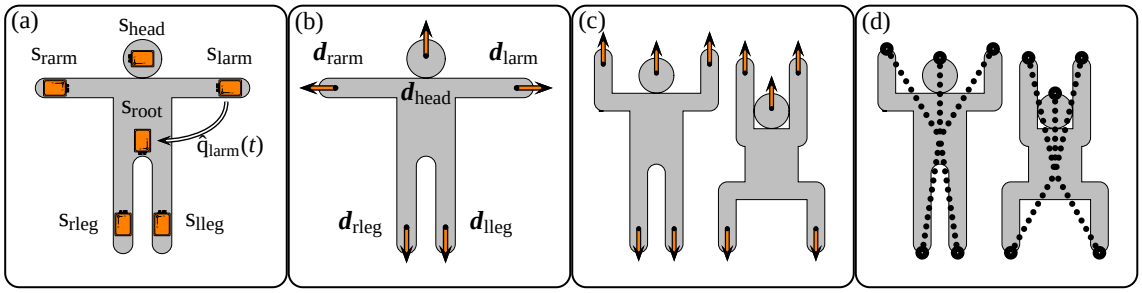

Figure 6.6. (a): Placement of the sensors on the body and normalized orientation *w.r.t.* s_{root}. **(b):** Body part directions used as inertial features for indexing the database. **(c):** Two poses that cannot be distinguished using inertial features. **(d):** The same two poses look different when using optical features.

disadvantage. This is because many orientation representations need special similarity metrics that are often incompatible to fast indexing structures, such as *k*-d trees. To this end, we use a vector $\hat{\boldsymbol{d}} \in \mathbb{R}^3$ that points in the direction of the bone of a body part, see Figure 6.6 (b). In our setup, these directions are co-aligned with the local X-axis of the sensor for all sensors except for the sensor s_{head}, where it is co-aligned with the local Y-axis. The normalized directions $\hat{\boldsymbol{d}}_b(t) := \hat{q}_b(t)[\boldsymbol{d}_b]$ are then stacked to serve as inertial-based query to the database. The retrieved pose is denoted as $\chi_{\mathcal{S}}^{\mathrm{DB}}$.

Selecting depth-based or inertial-based lookup. At first sight, it may seem that inertial features alone are sufficient to look up poses from the database, because they are independent from visibility issues. However, with our sparse set of six IMUs, the inertial data alone are often not discriminative enough to exactly characterize body poses. Some very different poses may induce the same inertial readings, and are thus ambiguous, see also Figure 6.6 (c). Of course, adding more IMUs to the body would remedy the problem but would starkly impair usablity and is not necessary as we show in the following. Geodesic extrema features are very accurate and discriminative of a pose, given that they are reliably found, which is not the case for all extrema in difficult non-frontal starkly occluded poses, see Figure 6.6 (d). Therefore, we introduce two reliability measures to assess the useability of depth-based features for retrieval, and use the inertial features only as fall-back modality for retrieval in case depth-based features cannot be trusted. We use the distances $\epsilon_i(t)$ of the geodesic extrema $i \in [1:5]$ at frame t *w.r.t.* the centroid of the point cloud which roughly lies at the center of the torso. For each end effector that distance does not change dramatically across poses in normal motion. When a geodesic extremum is not detected correctly, the computed distance $\epsilon_i(t)$ therefore typically differs significantly from $\bar{\epsilon}_i$. In practice, the distances can be obtained after the first pass of the modified Dijkstra's algorithm, presented in Baak *et al.* [2011]. This yields our first reliability measure

$$\epsilon(t) := \sum_{i=1}^{5} |\epsilon_i(t) - \bar{\epsilon}_i|, \tag{6.6}$$

The values of $\bar{\epsilon}_i$ for a specific actor are computed once from a short sequence of depth images in which geodesic extrema were detected reliably.

A second reliability measure is the difference between the purely depth-based computation of the global body pose similar to Baak *et al.* and the inertial sensors measured orientations. More

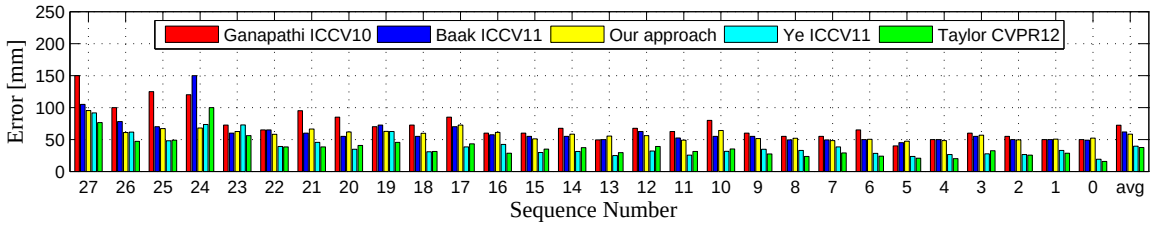

Figure 6.7. Evaluation on the Stanford dataset presented in Ganapathi *et al.* [2010]. **(red):** Ganapathi *et al.* [2010] **(blue):** Baak *et al.* [2011] **(yellow):** Our tracker. **(cyan):** Ye *et al.* [2011] (not real-time). **(green):** Taylor *et al.* [2012].

