TECHNISCHE UNIVERSITÄT MÜNCHEN

Computer Aided Medical Procedures & Augmented Reality / I16

# Machine Learning for Human Motion Analysis and Gesture Recognition

Loren Arthur Schwarz

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Florian Matthes

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Nassir Navab
2. Prof. Robert Pless, PhD
   Washington University, USA

Die Dissertation wurde am 5. Januar 2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 4. Juni 2012 angenommen.

# Abstract

In this thesis, we investigate machine learning methods for human motion analysis. We introduce algorithms for human pose estimation and activity recognition that do not build upon conventional cameras and that can cope with noisy and incomplete input data. We propose methods that capture human movements using body-worn inertial sensors or using a depth camera. In a training phase, the measurements from these modalities are complemented with precise and complete movement information recorded with a camera-based motion capture system. A novel approach for learning models of human motion is introduced based on non-linear dimensionality reduction and regression that enables human pose estimation and simultaneous activity recognition, given only the inertial sensor or depth data. As an application scenario, we present a customizable method for medical gesture-based human-machine interaction that fills an existing gap in the operating room, as confirmed by surgeons.

# Zusammenfassung

Diese Arbeit befasst sich mit maschinellem Lernen zur Analyse menschlicher Bewegungen. Es werden Algorithmen zur Erkennung von Körperhaltungen und Aktivitäten vorgestellt, die auf klassische Kameras verzichten und mit unvollständigen und verrauschten Daten zurecht kommen. Die vorgestellten Verfahren benutzen am Körper getragene Inertialsensoren oder eine Tiefenkamera als Eingabemedium. In einer Trainingsphase werden die Messungen dieser Modalitäten um präzise Bewegungsdaten ergänzt, die mit einem Kamerasystem erfasst werden. Ein neuartiger Ansatz zum Lernen von Bewegungsmodellen aus diesen Daten wird vorgestellt, der auf nicht-linearer Dimensionsreduktion und Regression beruht und es ermöglicht, Körperhaltungen und Aktivitäten allein aus den Sensormessungen bzw. den Tiefenbildern zu erkennen. Als Anwendung wird eine Methode zur gestenbasierten medizinischen Mensch-Maschine-Interaktion vorgestellt, für die Chirurgen zufolge im Operationssaal deutlicher Bedarf besteht.

# Acknowledgments

The time working on this thesis and being a member of the Chair for Computer Aided Medical Procedures and Augmented Reality at Technische Universität München will remain in my memories as a very precious and pleasurable one. What I learned during this period includes many theoretical concepts, a scientific approach to work and experiments, how to write papers and how to teach students. But what I learned also goes far beyond these aspects, and this is mainly due to the exceptional supervision and leadership by Prof. Dr. Nassir Navab. His depth of scientific understanding, creativity and capability of motivating and encouraging people are a benchmark that I will take with me into my future.

I also owe thanks to many colleagues at the chair whom I spent a lot of time with, including enlightening scientific discussions, several ups and downs, night shifts before deadlines, last-minute preparations before student exercises, as well as lunch breaks and tea ceremonies. First and foremost, I would like to thank Diana Mateus who, with her uncomplicated and friendly spirit, provided so much valuable advice and help to me. Together with her, it was great fun working our way through highly non-linear manifolds of paper submissions, motion capture experiments and teaching activities. I would also like to thank Max Baust, Selen Atasoy, Olivier Pauly and Slobodan Ilic for all the fruitful discussions on mathematical concepts and their potential use. Particular thanks also to Ali Bigdelou for a very streamlined and effective period of working together, where our complementary skills brought us into a completely new field of application, probably to the surprise of both of us. Thanks to Tassilo Klein for many fun times and for helping me in and out of all kinds of motivational crises. Thanks to Jürgen Sotke for all the laughs we had during serious discussions about nuclear power plants, windows and apples, or Loriot and Prokofiew, just to name a few of the recurring topics.

Last but not least, I would like to express my gratitude to my parents, Larissa and Egil Schwarz, who enabled me to pursue this path and supported me throughout, and to Claudia, who was always by my side and who always had an open mind for exchanging spontaneous thoughts on geeky computer science stuff and inspired me with several unorthodox ideas.

# Contents

# Part I

# Introduction

Chapter 1

# Motivation

Human motion is an unremarkable part of everyday life and yet an exciting subject to study. Involving hundreds of bones, joints and muscles, the motion of the human body has fascinated physicians and scientists throughout the centuries. Apart from serving the purposes of transportation and labor, human motion is also used for communication and can thus convey lots of information, be it intentional or implicit. Analyzing human motion therefore can have various benefits, ranging from understanding the physical interaction of bones and muscles to interpreting human behavior and intentions.

While the static anatomical structure and the mechanical properties of muscles and bones have been largely explored by scientists such as Leonardo da Vinci (1452-1519) and Galileo Galilei (1564-1642), the complex processes behind human motion remained unclear, if not miraculous, until the 19th century. One of the first steps towards modern human motion analysis can be attributed to Eadweard Muybridge (1830-1904), a British photographer. Muybridge succeeded in capturing multiple pictures of moving persons in a fast sequence (Figure 1.1). This way, human motion was for the first time broken down into distinct, observable body poses. Technological advances of the 20th century, including the refinement of photography and video technology, have resulted in improved approaches for capturing human movements, allowing scientists to complete their understanding of the body in motion. The advent of the computer then brought the quest for human motion analysis techniques that can automatically extract body poses and other motion information from images and sensor signals. As a consequence, novel and challenging applications have appeared, ranging from character animation in the movie industry, through surveillance systems that monitor human actions, to gesture-based approaches for human-computer interaction.

## 1.1 Human Motion Analysis

This work deals with three aspects of human motion analysis: pose estimation, action recognition and gesture recognition. In human pose estimation (Section 1.1.1), the objective is to digitally capture the exact movements of a person in three-dimensional space over time. Activity recognition (Section 1.1.2) aims at inferring an abstract notion of what a person is doing at each instant. Gesture recognition (Section 1.1.3) is the process of detecting a set of movements that are typically used for human-computer interaction. Our approach to these fields

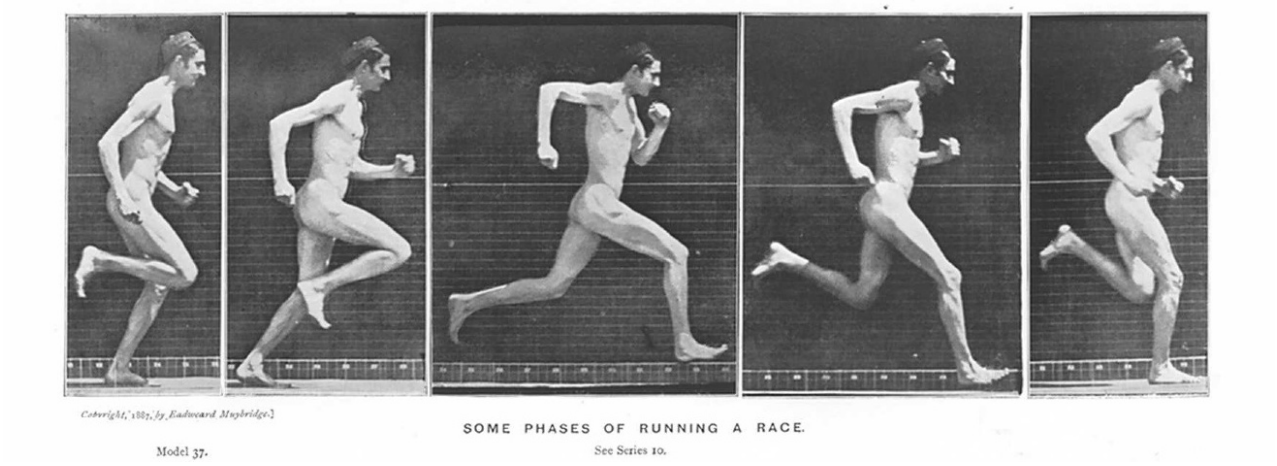



Figure 1.1: Photographs from "The Human Figure in Motion". Eadweard Muybridge, 1878.

is inspired by research in the computer vision community, where a general goal is to extract meaningful information from limited and noisy visual observations. We particularly investigate how to bring human motion analysis into scenarios where conventional camera-based systems face significant challenges. Consider as an example the operating room environment, where lighting is highly variable and occlusions by personnel and equipment disrupt the line of sight between cameras and persons to monitor. Long-term motion analysis in everyday environments is another example where classical cameras can hardly be used. In this work, we therefore focus on observations that come from novel depth cameras and inertial body-worn sensors (Section 1.2). To cope with the noisy and ambiguous data provided by these devices, we explore machine learning techniques that allow learning important characteristics of feasible human movements from training data (Section 1.3). The introduction concludes with an outline of existing and potential future applications for human motion analysis (Section 1.4).

### 1.1.1 Pose Estimation

Intuitively, a human pose is an instantaneous, spatial configuration of the body. As a pose can be characterized in an arbitrary level of detail, pose estimation methods typically use a human body model as a parametrizable representation of a real body. The complexity of such body models can vary strongly, depending on the particular application scenario (Figure 1.2). In many cases, a pure skeleton (or stick-figure) model is used consisting of limb segments that are connected through joints. Each joint can have one or more angles of flexion and a human pose is, in this case, fully determined by the angular configuration of all joints. Skeletal body models are sufficient if the articulated motion of the body is of main interest. In cases where the outer appearance of a human in motion is studied, volumetric models are more appropriate that include a representation of the deformable body surface. Throughout this work, we use a skeletal body model for its simplicity and flexibility (see Appendix A).

The problem of inferring a human pose from some type of observations is a classical parameter estimation problem. Given a set of observed quantities, such as image features or sensor measurements, and an assumed body model, the question is: Which parameters make the model fit best to the observations? It is clear that the complexity of this problem grows

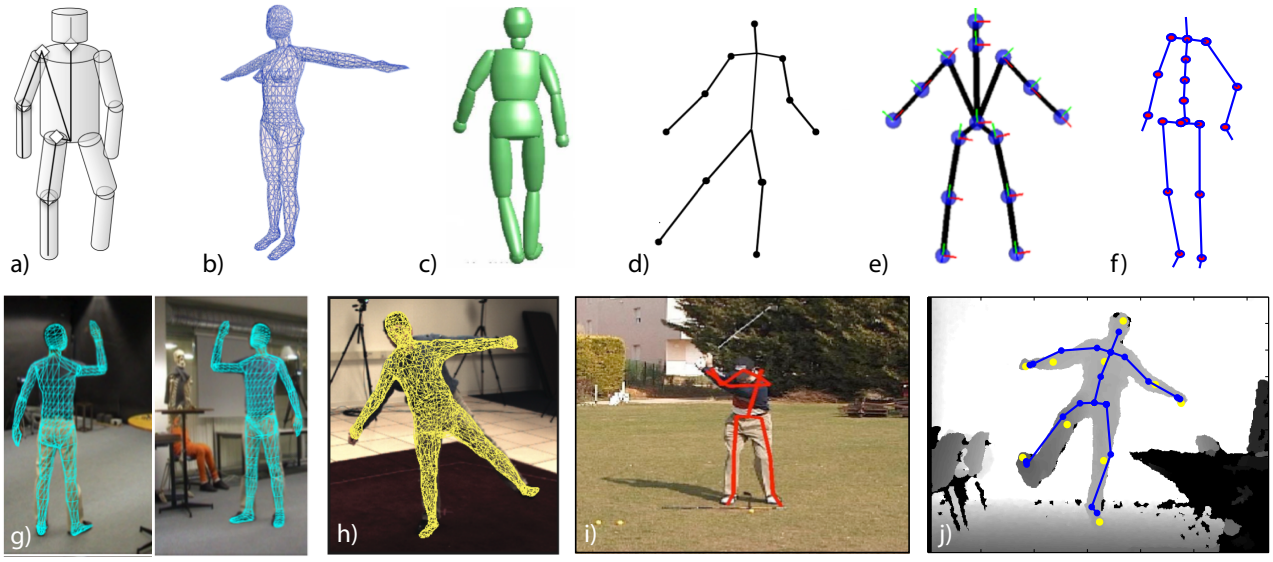



Figure 1.2: Human body models and pose estimation examples. a)-c) Volumetric human body models. d)-f) Skeleton or stick-figure human body models. g)-h) Pose estimation examples using volumetric body models. i)-j) Pose estimation examples using skeleton body models. Pictures from [86, 143, 41, 138, 12, 48, 45, 141].

with the number of parameters to estimate and with the level of noise in the observations. In terms of skeletal body models, pose estimation becomes an example of inverse kinematics. The guiding question is: Which set of joint angles allows a set of end effectors (e.g. hands, feet) to reach a set of given targets (e.g. image features)? Unfortunately, being an inverse problem, human pose estimation is ill-posed in many cases. This means that there might not always be a solution, or at least no unique solution, and that poses do not depend continuously on the observed features. The main reasons are the complexity of the human body and the fact that observations are often incomplete, noisy and ambiguous. As an example, consider a walking human that is seen from a side view. The perspective projection onto the 2D image plane causes depth cues to be lost and, depending on illumination, it can be difficult to tell which leg of the person is extended at a given moment. When other input modalities, such as inertial sensors, are used to capture human motion, there can be even less constraints that help estimating the human pose in 3D. As a result, finding the pose that best matches a given set of observations is challenging. Classical solution techniques, such as numerical optimization, can find local optima, instead of the globally best solution, or fail entirely. In this work, we concentrate on the machine learning approach, where the goal is to learn typical aspects of human motion in advance from example data to overcome the aforementioned challenges.

### 1.1.2 Activity Recognition

Activity recognition is a problem on a higher level of abstraction than pose estimation. In a pure activity recognition setting, the exact body poses of a person are only of limited interest. The objective is rather to assign an activity class, out of a set of expected activities, to the motion of a person at every time instant. The understanding of what an activity actually is, varies throughout the literature. Depending on the granularity of a given activity

recognition scenario, an activity can be anything from a repetitive movement fragment, such as raising an arm, to a sequence of complex movements serving one common purpose, e.g. cooking. In this work, we pursue a customary compromise and use the term *activity* for short but complete movements, such as walking, clapping or making a golf swing. Note that we will use the terms *action* and *activity* synonymously, as opposed to other authors that focus on distinguishing short motion fragments (actions) from long-term combinations thereof (activities). A systematic review of activity recognition is available, e.g. in [162].

Similar to pose estimation, activity recognition requires an underlying model that can determine several aspects: Which activities are expected? How are activities typically ordered? Which activities are more likely to occur than others? How are activities related to the observed features? A typical approach to build such a model is by means of statistical modeling, where some sort of example data is given. In a training phase, the example data is used to estimate answers to the questions above. This learning attempt is called *supervised*, if the training data contains previously known activity labels, or *unsupervised*, if no such prior knowledge is available. In this work, we will focus on supervised learning of activities. Given a learned activity model and a sequence of input observations, activity recognition can either be performed online or offline. In the former case, the aim is to recognize activities immediately at every time instant. In the latter case, a whole movement sequence is first recorded and can then be analyzed in its entirety. We address the online recognition case.

Challenges of activity recognition are mainly due to the complexity of possible activities and the variations between different executions of identical activities. It is also difficult to determine exact beginning and ending instants for activities when no future information is available. For instance, consider the case that two activities are known: waving and scratching the head, both performed with the right hand. While a person is raising his or her arm, there is hardly any possibility to know which of the two activities is performed. Only when the hand is close to the head there might be sufficient information to decide for either waving or scratching. Another typical challenge with activity recognition is how to handle anomalies, i.e. movements that do not belong to any of the expected activities. It is important but difficult not to attribute such unknown movements to any of the known activity classes. When the sensitivity for detecting anomalies is set too high, variations of expected movements might also be treated as an anomaly, which is undesired.

While activity recognition is a typical classification problem, in this work, we investigate activity recognition in a tight conjunction with pose estimation. Many activity recognition algorithms work on data that is the outcome of a pose estimation method (e.g. joint angles). Our objective is to target both problems simultaneously, aiming to realize a mutual benefit. We therefore do not rely on typical activity recognition ingredients, such as Hidden Markov Models (HMMs), but construct models that serve both, activity recognition and pose estimation.

### 1.1.3 Gesture Recognition

Gesture recognition is closely related to activity recognition. Given a series of observations, e.g. image features or sensor measurements, the goal is to automatically recognize body movements that belong to a certain set of gestures. As we have seen with activities, there is a similar uncertainty in the literature about how to define a gesture. While some authors see gestures as instantaneous body poses, such as holding up an index finger, other authors treat extended

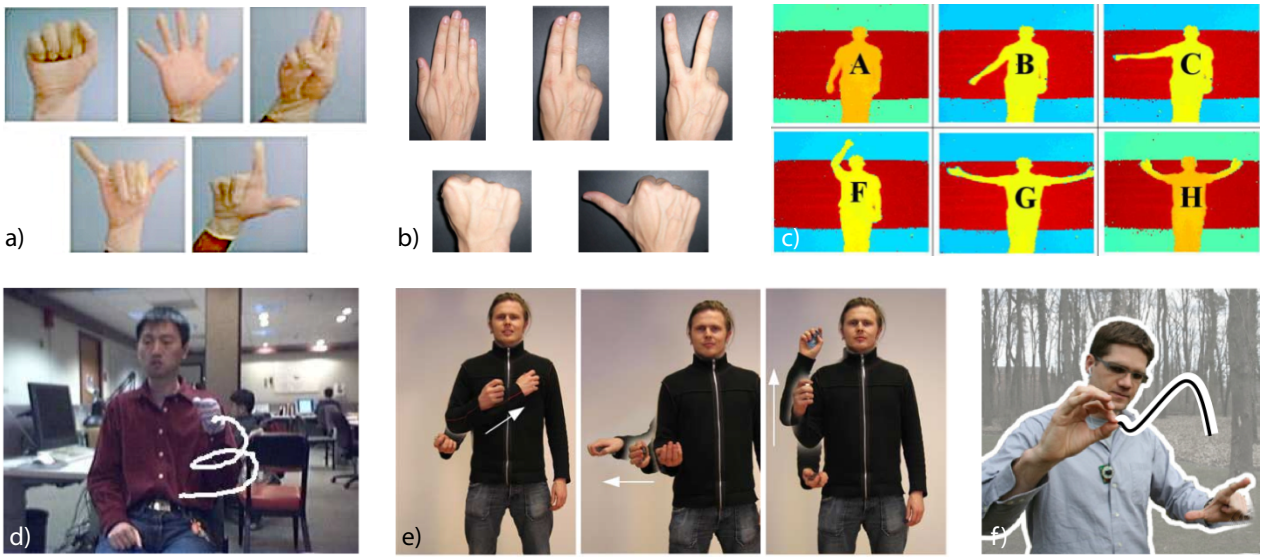



Figure 1.3: Examples of gestures used in typical gesture recognition systems. a)-c) Vocabularies of hand and full-body gestures used in categorical gesture recognition approaches. d)-f) Illustrations of spatio-temporal gestures. Pictures from [79, 61, 149, 7, 44, 57].

body movements as gestures, such as waving a hand. Exhaustive surveys of gesture recognition approaches for human-computer interaction are available in [73, 102].

Gesture recognition typically serves the purpose of letting persons interact with computerized systems without having to use the keyboard or mouse. Depending on what type of control is desired in a particular scenario, one can distinguish between *categorical* gestures and *spatio-temporal* gestures (Figure 1.3). Categorical gestures are used to issue discrete commands, such as "click the mouse". Consequently, gesture recognition systems need to identify specific hand or body poses in this case. Often it does not matter where these poses are performed and for how long. Gesture vocabularies specify the exact pose for each gesture and define which commands are associated to the gestures. For example, the hand poses in Figure 1.3 a) each represent one function that the user can trigger. When the system sees a hand with thumb and index finger extended, it will react in a pre-specified manner. On the other hand, spatio-temporal gestures do not directly correspond to discrete commands but are translated to some kind of movement in the system to be controlled. When recognizing spatio-temporal gestures, the exact location of a gesture matters, as well as the velocity and temporal extent. In Figure 1.3 e), the user can change the location of a virtual object with the movements of his right arm. In this work, we investigate an approach that combines categorical and spatio-temporal gestures to achieve a novel and flexible way of interacting with computerized systems.

## 1.2 Input Modalities

The first step in human motion analysis is to capture human movements using some type of input modality, or sensing device. A readily available solution for this purpose is to use optical motion capture systems. In this case, multiple cameras, often infrared cameras, are mounted around a designated tracking area. In order to detect and recognize individual body parts of a

moving person, reflective markers need to be attached to the person's limbs. Motion capture systems are able to track the three-dimensional location of the markers within the tracking area. A human body model is fitted to these marker locations using inverse kinematics, and the resulting joint angles are returned as the final motion capture output. When using an optical motion capture system, there is thus no need for further human pose estimation algorithms. Optical motion capture systems are used for many applications, such as industrial ergonomic studies, medical gait analysis or in the movie industry to transfer movements of a real person to virtual characters. We describe a motion capture system developed as a tool for this work in Appendix B. The system mainly serves the purpose of obtaining precise full-body motion data for training and as a ground truth for evaluating our proposed algorithms.

While very precise in their ability to track human movements, motion capture systems have disadvantages that surface when considering less controlled environments. The requirement to wear markers or even a specific motion capture suit is cumbersome when the person is pursuing some type of work or activity of everyday life. Recent marker-less motion capture systems remove the need for markers, however, at the cost of a very high sensitivity to visibility and lighting conditions. Typically, multiple regular cameras are arranged around a common area and the silhouette of a person is extracted in each of the camera views. This information is used to reconstruct 3D features that help estimating the human pose. Such marker-less systems often assume ideal illumination throughout the scene and clear, textureless backgrounds. Obviously, these assumptions are hard to fulfill in many real-life application scenarios. In this work, we therefore present motion analysis approaches that build upon particular types of input modalities: depth cameras (Section 1.2.1) and inertial sensors (Section 1.2.2). Throughout the thesis, we use the terms *observations* and *features* to refer to the input data for our algorithms extracted from the measurements of these modalities.

### 1.2.1 Depth Cameras

Recent technological advances have led to the development of cameras that allow acquiring dense, three-dimensional scans of a scene in real-time (Figure 1.4). Previously, 3D data was mainly available through multi-camera reconstruction systems, where several regular cameras are connected and depth is recovered by means of triangulation. Using infrared light, depth cameras are almost independent of lighting conditions and variations in visual appearance of a person, e.g. due to clothing. In every image pixel, these cameras provide a measurement of the distance from the camera sensor to the closest object surface. Using the known intrinsic camera parameters, the depth images can easily be converted into point clouds in metric space. This means that distances between points, e.g. on the surface of a person's body, correspond to true metric distances. There are two main types of depth cameras that operate based on different physical principles: Time of Flight (ToF) cameras and structured light cameras. ToF cameras require very sophisticated hardware and are thus comparably expensive. Structured light cameras are simpler to construct and therefore less expensive. The Microsoft Kinect [101], recently introduced in the gaming market, has helped depth imaging to spread rapidly, with a multitude of computer vision applications appearing in a short time.

ToF cameras measure the distance from a sensor array into the scene by means of modulated infrared light [88]. Rays from an infrared light flash are emitted and sent into the scene, where they hit object surfaces. Partially, these rays are reflected and travel back to the

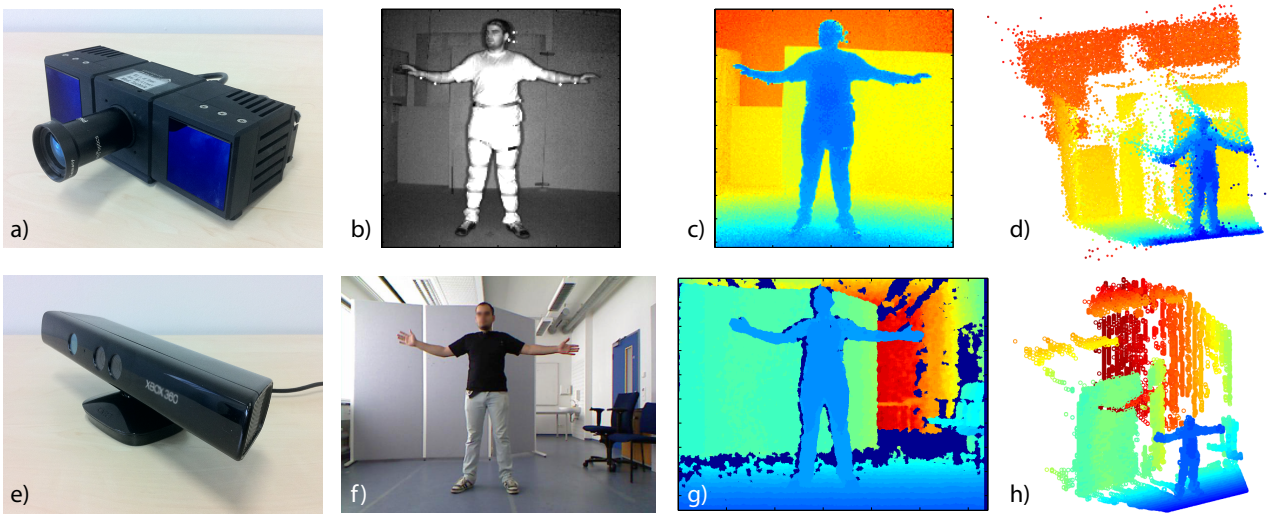



Figure 1.4: Examples of typical depth cameras. a) ToF camera with two infrared flashes and optics at the center [121]. b) ToF amplitude image. c) ToF depth image with color-coded depth (blue: close; red: distant). d) 3D Point cloud resulting from depth image. e) Kinect device with an infrared projector, an infrared camera and an RGB camera [101]. f) Kinect RGB image. g) Color-coded Kinect depth image. h) Resulting point cloud.

ToF camera. As shown in Figure 1.5 a), the returning rays exhibit a shift in the modulation phase that is measured in the sensor array. This phase shift is directly proportional to the distance the light traveled from the camera to an object and back. The term *time of flight* is thus misleading, as it is not actually the time that is measured. While ToF cameras have a relatively low resolution, they simultaneously provide depth and intensity images which can be used as regular grayscale images. The intensity images are created from the amplitude shift of incoming reflected light rays. Cameras based on the structured light principle project a known infrared light pattern into the scene and capture the projected pattern using a regular infrared camera, see Figure 1.5 b). The depth at any image location is estimated by evaluating the distortion of the projected pattern. The Kinect combines a structured light camera with a regular RGB camera that can be calibrated to the same reference frame.

Although depth cameras are very attractive for computer vision applications, their data is degraded by noise and various types of artifacts. ToF cameras currently have resolutions around $200 \times 200$ pixels, the Kinect has a VGA resolution of $640 \times 480$ pixels. Moreover, note that the 3D data captured by depth cameras is only available for object surfaces that are facing the camera. No 3D points are generated at locations that cannot be reached by the light rays emitted from the depth camera. The available data thus strongly depends on the view perspective of the depth camera. In fact, the data generated by ToF cameras or the Kinect is often referred to as being 2.5D (as opposed to 3D) for this reason. Nonetheless, this information facilitates problems such as segmenting a person from the background.

Typical noise in ToF cameras is due to systematic errors and other sources impacting the measurements [88]. For example, each ToF camera has a particular distance error that varies throughout the measurement range. Moreover, surfaces with low infrared reflectance properties (often black surfaces) result in inconsistent depth estimates. So-called "flying pixels" appear at object boundaries due to the fact that individual ToF camera pixels can receive

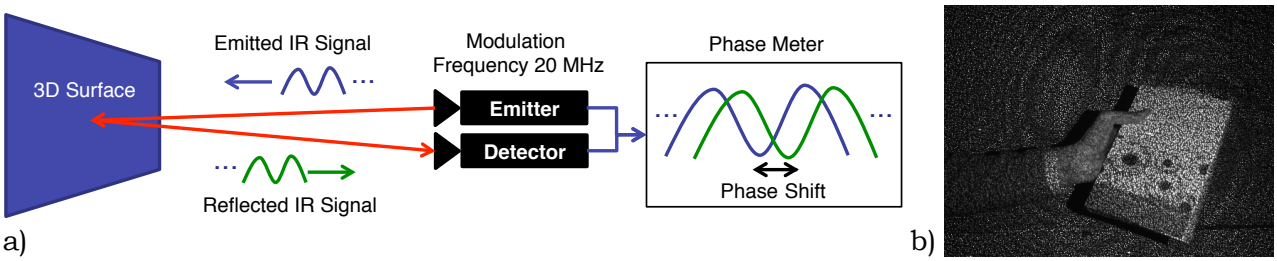



Figure 1.5: Working principles of depth cameras. a) The phase shift between emitted and received infrared light waves, as measured by a ToF camera, is proportional to the depth. b) Structured infrared light pattern projected by the Kinect and seen in the IR camera.

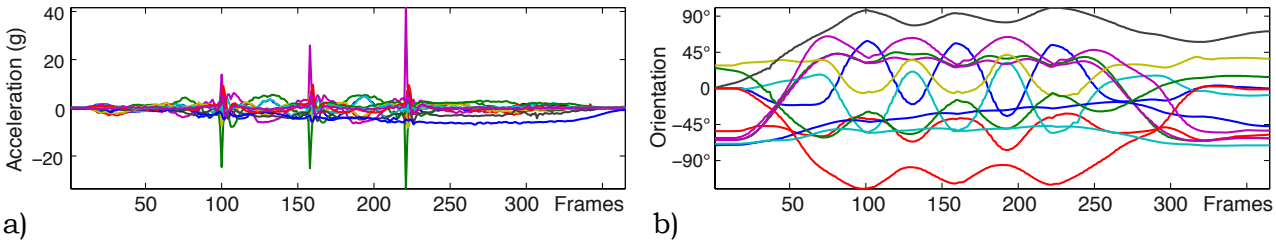



Figure 1.6: Accelerometers compared to orientation sensors. a) Accelerometer data from four inertial sensors attached to arms and legs of a person during a clapping movement with three repetitions. b) Orientation sensor data for the same clapping sequence.

multiple depth measurements. Finally, when observing a full human with a ToF camera, the currently low resolution leads to thin body parts being represented by just a few pixels. The Kinect provides higher resolution depth images, but different surface properties can also lead to depth irregularities. In addition, the structured light principle used by the Kinect causes shadow-like artifacts to appear around objects (or body parts) that are closer to the camera.

### 1.2.2 Inertial Sensors

Compared to cameras, inertial sensors provide an entirely different type of measurements, yet they are an interesting input modality for human motion analysis. When attached to the body of a person, wireless inertial sensors enable free movement without any consideration of line-of-sight or illumination. Originating from aviation and military applications, inertial sensors nowadays are almost ubiquitous – every modern mobile phone has an integrated inertial sensor. Wireless stand-alone sensors are currently available in small formats, such as $2cm^3$, and allow for a battery-based operation of several hours [161]. Exemplary sensors with a wireless receiver are shown in Figure 1.7 a). Continued miniaturization will lead to even smaller sensors in near future. Current early prototypes demonstrate the feasibility of integrating sensors into garments, such that they become barely noticeable for their users [107].

Simple inertial sensors measure acceleration in three orthogonal spatial directions. Such accelerometers are not very suitable for human pose estimation, as different body poses can result in almost identical sensor measurements, whenever the person does not move. Figure 1.6 a) shows accelerometer data for three clapping movements. There are significant peaks

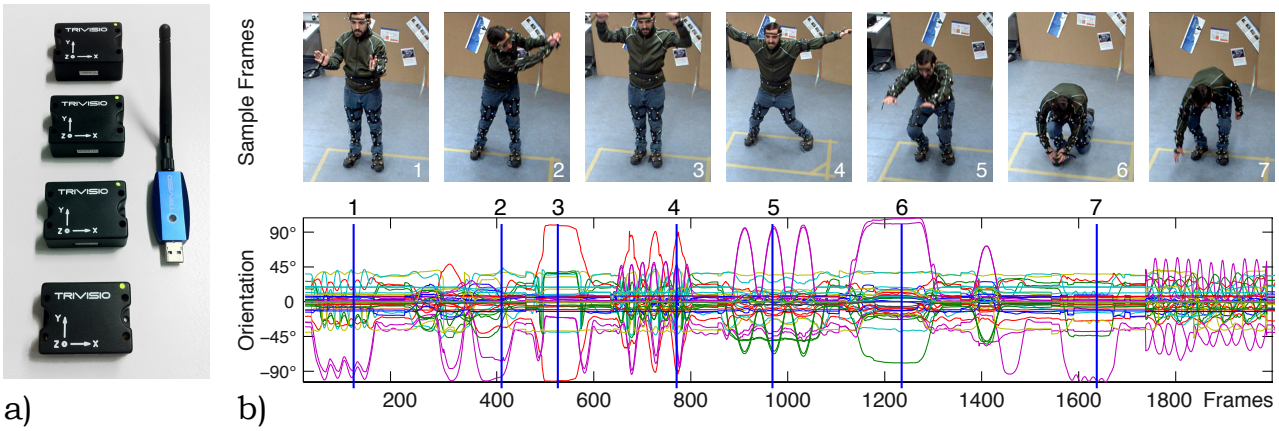



Figure 1.7: Inertial orientation sensors and their measurements. a) Four Trivisio wireless sensors and a USB receiver [161]. b) Images from an example movement sequence (top row) with recorded orientation sensor data (bottom row). Corresponding frames are indicated.

corresponding to the impacts, but the rest of the movements is almost indistinguishable. More advanced inertial sensors add gyroscopes that measure rotational acceleration along three axes. Together, such sensors can provide estimates of three-axial orientation. A familiar intuition behind these orientation measurements comes from airplanes whose orientation is typically expressed in terms of yaw (heading direction), pitch (vertical incline) and roll (rotation around the aircraft body). Figure 1.6 b) shows orientation data for the three clapping moves – the varying poses are clearly differentiable. A disadvantage of orientation sensors is their tendency to accumulate error, known as drift. Recent sensors therefore use additional information, for instance a magnetic compass, to complement their relative inertial measurements. Sensor fusion algorithms are employed to achieve stable orientation measurements [44].

Attaching several inertial orientation sensors to the limbs of a person, we obtain a continuous stream of orientation measurements with three values per sensor (or four, in quaternion representation). Typical sensors operate at around 60 Hz and each sensor can be addressed separately. This way, identifying individual body parts becomes straightforward, as opposed to the need of recognizing body parts from visual observations. Figure 1.7 b) shows pictures from an exemplary motion sequence of a person and the corresponding orientation data. Obviously, the data is smooth and varies significantly with different body poses. On the other hand, only a low-dimensional signal is recorded. A single frame in a video contains thousands of intensity values, whereas six orientation sensors result in a mere 18 values per frame.

## 1.3 Machine Learning

Having illustrated the aspects of human motion analysis and the input modalities we address in this work, we now outline the basics of our focused methodological approach. Machine learning refers to the process of generating knowledge from empirical example data, collected by observing a process or system of interest. Typically, examples of inputs and outputs of this system are available, but nothing or little is known about the underlying mechanisms

that relate inputs and outputs. In other words, the system is a black-box. Machine learning algorithms are given the observed data for training and try to infer the unknown principles of the system. In an ideal case, a machine learning algorithm does not only memorize the training samples, but is able to apply the learned principles to new data. The machine learning algorithm can thus be used to predict the behavior of the system with respect to previously unseen data. This ability to *generalize* is a crucial and challenging aspect of every machine learning algorithm. Once a model of the system's mechanisms has been learned, the need to analytically model all entities and processes involved in the system is alleviated. In many practical cases, creating such complete analytical models is an intractable task.

In terms of human motion analysis, machine learning can be helpful in several ways. First, static human body models, such as those shown in Figure 1.2, can be complemented with a model of dynamics by learning feasible human movements from training data. Such learned dynamics models can facilitate solving the ill-posed problem of estimating body poses from limited observations. Second, machine learning can be used to learn the appearance of human motion in arbitrary modalities, such as video or inertial sensor measurements. In this case, training requires motion data and the corresponding representation in the desired modality to be available simultaneously. After training, when new, previously unseen observations appear, the corresponding body poses can be estimated without modeling the complex imaging or sensor acquisition process. Third, machine learning can help categorizing human actions into classes and enable to recognize different types of activities from arbitrary observations.

Unfortunately, the boundaries of what machine learning methods can achieve are quite clear. The curse of dimensionality is one of the most important limiting factors: The higher the number of parameters in an estimation problem is, the more training data we need – increasing in an exponential fashion. In other words, there is never enough training data. Moreover, even well-designed and elaborate machine learning algorithms are prone to overfitting the training data, which results in poor generalization capabilities.

As briefly outlined before, machine learning methods can be generally categorized into supervised and unsupervised approaches. Supervised methods learn a mapping between a given set of inputs and the corresponding outputs. In human motion analysis, we will encounter this situation when learning the appearance of the human body in depth images and inertial sensor measurements. The term *supervised* implies that, during training, exemplary solutions to the problem at hand are available. Human activity recognition is another typical example of supervised learning, where the training data consists of motion information and associated activity labels. In *unsupervised* learning, a dataset is given for training without any additional information. The goal of an unsupervised learning algorithm is to learn specific model parameters that can best explain the data or simply to unveil the hidden, underlying structure of the data. In this work, we will deal with the latter situation in terms of manifold learning, used for extracting a low-dimensional representation of feasible human poses.

Another way of categorizing machine learning methods is in terms of generative and discriminative approaches. *Generative* learning methods attempt to model the state space of a given unknown system and the mapping from states to observations. In human motion analysis, states correspond to parametric representations of human poses, e.g. expressed using a body model. A generative method could learn a model of feasible human poses and a mapping from such poses to their corresponding appearance, e.g. in video. Pose estimation then consists of searching for poses generating an appearance that best matches the true image

observations. In contrast, *discriminative* learning methods do not model the system state explicitly, but instead focus on the direct mapping from observations to states. For human motion analysis, a discriminative method thus learns to directly predict human poses from observations. In this work, we investigate both, generative and discriminative approaches and discuss their respective properties and practical applicability.

## 1.4 Applications

There is a plethora of applications for human motion analysis methods. In the following, we give an overview of existing and potential application scenarios for the three subfields of human motion analysis focused in this work: human pose estimation, activity recognition and gesture recognition. The overview follows related scientific work of the last decade.

### 1.4.1 Entertainment and Animation

Methods for capturing human poses and movements play an important role in entertainment and animation applications. Motion capture systems are a commodity in the movie industry, where articulated virtual characters, such as human figures, robots or animals, need to be animated. To achieve a realistic appearance, such animations are rarely created by manipulating models on a timeline using keyboard and mouse. Instead, the motion of real persons is digitized and transferred to virtual characters – see Figure 1.8 a). A pose parameterization with joint angles is very suitable in this context, as these angles can be directly transferred to virtual body models, even if the proportions are not in an exact match with the real person [71]. Several techniques have been proposed to remove the need for a marker-based motion capture system, e.g. in [127, 151], where multi-camera systems are used instead, or in [147], where inertial sensors roughly capture human movements. Another typical animation task is to re-arrange existing motion capture sequences in animations, which includes modifying the timing, the style and the dynamics of a motion sequence. Recent methods allow animators to perform simple, abstract movements to vary these parameters, as explained e.g. in [157, 39, 65]. The central idea is that natural-looking animations can be best achieved by means of acted performances, instead of by tuning parameters.

Human motion tracking is currently strongly impacting the gaming industry. While classical gaming interfaces were based on keyboards, joysticks or other button-based controllers, human motion is increasingly used as a direct input. Several approaches to control virtual characters in games by means of inertial sensors on the body of the player have been proposed, e.g. in [144]. The Microsoft Kinect [101] has brought the possibilities of performance-based animation into gaming by capturing the player's movements with a depth camera in real-time [146, 51]. The Kinect also allows human movements to be used for user interface tasks. Human pose estimation can also help people in more serious entertainment scenarios, for instance when certain movements need to be learned or trained. In [90], a person wears a motion capture suit and can see his or her own martial arts moves in a head-mounted display, along with a virtual character that shows the corresponding perfect movement. A similar system is presented in [91] that employs inertial sensors and a single camera instead of a full marker-based motion capture system. Analogous approaches for learning dance movements also exist.

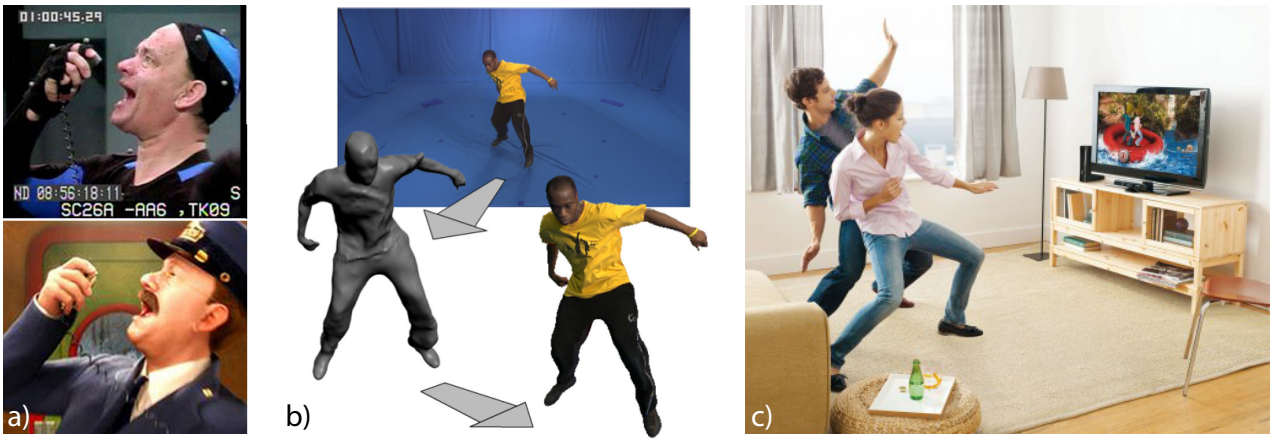



Figure 1.8: Human motion tracking for entertainment applications. a) Tom Hanks equipped with motion capture markers and the corresponding animated character (from the movie *The Polar Express*, 2004). b) Surface-based motion capture using chroma-keying. c) Kinect game illustration showing body tracking for two persons. Pictures from [67, 152, 101].

### 1.4.2 Industrial Monitoring and Surveillance

Industrial applications of human motion analysis mainly focus on two aspects: Monitoring and assisting workers in assembly tasks and evaluating ergonomics, either for achieving suitable work environments or for product design. When the current activity of an assembly line worker is known, his or her progress can be determined and specific assistive systems can be triggered. In [153, 154, 111], the authors present methods using inertial sensors and ultrasonic hand tracking that monitor the activity of workers at a car assembly line. The workflow is modeled beforehand with a finite state machine and workers are alerted when they omit one of the workflow steps. It is also possible to automatically provide suitable tools when a new workflow step begins. The authors of [97] describe a system for human activity recognition based on visual human tracking that can be used in robotic assistance scenarios. Here, a robot is performing tasks to help a human, e.g. an assembly line worker, and adapts its actions based on the behavior of the assisted person. One further step is taken in [145], where human motion is directly transferred to the movements of a robot. The authors learn a latent space representation that allows translating human poses into robotic control signals.

Human motion analysis also plays an important role in ergonomic studies. As part of the German public research project AVILUS (2008–2011), human body tracking methods were explored for the purpose of evaluating car and aircraft cockpit designs using virtual reality techniques [13]. A person, equipped with motion capture markers, is sitting on a chair and through a head-mounted display (HMD), he or she can see a virtual model of the surrounding cockpit. The movements of the person are tracked and transferred into the virtual scene so that interactions become possible, such as reaching for buttons or touching handles. The motion capture algorithm described in this work (Appendix B) was developed for the AVILUS project and was evaluated by a car manufacturer using the DTrack tracking system [2]. An anatomically correct anthropometric body model is presented in [22, 12]. This model can simulate the outer appearance for various body types and is therefore particularly suitable for ergonomic studies, where different body proportions can be evaluated automatically.

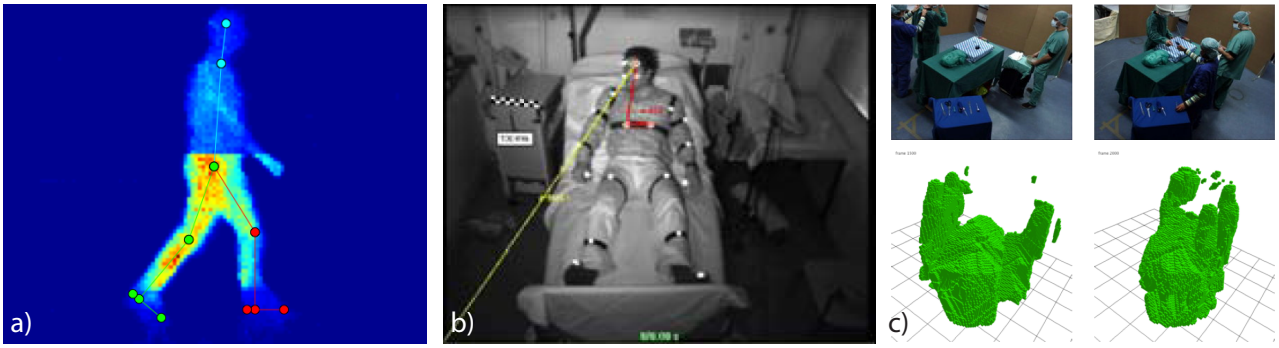



Figure 1.9: Human motion analysis for medical applications. a) Gait analysis using depth images from a ToF camera. b) Marker-based tracking of an epilepsy patient for analysis of motion patterns during seizures. c) Workflow recognition in the operating room based on visual data from a multi-camera 3D reconstruction system. Pictures from [31, 74, 115].

### 1.4.3 Medical Motion Analysis

In the medical domain, analyzing human motion can serve various purposes, ranging from disease diagnosis, through an evaluation of rehabilitation progress, to monitoring of elderly people during their everyday life. A common theme in many medical applications is to provide physicians with a quantification of human motion. Currently, it is typical practice that physicians visually assess the movements of a patient and derive judgments based on their personal experience and descriptive, disease-specific categorizations of mobility patterns. The aim of computer-based motion analysis in this context is to decrease the influence of subjective factors in diagnosis and to make assessments more repeatable.

An obvious use for tracking body movements is in orthopedic motion analysis. In [9], a system based on accelerometers is described for evaluating the gait coordination capabilities of osteoarthritis patients, in particular after hip arthoplasty. A gait analysis system using a time of flight camera to capture human motion is presented in [74]. Here, a simple skeleton model is fitted to the depth data showing a person on a treadmill from a side perspective. The algorithm returns parameters often used for gait analysis, such as cadence or range of step motion. The center of mass displacement during walking, another gait parameter of interest to physicians, is estimated by the method in [131] from accelerometer measurements. While the aforementioned methods are meant for a stationary application in a physician's practice, there are also several approaches for patient monitoring over extended periods of time in their everyday surroundings. Such approaches, often based on accelerometers, can be used to keep track of activity levels of cardiovascular disease patients. Research shows that sufficient physical activity is a key factor in prevention of cardiovascular events, such as stroke or myocardial infarction. In rehabilitation, motion analysis can be used to assist stroke patients in regaining their walking capabilities [181]. Another application is telemedicine for elderly care. Here, patients are equipped with wearable sensors that monitor their movements and their physical well-being. In case of any unexpected event, such as a fall, help can be requested automatically [78]. In [106], a method is proposed that recognizes activity changes of patients as well as walking periods. When prescribed activity levels are not reached, the patient can be informed and alerted accordingly.

Neurologists working with Parkinson's, epilepsy or Multiple Sclerosis patients increasingly use motion analysis methods, as these neurological diseases often cause movement disorders. Multiple Sclerosis patients suffer from a severe degradation of their motor abilities that can end in complete paralysis. Their disease progress, and thus the appropriate type of treatment, is determined from the patients' performance in several mobility assessment tests [116]. Capturing patient motion in these tests using cameras or inertial sensors can help to quantitatively compare the patients' mobility over time. Parkinson's patients suffer from symptoms such as tremor (shaking extremities), bradykinesia (slowed-down movements) and on-off-phases that can be partially regulated with medication [103]. Here, automatic motion analysis systems can help detecting inappropriate medication levels that result in specific measurable motor effects. The authors of [83] describe a system for automatic recognition of on-off-phases of Parkinson's patients from accelerometer data. Using such information, physicians can better determine the effect of medication on a particular patient than by means of irregular visual observations or patient self-assessments [116]. Research in epilepsy treatment shows that analyzing the duration and strength of epileptic seizures is crucial for correctly classifying the disease. In [31], a camera-based method is described that allows capturing seizure patterns quantitatively in long-term clinical epilepsy monitoring units. The authors also show that the extracted information can help determine where in the brain of a particular patient the neurological cause for epilepsy is located [110]. Other researchers focus on prediction of epileptic seizures from motion information combined with electroencephalogram (EEG) data [43].

A completely different medical application scenario for human motion analysis, and activity recognition in particular, is workflow monitoring. Modern healthcare is heavily concentrating on quality assurance and therefore most procedures follow a standardized workflow. The operating room (OR) is a typical example, where surgeries or interventions consist of several phases and multiple steps. Depending on the current step, personal roles can change and different tools and devices can be required. In [5, 6], methods are presented that can automatically determine the workflow phase based on accelerometers attached to the arms of a surgeon and sensors on several tools. The authors identify remaining surgery time prediction as a potential practical use and also automated documentation of surgeries [114]. A purely visual approach is taken in [115], where workflow phases of a surgery are learned and recognized using data from a multi-camera 3D reconstruction system, as shown in Figure 1.9 c).

### 1.4.4 Human-Machine Interaction

In human-machine interaction, there is a clear trend towards physical interface technologies that allow persons to use their body for communicating with a computerized system, instead of pressing keyboard buttons or using a mouse. Several authors use inertial sensors to capture human motion for gesture recognition [132, 44, 84]. In these approaches, inertial sensors suffice, as compared to using cameras, because full human body tracking is not desired. The authors of [132] propose a method that allows persons to control entertainment appliances, such as a television, with a Wii remote controller. This controller has been introduced for gaming and contains acceleration sensors. In [44], orientation sensors are proposed as a tangible user interface for controlling a media player. A personalizable gesture recognition system based on accelerometers is presented in [84] for design studios. Designers can use gestures to navigate in virtual design models during presentations to their customers. The method allows its users

to define their preferred gestures. A radical approach, called *imaginary interfaces*, is proposed in [57]. Here, a camera is attached to a necklace and tracks a person's hand movements in front of the body. The user can imitate a coordinate system with the fingers of one hand and use the other hand for interacting with an interface that the user knows and therefore has in his imagination. This way, the user can control a simple remote application, e.g. a sketch board, from any location, even without a screen – see Figure 1.3 f). The authors of [96] propose to use gestures, captured using accelerometers in a mobile phone, as a method for user authentication. A person-specific gesture pattern thus replaces conventional authentication approaches, such as entering a personal identification number. Another visionary application with acceleration sensors is introduced in [126]. The system aims at recognizing the cultural background of persons in communication situations from culture-specific gestures.

The operating room is a potentially very suitable environment for gesture recognition methods, as sterility requirements often do not permit surgeons to touch keyboards or mice [77]. In this context, several authors concentrate on using cameras for capturing hand gestures of surgical staff [85, 172, 52, 173]. The methods provide a set of gestures that the systems know and let the surgeon control intra-operative systems, such as image browsers. Such image browsers are widely spread in operating rooms, in particular during image guided interventions. To overcome the sensitivity of classical cameras to lighting conditions, recent work focuses on ToF cameras for gesture recognition. The authors of [61] propose to recognize upper-body gestures, see Figure 1.3 c). In [89, 149], hand gestures are detected with a ToF camera to allow for touch-less interaction with an intra-operative medical image viewer. In this work, we present a novel gesture-based interface approach for the operating room that uses body-worn inertial sensors and learns arbitrary gestures demonstrated by a surgeon.

Chapter 2

# Thesis Overview

Given the number of applications for human motion analysis presented before and the multitude of related methodological challenges, it is crucial to define the scope of this work in terms of its overall objectives. In the following, we give an overview of our objectives, the achieved contributions and the structure of the thesis. We also introduce our notational scheme.

## 2.1 Objectives

The overall goal behind the work presented in this thesis is to develop algorithms for human pose estimation, activity recognition and gesture recognition that can be applied in complex environments, such as the operating room, where traditional, camera-based methods face significant challenges. Consequently, we focus on inertial sensors and depth cameras as input modalities. We can formulate the guiding objectives behind this work as developing algorithms for human motion analysis that...

- can cope with sparse, noisy and ambiguous input information,
- make use of supplementary, informative observations in a training phase,
- learn prior models of human motion from training data,
- are applicable in real-time in challenging environments.

The rationale behind these objectives is as follows. Realistic application scenarios often do not allow using comprehensive input modalities and only very restricted observations are available. In particular, our use of inertial sensors and single depth cameras leads to very sparse and noisy input data. This issue can be facilitated by complementing the sparse information with additional input modalities, such as marker-based motion capture, that are only present during a training phase. Machine learning algorithms can use this comprehensive training data to extract models of human motion and to learn the relationship between the sparse input data and full human poses. As the computational complexity of machine learning algorithms concentrates on the training phase, we can achieve real-time performance at application time.

## 2.2 Contributions

This thesis incorporates the results of several research projects related to human pose estimation, activity recognition and gesture recognition. At a high level of abstraction, the major contributions, resulting from the objectives described above, can be summarized as follows:

- A novel algorithm for simultaneous human pose estimation and activity recognition using a discriminative machine learning approach. This algorithm models the mapping from input observations to human full-body poses using non-linear regression techniques that we train based on multiple-activity example data.

- A novel algorithm for simultaneous pose estimation and activity recognition using a generative machine learning approach. This algorithm builds upon dimensionality-reduced models of feasible human poses that we obtain using manifold learning. We propose an efficient Bayesian tracking approach that allows switching between multiple activities.

- A novel algorithm for gesture-based human-machine interaction in the operating room. This algorithm uses inertial sensors and allows surgeons to control intra-operative computerized devices without interfering with sterility requirements. The algorithm can learn arbitrary gestures from examples, as desired by a surgeon.

- Application of novel input modalities, wearable inertial sensors and depth cameras, that offer limited observations but enable human motion analysis in challenging settings. Our use of inertial sensors differs from existing work since we approach these devices as an alternative to computer vision input modalities.

## 2.3 Outline

The contents of each part of this thesis are structured as follows. The references given below refer to our original publications that the respective chapters build upon.

- Part I is an introduction to the topics of this thesis and the related background (Chapters 1 and 2). In Chapter 3, we discuss the most important mathematical techniques that our methods build upon. The methods covered are regression, where our focus is on kernel regression and Gaussian process regression, and dimensionality reduction, where we focus on manifold learning. The coverage is for the sake of completeness and familiar readers can skip this chapter.

- Part II focuses on activity recognition and pose estimation. In Chapter 4, we present a discriminative method for simultaneous activity recognition and pose estimation from body-worn inertial orientation sensor data [138]. Chapter 5 covers generative approaches for activity recognition and pose estimation based on manifold learning [139, 136, 140]. We present our evaluation on using inertial sensors and depth cameras as input. Chapter 6 is an excursus from the main theme of the thesis and deals with an alternative approach for pose estimation from depth data that is not based on machine learning [141, 142].

- Part III focuses on gesture recognition and brings concepts investigated in the previous chapters to the application scenario of the operating room. In Chapter 7, we describe our approach for human-computer interaction that uses inertial sensors to capture gestures and machine learning for gesture recognition. Surgeons can freely define gestures that are most suitable for controlling intra-operative computerized systems [16, 134, 135].
- Part IV concludes the thesis. Chapter 8 contains a summary and a discussion of our main learnings. Chapter 9 addresses related possible future research directions.
- Part V is an appendix and contains supplementary material. Appendix A introduces the human skeleton body model that is used throughout the thesis. In Appendix B, we describe the optical marker-based motion capture system that was developed as an introductory work to this thesis and that serves for acquisition of training and ground truth data. A list of authored and co-authored publications is given in Appendix C.

## 2.4 Notation

Throughout this work, we will adhere as much as possible to the following notational scheme:

| **Entity** | **Notation** | **Example** |
|---|---|---|
| Scalar values | Lower-case, plain letters | $a, b \in \mathbb{R}$ |
| Vectors | Bold-face lower-case letters | $\mathbf{x}, \mathbf{y} \in \mathbb{R}^2$ |
| Matrices | Bold-face capital letters | $\mathbf{A} \in \mathbb{R}^{m \times n}$ |
| Sets | Calligraphic font letters | $\mathcal{P} = \{1, 2, 3\}$ |
| Functions | Plain letters | $f(\mathbf{x})$ or $F(\mathbf{y})$ |

When indicating the dimensions of a matrix, e.g. $\mathbf{A} \in \mathbb{R}^{m \times n}$, the first index (here: $m$) refers to the number of rows and the second index (here: $n$) gives the number of columns of the matrix. Table 2.1 lists particular symbols used in this work and their general meaning.

| Symbol | Description |
|---|---|
| $\alpha$ | Discrete index of an activity or gesture. |
| $d_y$ | Dimensionality of full-body pose space. |
| $d_x$ | Dimensionality of observations, e.g. inertial sensor or camera data. |
| $d_z$ | Dimensionality of manifold embedding space. |
| $K$ | Number of skeleton joints. |
| $L$ | Number of anatomical landmarks. |
| $M$ | Number of considered activities or gestures. |
| $N$ | Number of given points in a dataset, e.g. poses or observations. |
| $n$ | Number of particles in the particle filter algorithm. |
| $T$ | Number of motion capture targets. |
| $t$ | Discrete time index, often used as a subscript. |
| $\mathbf{x} \in \mathbb{R}^{d_x}$ | Observation vector of $d_x$ dimensions, e.g. sensor or camera data. |
| $\mathbf{X} \in \mathbb{R}^{N \times d_x}$ | Matrix of $N$ observation vectors as rows. |
| $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ | Set of $N$ observation vectors. |
| $\mathbf{y} \in \mathbb{R}^{d_y}$ | Vector of $d_y$ joint angles representing a human full-body pose. |
| $\mathbf{Y} \in \mathbb{R}^{N \times d_y}$ | Matrix of $N$ human full-body pose vectors as rows. |
| $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ | Set of $N$ joint angle vectors. |
| $\mathbf{z} \in \mathbb{R}^{d_z}$ | Dimensionality-reduced human body pose in $d_z$ dimensions. |
| $\mathbf{Z} \in \mathbb{R}^{N \times d_z}$ | Matrix of $N$ dimensionality-reduced human poses as rows. |
| $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ | Set of $N$ dimensionality-reduced human poses. |
| $\hat{x}$ | Estimated quantity. |
| $\tilde{x}$ | Variation of a quantity. |
| $\bar{x}$ | Averaged quantity. |

Table 2.1: Notational scheme used throughout the thesis.

Chapter 3

# Mathematical Background

In this chapter, we introduce the main mathematical techniques used in our work. These techniques are regression and dimensionality reduction, where we focus on manifold learning.

## 3.1 Regression

Regression methods can be used to recover an unknown functional relationship $f(\mathbf{x}) = \mathbf{y}$, given noisy observations $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ of the output variable, at certain values of the input variable $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, where $\mathbf{y}_i \in \mathbb{R}^{d_y}$ and $\mathbf{x}_i \in \mathbb{R}^{d_x}$. Knowledge about this relationship allows predicting $\mathbf{y}$ for new values of $\mathbf{x}$. In the case of classical regression analysis, one makes an assumption on the type of the functional mapping $f$, e.g. that it is linear. Based on this assumption, a parametric model of appropriate type is fitted to the observed data points, for instance using least squares methods. Non-parametric regression methods do not assume a particular parametric form for the sought functional relationship $f$. Regression is in this case purely data-driven, i.e. statistics of the distribution of observed functional values are used to model the functional relationship and, thus, to predict functional values. In this work, we use non-parametric regression methods and thus our focus will be on kernel regression (KR) and Gaussian process regression (GPR).

### 3.1.1 Kernel Regression

Kernel regression refers to a class of non-parametric regression techniques where the value of the unknown function $f(\mathbf{x})$ at any location $\mathbf{x}$ is computed from the functional values of the known points $\mathbf{x}_i$ in the neighborhood of $\mathbf{x}$ using a kernel function for weighting. Among several kernel regression variants, a frequently used method is the Nadaraya-Watson estimator [105, 17]. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of locations where the values of $f$ are known, and let $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ be the corresponding functional values. Based on a kernel function, such as a Gaussian kernel $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\|(\mathbf{x}_i - \mathbf{x}_j)/\sigma\|^2)$ with a width parameter $\sigma$, the functional value $\hat{\mathbf{y}}$ at any location $\mathbf{x}$ can be estimated as

$$\hat{\mathbf{y}} = \hat{f}(\mathbf{x}) = \sum_{i=1}^{N} \frac{k(\mathbf{x}, \mathbf{x}_i)}{\sum_{j=1}^{N} k(\mathbf{x}, \mathbf{x}_j)} \mathbf{y}_i. \tag{3.1}$$

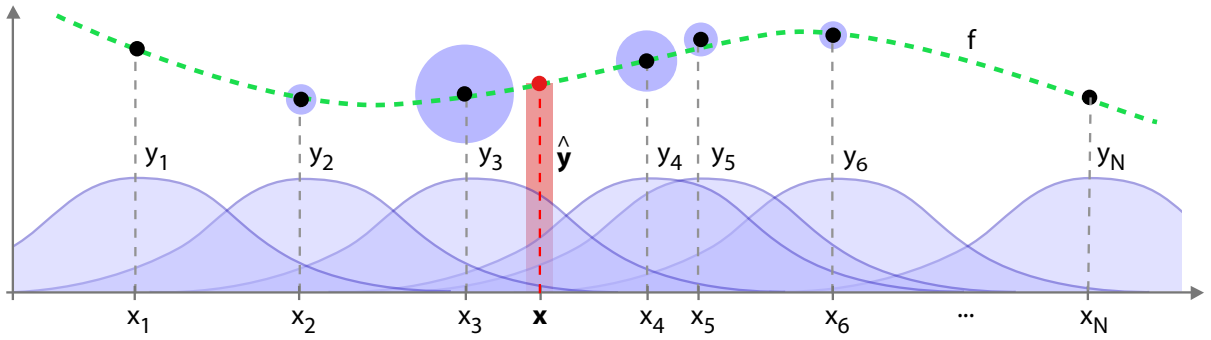



Figure 3.1: Illustration of the Nadaraya-Watson estimator. At the locations $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the noisy functional values $\{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ are known (black dots), but the function $f$ itself (dashed green line) is not available. The estimate for $\hat{\mathbf{y}} = f(\mathbf{x})$ is a weighted average of all known functional values $\mathbf{y}_i$. The blue shaded circles are proportional to the associated weight.

The estimate is a weighted sum of the functional values $\mathbf{y}_i$ for all $N$ known points, where the weights are given by the fraction. The largest contribution in the summation is attributed to functional values $\mathbf{y}_i$ at locations $\mathbf{x}_i$ in the vicinity of $\mathbf{x}$. For such locations, $k(\mathbf{x}, \mathbf{x}_i)$ will be maximal. Depending on the choice of $\sigma$, the influence of functional values at locations further away from $\mathbf{x}$ will decrease until becoming negligible. The denominator in Equation 3.1 ensures that the weights sum up to one. Therefore, the estimated function value is actually a convex combination of the known values. This is a favorable property, as it means that the estimates will always lie within the convex hull of the given data points. Figure 3.1 illustrates kernel regression with the Nadaraya-Watson estimator. In this one-dimensional example, a Gaussian is centered at each given input $\mathbf{x}_i$. Each Gaussian determines the influence range of a particular input point. This influence range can be adjusted by changing the width parameter $\sigma$, which is equal for all Gaussians. At a given new location $\mathbf{x}$, the values of the Gaussians for all $\mathbf{x}_i$ determine the influence of the respective function value $\mathbf{y}_i$ in the prediction of $\hat{\mathbf{y}} = f(\mathbf{x})$.

### 3.1.2 Gaussian Process Regression

A different kind of non-parametric regression method with interesting properties is based on Gaussian processes. A Gaussian process is a stochastic process, where every realization in space or time consists of a random variable with a normal (Gaussian) distribution. In addition, any finite subset of these random variables have a joint multivariate normal distribution [125]. Intuitively, Gaussian processes can be thought of as distributions over functions. In this context, the random variables of a Gaussian process are the function values $f(\mathbf{x})$, and the index set of the Gaussian process is the set of possible function inputs, i.e. the set where $\mathbf{x}$ is drawn from. Initially, we will assume that the functions we are interested in are real-valued, $f : \mathbb{R}^{d_x} \to \mathbb{R}$ and $d_y = 1$. The extension to vector-valued functions will follow (Section 4.3.1).

A Gaussian process can be uniquely determined by a mean function $\mu(\mathbf{x})$ and a covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. The covariance function determines the covariance between the random variables of the Gaussian process, i.e. the covariance of any pair of function values $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$, but is defined in terms of the inputs, $\mathbf{x}_i$ and $\mathbf{x}_j$. This means that the covariance between function values whose corresponding inputs are close will be high. The random

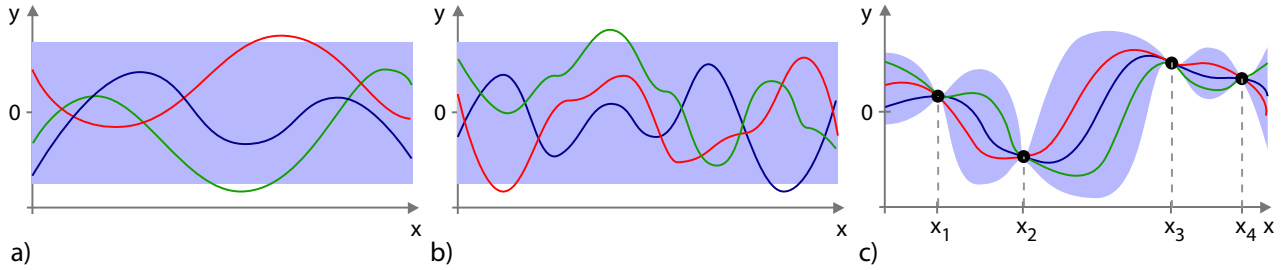



Figure 3.2: Illustration of Gaussian processes for regression. a) Three random functions sampled from the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{K})$, characterized by a specific covariance function. The shaded area represents the mean plus the standard deviation for each location, i.e. for each random variable of the Gaussian process. b) Three random functions sampled from a prior distribution with a different covariance function. c) Conditioned on known training data points (black dots), the distribution is restricted to functions that fit the data.

variable representing a function value $f(\mathbf{x})$ for an unknown input $\mathbf{x}$ will thus correlate well with the function values for close-by known inputs. A typical choice for the mean function is a constant zero $\mu(\mathbf{x}) = 0$. As a covariance function, different versions of the Gaussian are often used [41, 125], such as $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{1}{2}\|(\mathbf{x}_i - \mathbf{x}_j)\|^2)$ in the simplest case.

Once mean and covariance function of a Gaussian process are specified, a prior distribution over functions is determined. We can now draw samples from this distribution that will be individual functions – or more precisely, sets of random functional values following the mean and covariance functions of the Gaussian process. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of input points, where $\mathbf{x}_i \in \mathbb{R}^{d_x}$ and $\mathbf{X}$ is the corresponding matrix and let $\mathbf{K}$ be an $N \times N$ covariance matrix with entries $k_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$. A function sampled from the Gaussian process is then given by the vector

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}), \tag{3.2}$$

where $\mathbf{y} \in \mathbb{R}^N$, $\mathcal{N}(\cdot, \cdot)$ indicates a normal distribution and $\mathbf{0}$ is an $N$-dimensional vector of zeros. Note that each entry of $\mathbf{y}$ is the scalar functional value corresponding to one of the $N$ input points $\mathbf{x}_i$. However, the obtained functional values will not represent any function of particular interest, as we have not yet incorporated any information on known functional values $f(\mathbf{x}_i)$. Assuming we have a training dataset of observed functional values $\mathcal{Y} = \{y_1, \ldots, y_N\}$ corresponding to the inputs $\mathcal{X}$, where $y_i \in \mathbb{R}$ and $\mathbf{y} := [y_1, \ldots, y_N]^\top$, we can condition the prior distribution on this training data. The resulting posterior distribution will only represent functions that are consistent with the training observations. Figure 3.2 illustrates three functions drawn from the prior distribution $\mathcal{N}(\mathbf{0}, \mathbf{K})$ and three functions drawn from the posterior distribution conditioned on four training data points.

Let $\mathbf{x} \in \mathbb{R}^{d_x}$ and $y \in \mathbb{R}$ be a pair of input and functional values that are not part of the training data. The joint prior distribution of the training data and the new pair is then the extension of Equation 3.2, such that

$$\begin{bmatrix} \mathbf{y} \\ y \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{k}(\mathbf{x}) \\ \mathbf{k}(\mathbf{x})^\top & k(\mathbf{x}, \mathbf{x}) \end{bmatrix}\right), \tag{3.3}$$

where $\mathbf{k}(\mathbf{x})$ is a $N \times 1$ vector with entries $\mathbf{k}(\mathbf{x})_i = k(\mathbf{x}, \mathbf{x}_i)$. Conditioning this joint prior distribution on the observed (training) data as detailed in [125], we obtain a predictive posterior

distribution $\mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x}))$, with mean and variance given by

$$\mu(\mathbf{x}) = \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{y}, \tag{3.4}$$
$$\sigma(\mathbf{x}) = k(\mathbf{x},\mathbf{x}) - \mathbf{k}(\mathbf{x})^\top \mathbf{K}^{-1}\mathbf{k}(\mathbf{x}). \tag{3.5}$$

These equations are the core of Gaussian process regression. Given a training dataset of corresponding inputs $\mathcal{X}$ and functional values $\mathcal{Y}$, a prediction of the functional value $\hat{y} = f(\mathbf{x})$ for a new location $\mathbf{x}$ can be obtained using Equation 3.4, the predictive mean. In fact, this expression is a linear combination of functional values $\mathbf{y} = [y_1, \ldots, y_N]^\top$ from the training data, weighted with kernel terms that quantify the correlation of the new input $\mathbf{x}$ with the known training locations $\mathbf{x}_i$ and the correlation within the training data. Equation 3.5, the predictive variance, can be seen as an assessment of prediction uncertainty. Note that this term does not depend on the training observations $\mathbf{y}$. As illustrated in Figure 3.2 b), the variance is proportional to the distance of the unknown input $\mathbf{x}$ to any known training location $\mathbf{x}_i$. Intuitively, the prediction uncertainty is high where known inputs are far away.

A crucial ingredient of Gaussian process regression is the chosen covariance function $k(\mathbf{x}_i, \mathbf{x}_j)$. Although the regression model is non-parametric, the type of covariance function that is used determines the properties of the functions that a Gaussian process can represent. See Figure 3.2 a)-b) for an illustrative example of random functions modelled by Gaussian processes with different covariance functions. In practice, the simple Gaussian covariance function stated above would be replaced by an extended version, such as

$$k(\mathbf{x}_i, \mathbf{x}_j) = \alpha^2 \exp\left(-\frac{1}{2\beta^2}(\mathbf{x}_i - \mathbf{x}_j)^2\right) + \gamma^2 \delta_{ij}, \tag{3.6}$$

where $\delta_{ij}$ equals 1 if $i = j$ and 0 otherwise [125]. Here, the values $\alpha$, $\beta$ and $\gamma$ are introduced that are referred to as *hyper*parameters, to emphasize that Gaussian process regression is still non-parametric in the classical sense. The hyperparameter $\alpha$ influences the overall variance of the functions modeled by the Gaussian process. The hyperparameter $\beta$ determines the so-called length-scale of the modeled functions. The length-scale can intuitively be seen as an analogy to the frequency of a signal. The hyperparameter $\gamma$ represents the assumed noise level in the training data. Referring to the illustration in Figure 3.2 a), where random functions are drawn form a Gaussian process with a given covariance function, changing $\alpha$ would affect the amplitude of the functions. An increasing value of $\beta$ would cause the functions to vary more slowly, small $\beta$ would result in more oscillations, as in Figure 3.2 b). The choice of $\gamma$ would influence how closely the functions pass through the given points in Figure 3.2 c).

Suitable values for the hyperparameters are obtained by optimization over the training dataset. We minimize the negative log marginal data likelihood [125, 41] with respect to the hyperparameters $\theta = (\alpha, \beta, \gamma)$:

$$-\log p(\mathbf{y}|\mathbf{X}, \theta) = \frac{1}{2}\mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y} + \frac{1}{2}\log|\mathbf{K}| + \frac{N}{2}\log 2\pi, \tag{3.7}$$

In this sum, only the first term contains the training data observations $\mathbf{y}$. This term measures how well the Gaussian process model fits the training data. The second term in the sum can be seen as a measure of model complexity and serves the purpose of preventing overfitting [41]. The last term is a normalization constant. For details on solving this optimization problem numerically, we refer the interested reader to [125].

### 3.1.3 Other Methods

In this work, we focus on kernel regression and Gaussian process regression for the following reasons. Kernel regression according to the Nadaraya-Watson estimator is conceptually intuitive, expressed in a closed form (Equation 3.1) and does not require any iterative training procedure. The kernel width parameter $\sigma$ can be estimated from the variance of the input values $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ in the training data. Gaussian process regression requires optimization in the training phase to determine the hyperparameters. However, the appeal of Gaussian process regression lies in its probabilistic character. It allows not only obtaining an estimate of the functional value $f(\mathbf{x})$ for an unknown input $\mathbf{x}$ (Equation 3.4), but also to assess the prediction certainty for any new input by means of the predictive variance of the Gaussian process (Equation 3.5). In our method presented in Chapter 4, we use the predictive variance to derive a weighting scheme that combines multiple Gaussian process regression models, each specialized on a specific type of human motion data.

Another non-linear regression method with interesting properties is based on the Relevance Vector Machine (RVM) [160]. The RVM essentially follows the formulation of the Support Vector Machine (SVM) for regression [17], but expressed in a probabilistic Bayesian framework. The RVM requires a costly iterative optimization procedure to find suitable values for a set of hyper-parameters from training data. However, after training, the RVM bases its predictions on a very small number of *relevance vectors* that are prototypical examples from the training dataset [160]. This sparsity of the RVM allows for high computational efficiency with large training datasets and provides remarkable generalization properties. At the same time, the RVM is probabilistic, such that a predictive certainty is available, similar to Gaussian process regression. In Chapter 5, we employ regression in a generative model of human motions and compare the performance of kernel regression with that of Relevance Vector Regression.