precisely, we use the measure

$$\Delta(t) := \sum_{b \in \mathcal{B}} \delta(\hat{\mathrm{q}}_{\chi_{\mathcal{I}}^{\mathrm{DB}},b}(t), \hat{\mathrm{q}}_{\mathcal{S},b}(t)). \tag{6.7}$$

$\delta = \cos^{-1} |\langle \cdot, \cdot \rangle|$ measures the difference between rotations that we represent as quaternions, where $\langle \cdot, \cdot \rangle$ is the dot product treating quaternions as 4D vectors. The final retrieved pose is computed as

$$\chi^{\mathrm{DB}} := \begin{cases} \chi_{\mathcal{I}}^{\mathrm{DB}}, & \text{if } \epsilon(t) < \tau_1 \wedge \Delta(t) < \tau_2 \\ \chi_{\mathcal{S}}^{\mathrm{DB}}, & \text{otherwise} \end{cases} . \tag{6.8}$$

We found experimentally that $\tau_1 = 1.15$ and $\tau_2 = 4.0$ are good values for all motion sequences we tested.

6.5 Final Pose Estimation

The final pose computed by our algorithm is found in a late fusion step. We are running two local pose optimizations (Section 6.3), one using the database pose χ^{DB} as initialization for the optimizer, and one using the pose from the last frame χ^{last} as initialization. Here, we are only optimizing for those parameters that are part of χ_{act}. The resulting optimized poses are called $\chi_{\mathrm{opt}}^{\mathrm{DB}}$ and $\chi_{opt}^{\mathrm{last}}$. From those two, we select the best pose according to Equation (6.3). Those parameters that are not part of χ_{act} are taken over from $\chi_{\mathcal{S}}^{\mathrm{DB}}$. This way, even if body parts were occluded or unreliably captured by the camera, we obtain a final result that is based on actual sensor measurements, and not only hypothesized from some form of prior.

6.6 Evaluation

The C++ implementation of our tracker runs at around 30 fps on a PC with a 2.4 GHz Intel Core i7-2760 QM CPU. We qualitatively and quantitatively evaluate it and its components on several data sets and compare to related methods from the literature.

We use a pose database with 50 000 poses. 44 000 were kindly provided by Baak *et al.* [2011]. We include 6 000 additional poses that we recorded along with the evaluation data set (Section 6.6.2). These poses show similar types of motion, but are not part of the evaluation set. The pose database is recomputed for each actor once to match his skeleton dimension.

6.6.1 Evaluation on Stanford Dataset

We evaluate our tracker on the 28 sequences of the Stanford data set from Ganapathi *et al.* [2010]. This data set was recorded with a SwissRanger SR4000 time-of-flight camera and provides ground-truth marker positions from a Vicon motion capture system. However, the data neither contain a pose parametrized model of the recorded person nor inertial sensor data. We therefore estimated the size of the recorded person using a deformable shape model from a set of isolated depth frames obtained from the dataset, see Weiss *et al.* [2011] for details. Using the mesh of the fitted model, we designed a suitable skeleton with the same topology as required by our pose parametrized model. We tracked the whole dataset using an IK-tracker and the provided ground-truth marker positions as constraints. The obtained pose parameters were used to compute virtual sensor readings. Note, that there are a lot of manual preprocessing steps involved to make our tracker run on this data set, and each step introduces errors that are not part of the other tested trackers' evaluation (we copied over their error bars from the respective papers). We now, tracked the dataset using the provided depth frames as well as the virtual sensor readings with our tracker and computed the error metric as described in Ganapathi *et al.* [2010], Figure 6.7.

Discussion We used the mean errors according to the error metric described by Ganapathi *et al.* [2010] to compare our tracker to the ones of Ganapathi *et al.* [2010], Baak *et al.* [2011], Ye *et al.* [2011] which is not a real-time tracker, and Taylor *et al.* [2012]. By mean error, our tracker performs better than Ganapathi *et al.* [2010] and Baak *et al.* [2011] on most sequences, and is close to the others on all data (see comments at end). However, our tracker shows its true advantage on sequences with more challenging motion, 24–27, of which only 24 shows notable non-frontal poses, and periods where parts of the body are completely invisible. Here, one can see that other trackers fail, as the errors of most trackers roughly double with respect to the mean error on other sequences. In contrast, our tracker shows an increase of only about 15%, as it continues to follow the motion throughout the sequence. Please note the mean errors are not the best metric to asses our tracker, but are the only values reported in all other papers. The absolute mean errors of our tracker are likely biased by an overhead stemming from the preprocessing mentioned above, and mask its significant improvement on occluded poses.