## 3.2 Dimensionality Reduction

The objective of dimensionality reduction is to find a new representation for a set of given data points that has fewer dimensions than their original representation. We will denote the original dataset as $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$, where $\mathbf{y}_i \in \mathbb{R}^D$ and $N$ is the total number of given points. The corresponding points after dimensionality reduction will be denoted as $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, such that $\mathbf{z}_i \in \mathbb{R}^d$ and $d < D$, often even $d \ll D$. We will also use the matrix notation $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top$ and $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N]^\top$. In the desired case, the dimensionality-reduced representation preserves properties of the data points, such as their variance or their neighborhood relations, as much as possible. However, since fewer dimensions are used after dimensionality reduction, handling and analyzing the data becomes more efficient.

### 3.2.1 Linear Methods

In this work, we focus on manifold learning methods, i.e. on non-linear dimensionality reduction techniques. In order to motivate the need for non-linear dimensionality reduction, we first introduce typical linear methods, such as Principal Component Analysis (PCA) and Multi-dimensional Scaling (MDS), and consider their effect on linear and non-linear datasets.

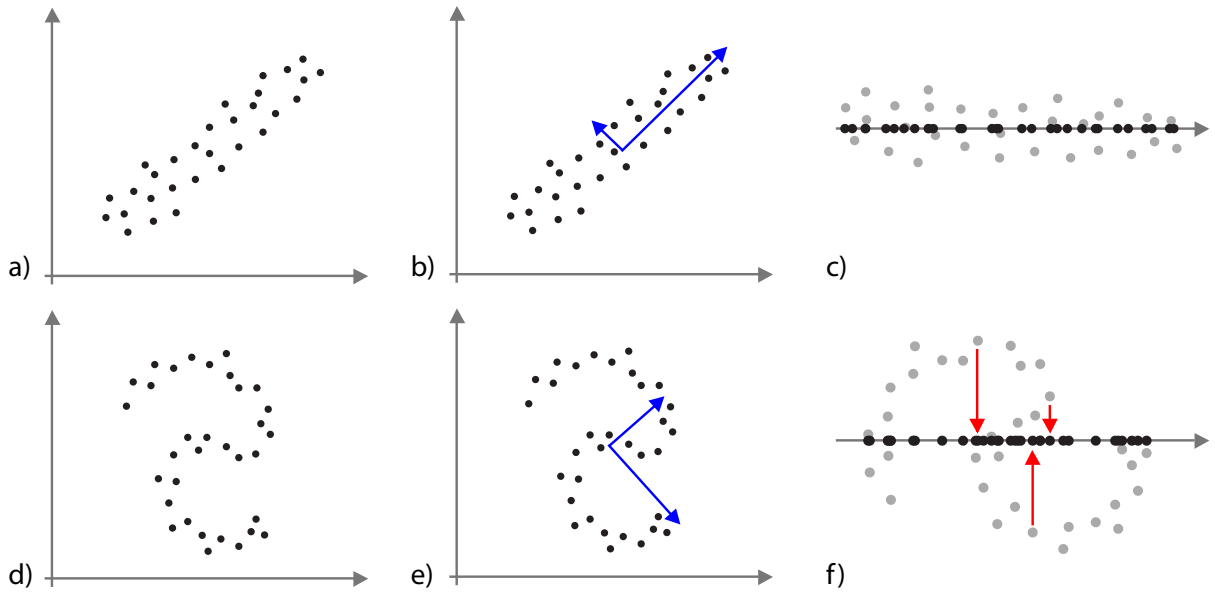



Figure 3.3: Illustration of the effect of dimensionality reduction using PCA. a) Approximately linear point distribution. b) Principal components (eigenvectors). c) Projection of original points onto first principal component. Points are distributed along the axis of largest variance. d) Non-linear point distribution. e) Principal components. f) Projection onto first principal component. Points from different parts of the structure underneath the data map together.

#### 3.2.1.1 Principal Component Analysis

PCA is a common linear dimensionality reduction method that can determine a linear projection mapping $\mathcal{Y}$ to new points $\mathcal{Z}$ in a subspace with different coordinate axes orientation. Dimensionality reduction is achieved by truncating this representation. After arranging the original points $\mathcal{Y}$ in a $N \times D$ matrix $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_N]^\top$, we subtract the mean from the points. Using the $N \times N$ centering matrix[1] $\mathbf{H}_N = \mathbf{I}_N - \frac{1}{N}\mathbf{1}\mathbf{1}^\top$, mean subtraction can be written concisely as $\mathbf{H}_N\mathbf{Y}$. Based on the $D \times D$ covariance matrix $\mathbf{C} = (\mathbf{H}_N\mathbf{Y})^\top(\mathbf{H}_N\mathbf{Y}) = \mathbf{Y}^\top\mathbf{H}_N\mathbf{H}_N\mathbf{Y} = \mathbf{Y}^\top\mathbf{H}_N\mathbf{Y}$, the linear PCA projection is determined from the eigenvalue decomposition $\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$. Here, $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues and $\mathbf{V}$ is an orthogonal matrix with the associated eigenvectors $\mathbf{v}_i$ as columns. The eigenvectors corresponding to the largest eigenvalues represent the directions of largest variance in the original point set $\mathcal{Y}$. Similarly, eigenvectors with less significant eigenvalues represent directions with smaller variance – see Figure 3.3. The coordinates of the dimensionality-reduced points can be obtained as the rows of the $N \times d$ matrix

$$\mathbf{Z} = (\mathbf{H}_N\mathbf{Y})\mathbf{V}_d, \tag{3.8}$$

where $\mathbf{V}_d$ is a $D \times d$ matrix with the $d$ eigenvectors corresponding to the $d$ largest eigenvalues of $\mathbf{C}$ as columns. The information that is omitted in the projection is limited to the $D - d$ dimensions in the dataset with smallest variance. With an appropriate choice of $d$, PCA thus allows explaining most of the variance of a dataset with significantly reduced dimensionality.

[1]Here, $\mathbf{I}_N$ is the $N \times N$ identity matrix and $\mathbf{1}$ denotes the $N \times 1$ vector consisting of ones.

#### 3.2.1.2 Multi-dimensional Scaling

As we have seen, the PCA projection is obtained from the eigenvalue decomposition of the covariance matrix for a given dataset. In some applications, we instead have a matrix of distances (or dissimilarities) between all pairs of points available. For dimensionality reduction, we might be more interested in preserving these distances than in maximizing variance. This idea leads to Multi-dimensional Scaling (MDS), an algorithm that can compute coordinates for a set of points in an arbitrary dimensionality $d$, such that the inter-point distances approximate those given in a Euclidean distance matrix. In particular, $d$ can be smaller than the dimensionality $D$ of the space where the Euclidean distances have been measured. As an illustrative example, MDS can be used to create a two-dimensional map of the cities of a country, given only the mutual distances between the cities.

Let $\mathbf{D}$ be an $N \times N$ matrix where the entries $d_{ij}$ are squared Euclidean distances between all pairs of points in a set $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ of $N$ points. The objective of MDS is to find the coordinates of a set of $d$-dimensional points $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$, such that the error

$$\mathcal{E}(\mathcal{Z}, \mathbf{D}) = \sum_{i=1}^{N} \sum_{j=1}^{N} d_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|^2 \tag{3.9}$$

is minimized. It can be shown that the solution to this minimization problem is determined from an eigenvalue problem [175]. To this end, the distance matrix $\mathbf{D}$ is first double-centered, resulting in $\mathbf{B} = -\frac{1}{2}\mathbf{H}_N\mathbf{D}\mathbf{H}_N$, where $\mathbf{H}_N$ is the centering matrix defined above. The eigenvalue decomposition of $\mathbf{B}$ into $\mathbf{B} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ leads to a diagonal matrix $\mathbf{\Lambda}$ containing the eigenvalues, and an orthogonal $N \times N$ matrix $\mathbf{U}$ with the corresponding eigenvectors $\mathbf{u}_i$ as columns. Assuming the eigenvalues are sorted in a decreasing order, we can extract as $\mathbf{\Lambda}_d$ the diagonal matrix with the $d$ largest eigenvalues and as $\mathbf{U}_d$ the $N \times d$ matrix with the corresponding eigenvectors as columns. The coordinates of the points approximating the Euclidean distances in $\mathbf{D}$ are then the rows of the $N \times d$ matrix

$$\mathbf{Z} = \mathbf{U}_d\mathbf{\Lambda}_d^{1/2}. \tag{3.10}$$

Instead of Euclidean distances, other dissimilarity measures can also be used, as discussed below. In fact, Euclidean distances in MDS lead to a solution that is identical to the PCA solution [175]. This relation becomes clear when observing that the double-centered squared distance matrix $\mathbf{B}$ can be rewritten as $\mathbf{B} = (\mathbf{H}_N\mathbf{Y})(\mathbf{H}_N\mathbf{Y})^\top$. This is the *Gram* matrix of the mean-subtracted points $\mathbf{H}_N\mathbf{Y}$. Recall that in PCA, the eigenvalue decomposition of the *covariance* matrix $\mathbf{C} = (\mathbf{H}_N\mathbf{Y})^\top(\mathbf{H}_N\mathbf{Y})$ is computed instead. As shown in [175], the non-zero eigenvalues of the Gram matrix are also the eigenvalues of the covariance matrix. The eigenvectors $\mathbf{u}_i$ of the Gram matrix and the eigenvectors $\mathbf{v}_i$ of the covariance matrix are related through $\sqrt{\lambda_i}\mathbf{u}_i = (\mathbf{H}_N\mathbf{Y})\mathbf{v}_i \ \forall i$. We can see that the left side of this equation corresponds to Equation 3.10, when considering an individual eigenvector, and the right side similarly corresponds to Equation 3.8, the PCA projection.

### 3.2.2 Manifold Learning

While PCA and MDS are highly valuable in many practical applications, they can only provide linear dimensionality reduction. In other words, a dataset with a non-linear structure

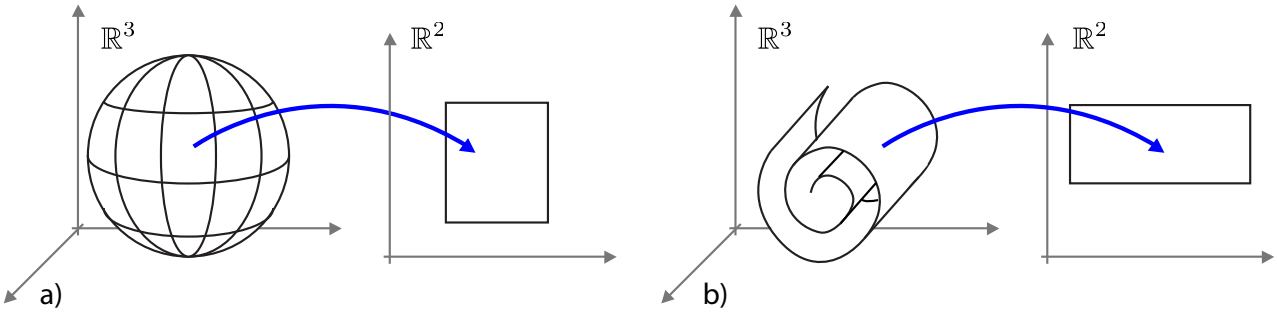



Figure 3.4: Examples of two-dimensional manifolds embedded in three-dimensional Euclidean space. a) A spherical surface in 3D with a local surface patch laid out as a 2D plane. c) A structure often called the Swiss roll in 3D and unfolded to a 2D plane.

will be projected to a linear subspace, with the risk of losing significant information. The variance of a dataset in each spatial dimension can be less informative than the underlying non-linear structure of the dataset, for instance through neighborhood relations among the points. However, PCA will in general not preserve this structure. Figure 3.3 illustrates the effects of PCA on an approximately linear dataset and a non-linear dataset that follows an underlying structure. The dataset in Figure 3.3 a) has a clear direction of largest variance that correlates with the structure of the data. However, the variance of the points in Figure 3.3 d) is not directly related to the non-linear structure of the dataset. Projecting the points onto this direction will disrupt the neighborhood relations between the points, as the *manifold* structure of the data will be disregarded. Similarly, MDS will not be able to account for the underlying manifold structure, as it approximates the Euclidean distances between all pairs of points, no matter whether they are in a spatial neighborhood.

Intuitively, a manifold is a generalization of a curve or surface to a higher dimension [94]. A manifold has an *intrinsic dimensionality* that can be thought of as the number of parameters required to uniquely specify a point on the manifold. In the context of dimensionality reduction, we are interested in situations where a manifold is embedded in a space that has more dimensions than the intrinsic dimensionality of the manifold. Manifold learning techniques attempt to reveal the manifold structure of a given dataset. The goal is to represent the data with as many dimensions as the dimensionality of the manifold, instead of using the full, probably wasteful dimensionality of its surrounding space.

More formally, a $d$-dimensional manifold is a topological space in which the neighborhood of each point is *homeomorphic* to the Euclidean space $\mathbb{R}^d$ [94]. Leaving the details of mathematical topology to the interested reader [40, 94], we will think of a $d$-dimensional manifold as a structure that is *locally similar* to the Euclidean space of $d$ dimensions. As an example, consider the earth as a sphere in 3D space. Locally, i.e. on the scale of meters and kilometers, we can easily represent the surface of the earth on a 2D plane. However, this local 2D structure globally resides in a 3D space. The earth surface can thus be thought of as a manifold with intrinsic dimensionality of two[2], embedded in a 3D surrounding space (Figure 3.4). In the context of data analysis, a typical observation is that points in a space of given

[2]Note that typical manifold learning methods might not be able to unfold a spherical 3D structure as in Figure 3.4 a), as there are no boundaries where a disconnection for unfolding could take place. The Swiss roll, illustrated in Figure 3.4 b), has clear boundaries that facilitate its unfolding to a 2D structure.

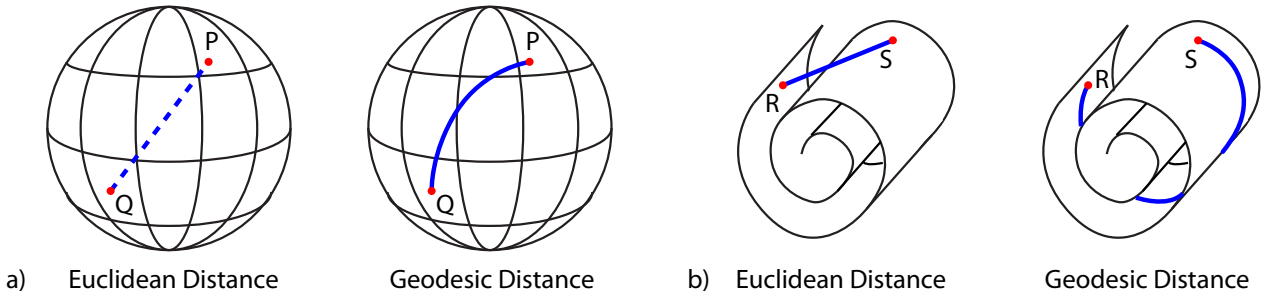



Figure 3.5: Illustration of the effect of measuring Euclidean and geodesic distances on 2D manifolds embedded in 3D space. a) Sphere: the Euclidean distance is measured through the interior of the sphere, the geodesic distance follows the surface. b) Swiss roll: the Euclidean distance takes a shortcut through space, the geodesic distance follows the planar manifold.

dimensionality lie on or close to a manifold of lower dimensionality. Consider human poses represented by vectors of joint angles. Each pose is a point in a space that has as many dimensions as there are joints. However, the vast majority of points in this space correspond to poses that are physiologically unfeasible. In particular, when dealing with a specific activity, such as walking, the corresponding poses will cluster near a manifold that has significantly fewer dimensions than the full pose space. It is often not easy to determine this intrinsic dimensionality of the data (see Section 3.2.3). In the walking example, we can assume that the manifold is one-dimensional, as a single parameter is sufficient to determine a particular pose in the cyclic walking pattern. In this work, we consider the three popular manifold learning methods Isomap, Locally Linear Embedding and Laplacian Eigenmaps.

#### 3.2.2.1 Isomap

Isomap, short for isometric feature mapping, was one of the first manifold learning approaches presented in the literature [156]. It is essentially an extension of Multi-dimensional Scaling. In the MDS algorithm, a point set is constructed based on a given matrix of Euclidean inter-point distances. However, these distances do not take into account the topology of the manifold structure that the points might be following. Figure 3.5 illustrates Euclidean distances in 3D between two points on a manifold and the corresponding distances along the 2D structure of the manifold. The distances along the manifold are referred to as *geodesic* distances[3]. In the Isomap algorithm, Euclidean distances between pairs of points in a dataset are replaced with geodesic distances, the remainder of the algorithm is identical to MDS.

A typical way of computing geodesic distances within a given set of points is by means of a graph. As before, let $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$ be a dataset of $N$ points with $\mathbf{y}_i \in \mathbb{R}^D$. We define a graph, where the given points are the vertices and edges are introduced whenever two points $\mathbf{y}_i$ and $\mathbf{y}_j$ are neighbors in the manifold structure. Such a graph can be represented by an $N \times N$ edge weight matrix $\mathbf{W}$ with entries

$$w_{ij} = \begin{cases} \|\mathbf{y}_i - \mathbf{y}_j\| & \text{if} \quad \mathbf{y}_i \in \mathcal{N}(\mathbf{y}_j) \\ \infty & \text{otherwise.} \end{cases} \tag{3.11}$$

[3]The term comes from geodesy, the discipline where the size and shape of the earth are studied. A geodesic is originally the shortest path between two points on the earth's surface, given by a segment of a large circle. In the general sense we are considering, geodesic distances are defined on arbitrary shapes in arbitrary dimensions.

Here, we assume that the geodesic distances can be approximated locally (i.e. between adjacent points) by means of Euclidean distances. The neighborhood $\mathcal{N}(\mathbf{y}_i)$ for a point $\mathbf{y}_i$ can be given by the $k$ nearest neighbors, or by an $\epsilon$-ball around $\mathbf{y}_i$. In practice, setting up the weight matrix amounts to computing a pair-wise distance matrix $\mathbf{D}$, followed by setting entries to $\infty$ wherever appropriate. Points that are located on topologically separated parts of the manifold should not share a common edge in the graph. In the example of Figure 3.5 c)-d), there would be no direct connection between the points $R$ and $S$ in the graph. Instead, the graph would approximate the surface of the Swiss roll.

Given the edge weight matrix $\mathbf{W}$, we can approximate the geodesic distances between all points in $\mathcal{Y}$ using all-pairs shortest paths algorithms, such as Dijkstra's algorithm [28] or the Floyd-Warshall algorithm [28]. These algorithms compute inter-point distances by following shortest paths along the edges in the graph and summing up the Euclidean edge weights. We then store the squares of the approximated geodesic distances in an $N \times N$ matrix $\mathbf{D}$. This matrix is then used as an input to MDS, instead of the Euclidean distance matrix. The remaining steps of MDS, the eigenvalue decomposition and the extraction of the final points from the eigenvalues and eigenvectors, are identical to MDS as described above.

#### 3.2.2.2 Locally Linear Embedding

Similar to Isomap, Locally Linear Embedding (LLE) also focuses on local neighborhoods of points in a given dataset [130]. However, LLE does not require costly shortest path computations. The key concept of LLE is that the local surrounding of any point $\mathbf{y}_i \in \mathcal{Y}$ is linear and, thus, that every such point can be represented as a linear combination of its neighboring points. For dimensionality reduction, a mapping is determined that preserves the local linear combination weights in a lower dimensionality. The assumption is that the local linear combinations of points will overlap and therefore approximate the non-linear manifold on a global scale. More formally, the objective of LLE is to find the entries $w_{ij}$ of an $N \times N$ weight matrix $\mathbf{W}$, such that the reconstruction error

$$\varepsilon(\mathbf{W}) = \sum_{i=1}^{N} \|\mathbf{y}_i - \sum_{\mathbf{y}_j \in \mathcal{N}(\mathbf{y}_i)} w_{ij}\mathbf{y}_j\|^2, \tag{3.12}$$

is minimized. Here, $\mathcal{N}(\mathbf{y}_i)$ is the set of $k$ nearest neighbors of a point $\mathbf{y}_i$. The optimal solution is sought under the constraint that the rows of $\mathbf{W}$ sum up to one. This means that every point is reconstructed in a convex combination of its neighbors, leading to a neighborhood representation that does not depend on the orientation of any coordinate frame [130]. The entries of the weight matrix can actually be computed in a closed form based on local covariance matrices for each considered point neighborhood. Given the weights, the objective is to find a set of points $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ with $\mathbf{z}_i \in \mathbb{R}^d$ that satisfy the linear combination weights in $d$ dimensions. The corresponding error to be minimized for point coordinates $\mathbf{z}_i$, while keeping the weights fixed, is

$$\mathcal{E}(\mathcal{Z}, \mathbf{W}) = \sum_{i=1}^{N} \|\mathbf{z}_i - \sum_{\mathbf{z}_j \in \mathcal{N}(\mathbf{z}_i)} w_{ij}\mathbf{z}_j\|^2. \tag{3.13}$$

As detailed in [130], this cost function can be rewritten such that the optimal solution is determined from the eigenvalue decomposition of the matrix $(\mathbf{I} - \mathbf{W})^\top(\mathbf{I} - \mathbf{W}) = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$.

The coordinates of the desired points are the rows of the $N \times d$ matrix $\mathbf{Z} = \mathbf{V}_d$, where $\mathbf{V}_d$ consists of the columns of $\mathbf{V}$ corresponding to the $d$ *smallest* non-zero eigenvalues. Note that in the case of PCA, we are looking for the eigenvectors corresponding to the *largest* eigenvalues, as PCA aims at maximizing variance.

#### 3.2.2.3 Laplacian Eigenmaps

Laplacian Eigenmaps is a manifold learning method that builds upon spectral graph theory, where the so-called graph Laplacian plays a crucial role [14]. The graph Laplacian is a matrix that encodes information about the structure of the graph, e.g. its connectedness properties. As in the case of Isomap, we consider the graph with the given points $\mathcal{Y}$ as vertices and with edges between points that are neighbors. Let $\mathbf{W}$ be an $N \times N$ matrix of edge weights

$$w_{ij} = \begin{cases} e^{-\|\mathbf{y}_i - \mathbf{y}_j\|^2/2\sigma^2} & \text{if} \quad \mathbf{y}_i \in \mathcal{N}(\mathbf{y}_j) \\ 0 & \text{otherwise.} \end{cases} \tag{3.14}$$

As before, the neighborhood $\mathcal{N}(\mathbf{y}_i)$ can be given by the $k$ nearest neighbors of $\mathbf{y}_i$ or by an $\epsilon$-ball around $\mathbf{y}_i$. The weight for an edge connecting neighboring points is expressed by a Gaussian kernel with a width parameter $\sigma$ that either has to be specified or that can be estimated based on given data [177]. Note that, as opposed to using Euclidean distances, the value of the Gaussian kernel will increase when points are more similar. As an alternative, the kernel-based weight in Equation 3.14 can also be replaced by a binary value of 1.

The objective of Laplacian Eigenmaps is to find the points $\mathcal{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_N\}$ in $d$ dimensions, such that $\mathbf{z}_i$ and $\mathbf{z}_j$ are close whenever $\mathbf{y}_i$ and $\mathbf{y}_j$ are close. This objective can be written as a minimization of the error

$$\mathcal{E}(\mathcal{Z}, \mathbf{W}) = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{ij} \|\mathbf{z}_i - \mathbf{z}_j\|^2. \tag{3.15}$$

In the summation, the contribution of the distance between $\mathbf{z}_i$ and $\mathbf{z}_j$ is high whenever the corresponding weight $w_{ij}$ is large, i.e. whenever the corresponding original points $\mathbf{y}_i$ and $\mathbf{y}_j$ are close. In the minimization, the distance between these points will thus be penalized stronger. The minimization problem in Equation 3.15 can be formulated as an eigenvalue problem by means of the graph Laplacian. Typically specific constraints are included, e.g. to prevent the trivial solution of $\mathbf{z}_i = \mathbf{0}\ \forall i$. The graph Laplacian is the matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal degree matrix with entries $d_{ii} = \sum_j w_{ij}$. Here, the degree matrix quantifies the connectivity for every point, i.e. the degree to which it is connected to other points. The desired embedding points $\mathcal{Z}$ can be determined from the solution of the generalized eigenvalue problem $\mathbf{L}\mathbf{v}_i = \lambda \mathbf{D}\mathbf{v}_i$. In matrix form, the desired points are $\mathbf{Z} = \mathbf{V}_d$, where $\mathbf{V}_d$ is an $N \times d$ matrix with columns $\mathbf{v}_i$ being the eigenvectors corresponding to the $d$ non-zero smallest eigenvalues. The low-dimensional points are the rows of this matrix.

### 3.2.3 Practical Considerations

The proposed manifold learning methods share the common objective of non-linear dimensionality reduction, but each of the methods has particular properties that make the methods

suitable for different settings. Each of the approaches is based on investigating neighborhood relations among the given points in their original, high-dimensional representation. As opposed to PCA, the manifold learning methods do not project the points onto a linear subspace, but find a low-dimensional point representation that implicitly minimizes some type of error. In all cases, this minimization problem is solved in closed form by means of an eigenvalue (or spectral) decomposition. For this reason, these methods are referred to as *spectral* methods.

#### 3.2.3.1 Preserved Properties of Original Data

The error that is minimized reveals which properties of the original dataset each of the techniques tries to preserve. Isomap preserves the geodesic distances between all points along the underlying manifold. In LLE, the local neighborhood structure for all points is preserved in the form of the local linear reconstruction weights. Laplacian Eigenmaps preserves neighborhood relations in the sense that low-dimensional points are positioned close to each other whenever the corresponding original points are close. We can therefore see that Isomap is a *global* method, in that it seeks a solution based on global constraints that involve all points simultaneously. LLE and Laplacian Eigenmaps are *local* methods that aggregate constraints on a local level, i.e. constraints between subsets of neighboring points. Local methods can be favorable when it is hard to satisfy all global constraints. An approximate global solution might, in such a case, be less valuable than one that fulfills the local constraints precisely. Human motion data is an example where local neighborhood properties between similar poses are crucial, while the relations between distant poses are of minor interest.

#### 3.2.3.2 Implicit Assumptions

While relatively simple to implement, manifold learning methods are based on a number of assumptions about the data that should be fulfilled for meaningful results. First and foremost, the so-called manifold assumption simply states that the data is indeed distributed on or close to a manifold embedded in a higher-dimensional space. This implies that the data points vary smoothly and continuously along the manifold [100]. In the case that this assumption is not fulfilled, manifold learning will not necessarily give results that are superior to a linear dimensionality reduction method, such as PCA. Determining whether a dataset fulfills the manifold assumption is, however, not straightforward. Intuitively, data often follows a manifold when the dominant type of transformation in the data only affects a limited number of underlying parameters. For example, consider the following examples of suitable data:

- Human motion data, e.g. represented by sets of joint angles, for selected activities, such as walking on a treadmill. Several authors have shown that feasible human poses lie on a manifold within the high-dimensional space of pose parameters [42, 72, 98, 19].

- Image datasets with a smooth variation in imaging parameters, such as viewpoint or lighting [156, 119, 120]. Manifold learning can distribute the images in a low-dimensional space such that the dimensions represent the directions of perceptual change in the dataset. Images are also ordered by the degree of variation for each parameter. New images can be synthesized by interpolating the given images along the manifold.

Figure 3.6: Temporal super-resolution by means of manifold learning. *Top row*: Frames inserted between two existing frames of a video sequence by insertion of new frames along the manifold structure connecting the given frames. *Bottom row*: Linear interpolation between the two given frames based on image intensities. Picture from [119].

- Videos where consecutive frames are characterized by a smooth variation of the depicted object. Examples include moving humans or animals, or faces with a varying expression. Manifold learning allows identifying repetitive scenes or achieving natural looking temporal super-resolution by interpolating frames along the underlying manifold instead of interpolating purely based on pixel intensities (see Figure 3.6) [130, 156, 118, 150, 10].
- Medical image datasets where individual images represent a given anatomical structure under varying physiological parameters, such as a beating heart or the lung under breathing deformations [170, 70, 50]. Manifold learning allows estimating the physiological parameters from the images only, e.g. for image-based respiratory breathing gating.
- Patches from medical images representing the same anatomy but obtained using different modalities, such as computed tomography (CT) or magnetic reconance imaging (MRI) [169]. Although the intensities of identical structures can vary non-linearly between the modalities, manifold learning allows to relate image patches across the modalities.

Another implicit assumption about the data made by manifold learning algorithms is that the given points provide a sufficiently dense sampling of the manifold [100]. When the number of available points is small with respect to the complexity of the manifold structure, the computed embedding can degenerate, e.g. resulting in disconnected components. Similarly, a non-uniform sampling of the manifold, with local aggregations of points, can lead to clustering in the low-dimensional representation.

#### 3.2.3.3 Out-of-Sample and Reconstruction Mappings

The manifold learning methods presented in this work have in common that they neither provide an explicit out-of-sample mapping, nor an explicit reconstruction mapping. This means that, once a manifold embedding for a given set of point has been computed, additional points cannot be added to the embedding in a principled way (out-of-sample mapping). On the other hand, it means that there is no transformation that maps any new point in embedding space to the original, high-dimensional representation (reconstruction mapping). An out-of-sample mapping exists in the case of PCA, where the given point set is used to compute a transformation that is also valid for new points. This transformation is given by the matrix of $d$

retained eigenvectors $\mathbf{V}_d$ in Equation 3.8. Isomap, LLE and Laplacian Eigenmaps directly extract the low-dimensional point coordinates from the eigenvectors of the respective eigenvalue decomposition. Mapping new points to an existing low-dimensional representation and back is, however, a common problem. A straightforward application is to project a query point into an embedding space for nearest neighbor classification.

Therefore, several out-of-sample extensions have been proposed in the literature for different manifold learning algorithms [60, 25]. For instance, in [25], an out-of-sample mapping for Laplacian Eigenmaps is presented that takes the form of the Nadaraya-Watson estimator (see Section 3.1.1). A new point is thus mapped to an existing embedding by treating the known original points $\mathcal{Y}$ and their dimensionality-reduced counterparts $\mathcal{Z}$ as outputs and inputs of a kernel regression problem. In fact, an analogous reconstruction mapping, projecting embedding points back to their high-dimensional representation, is also derived in [25]. This approach can also be used in conjunction with manifold learning methods other than Laplacian Eigenmaps. The Locality Preserving Projections (LPP) method [60] provides an explicit transformation mapping based on a similar derivation as in Laplacian Eigenmaps.

#### 3.2.3.4 Choice of Parameters

Manifold learning methods are largely non-parametric and data-driven, but a few parameters have to be considered before applying one of the methods to a given dataset. First and foremost, most algorithms require a specification of the dimensionality $d$ for the manifold embedding. This dimensionality should reflect the *intrinsic dimensionality* of the data, i.e. the number of parameters that are varied in the underlying process that generates the data. A classical example is an image dataset showing a 3D object from various orientations along two spatial axes [119, 156]. The full dimensionality of this dataset is $w \times h$, where $w$ and $h$ are the width and height of the images in pixels. However, the intrinsic dimensionality is $d = 2$, corresponding to the two underlying parameters that are varied in the dataset. Figure 3.7 shows the dataset after arrangement according to the coordinates of individual images in a 2D manifold embedding. Apart from the intuition, one way of estimating the intrinsic dimensionality in the case of Isomap, is to consider the decrease of residual error (Equation 3.9) when increasing the dimensionality [156]. Typically, there is a dimensionality value where adding dimensions to the manifold embedding does not improve the residual error any more. For methods other than Isomap, there is no guarantee that the error to be minimized (Equations 3.13 and 3.15 for LLE and Laplacian Eigenmaps) decreases with increasing dimensionality. Several heuristic methods have been proposed for estimating the intrinsic dimensionality of a dataset, e.g. based on graph-theoretic techniques [30, 29] or based on maximum likelihood estimation [95].

Another adjustable setting for all considered manifold learning methods is the definition of the neighborhood relation $\mathcal{N}(\mathbf{y}_i)$. This relation determines which points in the original dataset are neighbors for a given point $\mathbf{y}_i$. Changing the way the neighborhood is computed (e.g. $k$-nearest neighbors or $\epsilon$-ball) and modifying the neighborhood range (i.e. $k$ or $\epsilon$, respectively), can significantly impact the resulting manifold embedding. The importance of the neighborhood reflects that manifold learning methods generally rely on measuring dissimilarities only for close points. In many practical cases, only small dissimilarities are meaningful and large dissimilarities do not help in determining connections between given points. As an example, consider Figure 3.7, where distances between similar images are proportional to the

degree of parameter change, while distances between completely different images are mainly determined by a random overlap of intensity patterns. Similar to the intrinsic dimensionality parameter, a suitable choice of neighborhood is often also based on an intuition about the properties of the given data. Heuristic approaches also exist that adapt the neighborhood size based on the local density of the given points [177].

Once the way of determining a point neighborhood has been selected, a remaining open question is how to measure similarities between points. Point similarities are measured, both, to determine whether points are neighbors and to analyze the local relations between neighboring points. The straightforward choice of similarity measure is to use Euclidean distances. However, based on particular properties of the data at hand, other similarity measures might be more suitable [120]. For grayscale images, computing the sum of squared differences (SSD) between two images is equivalent to the Euclidean distance [170]. In medical imaging, this metric is sufficient for images from one imaging modality, but when multi-modal datasets are considered, more sophisticated similarity measures should be employed, such as normalized cross-correlation (NCC) or mutual information (MI). Another application-specific similarity measure is used in [10], where a method for analyzing endoscopic videos is presented. A sophisticated similarity measure is required to achieve rotational invariance with respect to the axis of the endoscope. An investigation of similarity measures for image-based manifold learning is given in [150]. In the case of human motion analysis, Euclidean distances, or variations thereof, between vectors of joint angles representing body poses are often used with satisfactory results [42, 100].

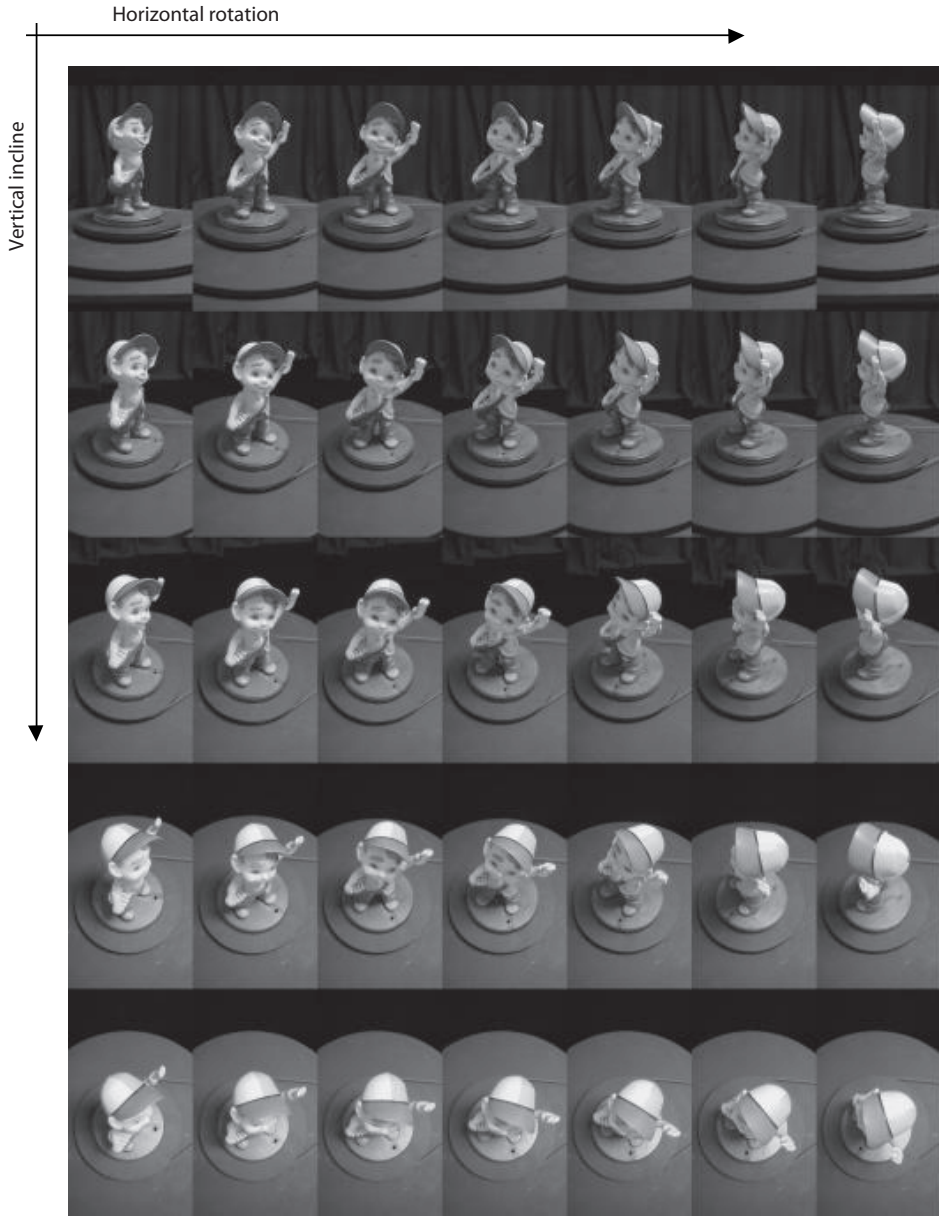



Figure 3.7: Images of an object under varying viewing angles around two spatial axes. The full dimensionality of this image space is $64 \times 32$, determined by the number of pixels in each image. The intrinsic dimensionality corresponds to the two varied underlying orientation parameters. Images are arranged according to their coordinates in a 2D manifold embedding using Isomap and post-processing [119], given an unordered input. Picture adapted from [119].

# Part II

# Activity Recognition and Pose Estimation

Chapter 4

# Regression for Human Motion Analysis

## 4.1 Introduction

In human pose estimation, we are given some type of limited observations $\mathbf{x} \in \mathbb{R}^{d_x}$ of a moving person, such as features from a depth image or inertial sensor measurements, and would like to predict the unknown full-body pose $\mathbf{y} \in \mathbb{R}^{d_y}$. Let $F(\mathbf{y}) = \mathbf{x}$ denote the natural functional mapping that represents the imaging process or the physical measurement process of inertial sensors. This mapping is lossy, as it projects high-dimensional poses to lower-dimensional observations, and is thus non-bijective. In particular, this means that the inverse mapping of $F$, the observation-to-pose mapping $F^{-1}(\mathbf{x}) = \mathbf{y}$ is non-functional. Intuitively, different body poses can have an almost identical appearance in images or sensor measurements, and determining a unique pose from such ambiguous observations is difficult. Yet this is exactly what discriminative pose estimation methods do by modeling the observation-to-pose mapping. In this work, we denote the direct mapping from observations to poses as $f(\mathbf{x}) = \mathbf{y}$.

The attractiveness of the discriminative approach, as opposed to the generative counterpart that we focus in Chapter 5, lies in its conceptual and computational simplicity. Learning the observation-to-pose mapping allows for a direct prediction of full-body poses, given only an appearance descriptor that can be easily extracted from each image or sensor measurement. In contrast, generative methods imply a search in pose space for the pose that best fits the observations. This typically requires statistical inference methods, such as a particle filter, or costly iterative optimization. Discriminative methods also do not explicitly model and track a system state but predict poses independently from each incoming observation. As a result, there is no need for initialization and pose estimation can quickly recover from failure immediately after an outlier observation.

Discriminative prediction of human poses has been studied for various types of observations, such as image features or 3D reconstruction data [53, 56, 145, 166]. Typically, non-linear regression methods are used for modeling the observation-to-pose mapping in a non-parametric way. The general idea is to use training datasets of corresponding full-body poses and feature descriptors for learning a regressor that allows predicting poses, given only novel feature descriptors. In our setting, we record synchronized full-body poses (using a motion capture system) and inertial sensor data. In contrast to previous work, we train multiple regressors, each for a distinct activity, and devise a mechanism for simultaneous activity recognition.

## 4.2 Related Work

Here, we give an overview of related work in the field of discriminative human pose estimation and activity recognition. Existing approaches can mainly be categorized based on the type of input observations and the regression method that are used. We then outline other work on sensor-based human motion analysis that is not necessarily related to machine learning.

**Input Observations.** Agarwal and Triggs [3, 4] learn a regression mapping from human silhouettes in images, represented using shape context descriptors, to full-body poses. Zhao et al. [179, 178] use SIFT features, extracted from multiple perspective images of a person, as an input for regression. Sun et al. [155] capture human motion using a multi-camera 3D reconstruction system and predict full-body poses from voxel-based features. Fossati et al. [45] use trajectories of body parts tracked in monocular videos as an input for regression. Okada and Soatto [113] propose a method that effectively selects Histogram of Oriented Gradients (HOG) features from images for discriminative pose estimation. Girshick et al. [51] extract distance-based features from Kinect depth data for general-activity regression of human poses. Our work is the first to use inertial sensor data similarly to optical features for estimating human full-body poses based on regression.

**Regression Methods.** Many existing discriminative approaches focus on variants of Relevance Vector Machine regression (RVM) [4, 155, 158] and Gaussian process regression (GPR) [45, 164, 179]. The basic regression techniques are often complemented with application-specific extensions to alleviate the ambiguity of the observation-to-pose mapping. The authors of [45] use a trained GPR model to obtain an initial pose estimate from image features and add a refinement step that optimizes the match between the predicted pose and the observations using a particle filter. In [4], an RVM is used for modeling the observation-to-pose mapping and, in addition, to obtain an auto-regressive mapping within pose space. This second regressor introduces temporal consistency by relating pose predictions to previous poses. In [113], multiple locally linear regressors are trained, based on the assumption that different visual features are relevant for different motion patterns. A support vector classifier is first employed, given a set of HOG features, to select one of the local regressors that then predicts a full-body pose. In [51], a random forest is trained for regression from depth features to full-body poses. Bissacco et al. [18] combine motion and appearance information, represented using Haar-like features, and use multiple regressors for pose prediction.

Except for [51], all methods mentioned so far focus on reconstructing poses for a single type of activity, such as walking. The reason for this limitation is that the observation-to-pose mapping $f(\mathbf{x}) = \mathbf{y}$ becomes even more ambiguous, as the number of considered poses increases. In fact, training a single regressor on poses corresponding to multiple activities is hardly feasible, except when enormous amounts of training data are available and a method is used that can efficiently handle the learned information at runtime [51]. Our approach is to combine multiple regressors, each trained on a specific activity, into one discriminative model. We propose two mechanisms to determine the contribution of each activity-specific regressor. One of these mechanisms uses the predictive variance of each GPR model, the other adds a Support Vector Machine (SVM) classifier trained to assign sensor measurements to activities.

**Inertial Sensors as Input Modality.** Human motion analysis based on wearable inertial sensors has been targeted by several authors. However, due to the sparsity of inertial sensor data, the focus was mostly on activity recognition [8, 80], as opposed to reconstructing full-body poses of a person. Existing methods for full-body tracking using inertial sensors [129, 167, 182, 183] try to recover the pose directly from the measurements, relying on physical models, inverse kinematics and stochastic filtering. For instance, Zhou et al. [182] use accelerometers to compute the pose of one arm based on kinematic constraints. Roetenberg et al. [129] fuse inertial and ultrasonic sensor streams using a Kalman filter, allowing to track position and orientation of the four limbs and the torso. A Kalman filter-based approach is also described by Zhu et al. [183], where magnetic sensors are used for tracking a single arm. Our method removes the need for complex, physical motion models, since learned prior knowledge on human motion is used to cope with the noisy and ambiguous sensor data.

Slyper and Hodgins [147] compare accelerometer measurements to a database of poses and replay motion sequences matching the measurements in an approximation of the true motion. The method proposed in [167] is able to track full-body poses using ultrasonic sensors and accelerometers. However, their approach relies on a combination of multiple different sensors that can be cumbersome for the user. A similarly low number of inertial sensors as in our case is used by Pons-Moll et al. [122]. In their recent work, orientation sensors serve to resolve ambiguities in a monocular video-based full-body pose tracking approach. While their system allows for activity-independent body tracking, the sensors only provide assistive constraints to complement the camera images. Our method does not require visual observations and allows for simultaneous activity recognition and full-body pose tracking from inertial sensors.

## 4.3 Activity Recognition and Pose Estimation using Inertial Sensors

In this section, we describe our discriminative approach for human pose estimation and simultaneous activity recognition from inertial sensor data. Challenges of such an an approach are mainly due to the properties of inertial sensors that typically exhibit high drift rates and provide only a limited number of constraints. When employing four inertial sensors, one on each limb of a person, we are able to record a continuous stream of a mere 12 values (3 orientation angles per sensor). Instead of trying to learn the observation-to-pose mapping for poses of multiple activities, we propose to learn separate, person-specific models for each considered activity. The training data consists of corresponding sets of sensor data and full-body joint angles that we record offline using motion capture. In the testing phase, only the sensor data is available for pose estimation, allowing the person to move freely and follow their daily routine (Figure 4.1). We introduce different techniques for combining the predictions of the activity-specific motion models and to classify the activities.

### 4.3.1 Method Overview

Let $\mathbf{x} \in \mathbb{R}^{d_x}$ be a vector containing the concatenated instantaneous measurements of all inertial sensors. Also, let $\mathbf{y} \in \mathbb{R}^{d_y}$ ($d_x \ll d_y$) denote a vector of joint angles that fully determine a pose in terms of an articulated body model. We assume that we are considering a set of $M$

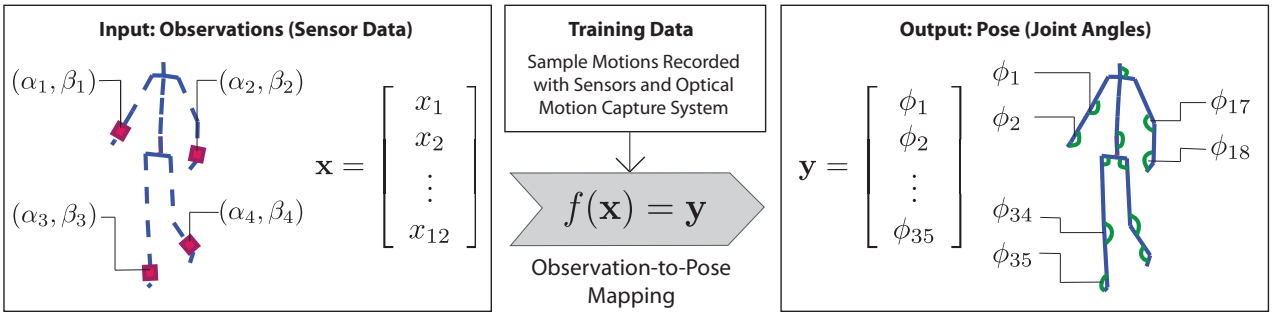



Figure 4.1: Schematic of the proposed pose estimation approach. A person is equipped with a few inertial sensors measuring orientation angles. Full body pose is predicted using a regressor that has been trained on the mapping between the sensor data $\mathbf{x}$ and the joint angles $\mathbf{y}$.

activities of interest, identified with an index $\alpha \in \{1, \ldots, M\}$. The sensor measurements for a specifc activity $\alpha$, consisting of $N_\alpha$ poses, are denoted as $\mathcal{X}^\alpha = \{\mathbf{x}_1^\alpha, \ldots, \mathbf{x}_{N_\alpha}^\alpha\}$ or arranged in a matrix $\mathbf{X}^\alpha = [\mathbf{x}_1^\alpha, ..., \mathbf{x}_{N_\alpha}^\alpha]^\top$. The corresponding pose vectors $\mathcal{Y}^\alpha = \{\mathbf{y}_1^\alpha, \ldots, \mathbf{y}_{N_\alpha}^\alpha\}$ are written as a matrix $\mathbf{Y}^\alpha = [\mathbf{y}_1^\alpha, ..., \mathbf{y}_{N_\alpha}^\alpha]^\top$. Let $\bar{\mathbf{y}}^\alpha$ denote the mean joint configuration for activity $\alpha$ and let $\bar{\mathcal{Y}}^\alpha$ be the set of the respective mean-subtracted pose vectors $\mathbf{y}_i^\alpha - \bar{\mathbf{y}}^\alpha$.

For each activity $\alpha$, we train a Gaussian process regression model $\mathcal{G}_\alpha = \{\mathcal{X}^\alpha, \bar{\mathcal{Y}}^\alpha, \theta_\alpha\}$ by optimizing the hyperparameters $\theta_\alpha$ according to Equation 3.7. Each model $\mathcal{G}_\alpha$ represents the observation-to-pose mapping $f_\alpha(\mathbf{x}_i^\alpha) = \mathbf{y}_i^\alpha$ that is valid for one particular activity. Given new sensor data $\mathbf{x}_t$ at time $t$ that is not part of the training data, we are interested in determining the activity $\hat{\alpha}_t$ the person is performing and in estimating the full-body pose $\hat{\mathbf{y}}_t$. We propose the following two approaches for combining the predictions of all $M$ activity-specific GPR models and to obtain an activity classification:

- The prediction of each GPR model is weighted according to the predictive variance of the model for the given sensor input. The predictive variances of all GPR models allow determining the activity $\hat{\alpha}$ performed by the person.

- An SVM classifier trained on sensor data is used to determine the activity $\hat{\alpha}$ performed by the person. The pose predictions of the individual GPR models are weighted in a sliding window according to the SVM classification.

Figure 4.2 shows how the inputs $\mathcal{X}$ and the pose predictions $\mathcal{Y}$ are related by the regression mapping. We will use this type of visualization throughout the thesis to allow for an easy comparison of the presented methods in terms of spaces and the corresponding mappings.

### 4.3.2 Activity Classification and Pose Inference

Since we train multiple GPR models $\mathcal{G}_\alpha$, one for each activity considered in the training dataset, we need to define how to obtain a single full-body pose estimate $\hat{\mathbf{y}}_t$ for each new sensor observation $\mathbf{x}_t$. Let us first see how to obtain the prediction $\hat{\mathbf{y}}_t^\alpha$ for one individual GPR model $\mathcal{G}_\alpha$. We use the predictive mean $\mu_\alpha(\mathbf{x}_t)$ of the model, in analogy to Equation 3.4 but extended to the vector-valued case. The predictive mean is essentially a weighted combination

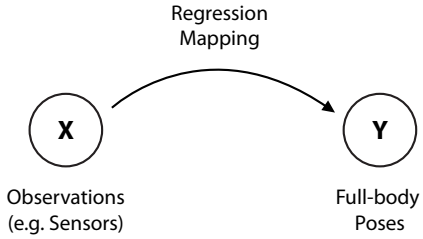



Figure 4.2: Diagram illustrating the relationship between observations ($\mathcal{X}$) and full-body poses ($\mathcal{Y}$), induced by the learned regression mapping. Compare also to Figures 5.4 and 7.4.

of the joint angle configurations in the respective training dataset:

$$\hat{\mathbf{y}}_t^\alpha := \mu_\alpha(\mathbf{x}_t) = \bar{\mathbf{y}}^\alpha + \mathbf{Y}^{\alpha\top}\mathbf{K}_\alpha^{-1}\mathbf{k}^\alpha(\mathbf{x}_t). \tag{4.1}$$

The notation $\hat{\mathbf{y}}_t^\alpha$ implies that this is only an estimate of the true, unknown joint angle configuration $\mathbf{y}_t$ corresponding to $\mathbf{x}_t$, as given by the model for activity $\alpha$. Here, $\mathbf{k}^\alpha(\mathbf{x})_i = k(\mathbf{x}, \mathbf{x}_i^\alpha)$ and $\mathbf{K}_\alpha$ is an $N_\alpha \times N_\alpha$ matrix with entries $k_{ij}^\alpha = k(\mathbf{x}_i^\alpha, \mathbf{x}_j^\alpha)$. We use a squared exponential covariance function $k(\cdot,\cdot)$ as defined in Equation 3.6. Equation 4.1 can be efficiently evaluated online for each incoming sensor reading, since $\mathbf{K}^{-1}$ can be precomputed. Note that the equation is defined separately for each activity $\alpha \in \{1, \ldots, M\}$ in the training dataset.

#### 4.3.2.1 Model Averaging using Predictive Variances

One possible approach to combine the full-body pose predictions of all activity-specific GPR models is by exploiting the ability of Gaussian processes to give an assessment of prediction uncertainty. This assessment allows telling which activity a particular observed pose most likely belongs to. Following Equation 3.5, we can write the predictive variance of model $\mathcal{G}_\alpha$ given a new sensor observation $\mathbf{x}_t$ as

$$v_t^\alpha := \sigma_\alpha(\mathbf{x}_t) = k(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{k}^\alpha(\mathbf{x}_t)^\top \mathbf{K}_\alpha^{-1}\mathbf{k}^\alpha(\mathbf{x}_t). \tag{4.2}$$

The quantity $\sigma_\alpha(\mathbf{x}_t)$ can be interpreted as a measure of how close the new sensor reading $\mathbf{x}_t$ is to the values in the training dataset of the model $\mathcal{G}_\alpha$. We can now define a simple classification rule that determines the activity $\hat{\alpha}_t$ that most likely matches the new pose. We simply select the activity for which the corresponding GPR model has the lowest predictive variance,

$$\hat{\alpha}_t = \underset{\alpha\in\{1,\ldots,M\}}{\arg\min}\ v_t^\alpha. \tag{4.3}$$

Chen et al. [27] present a scheme for combining predictions of several GPR models, where each model is trained on a different sub-part of one training dataset (bootstrap aggregation). We propose to transfer this model averaging scheme to our setting of multiple GPR models that are trained on distinct, activity-specific datasets. The pose estimate $\hat{\mathbf{y}}_t$ for a new sensor measurement $\mathbf{x}_t$ is then a weighted combination of the individual predictions $\hat{\mathbf{y}}_t^\alpha$ from all models,

$$\hat{\mathbf{y}}_t = \sum_{\alpha=1}^{M} w_t^\alpha \cdot \hat{\mathbf{y}}_t^\alpha, \tag{4.4}$$

where the weight $w_t^\alpha$ for a GPR model with index $\alpha$ is defined as

$$w_t^\alpha = \frac{(v_t^\alpha)^{-\lambda}}{\sum_{\alpha'=1}^{M}(v_t^{\alpha'})^{-\lambda}} \tag{4.5}$$

for a constant $\lambda \in \mathbb{N}^+$. The weights for the activity-specific models are proportional to the predictive variance $v_t^\alpha$ of each model (Equation 4.2). To ensure that the weights sum to one, and thus form a convex combination, each weight is divided by the sum of all weights. The weighting scheme assigns high importance to predictions from models that are certain about their estimate for the new pose. Lower weights are assigned to predictions from models showing a high predictive variance. The parameter $\lambda$ governs the smoothness of the weighting scheme. See Figure 4.3 b)-d) for an illustration of the weighting scheme.

An inherent advantage of this weighting approach is that transitions between different activities are smooth in the pose reconstruction. When the classification of subsequent frames changes from one activity to another, the influence of both corresponding GPR models is shifted gradually. Furthermore, even if the classification is incorrect, the prediction of the true model is not completely neglected and can positively influence the final pose estimation.

#### 4.3.2.2 Model Averaging using SVM Classification

As an alternative to combining the predictions of multiple GPR models by means of the predictive variance, we propose to use a multi-class SVM classifier based on the standard one-against-all approach [1]. The training data for the classifier is generated by concatenating the sensor measurement matrices $\mathbf{X}^\alpha$ over all activities in the training dataset into a matrix $\mathbf{X}$ and supplying the known class labels for each sensor measurement in a vector $\mathbf{C}$. After training the SVM, we estimate the current activity $\hat{\alpha}_t$ using the SVM classification for a given sensor measurement $\mathbf{x}_t$. The full-body pose prediction $\hat{\mathbf{y}}_t$ can be simply obtained from the corresponding GPR model $\mathcal{G}_{\hat{\alpha}_t}$. However, in order to increase the robustness with respect to scattered misclassifications, we add a sliding window over the sequence of incoming observations. All observations in a window of length $\omega$ prior to the current observation $\mathbf{x}_t$ are first classified using the SVM. The pose estimate $\hat{\mathbf{y}}_t$ is again a linear combination of the predictions from all models, as in Equation 4.4. However, the weight for an individual model $\alpha$,

$$w_t^\alpha = \frac{\#\alpha_\omega}{\omega} \tag{4.6}$$

now reflects the relative contribution of this model within the window. Here, $\#\alpha_\omega$ is the number of frames in the current window classified as activity $\alpha$. The more frames in a window are classified as a certain activity, the higher is the influence of the corresponding GPR model. Compared to the approach presented before that weights the contribution of all GPR models according to their predictive variance, this method selects a single GPR prediction most of the time. In the case of activity transitions, the contribution of two or more involved GPR models is shifted gradually. The window length parameter $\omega$ determines how quickly the influence of particular models shifts. See Figure 4.3 e)-f) for an illustration of this weighting scheme.

### 4.3.3 Evaluation

We performed several types of experiments in order to assess our method quantitatively and qualitatively. In the following sections, we first introduce the dataset we recorded in this

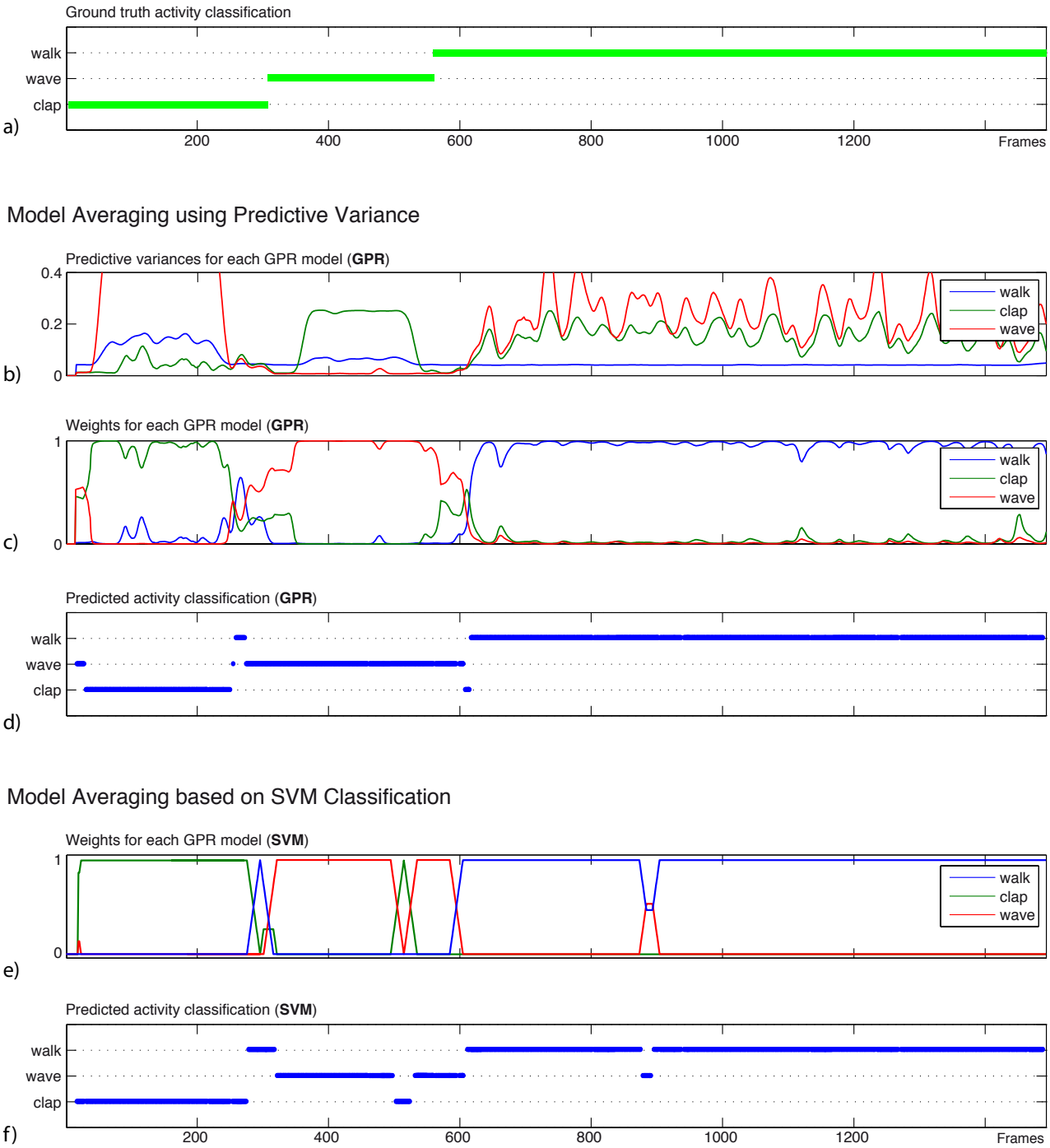



Figure 4.3: Illustration of the two proposed schemes for combining pose predictions of multiple activity-specific GPR models and for classifying activities. For clarity, only three activities are considered in this example (walk, wave, clap). a) True sequence of activities. b) Predictive variance for each of the three GPR models (Equation 4.2). c) Weights for the pose predictions of each GPR model (Equation 4.5). d) Predicted activity classification for the GPR approach (Equation 4.3). e) Weights for the pose predictions of each GPR model based on the SVM approach (Equation 4.6). f) Predicted activity classification for the SVM approach.

| | Activity | Description |
|---|---|---|
| 1. | Clap | Clapping both hands in front of the body. |
| 2. | Golf | A golf swing holding an imaginary golf bat in the hands. |
| 3. | Hurrah | Raising up both arms as in an expression of joy. |
| 4. | Jack | Jumping jack movement involving both arms and legs. |
| 5. | Knee | Repetitive knee bends with both arms extended forwards. |
| 6. | Laces | Kneeing down and binding shoelaces with both hands. |
| 7. | Pickup | Picking an object up from the floor with one hand. |
| 8. | Scratch | Scratching the head with one hand. |
| 9. | Walk | Walking in circles in a relaxed speed. |
| 10. | Wave | Waving with one hand above the head. |

Table 4.1: Activities included in the inertial sensor and motion capture dataset. Each activity starts and ends in an idle upright standing pose with both arms relaxed along the body.

work. This dataset is also used for the experiments in Chapter 5. Thereafter, we describe our experiments on pose estimation accuracy and precision of activity recognition.

#### 4.3.3.1 Inertial Sensor and Motion Capture Dataset

We recorded a dataset of corresponding full-body poses $\mathcal{Y}^\alpha = \{\mathbf{y}_1^\alpha, \ldots, \mathbf{y}_{N_\alpha}^\alpha\}$ and sensor measurements $\mathcal{X}^\alpha = \{\mathbf{x}_1^\alpha, \ldots, \mathbf{x}_{N_\alpha}^\alpha\}$ for each activity $\alpha \in \{1, \ldots, M\}$, using an optical marker-based motion capture system (Appendix B) that we synchronized with six wearable inertial orientation sensors [161]. Here, $N_\alpha$ is the number of frames we recorded for activity $\alpha$. In the dataset, a full-body pose is given by a vector $\mathbf{y}_i^\alpha$ of $d_y = 33$ dimensions representing the joint angles of our skeleton body model (Appendix A). An observation vector $\mathbf{x}_i^\alpha$ has $d_x = 12$ dimensions, representing pitch and roll for each of the sensors. We omit the yaw values that are measured by a compass, to be independent of a person's orientation with respect to magnetic north. Sensors were placed on the wrists, upper arms and shin-bones of each person. Figure B.1 b) shows an actor equipped with markers required by the infrared tracking system. The sensors were attached to the rigid frame of the motion capture targets. The dataset contains the $M = 10$ activities explained in Table 4.1. Each of the movements was recorded with 9 actors and 5 repetitions per actor. Every movement recording has a length of around 600 frames. For a systematic quantitative evaluation, we combined the movement recordings to sequences containing all 10 activities in a row. Figure 1.7 illustrates the acquisition process and shows the sensor data for one of the combined sequences. We made our dataset available for public download [133].

#### 4.3.3.2 Activity Classification

We evaluated the activity classification capabilities of our method, once for each of the two model averaging approaches described above. For brevity, we will refer to the approach combining multiple GPR model predictions and classifying activities using the predictive variance as *mGPR*. The other approach, using an SVM classifier for activity recognition, will be referred to as *mSVM*. Experiments were performed using the sequences containing all 10 activities that

we created from the movement recordings. As each activity was recorded separately per person and per repetition, ground truth activity labels were available. In a cross-validation manner, we used one of the available sequences for testing each time, while saving the remaning ones for training. Given that our dataset consists of 9 persons and 5 repetitions each, a total of 45 sequences were analyzed, each with approximately 2000 frames. As every movement recording begins and ends with an idle upright standing pose and this pose can be classified as any activity, we removed these idle standing frames from the testing data. To this end, we scanned the ground truth data for poses that are closer to a reference pose than a specified threshold. Computed over all sequences, this way 15% of all frames were removed.

Figure 4.4 shows the activity classification results for each of the two approaches in terms of confusion matrices. The values are averaged over all training sequences for all persons. Entry $(i, j)$ of a confusion matrix gives the relative number of frames belonging to activity $i$ that were classified as $j$. A perfect classification algorithm would generate a confusion matrix with ones on the diagonal and zeros everywhere else, as the diagonal entries represent the correct classification rates for each activity. For mGPR, we achieved an average correct classification rate of 70% and the confusion matrix has mostly marginal values off the diagonal. A very similar result can be found for the mSVM approach, while the average correct classification rate reaches a slightly higher 74%. Both, for mGPR and for mSVM, false positive classifications tend to concentrate on the walking activity. A possible explanation is that this activity involves movement of the whole body and consists of many distinct poses that have a similarity to poses of other activities. At the same time, both approaches achieve the highest correct classification rates for walking. Misclassifications also occur for both approaches between waving and scratching head. This behavior is inevitable, as both activities consist of raising the right hand up and differ only in a small portion of the movements. A significant difference between the two approaches can be seen in the correct classification rates for hurrah and jumping jack. These rather large movements are only correctly recognized in 50 to 60% of all frames for mGPR, while the mSVM approach achieves 82% in both cases.

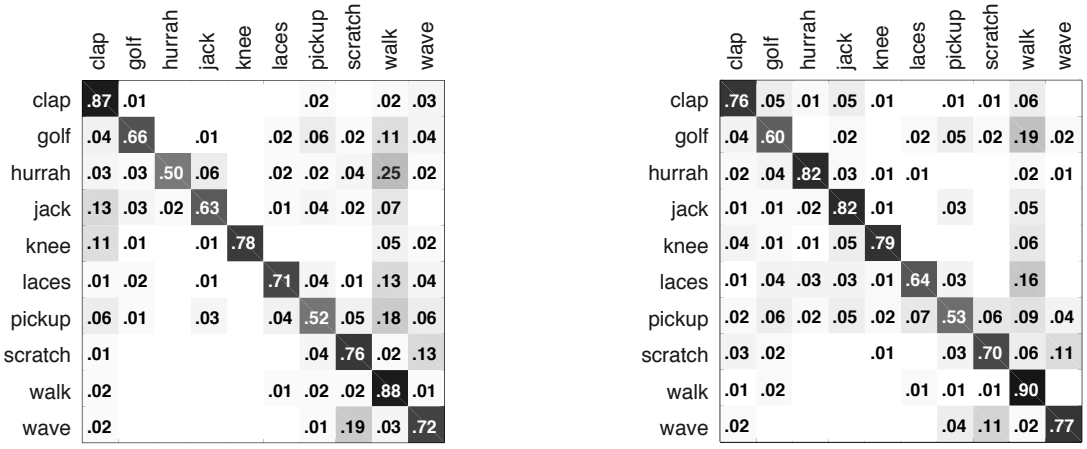
#### 4.3.3.3 Pose Estimation

In the cross-validation series of experiments described above, we also measured the precision of pose estimation, both for the mGPR and the mSVM approach. As a ground truth, we used the full-body pose motion capture data that is also available for the testing sequences. We devised two metrics to measure the deviation of the estimated pose $\hat{\mathbf{y}}_t$ from the ground truth pose $\mathbf{y}_t$ at each frame $t$. The angular error $e_{\text{ang}}(t)$ quantifies the average angular difference over all $d_y$ degrees of freedom of the skeleton model, corresponding to individual joints,

$$e_{\text{ang}}(t) = \frac{1}{d_y} \sum_{i=1}^{d_y} |\hat{\mathbf{y}}_t^{(i)} - \mathbf{y}_t^{(i)}|. \tag{4.7}$$

Here, the superscript notation in parentheses indicates individual vector entries. The distance error $e_{\text{dist}}(t)$ makes use of a forward kinematic function $f_{\text{kin}}^{(i)} : \mathbb{R}^{d_y} \to \mathbb{R}^3$ that computes the spatial location of the $i$-th joint, given a vector of joint angles. The metric quantifies the average Euclidean distance between estimated body joints and their ground truth locations,

$$e_{\text{dist}}(t) = \frac{1}{d_y} \sum_{i=1}^{d_y} \left\| f_{\text{kin}}^{(i)}(\hat{\mathbf{y}}_t) - f_{\text{kin}}^{(i)}(\mathbf{y}_t) \right\|, \tag{4.8}$$




(a) mGPR (b) mSVM

Figure 4.4: Activity classification results averaged over all experiments based on the GPR model averaging scheme (mGPR) and the SVM-based approach (mSVM). Entry $(i, j)$ of each of the confusion matrices corresponds to the relative number of frames belonging to activity $j$ that were classified as $i$. Matrix entries $< 0.01$ are hidden for clarity.

| | **mGPR** | | **mSVM** | |
|---|---|---|---|---|
| Activity | $e_{\text{ang}}$ | $e_{\text{dist}}$ | $e_{\text{ang}}$ | $e_{\text{dist}}$ |
| clap | 4.49 | 36.2 | 5.30 | 43.2 |
| golf | 7.34 | 64.4 | 8.33 | 69.9 |
| hurrah | 9.65 | 70.2 | 7.84 | 52.3 |
| jack | 7.48 | 56.4 | 7.90 | 51.1 |
| knee | 6.72 | 66.8 | 7.24 | 71.8 |
| laces | 8.03 | 79.1 | 9.22 | 90.3 |
| pickup | 6.88 | 62.2 | 7.54 | 67.7 |
| scratch | 4.28 | 30.3 | 5.34 | 37.4 |
| walk | 6.88 | 45.4 | 6.25 | 42.7 |
| wave | 4.55 | 26.8 | 4.82 | 30.1 |
| Average | **6.64** | **53.8** | **6.98** | **55.6** |

Table 4.2: Pose estimation accuracy for all considered activities, evaluated using the GPR approach (mGPR) and the SVM approach (mSVM) for averaging the predictions of all GPR models. Deviations from ground truth poses are provided as joint angles ($e_{\text{ang}}$ in degrees per joint) and as distances ($e_{\text{dist}}$ in millimeters per joint), averaged over all experiments.

Table 4.2 displays the results of our experiments in terms of both error metrics. The values are averaged over all frames of the testing sequences, including all persons and repetitions. Pose estimation precision is very similar for the mGPR and the mSVM approach with average angular errors of $\bar{e}_{\text{ang}} = 6.64°$ and $\bar{e}_{\text{ang}} = 6.98°$, respectively. The distance error is also in a similar range for both variants. The mSVM approach results in a slightly larger error $\bar{e}_{\text{dist}} = 55.6\text{mm}$, as opposed to $\bar{e}_{\text{dist}} = 53.8\text{mm}$ for the mGPR variant, but this effect is negligible. An explanation for this similar behavior is that both approaches predict poses using Gaussian process regression. While activity classification does influence how the predictions of individual activity-specific GPR models are combined, misclassifications mainly occur when poses are similar between different activities. In such cases, there is thus just a slight negative effect on the predicted poses. When inspecting pose estimation accuracy for individual activities, one notices that the lowest error is made for clapping, scratching head and waving. This is intuitive, as these movements only involve one or both arms, while the rest of the body remains still. On the other hand, higher errors can be found for hurrah, jumping jack or pickup. These movements are performed using the whole body and also exhibit larger dynamics, so that poses differ more between repetitions of the same activity.

Figure 4.5 visualizes the pose reconstruction result for four sample sequences, showing the predicted angles for the main joints, the corresponding ground truth and the angular error throughout the sequences. Overall, the pose estimation quality achieved in this work is comparable to the results of other state-of-the-art discriminative pose estimation methods, such as [4, 45, 148, 155, 179]. In fact, these methods use higher-dimensional input data, for instance human silhouettes captured in restrictive camera setups.

## 4.4 Discussion

In this chapter, we have presented a discriminative pose estimation method that allows reconstructing the full-body pose of a person, given only the limited observations of a few wearable inertial sensors attached to the person's limbs. Our method uses Gaussian process regression for learning the non-linear mapping between sensor observations and full joint angle configurations. We train our method on the movements of individual persons for each activity of interest. To cope with the ambiguous nature of the observation-to-pose mapping, a separate Gaussian process regression model is learned for each of the activities. We present two approaches for combining the pose predictions of each separate model into one final estimate. The first approach makes use of the predictive variance of each GPR model to assess which model is most certain about its prediction. The second approach relies on an additional SVM classifier that recognizes activities based on incoming sensor observations and gives the highest weight to the GPR model corresponding to the predicted activity.

Our experiments show that a few body-worn inertial sensors are sufficient for reconstructing full-body poses for movements the method has been trained on. The main limitation of the discriminative approach is that pose and activity predictions are based directly on the input sensor observations, without any underlying model of human motion. This leads to a high sensitivity to outlier measurements, which can be seen in the achieved classification and pose estimation results. In the next chapter, we will present a generative approach for human pose estimation and activity recognition to address this issue.