6.6.2 Evaluation Dataset

For more reliable testing of the performance of our tracker, we recorded a new dataset (Helten *et al.* [2013e]) containing a substantial fraction of challenging non-frontal poses and stark occlusions of body parts. Table 6.1 gives an overview over the six sequences of our evaluation dataset. While sequence D_1 contains comparably simple motions such as arm and leg rotations, the other five sequences are challenging for depth based trackers each in its own way. Sequence D_2 introduces considerable faster motions compared to Sequence D_1, including punching and kicking motions. However the motions are performed either by the arms or by the legs. In contrast, Sequence D_3 contains full body motions including jumping jacks, skiing motions, and squats. Especially the latter ones are interesting, because they induce inertial features that are almost equal over all phases of the motion. This stems from the fact that the arms and the lower legs do not change their orientation with respect to the trunk. In Sequence D_4 the arms touch the body at different locations which especially challenges the geodesic extrema-based database lookup as used by Baak *et al.*

Scene	Description	#Frames
D_1	Arm rotations, leg rotations, bending of upper body, and grabbing	1366
D_2	Punching, kicking, fast arm motion, and jumping	445
D_3	Jumping jacks, skiing, and squats	527
D_4	Arms at the hips, arms crossed, and hands behind head	930
D_5	Straight walking sidewards, and skiing sidewards	930
D_6	Circular walking, rotation on the spot, and moving arms behind the body	885

Table 6.1. Description of the six sequences from the evaluation dataset.

[2011]. In particular, this prevents correct geodesic extrema detection by introducing loops to the shape of the person. Sequence D_5 first introduces non-frontal poses including walking sidewards and skiing motions performed lateral *w.r.t.* the depth camera. Finally, Sequence D_6 completes our evaluation dataset by introducing walking in circles, rotating on the spot and selected occlusions of the arms.

For recording we used one Microsoft Kinect, six Xsens MTx IMUs as well as a PhaseSpace marker-based optical mocap system with 38 markers. The IMUs were strapped to the head, lower legs, the trunk, and forearms and are co-aligned with the assumed virtual sensors, see also Section 2.2.5. In the following, we assume that all data are temporally aligned and the Kinect data and the marker-based system are spatially aligned. We recorded 6 different sequences ($\mathrm{D}_1, \ldots, \mathrm{D}_6$) with varying difficulties including punching, kicking, rotating on the spot, sidewards and circular walking performed by one actor (See additional material for details). This totals in about 6 000 frames at 30 Hz. For all sequences we computed ground truth pose parameters and joint positions using the recorded marker positions and the same kinematic skeleton that we use in our tracker. For a qualitative evaluation of our tracker, also in comparison to previous approaches, we refer to Figure 6.2 and the accompanying video.

Discussion With this data, we quantitatively compare our tracker (hDB) to the Kinect SDK, as well as Baak *et al.* [2011]. We also quantitatively evaluate our tracker with only depth-based retrieval (dDB), and only inertial retrieval (iDB). To make results of very different trackers comparable, we introduce a new error measure based on joints. Since all trackers use a slightly different set of joints, we select for each tracker a subset of 16 joints that are close to semantic positions in the body such as the lower back, the middle of the back, the upper back, the head, the shoulders, the elbows, the wrists, the hips, the knees, and the ankles. Furthermore, as the corresponding joints from the different trackers do not lie at the exact same positions we need to normalize for this offset. We do this by calculating the average local displacement (*i. e.* local within the frame of the ground truth joint) of the joint relative to the corresponding ground-truth joint, and subtracting this offset from the position of the tracked joint, see also Section 5.3. The joint errors, for all sequences are depicted in Figure 6.9.

Figure 6.8 shows the average joint error for all tested trackers and algorithm variants on all 6 sequences. On the first four sequences which are easier and show no non-frontal poses, our final tracker (hDB) is among the best ones and, as expected, mostly comparable to Ganapathi's and Baak's methods. Importantly, it is always better than iDB and dDB. However, hDB outperforms all other approaches on the last two sequences, *e. g.* producing less than half the error (about 75 mm)

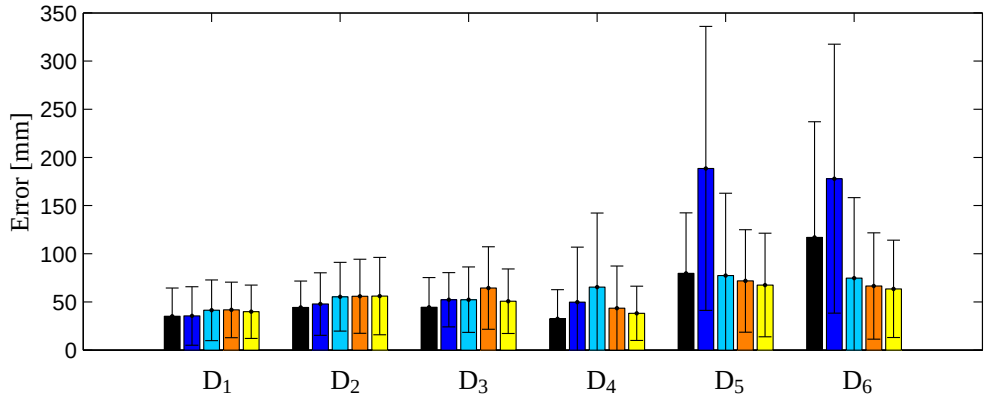

Figure 6.8. Average joint tracking error in millimeters for sequences $D_1, \ldots, D_6$ from our evaluation dataset, tracked with the joint tracker of the Kinect SDK **(black)**, Baak *et al.* **(blue)**, and our tracker with only depth-based DB lookup (dDB) **(light blue)**, only inertial-based DB lookup (iDB) **(orange)**, and the proposed combined DB lookup (hDB) **(yellow)**.

of Baak *et al.* [2011] with about 180 mm. The temporal error evolution of some representative joints in D_5 and D_6 are depicted in Figure 6.10 (a) for Kinect SDK, Baak *et al.*, and our algorithm. This clearly shows that our algorithm produces significantly lower errors than both others on certain spans of poses, which is masked in average error values. Finally, Figure 6.10 (b) shows the superiority of our tracker on selected time steps from that sequence, by visually comparing each result to ground truth joint locations (see video for more results). Error plots for the other joints and sequences can be found in the supplemental material. Here, we also included errors of our tracker, where one of the database-lookup strategies—either the depth-based or the inertial-based—was deactivated to show its impact on the overall performance. Our final tracker also performs consistently better than iDB and dDB illustrating the benefit of our fusion strategy. This is particularly evident in D_3 and D_4. Sequence D_3 contains squats, on which inertial-based feature lookup is ambiguous. D_4 contains motions where the arms touch the body at different locations. Here, the database lookup based on depth features fails.

6.7 Conclusions

In this chapter, we presented a hybrid method to track human full-body poses from a single depth camera and additional inertial sensors. Our algorithm runs in real-time and, in contrast to previous methods, captures the true body configuration even in difficult non-frontal poses and poses with partial and substantial visual occlusions. The core of the algorithm are new solutions for depth and inertial data fusion in a combined generative and discriminative tracker. In particular, we contributed with a visibility model that includes depth and inertial information to provide knowledge about what parts of the tracked person are visible in a given tracking situation. In particular, the visibility model provides information about which parts of the body are visible to the depth camera. This information is then used in the generative part of our proposed tracker to decide which information is more reliable for the motion reconstruction, the inertial data or the depth data. Furthermore, the visibility information is used to decide whether inertial or depth information is more reliable to look up a regularizing pose during discriminative tracking. We have demonstrated