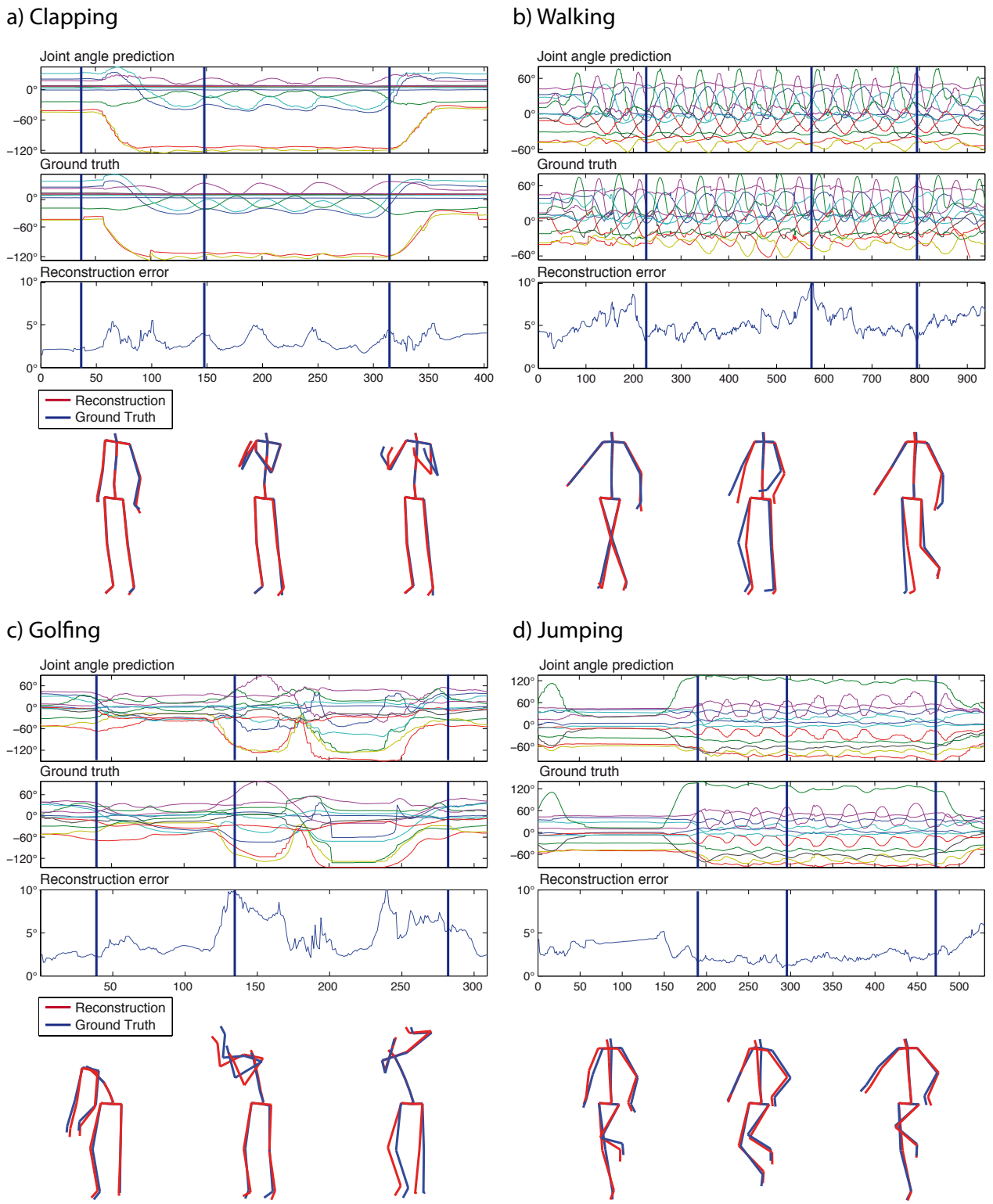



Figure 4.5: Illustration of pose estimation results for four sample sequences. For each sequence, the top graph shows the reconstructed angles for the main body joints over the length of the sequence. The second graph is the corresponding ground truth. The reconstruction error is plotted in the third graph. Skeletons are displayed for the frames indicated with black vertical bars (red: reconstructed pose, blue: ground truth pose).

Chapter 5

# Manifold Learning for Human Motion Analysis

## 5.1 Introduction

In the previous chapter, we investigated ways of predicting full-body poses $\mathbf{y} \in \mathbb{R}^{d_y}$ from limited observations $\mathbf{x} \in \mathbb{R}^{d_x}$ by modeling the observation-to-pose mapping $f(\mathbf{x}) = \mathbf{y}$. We proposed to use regression techniques to learn this discriminative mapping from training data. A disadvantage of such an approach is that we actually do not include a model of human motion in our predictions. In other words, we do not keep in mind which poses the human body can actually take and how feasible poses are related to each other. Instead, we solely focus on the appearance of body poses in an input modality, such as inertial sensor measurements. Problems arise particularly in the presence of outlier observations that the learned observation-to-pose mapping cannot handle correctly.

We now turn to another approach for human pose estimation that involves a generative model of human poses and movements. Here, we concentrate on learning feasible human poses from training data, allowing us to create a state space of potential body poses that we can then explore, given new input observations. In this state space, a vector $\mathbf{z} \in \mathbb{R}^{d_z}$ is a representation of a human pose that can *generate* both, its full pose vector $\mathbf{y}$ and the associated appearance vector $\mathbf{x}$. We use manifold learning to construct the state space in such a way that it allows reasoning meaningfully and efficiently about poses. For instance, the state space enables determining which poses are feasible and which poses most likely precede an activity switch. In fact, we will also model a state-to-observation mapping $F'(\mathbf{z}) = \mathbf{x}$ that is a functional, non-inverse mapping, as is the (unknown) natural pose-to-observation mapping $F(\mathbf{y}) = \mathbf{x}$ (see Section 4.1). Using the state-to-observation mapping, we can traverse the state space and look for poses that best match incoming observations.

The approach for pose estimation and activity recognition presented in this chapter is based on identifying the low-dimensional manifold of feasible human poses embedded within the high-dimensional space of pose parameters [42]. While the full pose space has as many dimensions as there are joints in the human body model, as little as two or three dimensions suffice to uniquely characterize points on the pose manifold [42, 72, 19, 98, 53]. As discussed in Section 3.2.2, manifold learning methods represent similar body poses with close-by low-dimensional manifold embedding points, leading to a compact parameterization of human poses. The non-linearity of human motion makes linear methods, such as PCA, hardly applicable [42].

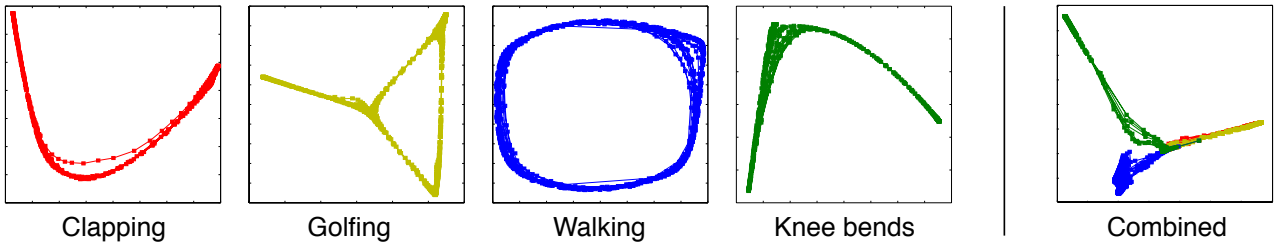



Figure 5.1: Manifold embeddings generated from human full-body pose data for different activities. Each point on the manifolds corresponds to a valid full-body pose. Clapping and knee bends: the movements have a linear, sequential pattern. Golfing: the movement consists of a linear part and a cyclic part. Walking: the movement has a cyclic, repetitive pattern. The embedding created from the pose data of all four activities is degenerated.

Human pose tracking based on manifold embeddings for single activities, e.g. walking, has been successfully demonstrated before [98, 42]. However, the generalization to include multiple activities is not straightforward [35]. In fact, a global embedding comprised of several activities will be dominated by inter-activity differences and characteristics of individual activities will not be represented sufficiently. This issue is illustrated in Figure 5.1. We therefore propose to combine multiple *activity-specific* manifold models into one compound motion model. This makes our method general, enabling us to track multiple activities, and at the same time specific, since specialized motion models for each activity are available.

The activity-specific motion models consist of manifold embeddings that capture the movement characteristics of each considered activity. The embeddings are generated from full-body pose training data that we acquire in a training phase using a motion capture system. In this chapter, we use as observations inertial sensor measurements and, in separate experiments, features extracted from depth images. We model the state-to-observation mapping $F'(\mathbf{z}) = \mathbf{x}$ using regression and analogously introduce a state-to-pose mapping $f'(\mathbf{z}) = \mathbf{y}$ that back-projects embedding points to their original full-body pose representation. As shown in Figure 5.2, these mappings allow predicting full-body poses and observation vectors from any given location in embedding space, resulting in a full generative human motion model.

We formulate the tracking problem within the Bayesian framework and employ a particle filter in manifold embedding space for state inference. Compared to optimization-based techniques, the particle filter increases robustness against local optima in state space by continuously tracking multiple pose hypotheses. We adapt the particle filter to the multiple-model setting by defining an efficient activity switching mechanism that governs the distribution of particles and their transition across different manifold embeddings. Our particle weighting scheme favors pose hypotheses that match the incoming observations (see Figure 5.2). Selecting the best-fitting pose hypothesis at each time allows us to simultaneously classify the performed activity and to estimate full-body poses. We also propose an effective measure of confidence for full-body pose predictions that enables us to detect movements that are not part of the training data. Recognizing such anomalies is crucial for an applicability of a training-based body tracking method in real-life scenarios.

As the method presented in this chapter is independent of the input modality that is used, we first describe the method as a general pose estimation and activity recognition framework (Section 5.3). Thereafter, we investigate its performance on input data coming from inertial

body-worn sensors (Section 5.4). Apart from an assessment of activity recognition rates and pose estimation accuracy, we also analyze the behavior of the proposed method for a large number of considered activities, the sensitivity to sensor placement on the body and the effect of stylistic movement variations. Finally, we similarly analyze our method for input observations coming from a depth camera (Section 5.5).

## 5.2 Related Work

In this section, we outline related work on generative human motion analysis methods. After highlighting various generative approaches proposed in the literature, we focus on applications of manifold learning for human motion analysis. The section concludes with an overview of methods using depth cameras for pose estimation and activity recognition.

**Generative Pose Estimation Methods.** Generative full-body pose estimation methods are based on the assumption that the mapping $F(\mathbf{y}) = \mathbf{x}$ from poses to observations is available. Human body tracking then requires searching for the most likely pose, given new observations. Methods using inverse kinematics for pose estimation follow this general approach [21, 159]. Here, a numerical optimization technique is used to find the full-body pose parameters such that a synthetic body model reaches a set of targets. These targets can be, for instance, locations of anatomical landmarks identified in images. The major difficulty in this case is the high dimensionality of full-body pose space that causes a high computational complexity for pose estimation. Several authors address this issue using sampling approaches, such as the particle filter [12, 128, 38]. While these methods are capable of reconstructing the body pose for arbitrary movements, the computational cost is still significant [26] and prevents applicability in real-time scenarios. Fortunately, complexity can further be reduced by restricting the search to learned low-dimensional subspaces of full-body pose space that capture characteristics of particular movement types [165, 98, 174]. Similar to Darby et al. [34] and Jaeggli et al. [72], we combine the advantages of sampling-based inference and dimensionality reduction by applying a particle filter in a low-dimensional motion model space.

**Manifold Learning for Motion Analysis.** Various methods have been employed for generating low-dimensional motion models from training data, e.g. PCA [128, 33] or Gaussian Process Latent Variable Models (GP-LVM) [165, 174]. We build upon manifold learning methods since they can capture non-linear characteristics of high-dimensional data by preserving local neighborhood relations. As has been shown in the literature [42], human motion data typically clusters in manifold structures, embedded in the high-dimensional full-body pose space. The particular manifold embedding technique to be used is not a crucial decision. For instance, Jaeggli et al. [72] employ Locally Linear Embedding (LLE) (Section 3.2.2.2). Blackburn and Ribeiro [19] rely on Isomap (Section 3.2.2.1) for generating manifold embeddings from human motion data. Lu and Sminchisescu [98] show promising results using Laplacian Eigenmaps (Section 3.2.2.3) for tracking human motion. Compared to GP-LVMs, manifold learning methods do not require expensive optimization in the training phase and ensure that similar high-dimensional points (i.e. full-body poses) are represented with close-by low-dimensional embedding points [92]. The latter property is valuable for low-dimensional pose tracking, since small, incremental movements in embedding space result in small pose changes.

The main challenge of methods based on learned, prior motion models is their limited ability to generalize from the training data. Lee and Elgammal [93] present an approach for relieving the dependence on visual observations that are very similar to those in the training data. To this end, the authors learn both, a *kinematic* manifold (representing full-body pose data) and a related *appearance* manifold (representing characteristics of visual observations). While this allows for a certain variation in the view-point of the camera, there is still the restriction to movement types that are part of the training data. Most techniques proposed in the literature therefore commonly focus on individual activities, such as walking [165, 98, 174, 81]. Attempts to track multiple activities have been made with a low-dimensional space that is shared by different activities [34] and with separate spaces for each considered activity [32, 72]. However, the computational complexity of these approaches grows significantly for more than two activities. Our method, integrating activity-specific manifold embeddings into a compound motion model, can cope with at least 18 activities (see Section 5.4).

When using a particle filter for state inference, a crucial algorithmic issue is how to distribute particles among multiple motion models and how to represent activity switching. Instead of applying separate particle filters on each activity-specific model, we follow Isard and Blake [69] and augment the classical particle filter with a discrete state variable, indicating a motion model for each particle. In the recent work of Darby et al. [34], Hidden Markov Models are learned for each activity in order to guide particle propagation along the learned low-dimensional embeddings. We achieve a similar goal by means of probabilistic, learned priors that, at the same time, allow us to control inter- and intra-activity particle distribution. Moreover, as opposed to our method, the authors of [34] extract observations from a minimum of two video camera views of a person.

**Inertial Sensors as Input Modality.** We refer the reader to Section 4.2 for an overview of related work on human pose estimation and activity recognition methods using inertial sensors as an input modality. In the literature, there is currently no work relying on inertial sensors in combination with a generative, learned human motion model, as proposed in this work.

**Depth Cameras as Input Modality.** Recently, several approaches for human activity recognition using ToF images have been proposed. For instance, Penne et al. use a ToF camera for hand gesture recognition with the aim of manipulating a medical 3D dataset [149, 89]. The method is based on a classifier trained to distinguish the surface appearance of a set of pre-defined hand poses. Holte et al. describe a technique for recognizing upper-body gestures, such as raising one or both arms, from ToF camera images [61]. Jensen et al. propose an approach for gait tracking using whole-body ToF images [74]. The method is based on localizing leg joint positions in range images of a person walking on a treadmill. Zhu et al. present a technique for upper-body tracking by fitting a body model to the ToF data, after identification of anatomical landmarks [184]. A more general, whole-body tracking approach is described by Plagemann et al. [117, 49]. Their method detects interest points in each ToF image and associates them with hands, head and feet of a body model. Shotton et al. [146] present a method for body joint detection in Microsoft Kinect depth images. Their method is based on a learning approach that classifies individual depth image pixels to one of multiple body parts, such as upper arm, elbow or thigh. As opposed to separately targeting activity recognition [149, 89, 61] and tracking [74, 184, 49, 146], we perform these tasks simultaneously.

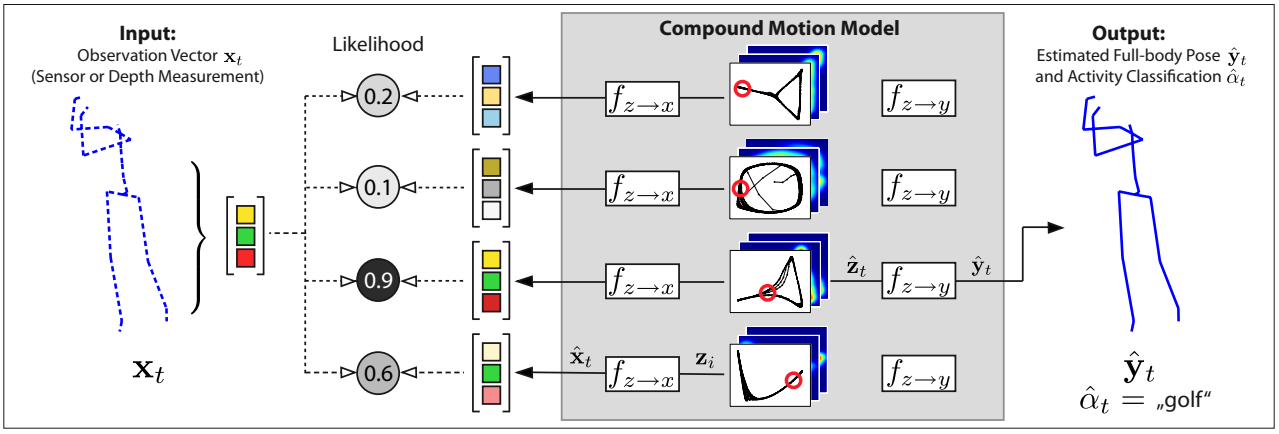



Figure 5.2: The compound motion model is comprised of several activity-specific models. Each of these consists of a manifold embedding of feasible poses, pose likelihood priors and learned mappings to observation space ($f_{z\to x}$) and full-body pose space ($f_{z\to y}$). We use a particle filter to track multiple pose hypotheses $\mathbf{z}_i$ for different activities (red circles). The likelihood of each pose hypothesis is evaluated by predicting an observation vector $\hat{\mathbf{x}}_i$ and comparing it against the incoming true observation $\mathbf{x}_t$. The highest-likelihood pose hypothesis is used for determining the activity $\hat{\alpha}_t$ and the estimated full-body pose $\hat{\mathbf{y}}_t$.

## 5.3 Generative Activity Recognition and Pose Estimation Framework

### 5.3.1 Method Overview

We address the problem of human full-body tracking and activity recognition from limited and noisy observations. In the following sections, we will use the term *observation* to refer to measurements from an arbitrary sensing modality, such as a depth camera or several inertial sensors. Given a set of $M$ activities of interest identified by an index $\alpha \in \{1, \ldots, M\}$, we start by building a compound motion model from training data containing both full-body poses $\mathbf{y} \in \mathbb{R}^{d_y}$ and observation vectors $\mathbf{x} \in \mathbb{R}^{d_x}$. Then, during testing, we wish to estimate the performed activity $\hat{\alpha}_t$ and the full-body pose $\hat{\mathbf{y}}_t$ at each time step $t$ only from novel incoming observations $\mathbf{x}_t$ (Figure 5.2). We formulate the tracking problem in a Bayesian framework using a generative model of feasible human poses.

For being able to cope with multiple activities, we define a *compound motion model* containing multiple activity-specific models. Each of these models is comprised of the following four components that are described in the upcoming sections:

1. **Manifold embedding**: A dimensionality-reduced representation of the full-body pose training data for activity $\alpha$. The manifold embedding efficiently parameterizes feasible human poses for this activity and provides a search space for pose tracking.

2. **Predictive mappings**: Regression mappings from the embedding space to full-body pose space and to observation space. The mappings allow predicting, from any point in the embedding space for activity $\alpha$, a (synthetic) measurement vector (for comparison to incoming observations) and a full-body pose (see Figure 5.4).

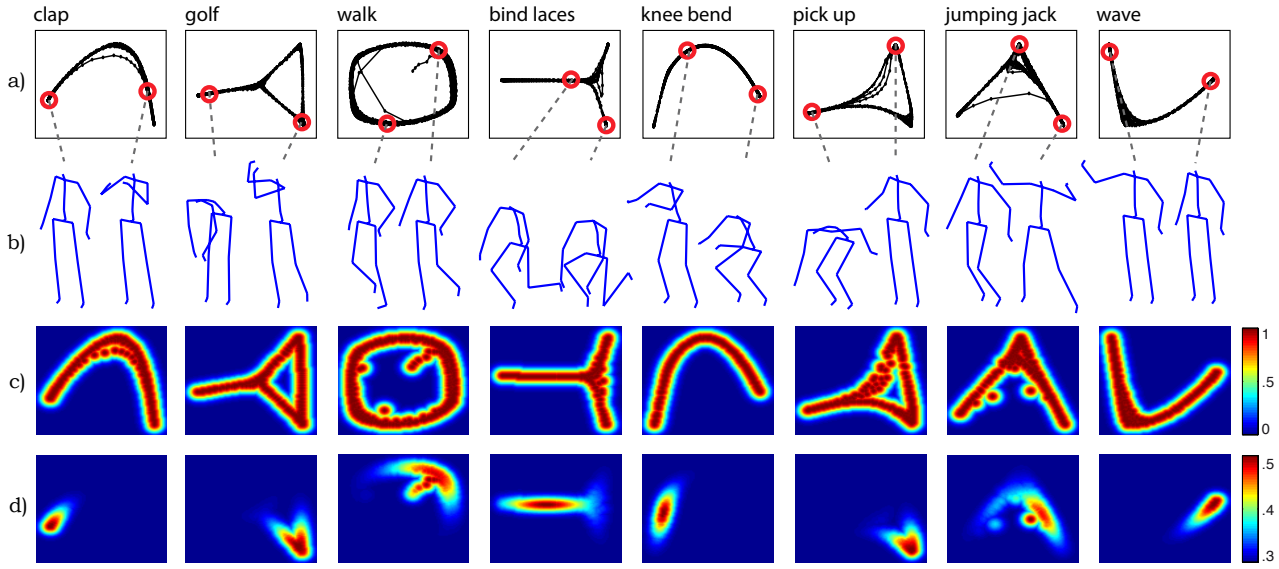



Figure 5.3: Components of learned motion models for 8 different activities. a) Two-dimensional manifold embeddings obtained using Laplacian Eigenmaps on full-body pose training data. b) Example full-body poses corresponding to embedding locations highlighted in red. c) Pose likelihood priors in embedding space. Blue colors represent low probabilities, red colors represent high probabilities. d) Activity switching priors in embedding space.

3. **Pose likelihood prior**: A probability distribution in manifold embedding space, giving the likelihood of any point to represent a feasible pose of activity $\alpha$. The likelihood prior allows further restricting search space during tracking.

4. **Activity switching prior**: Likelihood of a switch of activity from any pose in embedding space. The switching prior models the proposed activity switching mechanism and creates a link between the separate, activity-specific manifold embeddings.

We define the system state, to be estimated in every time step, as an activity index $\alpha \in \{1, \ldots, M\}$ and a pose $\mathbf{z} \in \mathbb{R}^{d_z}$ in low-dimensional embedding space ($d_z \ll d_y$). The state thus identifies an activity-specific motion model and a pose in its manifold embedding space. To find the state that best fits incoming observations in each time step, we employ a particle filter. As opposed to non-linear optimization methods that can get caught in local state space optima, the particle filter allows simultaneously tracking multiple pose hypotheses and selecting the most appropriate motion model.

### 5.3.2 Learning Low-dimensional Motion Models

In a training phase, each of the $M$ activity-specific motion models is learned from full-body pose data and corresponding observation vectors, i.e. inertial sensor measurements or depth camera features. The full-body data is acquired using a motion capture system (Appendix B) and has $d_y = 33$ dimensions, representing the joint angles of our simple skeleton model (Appendix A). In the following sections, we describe the components of each motion model. Figure 5.3 gives an illustration of the components for 8 different activities.

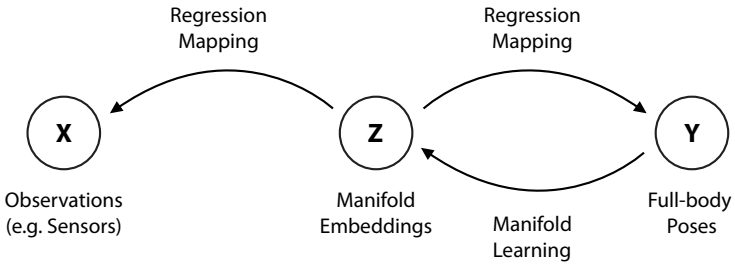



Figure 5.4: Diagram illustrating the relationship between observations ($\mathcal{X}$), full-body poses ($\mathcal{Y}$), manifold embeddings ($\mathcal{Z}$) and the regression mappings. Compare to Figures 4.2 and 7.4.

#### 5.3.2.1 Manifold Embedding

Let the set of full-body training poses for a single activity $\alpha$ be denoted as $\mathcal{Y}^\alpha = \{\mathbf{y}_1^\alpha, \ldots, \mathbf{y}_{N_\alpha}^\alpha\}$, where $N_\alpha$ is the number of poses for this activity. Learning the manifold underlying the full-body data gives us a low-dimensional representation $\mathbf{z}_i^\alpha$ for each pose $\mathbf{y}_i^\alpha$. We denote the set of all dimensionality-reduced points for activity $\alpha$ as $\mathcal{Z}^\alpha = \{\mathbf{z}_1^\alpha, \ldots, \mathbf{z}_{N_\alpha}^\alpha\}$. The low-dimensional embedding points $\mathcal{Z}^\alpha$ efficiently represent the *manifold of feasible poses* for activity $\alpha$ (Figure 5.3). Using this representation for tracking enables us to restrict the search space to likely poses, instead of exhaustively searching the high-dimensional full-body pose space.

While different manifold learning methods, such as Isomap or LLE, can be used in this context, we rely on Laplacian Eigenmaps [14] for the locality preservation property of this method. As discussed in Section 3.2.2, the objective of Laplacian Eigenmaps is to find low-dimensional point coordinates such that close points in the high-dimensional representation (i.e. human poses) are mapped to close-by low-dimensional points. This enables us to use the dimensionality-reduced embedding space for tracking subtle pose changes. Note that during tracking, poses are not required to coincide exactly with the known embedding points $\mathbf{z}_i^\alpha$, allowing us to track poses that deviate from the training data.

For the construction of the neighborhood graph in Laplacian Eigenmaps, we use the $k$-nearest neighbors approach and determine $k$ experimentally. As a similarity measure between any two poses $\mathbf{y}_i^\alpha$ and $\mathbf{y}_j^\alpha$, we define an extended Euclidean distance

$$d_{\text{joint}}(\mathbf{y}_i^\alpha, \mathbf{y}_j^\alpha) = \|\beta^\top(\mathbf{y}_i^\alpha - \mathbf{y}_j^\alpha)\|^2, \tag{5.1}$$

where $\beta \in \mathbb{R}^{d_y}$ is a constant weighting vector that gives a higher importance to joints that are close to the root of the skeleton between the hips [100]. This way, poses are considered close if the skeleton configuration in its general components is similar, while differences close to the end of each kinematic chain (e.g. at the hands) are more tolerable.

#### 5.3.2.2 Predictive Mappings

Intrinsically, manifold learning methods, and Laplacian Eigenmaps in particular, neither provide an out-of-sample mapping (to add new points to an existing embedding), nor a predictive mapping (for reconstructing high-dimensional representations for given embedding points) [81]. To relate poses $\mathbf{z}$ in low-dimensional embedding space to observations and to full-body poses, we therefore need to define mappings based on the training data. Similar to Kanaujia et

al. [81, 98], we use kernel regression according to the Nadaraya-Watson estimator (see Section 3.1.1). However, as an adaptation to our multiple-activity setting, we learn separate mappings for each of the manifold embeddings. The state-to-pose mapping $f^{\alpha}_{z\to y} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_y}$ allows us to predict a full-body pose $\hat{\mathbf{y}} = f^{\alpha}_{z\to y}(\mathbf{z})$ from any given location $\mathbf{z}$ in embedding space,

$$f^{\alpha}_{z\to y}(\mathbf{z}) = \sum_{i=1}^{N_\alpha} \frac{k_z(\mathbf{z}, \mathbf{z}^{\alpha}_i)}{\sum_{j=1}^{N_\alpha} k_z(\mathbf{z}, \mathbf{z}^{\alpha}_j)} \mathbf{y}^{\alpha}_i, \tag{5.2}$$

and the state-to-observation mapping $f^{\alpha}_{z\to x} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_x}$ similarly allows predicting an observation vector $\hat{\mathbf{x}} = f^{\alpha}_{z\to x}(\mathbf{z})$ from an embedding space location $\mathbf{z}$,

$$f^{\alpha}_{z\to x}(\mathbf{z}) = \sum_{i=1}^{N_\alpha} \frac{k_z(\mathbf{z}, \mathbf{z}^{\alpha}_i)}{\sum_{j=1}^{N_\alpha} k_z(\mathbf{z}, \mathbf{z}^{\alpha}_j)} \mathbf{x}^{\alpha}_i. \tag{5.3}$$

We use a Gaussian kernel function $k_z(\mathbf{z}_i, \mathbf{z}_j) \propto \exp(-\frac{1}{2}\|(\mathbf{z}_i - \mathbf{z}_j)/\sigma_z\|^2)$ with a width $\sigma_z$ determined from the standard deviation of the embedding points in the training data. Intuitively, the mappings compute a weighted average of the full-body poses $\mathbf{y}^{\alpha}_i$ and observation vectors $\mathbf{x}^{\alpha}_i$ in the training data, with weights proportional to the similarity of the embedding location $\mathbf{z}$ to each of the embedding points $\mathbf{z}^{\alpha}_i$.

In our experiments, we also use Relevance Vector Machine regression (RVM) for learning the predictive mappings [160, 158]. While in the case of kernel regression, the whole training dataset serves for a prediction from a new pose $\mathbf{z}$ in embedding space, RVM makes predictions only based on a small subset of the training data that is determined in a learning phase. Consequently, regression based on RVMs can be computationally more efficient. The performance of both methods in our full-body tracking method is evaluated in Section 5.4.

#### 5.3.2.3 Pose Likelihood Prior

Using the training data for each activity, we can derive the likelihood for arbitrary poses in low-dimensional embedding space. Intuitively, poses $\mathbf{z}$ that are close to the embedding points $\mathbf{z}^{\alpha}_i$ learned from training data should have the highest likelihood. A straightforward way of obtaining such a pose likelihood prior [98, 81] is to use a kernel density estimate (KDE),

$$p^{\alpha}_{\text{pose}}(\mathbf{z}) = \frac{1}{N_\alpha} \sum_{i=1}^{N_\alpha} k_z(\mathbf{z}, \mathbf{z}^{\alpha}_i), \tag{5.4}$$

where $k_z(\cdot,\cdot)$ is a Gaussian kernel as defined above. Using this definition, poses $\mathbf{z}$ are considered more likely if they are in a dense part of embedding space that contains *many* manifold embedding points. However, this implies an undesired bias towards poses belonging to slow movements. Such movements typically result in many very similar poses that are grouped together in embedding space. In order to remove this bias, we rewrite Equation 5.4 as

$$p^{\alpha}_{\text{pose}}(\mathbf{z}) = \max_{i\in\{1,\ldots,N_\alpha\}} k_z(\mathbf{z}, \mathbf{z}^{\alpha}_i). \tag{5.5}$$

This expression can be seen as a distance transform with a Gaussian distance function $k_z(\cdot,\cdot)$. The pose likelihood prior can be discretized and precomputed for computational efficiency.

#### 5.3.2.4 Activity Switching Prior

We also define a prior distribution $p^{\alpha}_{\text{switch}}(\mathbf{z})$ for every motion model that describes how likely a switch of activity is, given a pose $\mathbf{z}$ in embedding space. To ensure generality, we allow activity switching from any pose with a constant minimum probability $p_k$. However, we let the probability of switching increase for poses that typically occur between subsequent activities. In our experiments, the upright standing pose is used as an intermediate pose $\mathbf{y}_0$. Depending on the application scenario, other intermediate poses can also be defined. We denote the likelihood of switching activity from any embedding position $\mathbf{z}$ as

$$p^{\alpha}_{\text{switch}}(\mathbf{z}) = \min(1,\ k^{\alpha}_{y}(\mathbf{y}_0, f^{\alpha}_{z\to y}(\mathbf{z})) + p_k) \tag{5.6}$$

$$= \min(1,\ k^{\alpha}_{y}(\mathbf{y}_0, \hat{\mathbf{y}}) + p_k), \tag{5.7}$$

where $k^{\alpha}_{y}(\mathbf{y}_i, \mathbf{y}_j) \propto \exp(-\frac{1}{2}(\mathbf{y}_i - \mathbf{y}_j)^{\top}(\mathbf{\Sigma}^{\alpha}_{y})^{-1}(\mathbf{y}_i - \mathbf{y}_j))$ is a multivariate Gaussian kernel function with a diagonal covariance matrix $\mathbf{\Sigma}^{\alpha}_{y}$ derived from the training full-body poses $\mathbf{Y}^{\alpha}$. The vector $\hat{\mathbf{y}}$ is a predicted full-body pose. Equation 5.6 thus is proportional to the similarity of the full-body pose predicted from $\mathbf{z}$ to the intermediate pose $\mathbf{y}_0$.

### 5.3.3 Bayesian Tracking Using Multiple Motion Models

The testing phase of our method consists of tracking the pose in low-dimensional embedding space. For each time step $t$, we seek the most likely pose $\mathbf{z}_t$ in embedding space, given the observations $\mathbf{x}_t$. More formally, in a standard Bayesian tracking formulation with a Markovian assumption, one wishes to find the optimum of the posterior distribution

$$p(\mathbf{z}_t|\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_t)\ \ p(\mathbf{z}_t|\mathbf{x}_{t-1}), \tag{5.8}$$

comprised of the *observation model* $p(\mathbf{x}_t|\mathbf{z}_t)$, which represents the likelihood of an observation given a position in embedding space, and the *prior* $p(\mathbf{z}_t|\mathbf{x}_{t-1})$. The *prior* is modeled based on the posterior of the previous time step $p(\mathbf{z}_{t-1}|\mathbf{x}_{t-1})$ and an update given by a *dynamics model*:

$$p(\mathbf{z}_t|\mathbf{x}_{t-1}) = \int p(\mathbf{z}_t|\mathbf{z}_{t-1})\ \ p(\mathbf{z}_{t-1}|\mathbf{x}_{t-1})\ \ d\mathbf{z}_{t-1}. \tag{5.9}$$

The *dynamics model* $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ determines how pose estimates are updated from one time step to the next. Since we are using multiple motion models for tracking, we extend the tracking formulation with the discrete activity index $\alpha_t \in \{1, \ldots, M\}$. Equation 5.8 then becomes

$$p(\mathbf{z}_t, \alpha_t|\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_t, \alpha_t)\ \ p(\mathbf{z}_t, \alpha_t|\mathbf{x}_{t-1}). \tag{5.10}$$

Similar to [69], we also augment the dynamics model $p(\mathbf{z}_t|\mathbf{z}_{t-1})$ from Equation 5.9 with an activity index, yielding the factored dynamics model

$$p(\mathbf{z}_t, \alpha_t|\mathbf{z}_{t-1}, \alpha_{t-1}) = p(\mathbf{z}_t|\mathbf{z}_{t-1}, \alpha_t, \alpha_{t-1})\ \ p(\alpha_t|\mathbf{z}_{t-1}, \alpha_{t-1}). \tag{5.11}$$

The first term in Equation 5.11, which we refer to as *pose dynamics model*, governs the evolution of poses in their embedding space representation. The second term, named *activity transition model*, describes the activity switching process.

#### 5.3.3.1 Pose Dynamics Model

We define the new activity-aware pose dynamics model as follows:

$$p(\mathbf{z}_t|\mathbf{z}_{t-1}, \alpha_t, \alpha_{t-1}) = \begin{cases} p(\mathbf{z}_t|\mathbf{z}_{t-1}) & \text{if } \alpha_t = \alpha_{t-1}, \\ p^{\alpha_t}_{\text{pose}}(\mathbf{z}_t) & \text{else.} \end{cases} \tag{5.12}$$

When there is no switch of activity (i.e. $\alpha_t = \alpha_{t-1}$), dynamics are governed by a random walk in embedding space. This is modeled as a normal distribution centered at the previous pose in embedding space, $p(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{z}_t; \mathbf{z}_{t-1}, \mathbf{\Sigma}^{\alpha_t}_z)$. Here, $\mathbf{\Sigma}^{\alpha_t}_z$ is the diagonal covariance matrix of the low-dimensional representation $\mathcal{Z}^{\alpha_t}$ of the training poses $\mathcal{Y}^{\alpha_t}$. In the case of an activity switch (i.e. $\alpha_t \neq \alpha_{t-1}$), the dynamics model follows the pose likelihood prior $p^{\alpha_t}_{\text{pose}}(\mathbf{z})$ of activity $\alpha_t$ (see Section 5.3.2.3). In other words, the most likely poses after switching to a new activity $\alpha_t$ are those learned from the training data of this activity.

#### 5.3.3.2 Activity Transition Model

In our model, all sequences of consecutive activities are considered equally likely, such that $p(\alpha_t = j|\mathbf{z}_{t-1}, \alpha_{t-1} = i)$ is equal for all activity indices $j \neq i$. The probability of switching from a given activity $\alpha_{t-1}$ to any other activity then only depends on the previous pose $\mathbf{z}_{t-1}$ in embedding space. Thus, we state our activity transition model using the activity switching prior defined in Section 5.3.2.4 as

$$p(\alpha_t|\mathbf{z}_{t-1}, \alpha_{t-1}) = \begin{cases} 1 & \text{if } \alpha_t = \alpha_{t-1}, \\ p^{\alpha_{t-1}}_{\text{switch}}(\mathbf{z}_{t-1}) & \text{else.} \end{cases} \tag{5.13}$$

Our approach can be easily extended to model a Markov process on the level of activities, using a set of transition probabilities between all considered activities instead of Equation 5.13 [69]. In this work, however, we do not focus on particular sequences of activities and thus model switching to all activities as equally likely.