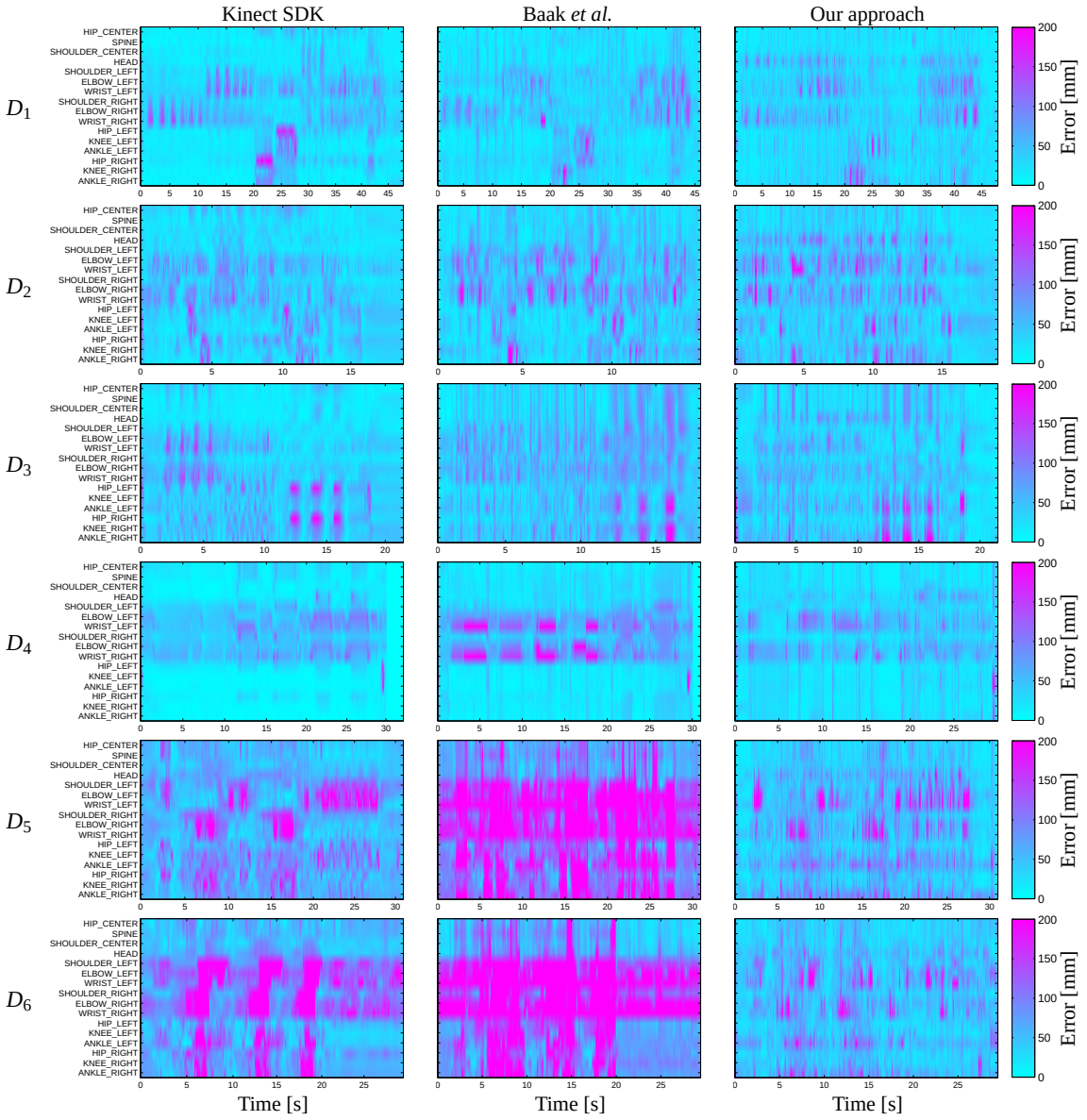

Figure 6.9. Joint Errors for all joints and all sequences $D_1, \ldots, D_6$ from the evaluation dataset. The three columns represent the three different trackers: **(left):** The Kinect SDK's joint tracker, **(middle):** the approach presented by Baak *et al.* [2011], and **(right):** our approach.

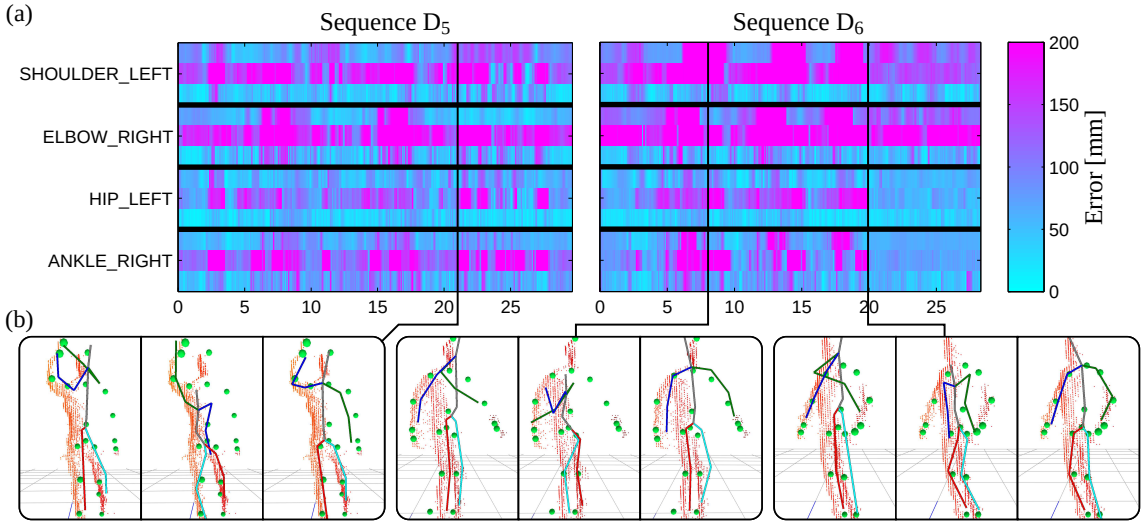

Figure 6.10. (a): Joint errors for selected joints over sequences D_5 and D_6 (time in seconds). Per joint there are three error rows: **(top)** Kinect SDK's tracker, **(middle)** Baak *et al.*, and **(bottom)** our approach. **(b):** Three challenging example poses from sequences D_5 and D_6. Input depth data, ground-thruth joint positions **(green dots)** and tracked **(skeleton)** shown from the side. Our approach **(right)** clearly outperforms the Kinect SDK's tracker **(left)**, and Baak *et al.*'s method **(middle).**

the performance of our tracker qualitatively and quantitatively on a large corpus of data that we provide to the community, and showed its clear advantages over other state-of-the-art methods.

Current limitations of our proposed tracking approach are for example the number of inertial sensors. While one sensor, used for estimating the global heading, improves the performance of our supposed tracker significantly, its full potential is only revealed using six inertial sensors. Here, the governing factor is the retrieval in our discriminative tracker. This is mainly because of the fact that we use an index solely based on inertial data if we detect that depth data is not sufficient for pose retrieval. One idea to reduce the number of sensors would be to introduce a hybrid retrieval approach that uses sparse inertial data in combination with depth cues for retrieval. In particular, using body part-based, detection algorithms such as proposed by Shotton *et al.* [2011] could be helpful. Another limitation is that inertial data is not yet included into the optimization scheme directly. Here, a prior similar as proposed by Pons-Moll *et al.* [2010] would improve the performance of our trackers for parts visible to the depth camera and one could better tackle tracking issues with rotational ambiguities of the extremities.