#### 5.3.3.3 Observation Model

Our observation model $p(\mathbf{x}_t|\mathbf{z}_t, \alpha_t)$ represents the likelihood of an observation $\mathbf{x}_t$, given a point $\mathbf{z}_t$ in the embedding space of activity $\alpha_t$. We define it as a product of three terms – a prediction term, a pose smoothness term and a pose likelihood prior:

$$\begin{aligned} p(\mathbf{x}_t|\mathbf{z}_t, \alpha_t) &= k^{\alpha_t}_x(\mathbf{x}_t, f^{\alpha_t}_{z\to x}(\mathbf{z}_t)) \cdot k^{\alpha_t}_y(\hat{\mathbf{y}}_{t-1}, f^{\alpha_t}_{z\to y}(\mathbf{z}_t)) \cdot p^{\alpha_t}_{\text{pose}}(\mathbf{z}_t) && (5.14) \\ &= k^{\alpha_t}_x(\mathbf{x}_t, \hat{\mathbf{x}}) \cdot k^{\alpha_t}_y(\hat{\mathbf{y}}_{t-1}, \hat{\mathbf{y}}) \cdot p^{\alpha_t}_{\text{pose}}(\mathbf{z}_t). && (5.15) \end{aligned}$$

The prediction term $k^{\alpha_t}_x(\mathbf{x}_t, f^{\alpha_t}_{z\to x}(\mathbf{z}_t))$ uses the learned mapping $f^{\alpha}_{z\to x}(\mathbf{z})$ to predict observations from a pose $\mathbf{z}_t$. The likelihood of $\mathbf{z}_t$ based on this term is maximal if the prediction perfectly matches the true observation $\mathbf{x}_t$. In order to reduce the influence of outlier observations, the smoothness term $k^{\alpha_t}_y(\hat{\mathbf{y}}_{t-1}, f^{\alpha_t}_{z\to y}(\mathbf{z}_t))$ penalizes embedding locations if their predicted full-body pose differs strongly form the previous pose $\hat{\mathbf{y}}_{t-1}$. The pose likelihood prior, defined in Section 5.3.2.3, encourages poses that are close to the training data. The multivariate Gaussian kernel functions $k^{\alpha}_x(\cdot,\cdot)$ and $k^{\alpha}_y(\cdot,\cdot)$ have diagonal covariance matrices $\mathbf{\Sigma}^{\alpha}_x$ and $\mathbf{\Sigma}^{\alpha}_y$ determined from the training observations $\mathcal{X}^{\alpha}$ and full-body poses $\mathcal{Y}^{\alpha}$ of activity $\alpha$.

### 5.3.4 State Inference Using a Particle Filter

Our goal is to estimate the most likely activity $\hat{\alpha}_t$ and pose $\hat{\mathbf{z}}_t$ in the low-dimensional representation after making observations $\mathbf{x}_t$ at each time $t$. We employ a particle filter [68, 69] to sample the posterior density in Equation 5.10. The particle filter algorithm allows estimating Bayesian models where an exact inference is not feasible and where the assumptions of the Kalman filter (linearity, Gaussian distribution) are not met. A potentially large number of particles is distributed and propagated through state space to approximate the posterior density of the Bayesian model. The computational complexity of the particle filter grows with the dimensionality of the state space. In the high-dimensional space of full-body pose parameters, particle filtering can be prohibitive [12]. However, it is computationally efficient in our setting, as we employ the particle filter in the low-dimensional manifold embedding space.

Each individual particle can be seen as a state hypothesis. As the system state in our context is given by an activity index $\alpha$ and a location $\mathbf{z}$ in the corresponding manifold embedding space, we model each of $n$ particles as a pair $\mathbf{p}_t^i = (\alpha_t^i, \mathbf{z}_t^i)$, $i \in \{1, \ldots, n\}$. Initially, all particles are distributed across the activity-specific manifold embeddings of our compound motion model. The initial particle locations are randomly chosen by sampling the likelihood prior distribution $p_{\text{pose}}^{\alpha}(\mathbf{z})$ for each activity. Every particle has an associated weight $w_t^i$ that quantifies its relative importance or, more intuitively, the suitability of the state hypothesis represented by the particle. The initial weights are selected uniformly.

We follow the sequential importance resampling (SIR) variant of the particle filter that consists of the following steps, performed at each time step $t$ (see also Figure 5.5):

1. **Resample**: Given the particle locations and weights from the previous time step, $\{\mathbf{p}_{t-1}^i, w_{t-1}^i\}$, we obtain the new set of locations $\{\tilde{\mathbf{p}}_{t-1}^i\}$ by selecting $n$ new particles from the previous locations with a probability that is proportional to the given weights.

2. **Predict**: A prediction $\mathbf{p}_t^i$ is computed for each particle by applying the dynamics model, i.e. by sampling from $p(\mathbf{z}_t, \alpha_t | \tilde{\mathbf{z}}_{t-1}^i, \tilde{\alpha}_{t-1}^i)$. This implies switching the $i$-th particle to a randomly chosen other activity with a probability of $p_{\text{switch}}^{\alpha^*}(\tilde{\mathbf{z}}_{t-1}^i)$, where $\alpha^* := \tilde{\alpha}_{t-1}^i$. For particles that do not switch activity, the prediction step is a random walk.

3. **Evaluate**: Each particle $\mathbf{p}_t^i$ at its new location is weighted by evaluating the observation model and storing the associated weight as $w_t^i = p(\mathbf{x}_t | \mathbf{z}_t^i, \alpha_t^i)$. Finally, we normalize the weights such that $\sum_n w_t^i = 1$, as the particle weights represent a probability distribution.

The distribution of particles across the activity-specific manifold embeddings at each time $t$ gives an insight on which activity is most likely being performed. We only consider the particles with a weight higher than a threshold and estimate the activity $\hat{\alpha}_t$ as the most frequent activity among these particles. Let $W$ denote a set of indices corresponding to the highest-weight particles with the estimated activity index $\hat{\alpha}_t$. We let these particles contribute to the pose estimate $\hat{\mathbf{z}}_t$ in low-dimensional space:

$$\hat{\mathbf{z}}_t = \sum_{i \in W} \frac{w_t^i}{\sum_{j \in W} w_t^j} \cdot \mathbf{z}_t^i. \tag{5.16}$$

Here, the pose hypotheses $\mathbf{z}_t^i$ of all particles with indices in $W$ are averaged using the respective particle weights. In order to obtain a convex combination, the weights are normalized to sum

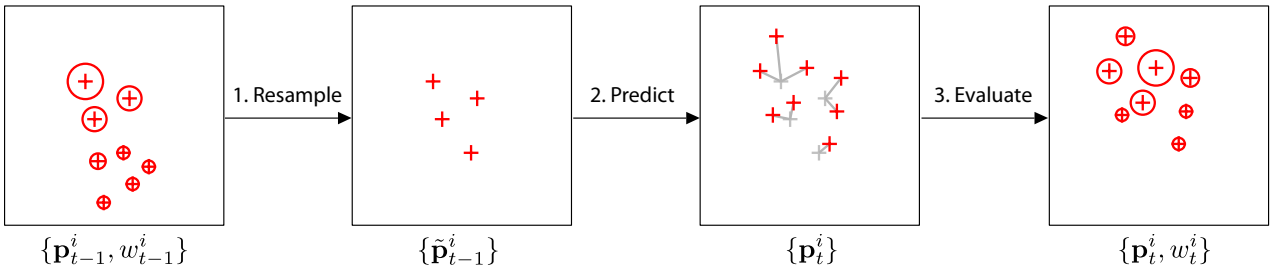



Figure 5.5: Illustration of the particle filter algorithm for $n = 8$ particles. Particles are indicated as crosses, the diameter of the circles represents their weights. The particles from time step $t-1$ are first resampled according to the weights. Note that some particle can be sampled multiple times or not at all. The resampled particles are predicted, usually by means of a random walk. In the evaluation step, the new weights for time step $t$ are computed.

to unity. Note that the index set $W$ only contains particles of one particular activity. Given $\hat{\mathbf{z}}_t$, we predict the full-body pose $\hat{\mathbf{y}}_t$ using the state-to-pose mapping (Equation 5.2) as

$$\hat{\mathbf{y}}_t = f_{z\to y}^{\hat{\alpha}_t}(\hat{\mathbf{z}}_t). \tag{5.17}$$

Figure 5.6 illustrates the tracking algorithm and, in particular, how particles sample the activity-specific manifold embeddings. In the exemplary sequence, the person switches from waving to golfing. The compound motion model has been trained on ten activities (see Section 5.4). The illustration shows four of the ten manifold embeddings for several frames of the sequence. Initially, the person is waving and most particles are concentrated around a pose on the *waving* embedding. A small number of particles also samples all other manifolds for the case that a sudden change of activity occurs. As the person leans forward for golfing, particles quickly start accumulating on the *golfing* manifold, since the predicted observations of these particles increasingly match the real measurements. Subsequent resampling steps cause the majority of particles to follow. The proposed compound motion model is thus capable of quickly adapting to activity changes. A video illustration is available online [133].

### 5.3.5 Anomaly Detection

The compound motion model allows us to derive a measure of confidence for full-body pose predictions. This measure can be used to detect anomalies in the movements of a testing person, i.e. movements that are not part of the training dataset. In such a situation, the activity classification and full-body pose predictions given by the algorithm are most likely inaccurate and can be disregarded. We define the predictive confidence $c_t \in \mathbb{R}$ at each time step $t$ as

$$c_t = \log(k_x^{\hat{\alpha}_t}(\mathbf{x}_t, f_{z\to x}^{\hat{\alpha}_t}(\hat{\mathbf{z}}_t))) = \log(k_x^{\hat{\alpha}_t}(\mathbf{x}_t, \hat{\mathbf{x}})), \tag{5.18}$$

where $k_x^\alpha(\cdot,\cdot)$ is a multivariate Gaussian kernel function with a covariance matrix $\mathbf{\Sigma}_x^\alpha$ derived from the observations $\mathcal{X}^\alpha$ in the training data. The confidence is maximal when the observation vector $\hat{\mathbf{x}}$, predicted from the current pose $\hat{\mathbf{z}}_t$ using the state-to-observation mapping $f_{z\to x}^{\hat{\alpha}_t}$, approaches the true observed measurement $\mathbf{x}_t$. We declare movements anomalous if $c_t$ drops below a threshold for a given number of subsequent frames.

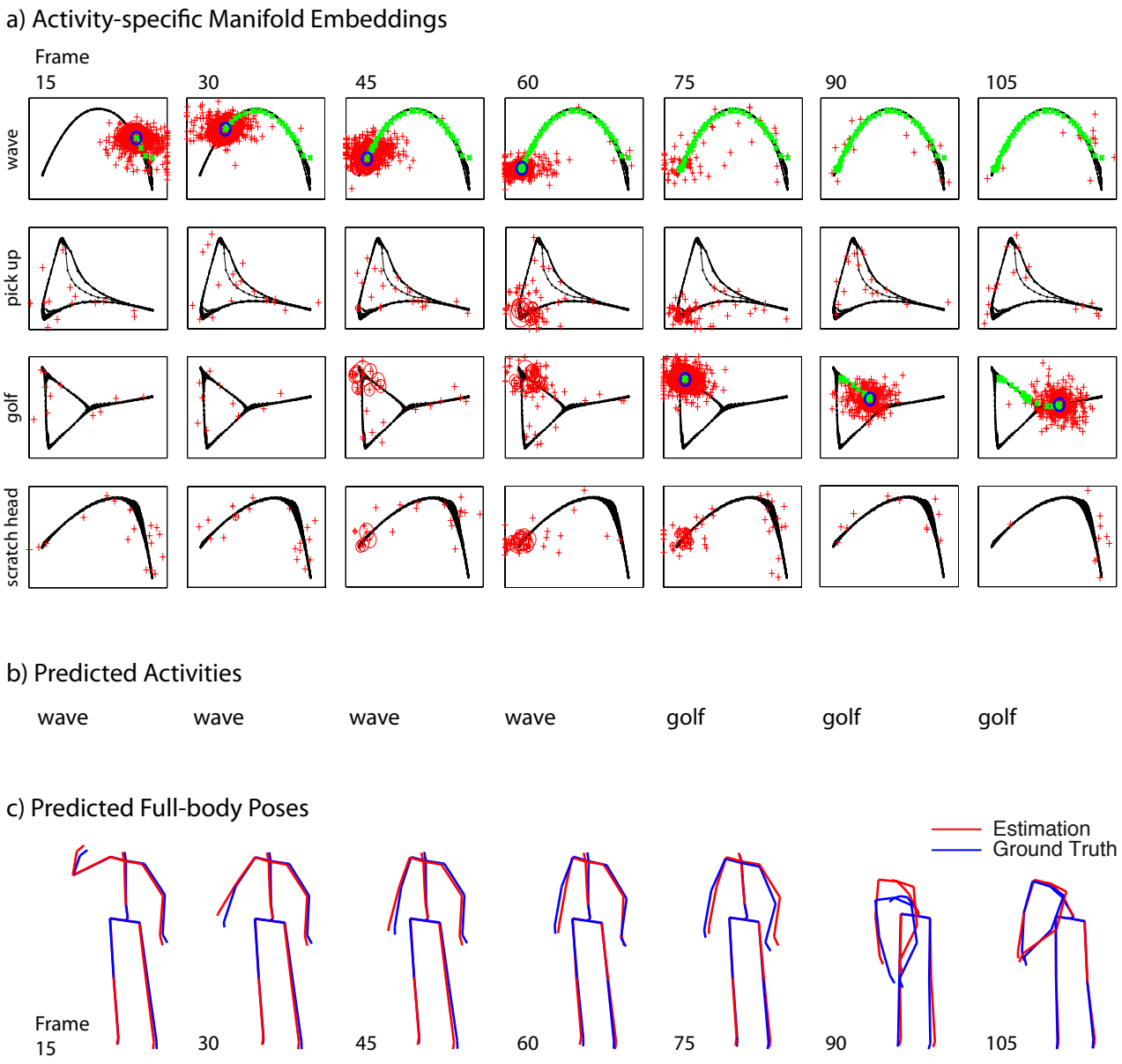



Figure 5.6: The particle filter-based activity switching mechanism on a sample sequence changing from waving to golfing. a) Four of the ten activity-specific manifold embeddings (*wave*, *pick up*, *golf* and *scratch head*) are displayed for several frames. Particles are shown as red crosses. The particle representing the state estimate $\hat{\mathbf{z}}_t$ in each of the frames is circled in dark. Green crosses indicate the state trace of previous frames. b) Predicted activities $\hat{\alpha}_t$, as determined from the particles with highest weight. c) The corresponding predicted (red) and ground-truth body poses (blue).

## 5.4 Using Inertial Sensors as Input Modality

Approaches for tracking the three-dimensional pose of a person's body have mostly been studied in the field of computer vision, where observations are typically image features, such as human silhouettes [165, 42, 12, 4]. Vision-based methods depend on illumination, view-point and line-of-sight between the tracked person and one or more cameras. Such constraints are not practicable when specific activities need to be studied over extended periods of time. Exemplary applications include industrial ergonomic studies or motion analysis for medical diagnosis of motion disorders, e.g. Multiple Sclerosis [116]. These scenarios require the recovery of full-body motion for a set of activities of interest, while the subjects move freely. In the following sections, we describe our experiments on recognizing activities and tracking full-body poses using the proposed framework and inertial sensors as an input modality. In Section 5.5, we will show our experiments for input observations coming from a depth camera.

### 5.4.1 Evaluation

After assessing the performance of activity classification and full-body pose estimation, we evaluate the anomaly detection approach. Experiments using a varying number of particles are described thereafter. We investigate the scalability of our method with respect to a large number of considered activities, the sensitivity to sensor placement on the body and the sensitivity to stylistic variation of movements.

#### 5.4.1.1 Dataset and Training Setup

We used the dataset that was also employed in the evaluation of our discriminative sensor-based pose estimation method, see Chapter 4. An illustration of the sensor data is shown in Figure 1.7. For training, we used corresponding sequences of sensor and motion capture data to learn our compound motion model. From the full-body pose data, we learned manifold embeddings $\mathcal{Z}^\alpha$ of $d_z = 2$ dimensions for each of the $M = 10$ activities in the dataset. During testing, the motion capture data served as ground truth and only sensor data was used as an input. The testing data consists of 5 sequences per actor containing all activities performed in a row. While testing on any of these sequences, the remaining 4 sequences per actor were used for training (cross-validation scheme). We used two regression methods for comparison. The mappings from manifold embedding space to observations ($f^\alpha_{z\to x}$) and to full-body pose space ($f^\alpha_{z\to y}$) were defined using kernel regression (KR) and Relevance Vector Machine regression (RVM). Unless denoted otherwise, results were obtained using $n = 400$ particles.

#### 5.4.1.2 Activity Classification

As outlined in section 5.3.4, we classify the activity performed in each frame based on the activity indices of the particles with the highest weight. Once this activity is estimated, the activity-specific motion model used for predicting the full-body pose is determined. Wrong activity classification can therefore adversely affect tracking results.

Figure 5.7(a) shows classification results for one of the testing sequences. As can be seen in the top graph, misclassifications are scarce and mainly occur at the beginning and end of activities, where the person is standing idle. These frames can be classified as any activity,

since the training data for every activity contains frames of idle standing. The bottom graph of Figure 5.7(a) reveals the number of particles per activity-specific manifold embedding. For clarity, we only show four of ten manifold embeddings that were learned. The distribution of particles across the embeddings is an indicator of how well a particular motion model fits the sensor data. We can see that a small number of particles continuously samples all of the manifold embeddings. High concentrations of particles on a particular manifold embedding occur almost exactly when the corresponding activity is performed.

Figure 5.8 gives the classification results for all activities over all testing sequences, evaluated using KR and RVM for the predictive mappings. On average, we achieved a correct classification rate (i.e. a true positive rate) of 79% for all non-idle frames (RVM) and 89% (KR), respectively. As in our evaluation of the discriminative sensor-based pose estimation method in Section 4.3.3, the quantitative results do not contain the classifications for the idle-standing frames. Although the RVM approach is less precise, both confusion matrices are mostly diagonal. Significant confusion only occurs between the activities *waving* and *scratching head*. In fact, both consist of raising the right arm close to the head. Misclassification in this case therefore does not necessarily affect the precision of full-body pose estimation.

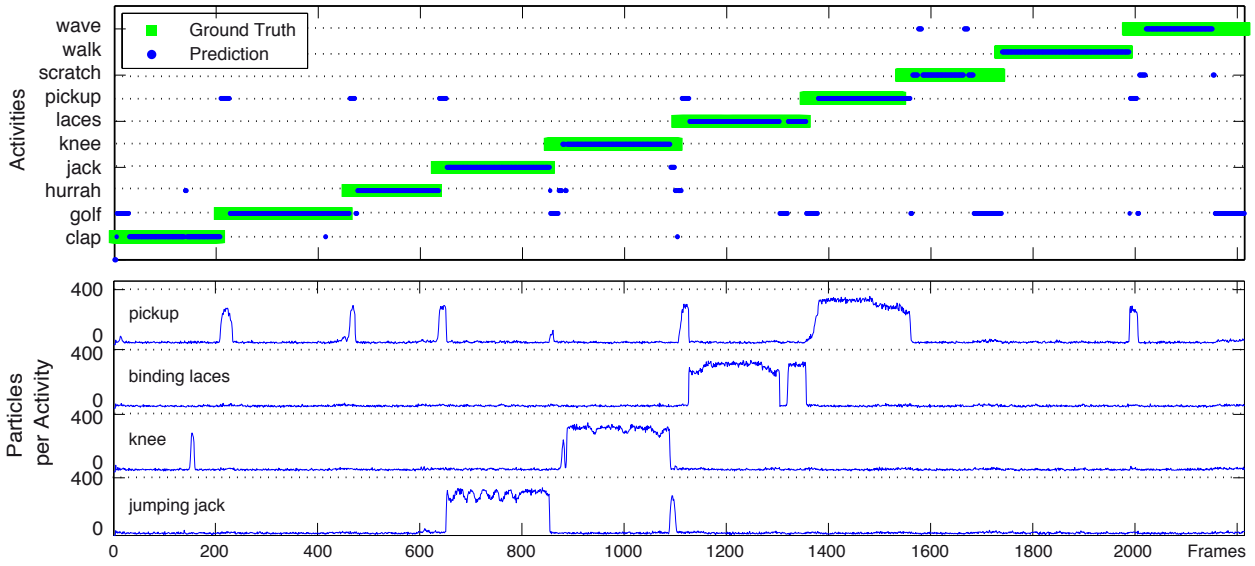
#### 5.4.1.3 Pose Estimation

We measured how precisely the poses estimated by our method match the ground truth using the two metrics presented in Chapter 4. The angular error $e_{\text{ang}}$ (Equation 4.7) gives the deviation from the ground truth in terms of joint angles. The distance error $e_{\text{dist}}$ (Equation 4.8) is the difference in 3D space between predicted joint locations and the ground truth. Using the RVM approach, we achieved $\bar{e}_{\text{ang}} = 7.54°$ per joint and $\bar{e}_{\text{dist}} = 55.3\text{mm}$, averaged over all frames of the testing sequences. The kernel regression approach allowed for a higher pose estimation accuracy with $\bar{e}_{\text{ang}} = 6.23°$ per joint and $\bar{e}_{\text{dist}} = 45.2\text{mm}$. As shown in Table 5.3, the deviation from the ground truth only increases for fast movements with a large variability, such as *jumping jack* or *walking*. Figure 5.9 provides a qualitative view of the achieved results. Our approach compares favorably to state-of-the-art pose estimation methods working on visual image data, e.g. [4, 167, 155], where similar average angular and distance errors are reported as achieved with our method from inertial sensor data.

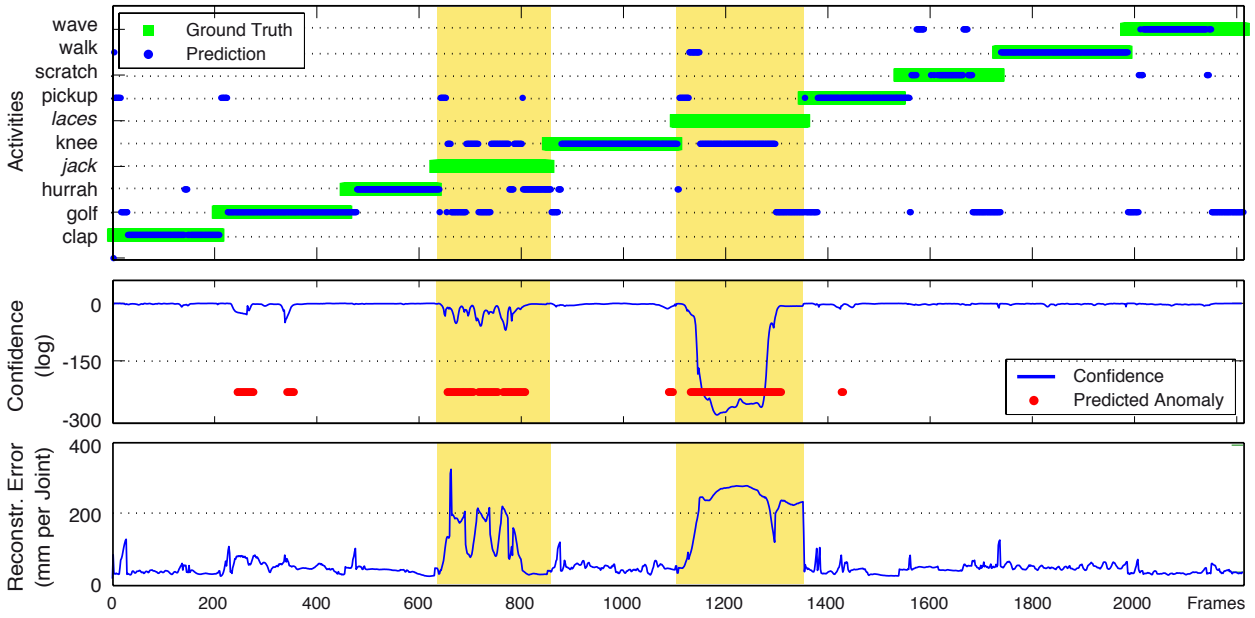
#### 5.4.1.4 Anomaly Detection

In order to evaluate the capability of our method to detect anomalous movements (belonging to activities that are not part of the training data), we performed a series of experiments on the testing sequence shown in Figure 5.7(a), using the kernel regression method. We repeatedly ran our algorithm on the sequence, each time leaving out one of the activities from the training data, thus simulating situations where different unknown activities are performed. Results are reported in Table 5.2 in terms of true positive ($r_{\text{TP}}$) and false positive rates ($r_{\text{FP}}$). In an ideal case, all frames corresponding to the anomalous activities should be recognized as such, resulting in $r_{\text{TP}} = 1$. Conversely, $r_{\text{FP}} = 0$ would indicate that no frames with known activities are falsely labelled as anomalies.

Depending on which activity was left out from the training data set, our approach achieves up to $r_{\text{TP}} = 0.97$. On average, $r_{\text{FP}} = 0.04$. Note that removing particular activities from the training data results in very low true positive rates, such as $r_{\text{TP}} = 0.13$ for *waving* and $r_{\text{TP}} = 0$

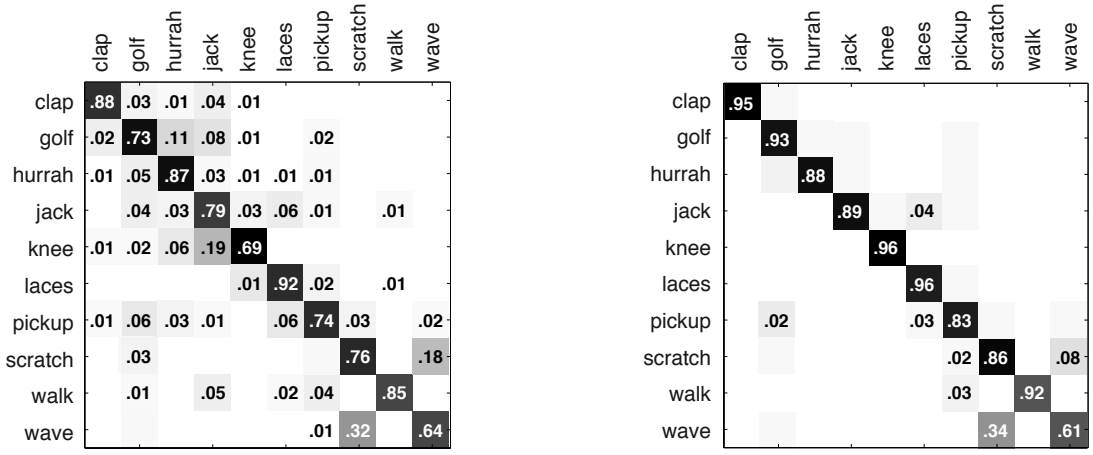



(a) Activity classification with all activity-specific motion models present.




(b) Activity classification and anomaly detection with missing motion models for *laces* and *jumping jack*.

Figure 5.7: Activity classification results for a testing sequence with 10 activities. a) *Top:* Ground truth classification and predicted activities for each frame of the sequence. *Bottom:* Number of particles per frame sampling four of the activity manifolds (400 particles in total). b) *Top:* Ground truth classification and predicted activities with missing learned models for two activities. *Middle:* Confidence $c_t$ (Equation 5.18) plotted over the length of the sequence. The periods recognized as belonging to anomalous activities are indicated in red. *Bottom:* Full-body pose reconstruction error $e_{\text{dist}}$ (Section 5.4.1.3) plotted over the length of the sequence.




(a) RVM (b) KR

Figure 5.8: Activity classification results averaged over all experiments based on kernel regression (KR) and Relevance Vector Machine regression (RVM). Entry $(i, j)$ of each of the confusion matrices corresponds to the relative number of frames belonging to activity $j$ that were classified as $i$. Matrix entries $< 0.01$ are hidden for clarity.

| | **RVM** | | **KR** | |
|---|---|---|---|---|
| Activity | $e_{\text{ang}}$ | $e_{\text{dist}}$ | $e_{\text{ang}}$ | $e_{\text{dist}}$ |
| clap | 5.62 | 40.5 | 4.95 | 37.8 |
| golf | 8.17 | 71.8 | 6.10 | 51.2 |
| hurrah | 7.53 | 44.0 | 6.79 | 40.2 |
| jack | 9.68 | 63.9 | 8.80 | 58.1 |
| knee | 10.69 | 88.0 | 4.87 | 45.6 |
| laces | 7.47 | 73.2 | 5.90 | 60.5 |
| pickup | 6.67 | 59.8 | 5.90 | 51.4 |
| scratch | 4.41 | 28.5 | 4.43 | 27.7 |
| walk | 10.16 | 54.7 | 9.65 | 50.7 |
| wave | 5.01 | 28.8 | 4.93 | 28.5 |
| Average | **7.54** | **55.3** | **6.23** | **45.2** |

Table 5.1: Pose estimation accuracy for all considered activities, evaluated using kernel regression (KR) and Relevance Vector Machine regression (RVM) for the predictive mappings $f_{z\to y}$ and $f_{z\to x}$. Deviations from ground truth poses are provided as joint angles ($e_{\text{ang}}$ in degrees per joint) and as distances ($e_{\text{dist}}$ in millimeters per joint), averaged over all experiments.

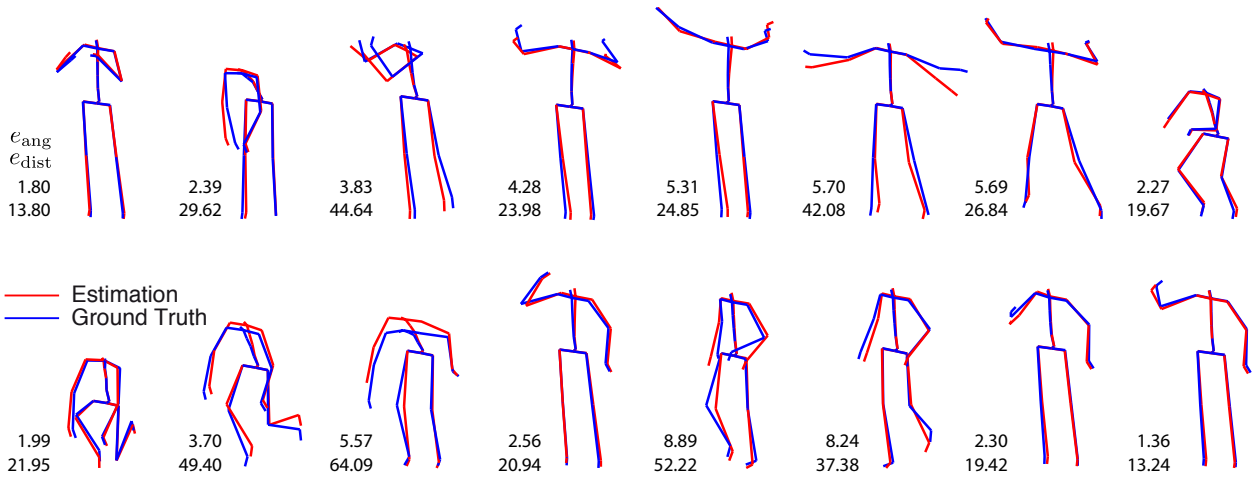



Figure 5.9: Illustration of full-body pose estimation results for selected frames of a testing sequence. The predicted poses (red) are overlaid ontop of the ground truth poses (blue) obtained using a motion capture system. The indicated numbers for each of the poses are the instantaneous angular error $e_{\text{ang}}$ in degrees per joint (top) and the distance error $e_{\text{dist}}$ in millimeters per joint (bottom).

| | **KR** | |
|---|---|---|
| Activity | $r_{\text{TP}}$ | $r_{\text{FP}}$ |
| clap | 0.24 | 0.04 |
| golf | 0.86 | 0.01 |
| hurrah | 0.88 | 0.04 |
| jack | 0.86 | 0.04 |
| knee | 0.83 | 0.05 |
| laces | 0.97 | 0.05 |
| pickup | 0.92 | 0.04 |
| scratch | 0.00 | 0.04 |
| walk | 0.84 | 0.04 |
| wave | 0.13 | 0.04 |
| Average | **0.65** | **0.04** |

Table 5.2: Anomaly detection results for experiments with activities intentionally removed from the training data. True positive and false positive rates indicating the relative number of frames correctly or incorrectly classified as anomaly, respectively.

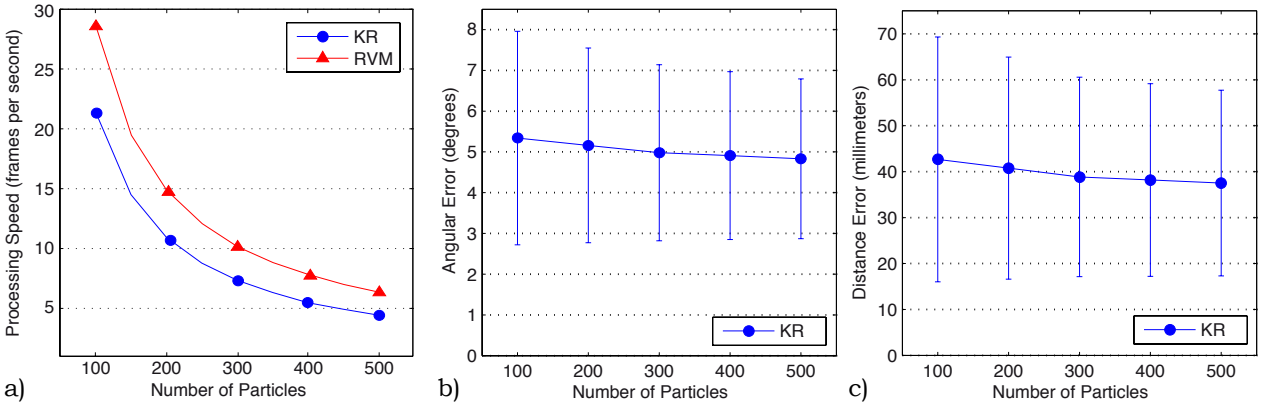



Figure 5.10: Influence of the total number of particles on the tracking algorithm. a) Average frame rates for experiments with 100, 200, 300, 400 and 500 particles. b) Average angular error $e_{\text{ang}}$ with standard deviations for each of the experiments with varying number of particles. Error and standard deviation decrease slightly for higher numbers of particles. c) A similar behavior is present for the average distance error $e_{\text{dist}}$.

for *scratching head*. This effect is due to the similarity of these two activities. For instance, when *scratching head* is missing in the training data, the corresponding frames are classified as *waving* with a sufficiently high confidence measure $c_t$. As a result, no anomaly is reported. Figure 5.7(b) illustrates the anomaly detection process for the situation when two activities, *binding laces* and *jumping jack*, were intentionally removed from the training data. Obviously, the frames where these activities are performed, are misclassified. However, the confidence $c_t$ clearly drops during these periods, allowing our method to recognize the anomaly. Frames falsely detected as anomaly occur only occasionally (e.g. during *golfing*) and can be caused by movements that deviate significantly from the training data. In this case, however, the full-body prediction is not negatively affected, since the activity is correctly classified and the correct motion model is used. The error plot in Figure 5.7(b) shows that the full-body pose prediction strongly deviates from the ground truth only when the person is performing the *jumping jack* and *binding laces* movements that were missing in the training data.

#### 5.4.1.5 Computational Performance and Number of Particles

We evaluated the effect of varying the total number of particles $n$ in terms of processing performance and pose estimation quality. The experiments described so far were repeated with different values for $n$, ranging from 100 to 500. The kernel regression-based approach was used and the data of *one* actor. As shown in Figure 5.10, the frame rates achievable with our Matlab implementation decrease with more particles. However, nearly interactive frame rates can be achieved with a reasonable number of particles, such as 200 or less. For comparison, we included the frame rates for the RVM-based approach in Figure 5.10 a). RVM regression is able to speed up the overall algorithm by a factor that is almost constant for all numbers of particles. It is worth noting that the pose reconstruction quality, measured as $e_{\text{ang}}$ and $e_{\text{dist}}$, deteriorates only slightly for small numbers of particles. Likewise, the correct classification rates decrease slightly from 92% for 500 particles to 89% for 100 particles.

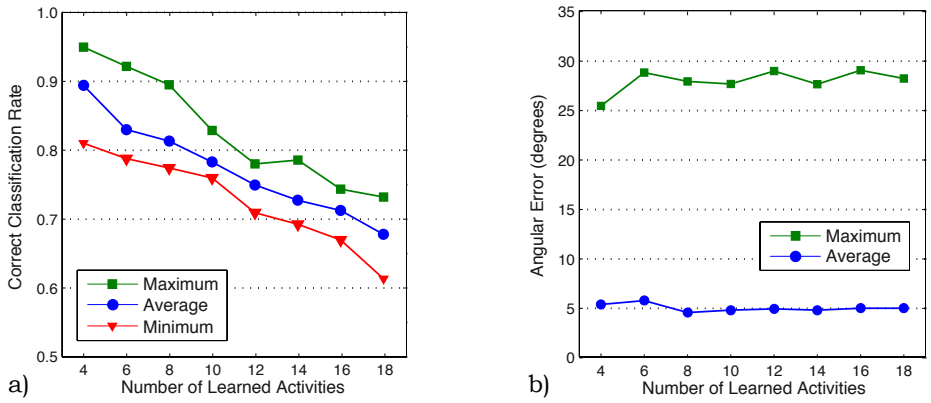



Figure 5.11: Sensitivity of activity recognition and pose estimation to the number of activities learned simultaneously. a) Activity classification rates for different numbers of simultaneously considered activities. b) Pose estimation accuracy for different numbers of activities.