Chapter 7

Summary and Outlook

In this thesis, we have presented several techniques for processing and reconstructing human motion data that originates from different sensor modalities. A key aspect was that different sensor modalities provide different kinds of motion data and have specific advantages and disadvantages. For example, optical mocap systems provide motion data with the highest precision and descriptiveness. On the downside, they are expensive, difficult to setup and maintain, and pose constraints on the recording location and lighting conditions. Because of these properties, optical systems can be afforded only by a small number of people and are mainly used in high-budget movie and game productions. Inertial sensors, in contrast, are less expensive and pose considerably less constraints on the recording location. In particular, such sensors are completely independent from optical cues, which renders them immune to occlusion-based errors and lighting related problems. This makes them interesting for applications in sports science or medical home rehabilitation scenarios, which often take place in spacious and uncontrolled environments or even outside. Furthermore, because of their small size, inertial sensors have been employed in modern consumer electronics, such as smartphones or game consoles, as an additional input modality. Unfortunately, the data they provide is not as rich as the data obtained from optical systems. In addition, if used for full body recording of motion data, many inertial sensors must be placed on the body which renders them impracticable for home user applications such as full body control of video games or augmented reality applications. With depth sensing devices such as the Microsoft Kinect, an alternative sensor type has revolutionized the market. By providing real-time 3D geometry information in an inexpensive and easy to use manner, full-body human motion tracking can now be applied in home user environments. However, even state-of-the-art methods still suffer from various challenges such as ambiguities implied by the low resolution and noisiness of the data or missing information in the case of occluded body parts.

In this context, we have contributed in several aspects. As a first contribution, we systematically analyzed the expressiveness of several mid-level representations for the comparison of motion data originating from different sensor modalities. In a cross-modal scenario, we took a close look on features that can be derived from both inertial sensors and optical sensors. We showed that features based on orientation data that can be deduced from the measurements of inertial sensors is outperforming other representations such as, *e. g.*, local accelerations. We discussed the application of these techniques in the context of real-time full-body motion reconstruction.

As a second main contribution, we showed a practical application of these techniques in the con-

text of automatic classification of sports motions. We considered the scenario of trampoline motions, where the athlete has to perform a sequence of predefined jumps. This scenario was especially well-suited for the utilization of inertial sensors because of the highly dynamic and spacious character of trampoline motions that can hardly be captured using optical mocap devices such as marker-based systems. We contributed with a set of discriminative features based on inertial sensor data and an efficient DTW-based learning procedure based on motion templates. Also, we showed how different masking techniques improve the classification accuracy by enhancing or suppressing certain parts of the motion templates. In particular, the masking allows for controlling the sensitivity of motion templates to variations within one class.

As a third main contribution, we developed techniques to improve the performance of real-time depth-based human motion trackers as used in home consumer scenarios. We introduced a novel algorithm for estimating the shape of a person from only two sequentially taken depth images. In contrast to previous approaches, we used pure 3D-features and a combination of point and plane constraints to obtain comparable shape reconstruction results. Opposed to previous approaches the running time could be reduced from about one hour to about one minute. The estimated shape is important for many model-based depth tracker and is indispensable for augmented reality applications such as virtual try-on. To demonstrate this, we described how the estimated shape can be included in existing model-based tracking approaches. In comparison to previous tracking approaches, we could achieve an increased joint tracking accuracy as well as a better approximation of the depth image.

Finally, we studied one important drawback of current state-of-the-art depth tracking approaches that stem from the limited information provided by monocular depth data. In particular, we took a deeper look on how to deal with tracking errors that stem from the occlusion of body-parts. To tackle this problem, we proposed the usage of an additional sensor modality to provide complementary information that is not subject to occlusions. Inertial sensors have turned out to fulfill these requirements and provide rich information that can be utilized in several components of existing depth-based trackers to improve tracking results. As example we employed the tracker presented by Baak *et al.* [2011], which is a hybrid tracking approach fusing discriminative and generative tracking concepts. We showed that both concepts can be enriched by the data provided by inertial sensors to increase tracking performance. Especially, in tracking situations with non-frontal poses and/or occluded body-parts, we could achieve substantial improvements compared to other state-of-the-art depth trackers.

Outlook. We see several directions for further research. In general, dealing with different sensor modalities is an important direction of research not only in computer animation and robotics but also in the domains of medical rehabilitation and sport sciences. For example, in sport sciences, most experimental setups consist of various types of sensors such as optical marker-based systems, inertial sensors, force plates, high-speed cameras and EMG-sensors that measure muscle activity. However, in most cases these sensor modalities are considered independently without fusing them in an unified model. One first step, would be to combine modalities that provide similar data. An example are optical marker-based systems and high-speed cameras. While the frame rate of the former is restricted to about 120 Hz, the later achieves frame rates of more than 1 000 Hz. Even if high-speed cameras are only used in sparse numbers, they can increase the overall temporal resolution of the traditional marker-based system. Another example would be the usage of inertial sensors in combination with force plates or EMG-sensors to obtain a better impression which

forces act on a specific part of the body. In this context, elaborate body models, such as the OpenSim[1] show already promising results. On the downside, their main input modality are marker positions obtained by marker-based systems.

In most situations, optical systems are chosen because of their superior precision compared to other systems, but their specific requirements constrain their applicability to lab environments. Another drawback is that the placement of markers possibly restricts an athlete in the way he or she can perform the motion to be recorded. The application described in Chapter 4 was one example, where optical systems could not be used because of the high dynamics and required volume of trampoline motions. We showed that inertial sensors were much better suited in this context. However, one single sensor modality might in some scenarios not be sufficient to solve the task. One example would be if one not only wants to classify trampoline jumps but also wants to exactly reconstruct the motion for further analysis. In this context, the data provided by the inertial sensors alone is not sufficient. In particular, the global position of the athlete could not be reconstructed. Here, a small number of visual cues obtained from intensity or depth cameras may suffice to reconstruct the motion.