#### 5.4.1.6 Scalability to Many Activities

Learning activity-specific motion models can potentially lead to degeneracy problems when a large number of activities is considered simultaneously. In such cases, a lot of overlapping poses between different activities can be expected, causing confusion for the activity recognition method. To evaluate the behavior of the proposed method for different numbers of activities, we used the dataset of 18 gestures listed in Table 7.1, performed by one subject in four repetitions. The distinction between activities and gestures is irrelevant, except for the fact that the gestures are more prone to being confused than activities, as only arm movements are included. Examples of gestures in this dataset are raising and moving one or both arms horizontally in front of the body, tracing out circles with one or both arms, etc. In a series of experiments, we randomly selected subsets of $\mu \in \{4, 6, 8, 10, 12, 14, 16, 18\}$ activities from the 18 available ones for training and for testing. Different recordings were used for training and for testing. Each random selection of $\mu$ activities was repeated 10 times, the results were averaged. Figure 5.11 shows that varying the number of considered activities affects classification rates, but does not affect the accuracy of pose estimation. Moreover, the confusion in activity classification is moderate, even for as many as 18 simultaneously learned activities.

#### 5.4.1.7 Sensitivity to Sensor Placement

In order to evaluate how sensitive our method is to an exact placement of the inertial sensors, we performed an experiment with one actor and 8 activities, varying the position of the sensors on the person's body. Using a training sequence containing all activities, recorded with the sensors in a reference position, we tested on sequences with incremental rotational and translational deviation of the sensors. Translation was changed to 4, 8 and 12 cm (along arms and legs) from the reference position, rotation (around the arm and leg axis) was changed to approximately 30, 60 and 90 degrees. Figure 5.12 a) shows that activity recognition is more sensitive to rotational deviation than to translations. However, rotations up to 30° and translations up to 4 cm almost do not affect correct classification rates. We argue that rotational deviations close to 90°, where recognition rates drop below 60%, are not to be

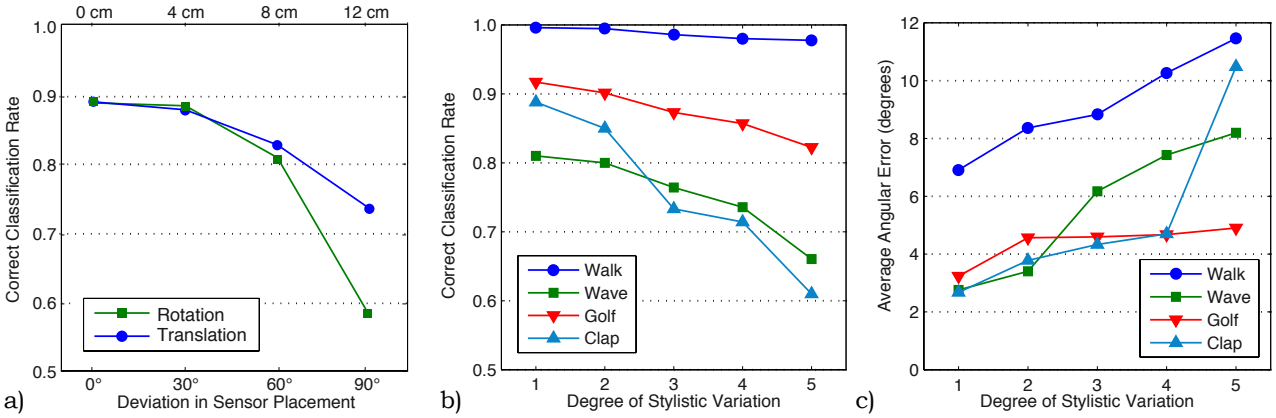



Figure 5.12: Sensitivity to sensor placement and stylistic variation. a) Activity classification rates for incremental rotational and translational deviations in sensor placement from a reference position. b) Activity classification rates for five incremental degrees of stylistic variation for four exemplary actions. c) Accuracy of pose estimation for the same settings.

expected in real-life scenarios, where sensors can shift and rotate slightly while being worn. The results also imply that the sensors can be taken off and on again without new training.

#### 5.4.1.8 Sensitivity to Stylistic Variation

Although our method is based on person-specific training data, stylistic variations in performing the learned movements will likely occur in practical applications. To assess the capability of the proposed approach to cope with such variations, we performed a series of experiments. For four exemplary actions (*walk*, *wave*, *golf* and *clap*) we recorded six additional sequences, where the actor was varying movement style with incremental severity. We used the first sequence for training and tested on the other five, as shown in Figure 5.12 b)-c). In variation 1, the person performed the movements as similar to the training data, as possible. Classification rates range from almost 100% for *walk* to above 80% for *wave*. The average angular error lies, for variation 1, between 3° per joint per frame and 7°. The variations for the *walk* action included striding, jogging and simulated running on a treadmill, while the trained style was very slow and regular walking. At the most severe degree of variation, the person moved in a very sloppy way. *Waving* and *clapping* were performed with different amplitudes and in different spatial locations. Figure 5.12 b)-c) illustrates that stylistic variation does affect activity classification rates and precision of pose estimation. However, classification rates remain above 70% for most of the stylistic variations. In the case of *walk*, above 95% of frames were correctly classified, even for the treadmill running style. The angular error between estimated and ground truth poses increases to 5° for *golf* and to 12° for *walk*. As all predicted poses are reconstructed from the known poses in the training data, this behavior can be expected.

## 5.5 Using Depth Cameras as Input Modality

Depth cameras, such as Time-of-Flight (ToF) cameras or the Kinect, are an attractive imaging modality, as they provide dense 3D scans of a scene in real-time. In this section, we discuss the

results of our evaluation of the generative activity recognition and pose estimation framework on data from a ToF camera. Before explaining the experimental setup and outcome, we describe our feature extraction approach. While the inertial sensor data is sparse enough to be used in its entirety without dedicated feature extraction, we need to define a way of obtaining feature descriptors usable in our method from raw depth images. To prevent being dependent on error prone body part detection in ToF data, we design a global feature representation describing the human body shape at each instant. As opposed to feature descriptors commonly used for 3D shape recognition [76, 61], our feature representation varies continuously with the performed motion.

### 5.5.1 Feature Extraction

Approaches for pose tracking that depend on detecting body parts in the depth data, e.g. [117, 49], are susceptible to noise and self-occlusions. Another strategy for feature extraction is to use general 3D shape descriptors, such as shape contexts or spin images [76]. This type of features have been used for gesture recognition from depth data [61]. However, these descriptors are mainly suitable for shape classification and do not adequately represent subtle pose changes, as required for full-body tracking. We propose a simple but general feature descriptor that, first, does not rely on body-part detection and, second, varies smoothly with the movements of a person. In fact, it is the generative manifold learning-based motion model that enables full-body tracking from such simple depth cues. Given a ToF depth map $\mathbf{D}_t$ at time $t$, we obtain a feature vector $\mathbf{x}_t \in \mathbb{R}^{d_x}$ in three steps:

1. **Preprocessing** to remove noise and to convert the depth map to 3D information.
2. **Segmentation** of the person from the scene background.
3. **Extraction** of simple features related to the boundaries of the observed 3D point cloud.

We apply a median filter to the depth map to remove noise. The filtered depth map is then transformed to a 3D point cloud using the intrinsic parameters of the ToF camera [58]. The coordinates $(X, Y, Z)$ of a 3D point corresponding to a depth map pixel $\mathbf{D}_t(x, y)$ can be computed as

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} ((x - c_x) \cdot \mathbf{D}_t(x, y)) \ / \ f_x \\ ((y - c_y) \cdot \mathbf{D}_t(x, y)) \ / \ f_y \\ \mathbf{D}_t(x, y) \end{pmatrix}, \tag{5.19}$$

where $(c_x, c_y)$ is the principal point of the ToF camera and $(f_x, f_y)$ is its focal length. We will denote the set of 3D points obtained this way as $\tilde{\mathbf{D}}_t$. Note that this representation of the scene is invariant to scaling that can be caused, for instance, by the person moving towards or away from the camera. We proceed in an analogous way with a background depth map $\mathbf{D}_{\text{bg}}$ that we process using Equation 5.19 to obtain the 3D point cloud $\tilde{\mathbf{D}}_{\text{bg}}$.

In order to segment a person in front of the ToF camera, we perform static background subtraction [87], as illustrated in Figure 5.13 a)-c). We discard all 3D points in $\tilde{\mathbf{D}}_t$ if their $Z$ coordinate is close to that of the corresponding points in $\tilde{\mathbf{D}}_{\text{bg}}$. The centroid $\mathbf{c}_t$ of the remaining 3D points is computed and the 3D space occupied by the points is divided into $b$ cells, followed by determining the bounding box of each cell. We then extract the vectors $\mathbf{v}_{t,k}$ for $k \in \{1, \ldots, b\}$, connecting the centroid to the bounding box corners that are furthest apart

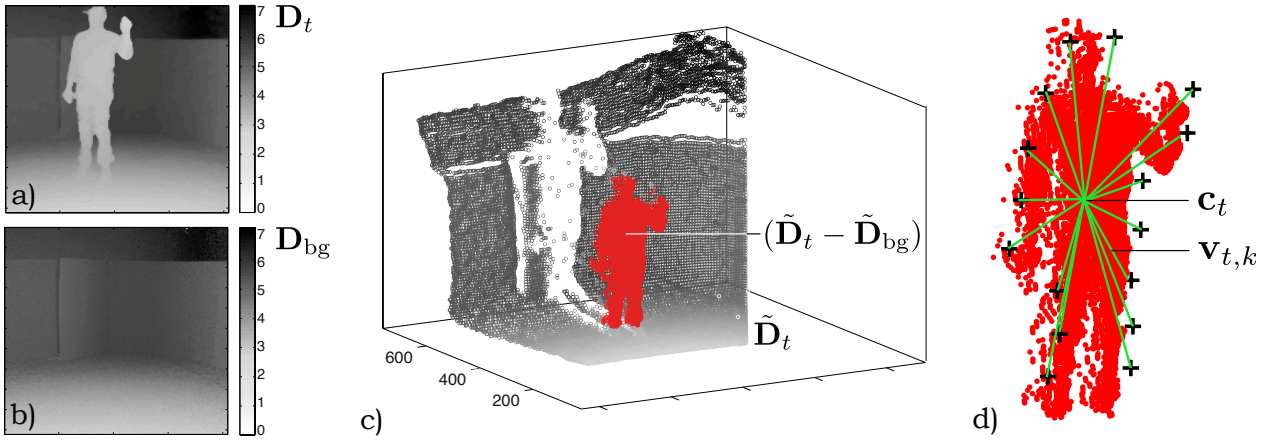



Figure 5.13: ToF feature extraction process. a) Filtered depth image, b) background image, c) 3D segmented foreground, d) feature descriptor based on 3D vectors (lines) from the centroid of the segmented person to extremal points on the surface boundary (crosses).

and closest to the camera – see Figure 5.13 d). These corner points are equivalent to a sparse silhouette representation but in 3D. The feature descriptor is then given by $\mathbf{x}_t = [\mathbf{v}_{t,1} \dots \mathbf{v}_{t,b}]^\top$. Using relative vectors from the centroid to the corner points $\mathbf{v}_{t,k}$ as entries of the feature descriptor makes the descriptor invariant to translations of the person parallel to the camera. For noise reduction, we apply a moving average that includes the previous feature vector $\mathbf{x}_{t-1}$.

### 5.5.2 Evaluation

The following sections present the results of our evaluation. Since a synchronized dataset of depth images and motion capture data was not available online at the time of writing, we recorded our own dataset using a PMDVision CamCube ToF camera, see Figure 1.4 a) [121], and an ART Dtrack2 tracking system [2]. The ToF camera has a resolution of 204 × 204 pixels. Depth features were extracted from the ToF images by subdividing the background-subtracted 3D point cloud into 16 cells (8 vertical, 2 horizontal). The feature descriptor thus has $d_x = 16 \times 3 = 48$ dimensions. As in our experiments with the inertial sensor data, we generated manifold embeddings of $d_z = 2$ dimensions from the full-body pose motion capture data using Laplacian Eigenmaps.

Our ToF-based dataset is structured similarly to the inertial sensors dataset described before. We considered $M = 10$ activities: clapping, golfing, hurrah (arms up), jumping jack, knee bends, picking something up, punching, scratching head, playing the violin and waving. Each of the movements was recorded 6 times with 10 actors. The testing data consists of 6 sequences per actor containing all activities in a row ($\sim$ 1500 frames per sequence). Only the depth features were used for testing, the motion capture data served as ground truth. Our experiments on classification (Section 5.5.2.1) and pose estimation (Section 5.5.2.2) were performed in a cross-validation scheme, i.e. each testing sequence was generated from one of the recordings per activity and actor, using the remaining five for training. All presented experiments have been performed with $n = 300$ particles. In the evaluation on depth data, we used kernel regression (KR) for modeling the predictive mappings from the manifold embeddings to observation space and to full-body pose space.

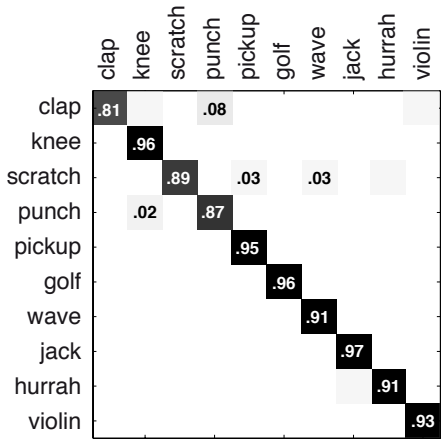



Figure 5.14: Activity classification results averaged over all experiments. Entry $(i, j)$ of the confusion matrix corresponds to the relative number of frames belonging to activity $j$ that were classified as $i$. Matrix entries $< 0.01$ are hidden for clarity.

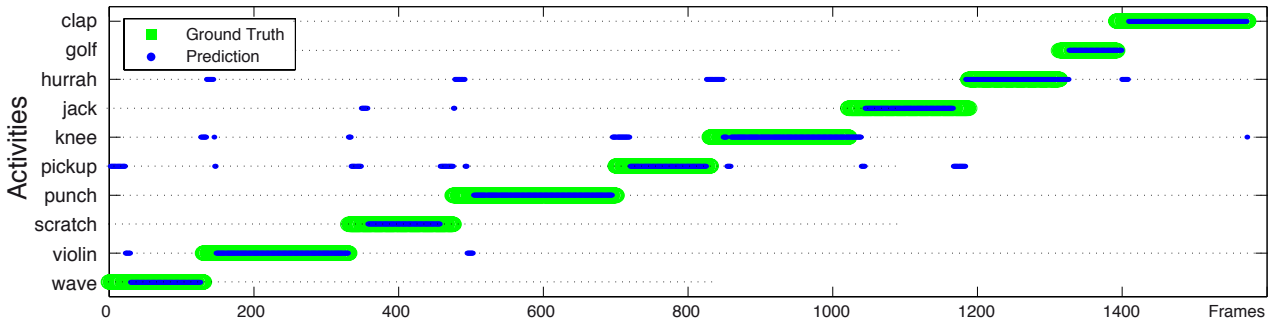



Figure 5.15: Activity classification results for one of the testing sequences. Ground truth classification (green) and predicted activities (blue) are shown for each frame of the sequence.

#### 5.5.2.1 Activity Classification

Figure 5.15 shows exemplary activity classification results for *one* of the testing sequences. Misclassifications mainly occur at the beginning and end of activities, which corresponds to periods of the idle standing pose that is common to all activities. The confusion matrix in Figure 5.14 gives the classification rates for all activities over *all* testing sequences. On average, we achieved a correct classification rate of 92% for all non-idle frames. As in our experiments on inertial sensor-based input data, we disregarded the classifications for all idle-standing frames in our quantitative results. The matrix is mostly diagonal, minor confusion only occurs between activities that consist of similar full-body poses, such as *waving* and *scratching head*. Misclassification in these cases can hardly be avoided but does not necessarily affect the precision of full-body pose estimation.

#### 5.5.2.2 Pose Estimation

We measured how precisely the poses estimated by our method match the ground truth using the two metrics presented in Chapter 4. The angular error $e_{\text{ang}}$ (Equation 4.7) gives the

| | KR | |
|---|---|---|
| Activity | $e_{\text{ang}}$ | $e_{\text{dist}}$ |
| clap | 3.00 | 20.1 |
| golf | 4.02 | 37.4 |
| hurrah | 5.47 | 28.5 |
| jack | 8.78 | 53.1 |
| knee | 4.64 | 41.3 |
| pickup | 3.51 | 29.9 |
| punch | 3.67 | 23.8 |
| scratch | 2.56 | 15.7 |
| violin | 3.64 | 24.8 |
| wave | 2.83 | 16.2 |
| Average | **4.21** | **29.1** |

Table 5.3: Pose estimation accuracy for all considered activities. Differences to ground truth poses are shown as joint angles ($e_{\text{ang}}$ in degrees per joint) and as distances ($e_{\text{dist}}$ in millimeters per joint), averaged over all experiments. Standard deviations $\sigma_{\text{ang}}$ and $\sigma_{\text{dist}}$ are provided.

deviation from the ground truth in terms of joint angles. The distance error $e_{\text{dist}}$ (Equation 4.8) is the difference in 3D space between predicted joint locations and the ground truth. Averaged over all frames of the testing sequences, we achieved $\bar{e}_{\text{ang}} = 4.21°$ per joint and $\bar{e}_{\text{dist}} = 29.1$mm. As shown in Table 5.3, the deviation from the ground truth only increases for fast movements with a large variability, such as *jumping jack*. Figure 5.16 illustrates the achievable accuracy for a qualitative inspection. Our results are comparable to other state-of-the-art methods using visual observations as input, e.g. [4, 167].

## 5.6 Discussion

In this chapter, we have presented a framework for simultaneous activity recognition and full-body pose estimation from limited and noisy observations. In order to cope with the sparsity and the often ambiguous nature of the measurements, we learn a compound model of human motion from full-body pose training data. The method is efficient, since we track poses in a low-dimensional space of manifold embeddings and use non-linear regression to relate the embedding space to observations and to full-body poses. We have evaluated the method with inertial sensor data and depth images from a ToF camera as an input. Our experiments showed that the method can recognize motions of multiple activities and track human full-body poses from inertial sensor data and depth images. Furthermore, the method tolerates stylistic variation of movements and deviations in sensor placement. We conclude the chapter with a discussion of a few key insights.

**Tracking Unknown Movements.** Our method requires that the motion model is initially trained on a set of activities that are expected to occur in a particular application scenario. While the model allows stylistic variation between instances of the same motion (Section

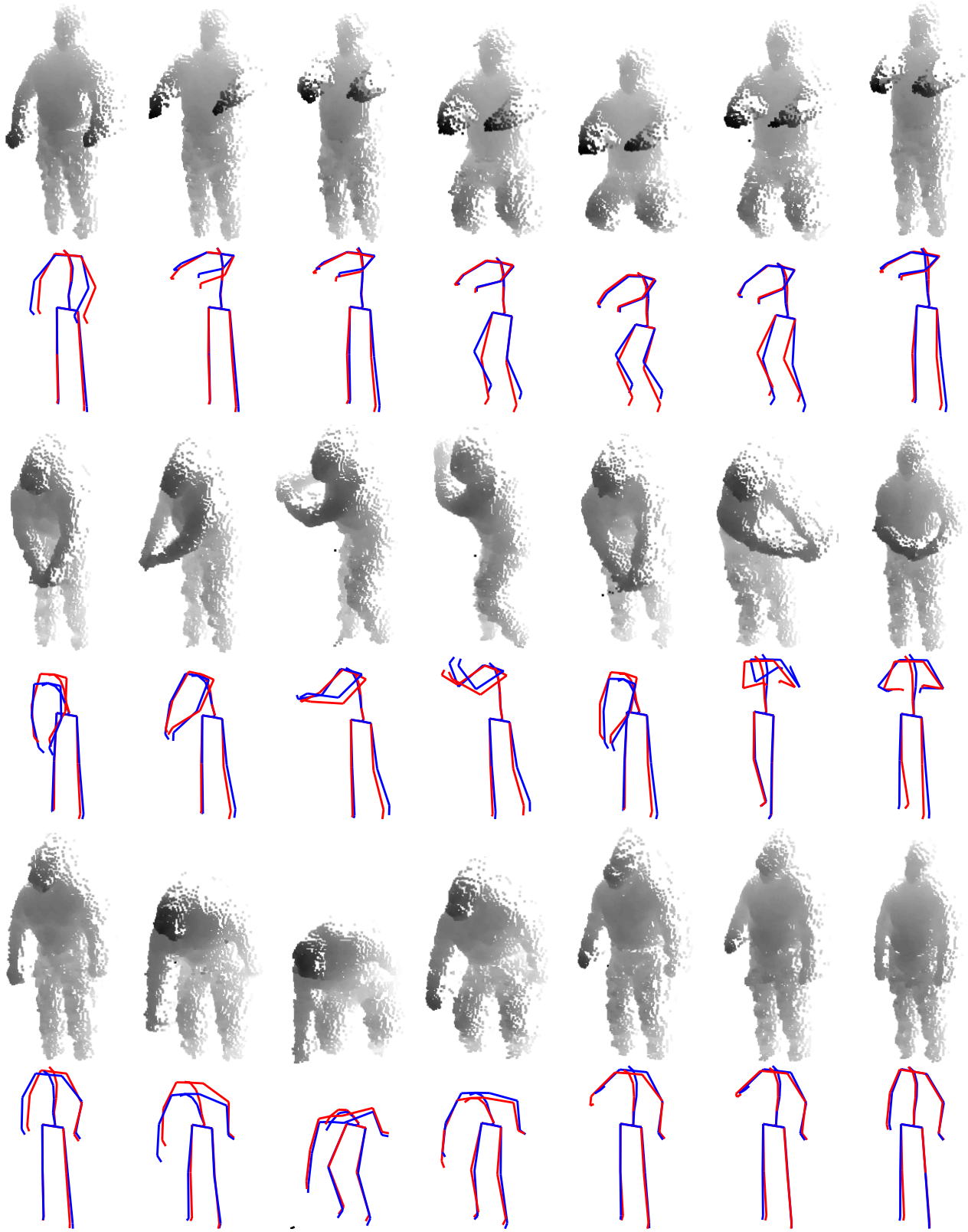


Figure 5.16: Illustration of input ToF data and corresponding pose estimation results for three activities (knee bends, golfing, picking something up). Estimated (red) and ground truth poses (blue) are shown below each of the ToF samples.

5.4.1.8), completely unseen movements will not be reconstructed precisely. However, our method will still provide a pose estimate that matches the new observations as close as possible. When the predictive confidence is low, the movement will be labeled as anomalous. In an application scenario where a correct action recognition is vital, this would allow the pose estimation system to be deactivated in order to prevent erroneous predictions. As soon as a known movement is performed, the multiple-hypothesis tracker will furthermore quickly recover the correct pose. We also do not require pose initialization, since when tracking begins, particles are distributed to sample all learned feasible poses.

**Properties of the Particle Filter.** As shown in Section 5.4.1.5, the total number of particles is not a critical parameter to our method. The main reasons for this favorable behavior are twofold: (1) the low dimensionality of search space as compared to methods operating in a high-dimensional parameter space and (2) the fact that full-body pose predictions are generated by regression mappings from training data. The latter property ensures that full-body poses will never deviate significantly from the known movements. In fact, when using kernel regression, the space of possible pose predictions is restricted to a convex combination of the poses in the training data. Moreover, small variations in embedding positions, as realized by using larger numbers of particles, only have a small influence on full-body pose predictions.

The only constraint on the number of particles is that there are sufficiently many particles on every manifold embedding, even when most particles follow a high-likelihood hypothesis. This is to ensure that a sudden change of activity by the person can be recognized quickly. Therefore, the appropriate total particle number mainly depends on the minimum activity switching probability $p_k$ (Section 5.3.2.4) and the number of considered activities $M$. Let us assume we use $n = 400$ particles and $p_k = 0.3$, as in our experiments. In this case, $0.3 \cdot 400 = 120$ particles will, in each step of the particle filter, randomly switch away from their current manifold embedding and and will be distributed across 9 of the 10 embeddings. The remaning $400 - 120 = 280$ particles will follow a high-likelihood pose hypothesis and will therefore be concentrated on one manifold embedding. We would thus constantly expect $120/9 \approx 13$ particles on each of the 9 other embeddings. In our experiments, this number was sufficient for allowing our method to quickly catch up with activity switches.

**Invariance Properties of Depth and Inertial Sensor Data.** The depth images from ToF cameras allow us to easily segment the foreground and to extract features that are invariant to a translation of the person in front of the camera. We do not explicitly address rotational invariance, e.g. to handle a person turning around the vertical axis. We argue that this is sufficient for interactions where the person is facing the camera. Although the sequences in the experiments included small variations, more severe rotations of the person would require to include lateral poses in the training data. Experiments using a 2D version of our depth descriptor confirm the advantage of using the 3D information, as well as the ability of our simple depth descriptor to capture the 3D information. Obviously, the general nature of our activity recognition and pose estimation framework also allows selecting different, more advanced 3D depth features. When using wireless inertial sensors as a source of input observations, the person is almost completely unlimited in his or her movements. As long as the wireless reception between the sensors and the receiver is maintained, neither translational nor rotational movements of the person have any effect on the measured sensor observations.

**Comparison to the Discriminative Method.** It is worthwhile comparing the activity recognition and pose estimation framework to the discriminative method presented in Chapter 4. Comparability is straightforward, as we used the same sensor-based training and testing dataset in both cases and both algorithms also perform simultaneous activity recognition and pose estimation. The generative method based on manifold learning achieves a correct classification rate of 89% of all non-idle frames. Both versions of the discriminative method (mGPR and mSVM) can only correctly recognize the performed activity in 70% and 74% of all frames, respectively. In terms of pose estimation accuracy, the discriminative method reaches the RVM variant of the generative approach. However, in the KR version, the generative method is clearly superior. While the discriminative method achieves $\bar{e}_{\text{ang}} = 6.64°$ per joint and a distance error of $\bar{e}_{\text{dist}} = 53.8$mm, the results for the manifold-based algorithm are $\bar{e}_{\text{ang}} = 6.23°$ and $\bar{e}_{\text{dist}} = 45.2$mm per joint.

The improvement by the generative approach is even more evident when considering the maximal angular error over all testing sequences. While the discriminative method leads to a maximum angular error of 80.8°, the generative method only deviates from the ground truth poses by a maximum of 46.5° per joint per frame. A detailed inspection of pose estimation results shows that the proposed low-dimensional, generative motion model increases the robustness against outlier sensor measurements. In such cases, the particle filter introduces a temporal coherence between subsequent frames, while the discriminative method more easily loses track in single frames and confuses activities. In the light of practical applicability, the gained advantage of the generative method over the discriminative approach is considerable, given that even spurious misclassifications can negatively affect the user experience. However, due to its flexibility and computational simplicity, the discriminative method is suitable as an initialization method for other pose estimation approaches.

**Inertial Sensors Compared to Depth Camera Data.** Our experiments demonstrate that depth camera data can lead to better results, both in terms of activity recognition and pose estimation. While the best average angular and distance error values for the sensor-based dataset are $\bar{e}_{\text{ang}} = 6.23°$ and $\bar{e}_{\text{dist}} = 45.2$mm per joint, the depth data allows for a notably better $\bar{e}_{\text{ang}} = 4.21°$ and $\bar{e}_{\text{dist}} = 29.1$mm per joint. Using depth features, our generative method correctly classified 92% of all frames in the testing dataset, compared to a maximum of 89% for the inertial sensor dataset. The total number of considered activities, testing persons and frames is analogous in both, the inertial sensor dataset and the ToF dataset.

In interpreting these results, one has to keep in mind that the observation vectors extracted from the depth images contain more information than the inertial sensor data. As stated before, we employed only six inertial sensors to provide an unobtrusive way of capturing a person's movements. Given that we used two streams of orientation values per sensor (omitting the yaw value to be independent of magnetic north), activity recognition and pose estimation takes place based on a merely 12-dimensional input. On the other hand, the feature vectors we obtain from the depth data are 48-dimensional, and thus more informative. While pose estimation and activity recognition from depth data has attractive applications, we believe that the (only slightly worse) results we achieved with the inertial sensors are quite noteworthy.

Chapter 6

# Graph-based Human Pose Estimation using Depth Data

## 6.1 Introduction

In the previous chapters, we have investigated machine learning methods for human pose estimation and activity recognition. We have approached the problem of predicting human body poses and classifying activities from a discriminative and a generative perspective. Manifold learning proved to be a valuable technique for creating low-dimensional prior models of human motion that can help to cope with the challenges of sparse and noisy observation data. We have also seen the limitations of learning-based methods related to generalization that are mainly due to the dependence on the amount and the quality of the training data.

We now consider an entirely different approach to human pose estimation that does not rely on machine learning. In this excursus from the main theme of our work, the objective is to make maximum use of the depth data delivered by a ToF camera or Kinect. In our previous experiments with depth data, we used a straightforward feature extraction mechanism and showed how to overcome the simplicity of these features using a learning-based method. Here, we do more justice to the depth data and explore how far we can go in pose estimation, given this data alone. In fact, the data provided by ToF cameras and the Kinect delivers both, 3D coordinates and 2D grid information. Additionally, ToF cameras generate grayscale images and the Kinect produces RGB images that are easily co-registered with the depth data. In this chapter, we do not address the activity recognition problem, but focus on tracking the limbs of a person. The application scenario we target is gesture-based human-machine interaction, where a person is facing a screen and also a depth camera mounted nearby.

Human gestures are a natural means of communication and allow complex information to be conveyed. Using gestures for interaction with computers can be of great benefit, particularly in scenarios where traditional input devices are impractical. The method we present in this chapter is based on robustly identifying anatomical landmarks in the depth data that then serve as targets for fitting a skeleton. We propose to represent the background-subtracted depth data by means of a graph that facilitates the detection of body parts. In addition, we use the optical flow between consecutive intensity or RGB images for depth disambiguation when body parts occlude each other.

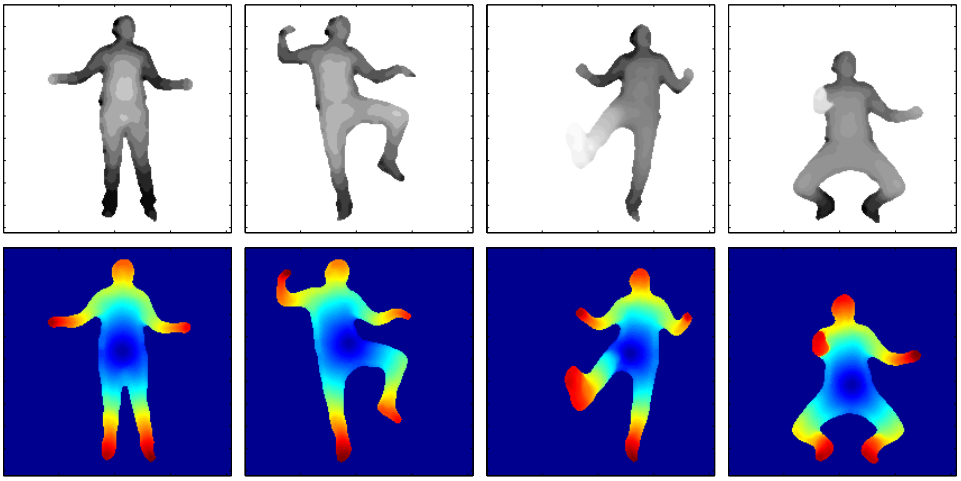


Figure 6.1: Illustration of the robustness of geodesic distances against pose changes. *Top:* Background-subtracted depth images for various poses. *Bottom:* Geodesic distances from the body center to all other surface points. Colors range from blue (zero distance) to red (maximal distance). Note that the distance to hands and feet remains almost constant across all poses.

Representing the 3D points on the surface of a person as a graph allows us to measure geodesic distances between different points on the body. While the Euclidean distance between two body points is measured through 3D space and thus can vary significantly with body movement, the geodesic distance is defined along adjacent graph nodes, i.e. along the surface of the body. Consequently, the geodesic distance between two points on the body, e.g. the centroid and an extremity, can be assumed constant, independent of body posture [117, 104] (Figure 6.1). We can therefore extract anatomical landmarks by searching for points at mutual geodesic distances that correspond to the actual measurements of a person. Using only depth and intensity data also decreases our dependence on visual appearance. Thus, we avoid typical problems that arise when using intensity-based feature descriptors for interest point detection, e.g. lack of texture and illumination or perspective changes.

When body parts occlude each other, the graph constructed from the 3D points can degenerate. In this case, separating the occluding body part from the part behind it becomes difficult, leading to undesired graph edges. These edges between points on different body parts result in erroneous geodesic distances, and consequently, lead to undetected anatomical landmarks. We address this issue by taking into account the motion occurring between subsequent frames. In particular, we identify and remove the undesired graph edges based on optical flow fields that are computed from the intensity or RGB images.

The method presented in this chapter takes full advantage of the available information by simultaneously using depth images (for segmentation and generation of 3D points), intensity or RGB images (for optical flow) and the graph-based representation (for geodesic distances). Compared to other depth camera-based human body tracking approaches, ours is able to quickly recover from tracking failures due to the robust anatomical landmark detection approach. We use a ToF camera and a Kinect device in our experimental evaluation. The experiments show that the method is able to effectively track the full-body pose in several ToF and Kinect sequences containing various human movements.

## 6.2 Related Work

Techniques for human pose estimation from visual observations can be broadly categorized into learning-based approaches that facilitate the problem by means of training data (e.g. [164, 72, 155, 136]) and approaches that estimate human poses from observed features without prior knowledge. We provided a literature overview of learning-based methods for human pose estimation in Chapter 5, including a section on approaches using depth data as input.

**Visual Human Pose Estimation Without Learning.** Methods that do not rely on machine learning (e.g. [82, 184, 12, 122, 75, 108]) are typically highly dependent on a reliable feature extraction, as the appearance of the human body is heavily affected by illumination and pose changes, and by noise in the observation data. As opposed to learning-based methods, there is no prior knowledge of feasible human movements or human appearance that can help disambiguate insufficient observations. Most pose estimation methods without prior motion or appearance models therefore build upon the rich data obtained from multi-view reconstruction systems. For instance, Kehl and van Gool [82] use a multi-camera setup and generate 3D volumetric reconstructions for human pose estimation. Gall et al. [47] fit a skeleton to multi-camera data by repeatedly deforming and refining a surface mesh that is rigged to an articulated body model. Other approaches focus on tracking the surface of a person in 3D [62, 23, 24]. These methods are able to recover fine details of the outer appearance, including clothing. The key disadvantage is the remaining requirement of a multi-camera reconstruction system. Obviously, the quality of the depth data we obtain from a ToF camera or the Kinect cannot match that of a 3D reconstruction system. However, the data is available without the complexity of 3D reconstruction algorithms in real-time.

When no learned models of human motion are available, efficient state inference techniques also play a crucial role to deal with the high dimensionality of full-body pose space. Bandouch et al. [12] employ a particle filter for state estimation in a multi-view setup. However, their approach reaches low, non-interactive frame rates. The authors of [46] address this problem by combining particle filtering with local optimization on multiple hierarchical layers. In [122], inertial sensors are attached to the human body to complement visual observations and thus to guide the search for correct human poses.

**Using ToF Cameras for Human Pose Estimation.** Several authors have recently explored the use of ToF cameras for an analysis of human motions (e.g. [20, 149, 74, 185, 136]). In [149], a system is described that recognizes simple hand gestures for navigation in medical imaging applications. Hand movements are also tracked in [99]. The method of Jensen et al. [74] allows tracking the movement of legs in side-views for medical gait analysis. Holte et al. [61] propose a method that integrates ToF range and intensity images for human gesture recognition. Their approach is not used for pose tracking, but is able to classify upper-body gestures, such as raising an arm. The authors avoid identifying anatomical landmarks by using a global pose descriptor. Zhu et al. [184] present a full-body pose estimation system that relies on fitting templates for each body part to the ToF data. In [185], the authors combine a template fitting technique based on dense point correspondences with an interest point detector for increased robustness. While the approach can track full-body motion, it relies on an independent, heuristic treatment for each body part.

**Using the Kinect for Human Pose Estimation.** After its recent appearance, the Microsoft Kinect has caused a multitude of applications to appear, including approaches for human body tracking. For instance, Hu et al. [66] extract leg movement information for medical gait analysis using a Kinect that is mounted on a moving walker. While appropriate for their intended purpose, this method cannot track full-body poses. Shotton et al. [146] present a method for body joint detection in Kinect depth images. Their method is based on a learning approach that classifies individual depth image pixels to one of multiple body parts, such as upper arm, elbow or thigh. A random forest classifier is trained with a large database of labeled depth images. The training database is generated by applying motion capture data to a variety of synthetic body models, thus obtaining training data for various body proportions. In [51], the authors present an alternative pose estimation technique based on random forest regression. While both approaches decrease the sensitivity to person-specific appearance, they still depend on the movements contained in the training dataset. In addition, the performance of the algorithms is to a large extent due to the volume of the used training dataset and to the exceptional computational capabilities available during the training phase [51, 146].