In the context of personalized tracking, a further direction of research is the real-time estimation of both shape and pose at the same time. This would render the pre-processing step for obtaining a personalized model not necessary anymore. Furthermore, it would enable other interesting applications such as real-time acquisition of appearance features of a person for identification purposes. To this end, a more robust finding of correspondences between model and depth data would be necessary, as for example used by Taylor *et al.* [2012]. Also, after approximating the pose and overall shape of a person, one could derive further interesting geometric information such as high detailed reconstruction of the body surface or identification of time variant surface features such as cloth folds. Finally, an exact approximation of the surface would enable the estimation of material or lighting parameters similar as in performance capture approaches.

A current limitation of our proposed combined depth/inertial tracking approach concerns the number of inertial sensors required. While already one additional inertial sensor, used for estimating the global heading, significantly improves the performance of our tracker, its full potential is only developed using six inertial sensors. One idea to reduce the number of sensors is to introduce a hybrid retrieval approach that uses sparse inertial data in combination with depth cues. In particular, using detection algorithms for body part detection such as proposed by Shotton *et al.* [2011] could be helpful. Another limitation is that inertial data is not yet included into the optimization scheme directly. A prior similar to the one proposed by Pons-Moll *et al.* [2010] may improve the performance of our tracker for parts that are visible to the depth camera and one could better tackle tracking issues with rotational ambiguities of the extremities.

[1] http://opensim.stanford.edu

Bibliography

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

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

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.

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.

Andreas Baak. *Retrieval-based Approaches for Tracking and Reconstructing Human Motions*. PhD thesis, MPI Informatik and Universität des Saarlandes, 2012.

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

P. Boissy, S. Choquette, M. Hamel, and N. Noury. User-based motion sensing and fuzzy logic for automated fall detection in older adults. *Telemedicine Journal and eHealth*, 13(6):683–694, 2007.

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.

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.

Alex Butler, Shahram Izadi, Otmar Hilliges, David Molyneaux, Steve Hodges, and David Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI)*, pages 1933–1936, 2012.

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

Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10:145–155, 1992.

James E. Davis, Marcin Grzegorzek, Bernd Jähne, Reinhard Koch, Andreas Kolb, Ramesh Raskar, and Christian Theobalt, editors. *Lecture Notes in Computer Science (8200): Time-of-Flight Imaging: Algorithms, Sensors and Applications*. Springer-Verlag Berlin Heidelberg, 2013.

Edilson de Aguiar, Christian Theobalt, Sebastian Thrun, and Hans-Peter Seidel. Automatic conversion of mesh animations into skeleton-based animations. *Computer Graphics Forum (Proceedings Eurographics)*, 27(2)(2):389–397, 2008.

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.

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

Jonathan Deutscher, Andrew Blake, and Ian Reid. Articulated body motion capture by annealed particle filtering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 126–133, 2000.

Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Transactions on Graphics (TOG)*, 22(3):409–416, 2003.

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.

Jürgen 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, 2009.

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)*, pages 755–762, 2010.

Varun Ganapathi, Christian Plagemann, Daphne Koller, and Sebastian Thrun. Real-time human pose tracking from range data. In *European Conference on Computer Vision (ECCV)*, pages 738–751, 2012.

Ross Girshick, Jamie 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.

Jason Harding, Colin G. Mackintosh, Allan G. Hahn, and Daniel A. James. Classification of aerial acrobatics in elite half-pipe snowboarding using body mounted inertial sensors. *The Engineering of Sport 7*, 2:447–456, 2008.

Nils Hasler, Carsten Stoll, Martin Sunkel, Bodo Rosenhahn, and Hans-Peter Seidel. A statistical model of human pose and body shape. *Computer Graphics Forum (Proceedings of Eurographics 2008)*, 2(28):337-346, 2009.

Thomas Helten, Heike Brock, Meinard Müller, and Hans-Peter Seidel. Classification of trampoline jumps using inertial sensors. *Sports Engineering*, 14:155–164, 2011.

Thomas Helten, Meinard Müller, Jochen Tautges, Andreas Weber, and Hans-Peter Seidel. Towards cross-modal comparison of human motion data. In *DAGM-Symposium*, pages 61–70, 2011.

Thomas Helten, Andreas Baak, Gaurav Bharaj, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Personalization and evaluation of a real-time depth-based full body tracker. In *Proceedings of the third joint 3DIM/3DPVT Conference (3DV)*, 2013.

Thomas Helten, Andreas Baak, Gaurav Bharaj, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Personalized depth tracker dataset. `http://resources.mpi-inf.mpg.de/PersonalizedDepthTracker`, 2013.

Thomas Helten, Andreas Baak, Meinard Müller, and Christian Theobalt. Full-body human motion capture from monocular depth images. In James Davis, Marcin Grzegorzek, Bernd Jähne, Reinhard Koch, Andreas Kolb, Ramesh Raskar, and Christian Theobalt, editors, *LNCS 8200, Time-of-Flight Imaging: Algorithms, Sensors and Applications*, pages 188–206. Springer-Verlag Berlin Heidelberg, 2013.

Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Real-time body tracking with one depth camera and inertial sensors. In *IEEE International Conference on Computer Vision (ICCV)*, 2013.

Thomas Helten, Andreas Baak, Gaurav Bharaj, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. Inertial depth tracker dataset. `http://resources.mpi-inf.mpg.de/InertialDepthTracker`, 2013.

Arjun Jain, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. Moviereshape: Tracking and reshaping of humans in videos. *ACM Transaction on Graphics (TOG)*, 29(5), 2010.

Doug L. James and Christopher D. Twigg. Skinning mesh animations. In *ACM SIGGRAPH*, pages 399–407, 2005.

Rudolph E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering*, 82(Series D):35–45, 1960.

Bob Kemp, Ad J. M. W. Janssen, and Bob van der Kamp. Body position can be monitored in 3D using miniature accelerometers and earth-magnetic field sensors. *Electroencephalography and Clinical Neurophysiology/Electromyography and Motor Control*, 109(6):484–488, 1998.

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.

Andreas Kolb, Erhardt Barth, Reinhard Koch, and Rasmus Larsen. Time-of-flight sensors in computer graphics. *Computer Graphics Forum (CGF)*, 29(1):141–159, 2009.

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.

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

Jihong Lee and Insoo Ha. Real-time motion capture for a human body using accelerometers. *Robotica*, 19(06):601–610, 2001.

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.

Miao Liao, Qing Zhang, Huamin Wang, Ruigang Yang, and Minglun Gong. Modeling deformable objects from a single depth camera. In *International Conference on Computer Vision (ICCV)*, pages 167–174, 2009.

Tao Liu, Yoshio Inoue, and Kyoko Shibata. Development of a wearable sensor system for quantitative gait analysis. *Measurement*, 42(7):978–988, 2009.

Yebin Liu, Carsten Stoll, Jürgen 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.

Henk J. Luinge and Peter H. Veltink. Measuring orientation of human body segments using miniature gyroscopes and accelerometers. *Medical and Biological Engineering and Computing*, 43(2):273–282, 2005.

Andrew Maimone and Henry Fuchs. Reducing interference between multiple structured light depth sensors using motion. In *IEEE Virtual Reality Short Papers and Posters (VRW)*, pages 51–54, 2012.

Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven Gortler, and Leonard McMillan. Image-based visual hulls. In *ACM SIGGRAPH*, pages 369–374, 2000.

Meinard Müller and Sebastian Ewert. Towards timbre-invariant audio features for harmony-based music. *IEEE Transactions on Audio, Speech, and Language Processing (TASLP)*, 2009.

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

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.

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`.

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

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

Yuji Ohgi, Hiroshi Ichikawa, and Chikara Miyaji. Microcomputer-based acceleration sensor device for swimming stroke monitoring. *JSME International Journal Series C, Mechanical Systems, Machine Elements and Manufacturing*, 45(4):960–966, 2002.

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

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.

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.

Gerard Pons-Moll, Andreas Baak, Jürgen 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.

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

Angelo Sabatini, Chiara Martelloni, Sergio Scapellato, and Filippo Cavallo. Assessment of walking features from foot inertial sensing. *IEEE Transactions on Biomedical Engineering*, 52(3):486–494, 2005.

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.

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, 2008.

Ken Shoemake. Animating rotation with quaternion curves. *ACM SIGGRAPH Computer Graphics*, 19(3):245–254, 1985.

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.

Ronit Slyper and Jessica Hodgins. Action capture with accelerometers. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2008.

Jonathan Starck and Adrian Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1387–1394, 2005.

Jonathan Starck and Adrian Hilton. Correspondence labelling for wide-timeframe free-form surface matching. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1–8, 2007.

Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.

Carsten Stoll, Nils Hasler, Jürgen 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.

Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernd Eberhardt. Motion reconstruction using sparse accelerometer data. *ACM Transactions on Graphics (TOG)*, 30(3), 2011.

Jochen Tautges. *Reconstruction of Human Motions Based on Low-Dimensional Control Signals*. Dissertation, Universität Bonn, 2012.

Jonathan Taylor, Jamie Shotton, Toby Sharp, and Andrew W. Fitzgibbon. The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (TOG)*, 27(3):1–9, 2008.

Robert Y. Wang and Jovan Popovic. Real-time hand-tracking with a color glove. *ACM Transactions on Graphics (TOG)*, 28(3), 2009.

Xiaolin Wei, Peizhao Zhang, and Jinxiang Chai. Accurate realtime full-body motion capture using a single depth camera. *ACM Transactions on Graphics (TOG)*, 31(6), 2012.

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.

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.

Genzhi Ye, Yebin Liu, Nils Hasler, Xiangyang Ji, Qionghai Dai, and Christian Theobalt. Performance capture of interacting characters with handheld Kinects. In *European Conference on Computer Vision (ECCV)*, pages 828–841, 2012.

Li Zhang, Brian Curless, and Steven M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 367–374, 2003.

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.

Jakob Ziegler, Henrik Kretzschmar, Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Accurate human motion capture in large areas by combining IMU- and laser-based people tracking. In *IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 86–91, 2011.