**Graph-based Human Pose Estimation.** Similar to our approach, Plagemann et al. [117] use a ToF camera and create a graph representation of the 3D data for detection of anatomical landmarks. Their technique extracts interest points with maximal geodesic distance from the body centroid and classifies them as hands, head and feet using a classifier trained on depth image patches. The method does not explicitly address the problem of self-occlusions between body parts and reportedly struggles in such situations [117]. Without modifying the interest point detection technique, the authors add in [49] a pose estimation method embedded in a Bayesian tracking framework. Our proposed method uses optical flow measured in ToF intensity images to cope with body self-occlusions.

**Optical Flow for Depth Disambiguation.** Optical flow has been used in [37] for motion estimation and segmentation of a person in a monocular pedestrian tracking application. Okada et al. [112] describe a person tracking method that combines disparity computation in a stereo setup with optical flow. Similar to our approach for disambiguation in the case of self-occlusions, an interest region map is propagated through the tracking sequence using the computed flow vectors. While this method allows tracking the bounding boxes of the head and upper body, our technique estimates the joint angles of a full skeleton body model in every frame. To the best of our knowledge, using optical flow for segmentation of occluding body parts in depth-image based human body tracking is a novel approach, enabling us to track arbitrary full-body movements from a frontal perspective.

## 6.3 Geodesics and Optical Flow for Pose Estimation

### 6.3.1 Method Overview

We are given a sequence $\{\mathcal{T}_t\}_{t=1}^{N_D}$ of $N_D$ depth measurements, where each $\mathcal{T}_t = (\mathbf{D}_t, \mathbf{I}_t)$ consists of a depth image $\mathbf{D}_t$ and an intensity image $\mathbf{I}_t$, both of size $n_x \times n_y$. When using a ToF camera, $\mathbf{I}_t$ denotes the grayscale amplitude image. For the Kinect device, $\mathbf{I}_t$ refers to the RGB image. While ToF cameras acquire both images with the same sensor, the Kinect uses two

separate cameras for the depth and the RGB images. Thus, the RGB image first needs to be aligned to the reference frame of the depth image, e.g. using stereo calibration techniques. Alternatively, the Kinect programming interface by PrimeSense [124] also provides methods that allow directly accessing color pixels corresponding to given depth image coordinates based on an internal matching procedure. In the following, we assume that there is a correspondence between any depth pixel $\mathbf{D}_t(i,j)$ and an intensity image pixel $\mathbf{I}_t(i,j)$.

Given a sequence of depth and intensity images of a person, our goal is to estimate the full-body pose $\hat{\mathbf{y}}_t \in \mathbb{R}^{d_y}$ of the person at each frame $t$, parameterized by the $d_y$ joint angles of our simple skeleton model (Appendix A). We assume that, in an initialization step, the person faces the camera for a few seconds and takes on a T-pose with the arms extended sideways. During this phase, approximate limb lengths of the person are captured for calibration. The main method consists of the following steps, performed in each frame (see Figure 6.2):

1. **Depth image preprocessing**. The depth image is first median filtered to reduce the amount of noise. In addition, a 3D point cloud representation is computed from the depth data (Section 6.3.2).

2. **Graph construction from depth data**. We create a graph with the 3D points as vertices and introduce edges whenever two 3D points fulfill a specific spatial neighborhood property (Section 6.3.3).

3. **Creation of geodesic distance map**. The graph is used to generate a map of geodesic distances from the body center to all other body points. The distances represented in this map can be assumed invariant to articulation changes (Section 6.3.3.1).

4. **Localization of anatomical landmarks**. We locate $L$ anatomical landmarks $\mathcal{P}_t = \{\mathbf{p}_i^t\}_{i=1}^L$ using the geodesic distance map, where $\mathbf{p}_i^t \in \mathbb{R}^3$, and determine the discrete landmark labels $\lambda(\mathbf{p}_i^t)$, e.g. *head*, *left knee*, *right hand* (Section 6.3.3.2).

5. **Disambiguation using optical flow**. The optical flow between the previous and the current frame, measured using the intensity images $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$, is used to track body parts that occlude each other (Section 6.3.4).

6. **Skeleton fitting to anatomical landmarks**. We finally employ a model-based skeleton fitting approach to estimate the full-body pose that best fits the extracted anatomical landmarks (Section 6.3.5).

### 6.3.2 Depth Image Preprocessing

After median-filtering the incoming depth image $\mathbf{D}_t$, we subtract a previously recorded background image to segment the person in the foreground. We then transform the segmented depth image into a 3D point cloud based on the known intrinsic parameters of the depth camera (see Section 5.5.1). Let $\mathcal{X}_t = \{\mathbf{x}_{ij}\}$ denote the resulting set of $n_x n_y$ 3D points. We assume that, after creating the 3D point cloud, we are still able to identify the 2D depth image coordinates belonging to a given 3D point. Our notation indicates that the point $\mathbf{x}_{ij}$ corresponds to the depth image pixel with coordinates $(i,j)$.

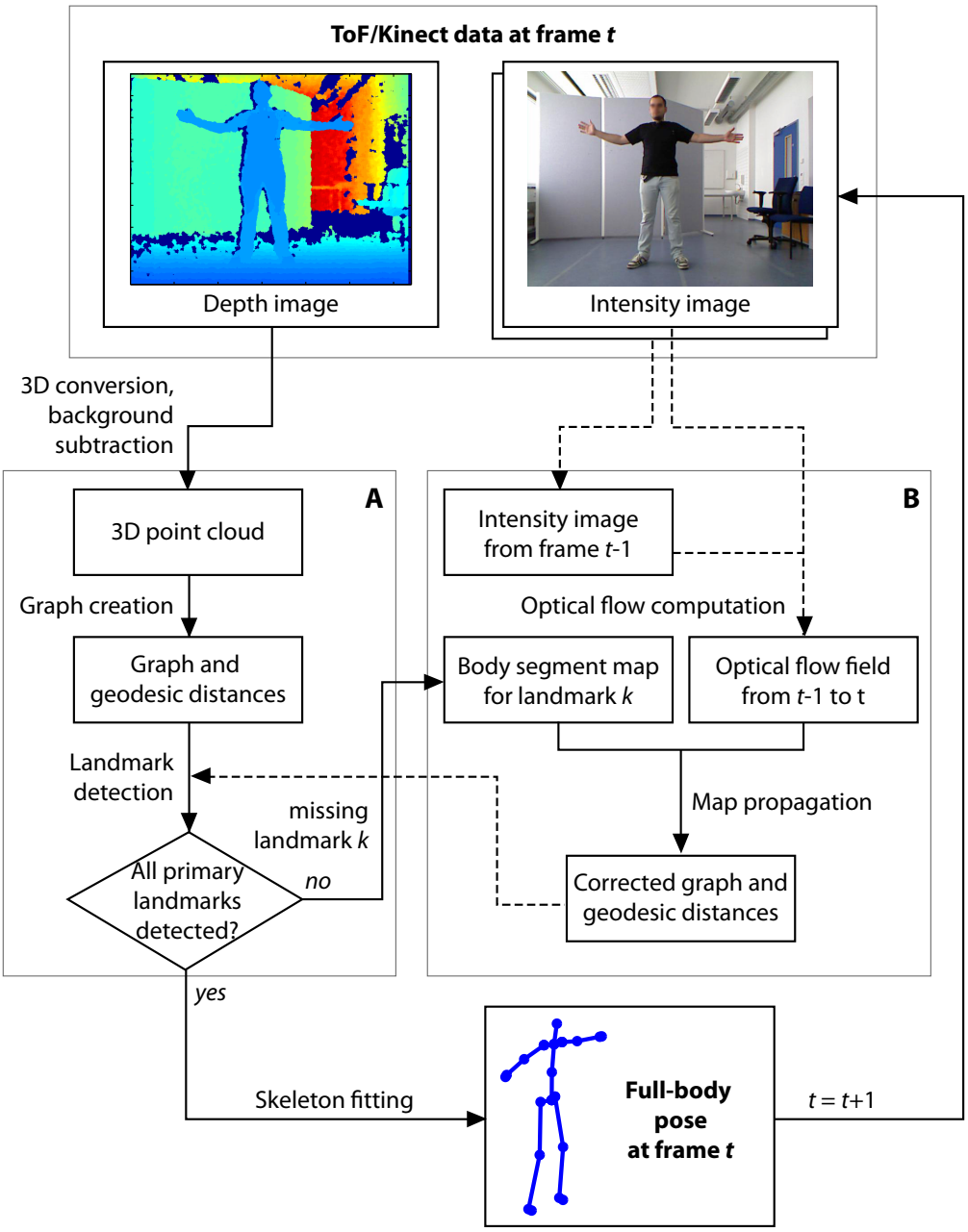



Figure 6.2: Schematic of the depth-image human body tracking method. In each frame $t$, the algorithm constructs a geodesic distance graph based on the 3D-converted depth image and extracts anatomical landmarks (A). For each undetected landmark $k$, the disambiguation process using optical flow on the intensity image is executed (B). The corrected geodesic distance graph for a body part $k$ allows detecting the missing anatomical landmarks.

### 6.3.3 Graph Construction from Depth Data

We construct a graph $\mathcal{G}_t = (V_t, E_t)$, where $V_t = \mathcal{X}_t$ are the vertices and $E_t \subseteq V_t \times V_t$ are the edges. Whether two vertices, i.e. 3D points, are connected with an edge or not is based on their spatial distance in 3D and on their vicinity in the 2D depth image. We thus define the set of edges as

$$E_t = \{(\mathbf{x}_{ij}, \mathbf{x}_{kl}) \in V_t \times V_t \mid \|\mathbf{x}_{ij} - \mathbf{x}_{kl}\| < \delta \wedge \|(i,j)^\top - (k,l)^\top\|_\infty \leq 1\}, \tag{6.1}$$

where $\|\cdot\|$ is the Euclidean and $\|\cdot\|_\infty$ is the maximum norm and $(i,j)^\top$, $(k,l)^\top$ are the 2D coordinates of the two points $\mathbf{x}_{ij}, \mathbf{x}_{kl}$ in the depth image. For each edge $e = (\mathbf{x}, \mathbf{x}') \in E_t$, we store a weight $w(e) = \|\mathbf{x} - \mathbf{x}'\|$. We thus connect points with a 3D Euclidean distance of less than $\delta$ that project to neighboring pixels in 2D. Incorporating the 2D neighborhood allows us to efficiently construct the graph in linear time by traversing the depth image once. The classical way of constructing a neighborhood graph from a set of 3D points would first require the expensive computation of all pairwise point distances in 3D.

#### 6.3.3.1 Creation of Geodesic Distance Map

Using $\mathcal{G}_t$, we are able to measure geodesic distances between different body locations, as opposed to measuring Euclidean distances between the corresponding 3D points. The advantage of geodesic distances is that the local topology of the body is taken into account (see also Section 3.2.2.1). The geodesic distance $d_\mathcal{G}(\mathbf{x}, \mathbf{x}')$ between two points $\mathbf{x}, \mathbf{x}' \in V_t$ is given by

$$d_\mathcal{G}(\mathbf{x}, \mathbf{x}') = \sum_{e \in SP(\mathbf{x}, \mathbf{x}')} w(e), \tag{6.2}$$

where $SP(\mathbf{x}, \mathbf{x}')$ contains all edges along the shortest path between $\mathbf{x}$ and $\mathbf{x}'$. Intuitively, the geodesic distance between two locations on the body is thus the length of the shortest path over the body surface. Our central assumption is that all anatomical landmarks remain at a nearly constant geodesic distance from the body center of mass, independent of body pose [117]. Let $\mathbf{c}^t$ denote the 3D point that is closest to the centroid of the segmented point cloud $\mathcal{X}_t$. We then define the geodesic distance map $\mathbf{M}_t$, of the same size as the depth image $\mathbf{D}_t$, as

$$\mathbf{M}_t(i,j) = d_\mathcal{G}(\mathbf{c}^t, \mathbf{x}_{ij}). \tag{6.3}$$

Examples of geodesic distance maps are given in Figure 6.1. To determine the geodesic distance from the body center for the 3D point belonging to the depth image pixel $\mathbf{D}_t(i,j)$, we evaluate the geodesic distance map at $\mathbf{M}_t(i,j)$. Given a single source point, the shortest paths to all other graph points can be computed using Dijkstra's single source shortest paths algorithm [28]. In an efficient implementation, this algorithm has a complexity of $O(|V_t| \cdot \log |V_t|)$.

#### 6.3.3.2 Detection of Anatomical Landmarks

Having constructed the graph $\mathcal{G}_t$ and the geodesic distance map $\mathbf{M}_t$ in frame $t$, we proceed by locating $L = 11$ anatomical landmarks $\mathcal{P}_t = \{\mathbf{p}_i^t\}_{i=1}^L$ and determining their discrete labels $\lambda(\mathbf{p}_i^t)$. We distinguish between *primary* landmarks $\mathcal{P}_t' \subset \mathcal{P}_t$ (body center, head, hands, feet) and *secondary* landmarks $\mathcal{P}_t'' \subset \mathcal{P}_t$ (chest, knees, elbows).

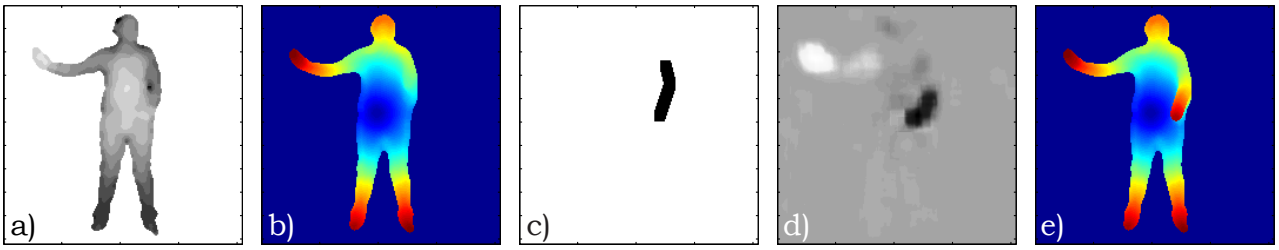



Figure 6.3: Illustration of the depth disambiguation approach using optical flow. a) Background-subtracted ToF depth image with a hand in front of the torso. b) Geodesic distance map computed using the graph $\mathcal{G}_t$ with the origin at the body center. The occluding arm is too close to the torso for being separated. c) Body segment map for the arm obtained in the previous frame. d) Optical flow field ($x$-component) from previous to current frame. e) Geodesic distance map after removal of undesired edges in $\mathcal{G}_t$. The arm is now separated and has the expected geodesic distance from the body center.

The first *primary* anatomical landmark is given by the body center of mass $\mathbf{c}^t$. To extract the extremities, we use the geodesic distance map to select all points $\mathbf{x}$ with $d_G(\mathbf{x}, \mathbf{c}^t) > \tau$. Here, $\tau$ is a person-specific threshold that approximates the distance from the body center to the shoulders. We therefore obtain spatially isolated sets of points that we treat as belonging to different limbs. For each of these isolated sets, we store the point with largest geodesic distance from the body center, yielding the set of primary anatomical landmarks $\mathcal{P}'_t$.

Given the locations of the primary anatomical landmarks, we need to determine their labels in order to detect the secondary landmarks, i.e. the chest, elbows and knees. During initialization, where the person takes on a T-pose, we create an initial labeling of the anatomical landmarks. Each landmark $\mathbf{p}_i^0$ detected in the initialization frame ($t = 0$) is assigned an appropriate label $\lambda(\mathbf{p}_i^0)$ based on the assumed T-pose. In any subsequent frame $t$, we determine the labels for the primary landmarks by matching the detected positions $\mathbf{p}_i^t$ to the known landmarks in the previous frame. The label for the $i$-th landmark is thus

$$\lambda(\mathbf{p}_i^t) = \lambda(\tilde{\mathbf{p}}^{t-1}), \text{ where } \tilde{\mathbf{p}}^{t-1} = \underset{\mathbf{p} \in \mathcal{P}'_{t-1}}{\arg\min} \|\mathbf{p}_i^t - \mathbf{p}\|. \tag{6.4}$$

We can then extract the location of the *secondary* landmarks $\mathcal{P}''_t$, i.e. the chest, elbows and knees, by measuring geodesic distances from the localized primary anatomical landmarks. That is, we select points on the body as the chest, the elbows and the knees that are located at respective distances from the body center, the hands and the feet.

### 6.3.4 Depth Disambiguation Using Optical Flow

In cases when the extremities are clearly separated from each other, the graph-based landmark identification approach allows us to detect all primary and secondary landmarks. However, when body parts occlude each other, the graph $\mathcal{G}_t$ will likely contain edges that connect points on different body parts. In such a situation, two points $\mathbf{x}, \mathbf{x}' \in V_t$ on distinct body parts can easily satisfy the two conditions of Equation 6.1. Consequently, the geodesic distances will be computed inappropriately. Figure 6.3 b) gives an example where an arm in front of the torso

is connected to the upper body and the geodesic distance from the body center to the hand is underestimated. Without correction, anatomical landmarks on the arm cannot be detected.

We therefore propose a disambiguation approach that makes use of movement occurring between frames. Assuming that distinct body parts move separately, this approach allows us to disconnect points belonging to different body parts. We introduce a binary map indicating the location of the entire *occluding* body segment in the depth image. This map is propagated from frame to frame using optical flow, until the body parts become separable again.

#### 6.3.4.1 Creation of Body Segment Map

Let $\mathbf{p}_m^t \in \mathcal{P}_t'$ be the location of a primary anatomical landmark at time $t$ and let $b_m^t$ denote the corresponding body part, i.e. an arm or a leg. We define the body segment map $\mathbf{S}_t^m$ for $b_m^t$ to be a binary image of the same size as the depth image $\mathbf{D}_t$, such that

$$\mathbf{S}_t^m(i,j) = \begin{cases} 1 & \text{if } d_{\mathcal{G}}(\mathbf{p}_m^t, \mathbf{x}_{ij}) < \mu, \\ 0 & \text{otherwise.} \end{cases} \tag{6.5}$$

All pixel locations $(i, j)$ in the map are assigned a value of 1 if the geodesic distance between their corresponding 3D point $\mathbf{x}_{ij}$ and the landmark $\mathbf{p}_m^t$ does not exceed $\mu$. This threshold is chosen based on the length of the person's limbs (determined during initialization), such that the entire body segment $b_m^t$ is included in the segment map. Figure 6.3 c) shows a body segment map for the person's left arm.

#### 6.3.4.2 Map Propagation Using Optical Flow

For every primary landmark $\mathbf{p}_m^t$ that is not detected using the approach described in section 6.3.3.2, we obtain the corresponding body segment map $\mathbf{S}_{t-1}^m$ from the previous frame. If the landmark was detectable in that frame, we construct the map according to Equation 6.5, otherwise we assume that the map is available from previous propagation steps. Let $\mathcal{F}_t = (\mathbf{F}_{t,x}, \mathbf{F}_{t,y})$ denote the optical flow between the intensity images $\mathbf{I}_{t-1}$ and $\mathbf{I}_t$. Note that we use the ToF grayscale amplitude image or the Kinect RGB image, respectively. $\mathbf{F}_{t,x}(i,j)$ is the $x$-component of the estimated movement for pixel $(i, j)$ between the two images, and similarly for $y$. While different methods for obtaining $\mathcal{F}_t$ can be used, we employed the classical Horn-Schunck method with satisfactory results [63]. Figure 6.3 d) shows an exemplary flow field. We use the optical flow to update the map $\mathbf{S}_{t-1}^m$ such that it reflects the assumed position of body part $b_m^t$ in frame $t$. The propagated map $\mathbf{S}_t^m$ is computed so that

$$\mathbf{S}_t^m(i + \mathbf{F}_{t,x}(i,j),\ j + \mathbf{F}_{t,y}(i,j)) = \mathbf{S}_{t-1}^m(i,j). \tag{6.6}$$

A set of image processing steps are applied to the propagated map, including morphological operations, to remove noise and cavities caused by artifacts in the optical flow field.

#### 6.3.4.3 Removal of undesired graph edges

Using the updated and corrected map, we can remove the undesired edges in the graph $\mathcal{G}_t$ that connect points on body segment $b_m^t$ to the body part in the background, e.g. the torso. We update the set of edges as $E_t = E_t - F$, with

$$F = \{(\mathbf{x}_{ij}, \mathbf{x}_{kl}) \in E_t \mid \mathbf{S}_t^m(i,j) \neq \mathbf{S}_t^m(k,l)\}, \tag{6.7}$$

where $\mathbf{x}_{ij}$ is the 3D point corresponding to the location $(i, j)$ in the body segment map. In other words, all edges are removed where one point lies within the body segment map and the other point does not. Figure 6.3 e) illustrates the geodesic distances from the body center after the edges between the occluding arm and the torso have been disconnected. The corrected graph allows us to identify the primary and secondary anatomical landmarks on body segment $b_m^t$ by re-computing the geodesic distances from the body center and selecting points with a maximal distance, as described in Section 6.3.3.2.

In the situation that multiple primary anatomical landmarks cannot be detected, we repeat the process described above for every missing landmark, each time propagating the appropriate body segment map and disconnecting undesired graph edges. Note that the map propagation step, although based upon optical flow between subsequent frames, does not fail when there is no movement. In such cases, the optical flow field is close to zero and the body segment map simply remains unchanged.

### 6.3.5 Skeleton Fitting Using Inverse Kinematics

Once the anatomical landmarks $\mathcal{P}_t$ have been identified and labeled in frame $t$, we estimate the full-body pose parameters $\hat{\mathbf{y}}_t \in \mathbb{R}^{d_y}$ by fitting a skeleton to the detected points. Our skeleton model is described in Appendix A and has $d = 33$ degrees of freedom. It consists of $K = 22$ joints, denoted as $\{\mathbf{s}_i\}_{i=1}^K$, that are distributed over five kinematic chains. Starting with the torso chain that is registered in the body centroid $\mathbf{c}^t$, the full-body pose is determined, intuitively, by attracting selected joints of the kinematic chains (effectors) to the locations of the anatomical landmarks (targets). To match the $L = 11$ anatomical landmarks extracted from the depth images, we select the following joints from the $K$ body model joints as effectors: the hands, elbows, feet, knees, the head, the pelvis and one of the spine joints. Formally, our objective is to find the optimal joint angle configuration $\hat{\mathbf{y}}_t$ such that the residual error

$$\mathcal{E}(\mathbf{y}, t) = \sum_{i=1}^{L} \left\| \mathbf{p}_i^t - f_{\text{kin}}^{(i)}(\mathbf{y}) \right\| + c(\mathbf{y}) \tag{6.8}$$

is minimized. Here, $f_{\text{kin}}^{(i)} : \mathbb{R}^{d_y} \rightarrow \mathbb{R}^3$ is a forward kinematic function that computes the 3D position of the $i$-th joint, given a vector $\mathbf{y}$ of joint angles. We assume that the $i$-th joint is the effector corresponding to the $i$-th anatomical landmark. The term $c(\mathbf{y})$ penalizes joint angle configurations that violate a set of constraints (see Appendix B). This term increases polynomially when any of the joint angles approaches its pre-specified lower and upper limits.

To determine the optimal joint angle configuration $\hat{\mathbf{y}}_t$ at each time step $t$, we employ an iterative Gauss-Newton optimization approach that, starting with an initial value $\hat{\mathbf{y}}_t^0$, computes updates $\Delta_{\mathbf{y}}$ such that $\hat{\mathbf{y}}_t^{i+1} = \hat{\mathbf{y}}_t^i + \Delta_{\mathbf{y}}$, until convergence. In each frame, we use the joint angles of the previous frame as an initial value, $\hat{\mathbf{y}}_t^0 = \hat{\mathbf{y}}_{t-1}$. Assuming incremental body movement between subsequent frames, this increases convergence rates and decreases the probability of hitting local minima. In addition to the estimated joint angles $\hat{\mathbf{y}}_t$, the skeleton fitting step also gives us the corresponding 3D locations $\{\hat{\mathbf{s}}_i\}_{i=1}^K$ of all skeleton joints. Note that these locations will be close to the location of the anatomical landmarks, but there will not necessarily be a coincidence. This is a favorable effect, as it will limit the impact of outlier landmark detections.

### 6.3.6 Evaluation

We evaluated our body tracking method using a ToF camera and a Microsoft Kinect device (see Figure 1.4). The ToF camera is a PMDVision CamCube [121] with a resolution of $204 \times 204$ pixels for both, intensity and depth images. The Kinect [101] captures depth and RGB images at a resolution of $640 \times 480$ pixels. We performed the following two sets of experiments:

- **ToF data**: To assess our method on ToF data, we compared the skeleton predictions produced by our method to ground truth skeleton data obtained using a marker-based motion capture system (Appendix B). 20 testing sequences were recorded using the ToF-camera. Each of the sequences consists of around 400 frames, captured at a frequency of 10 Hz. The recorded movements range from simple motions, such as waving an arm, trough complex full-body movements with occlusions between body parts. Figure 6.7 gives an overview of the movements in our ToF-based evaluation set.
- **Kinect data**: To assess our method on Kinect data, we used the skeleton information generated by the NITE skeleton tracker from PrimeSense [124] as a reference. The Kinect-based data we used for evaluation consists of 10 testing sequences, each approximately of 600 frames in length, recorded at a frequency of 30 Hz. The movements are similar to those performed in the ToF dataset.

In our experiments, the depth and intensity images were pre-processed using a median filter to decrease the level of noise. We segmented the person from the background in each frame by subtracting a static depth image of the lab acquired beforehand. At each time step, we low-pass filtered each of the spatial optical flow field components. Our current Matlab implementation reaches tracking rates of 2-6 frames per second for ToF data. Performance on the Kinect data is slightly lower, given its higher resolution and thus higher complexity in the graph computations.

#### 6.3.6.1 Full-body Pose Estimation on ToF Data

For recording ground truth full-body poses, we used an optical marker-based motion capture system based on the ART Dtrack 2 tracking system [2] (Appendix B). The motion capture system was synchronized with the ToF camera and registered to its coordinate frame. Motion capture markers were placed on the back of the person to prevent interference with the depth measurements. The motion capture system provides the true 3D positions $\{\mathbf{s}_i\}_{i=1}^{K}$ of the $K = 22$ body joints of the skeleton model described in section 6.3.5. As an error metric, we computed in every frame the average Euclidean distance between the estimated and true locations of these body joints. We define the distance error for our ToF experiments as

$$e_{\text{tof}}(t) = \frac{1}{K} \sum_{i=1}^{K} \|\mathbf{s}_i - \hat{\mathbf{s}}_i\|, \tag{6.9}$$

where $\hat{\mathbf{s}}_i$ is the estimated 3D position of the $i$-th skeleton joint and $\mathbf{s}_i$ is the corresponding ground truth. Note that, even in the case of a perfectly estimated full-body pose, $e_{\text{tof}}(t)$ will not be zero, since the markers of the motion capture system do not coincide with our detected anatomical landmarks. Moreover, the motion capture system fits the skeleton to assumed locations of joints within the body, whereas our fitting targets are on the body surface.

Averaged over all testing sequences, our method achieved a distance error of $\bar{e}_{\text{tof}} = 70.1$ mm with a standard deviation of 9.8 mm. Figure 6.4 shows plots of the distance error over the length of two typical testing sequences. The overlaid full-body pose prediction and ground truth for selected frames allows for a better interpretability of the results. The left graph corresponds to one of the easier sequences where only the arms are moved, however, including body self-occlusions. In this case, the distance error averaged over all joints is around 50 mm. The maximum error in each frame rarely exceeds 100 mm. On the right side, results are shown for a more difficult sequence including full-body movement. Especially when legs are raised, the average error increases to around 100 mm. The effect of the maximal value of 250 mm for individual joints is visualized in example 7 (Figure 6.4), where the position of the right knee deviates from the ground truth. This being a worst-case example, our method compares favorably to current state-of-art methods for ToF-based full-body pose tracking (e.g. [49]).

#### 6.3.6.2 Full-body Pose Estimation on Kinect Data

The open-source NITE framework, maintained by PrimeSense [124], provides a skeleton tracking algorithm that we used as a reference for evaluating the performance of our method on Kinect data. For this purpose, we recorded the NITE skeleton predictions for each frame together with the corresponding depth and intensity images. In our ToF-based experiments (Section 6.3.6.1), we used the same body model for skeleton fitting and for the motion capture ground truth, leading to a one-to-one correspondence between estimated and ground truth joints. The skeleton model employed in the NITE framework, however, is structured differently and only consists of $\tilde{K} = 15$ joints. We denote the locations of these joints as $\{\tilde{\mathbf{s}}_i\}_{i=1}^{\tilde{K}}$. To measure the deviation of joint locations predicted by our method, we manually created an assignment of matching joints between the two body models. Let $\kappa(i)$ be the index of the NITE joint corresponding to the $i$-th joint of our model. We define the distance error for our Kinect experiments as

$$e_{\text{kinect}}(t) = \frac{1}{\tilde{K}} \sum_{i=1}^{\tilde{K}} \|\mathbf{s}_i - \tilde{\mathbf{s}}_{\kappa(i)}\|. \tag{6.10}$$

Averaged over all Kinect-based testing sequences, our method achieved a distance error of $\bar{e}_{\text{kinect}} = 108.4$ mm. This value is higher than in the ToF-based experiments, as there is no exact correspondence in body joints between the two used body models. We observed that a deviation of approximately 30 mm was present even when the predicted skeleton and the NITE skeleton visually were in the same pose. Figure 6.5 shows a plot of the distance error over a typical Kinect-based testing sequence involving full-body movements and hand occlusions. For a better understanding of the difference between the averaged and maximal error, Figure 6.6 provides the deviations for each of the $\tilde{K} = 15$ joints that were compared. Most notably, the deviations for body parts that are crucial for gesture-based interaction, such as the hands and elbows, are below 100 mm. There is a significantly higher error for both knees that is mainly due to a different location of the knee joint between both skeleton models. As can be seen in Figure 6.8, the NITE skeleton tracker tends to underestimate the lower leg length, placing the knee joint lower than in our model.

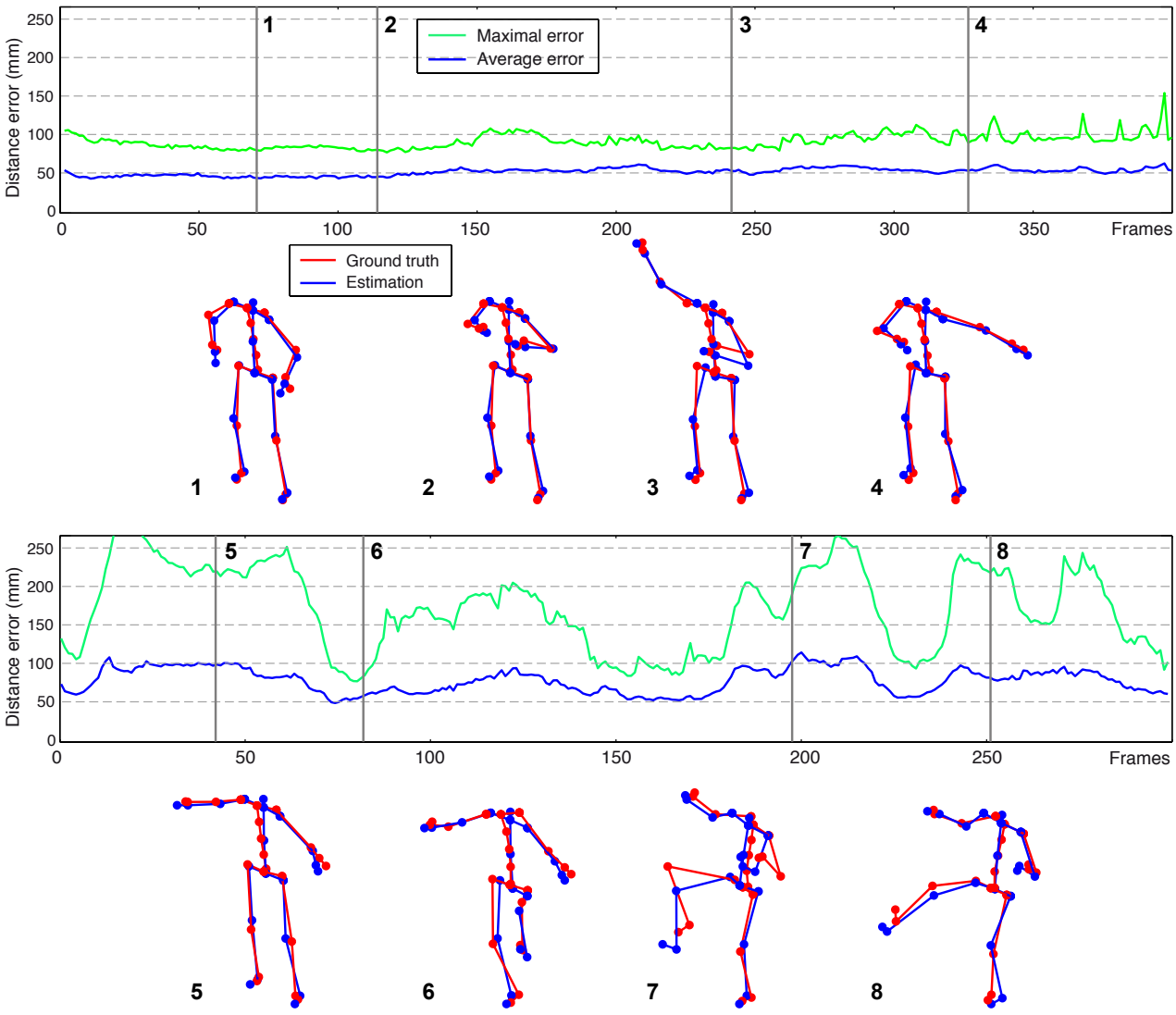



Figure 6.4: Illustration of quantitative pose estimation results for ToF depth images. The two graphs show the distance error $e_{\text{tof}}(t)$ over the length of two exemplary testing sequences. The average error over all joints is plotted (blue), along with the maximum error in each frame (green). *Left:* Typical sequence where only hands are moved, including self-occlusions. *Right:* Typical sequence involving full-body movement. Results for selected frames are visualized below with overlaid estimated (blue) and ground truth poses (red).

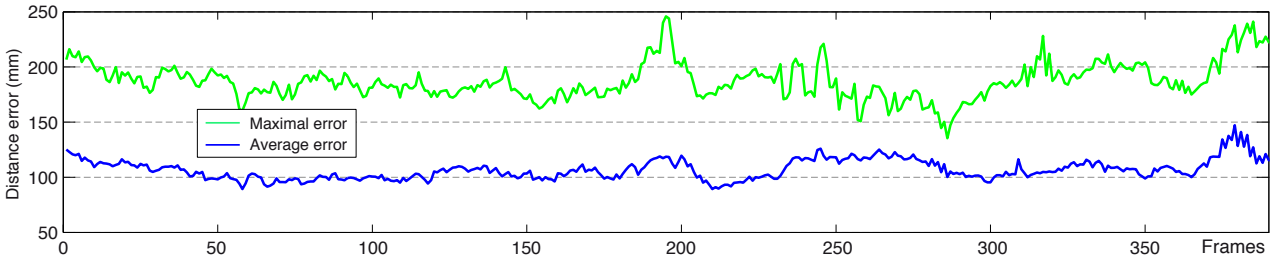



Figure 6.5: Quantitative pose estimation results on Kinect depth data. Distance error $e_{\text{kinect}}(t)$ over the length of one exemplary testing sequence involving arm and full-body movements. The average error over all joints is shown blue, the maximum error in each frame is green.

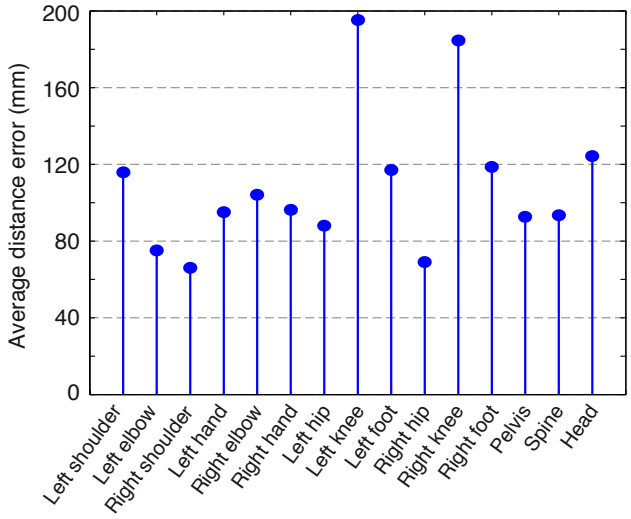



Figure 6.6: Quantitative pose estimation results on Kinect depth data. Distribution of distance error across the 15 joints used for comparison to the NITE skeleton model.

#### 6.3.6.3 Qualitative Assessment

Figure 6.7 provides example images from our ToF-based testing sequences for a qualitative assessment of our method. As can be seen, the estimated full-body poses match with the ToF depth images. Poses are predicted faithfully, even in cases where arms or legs move towards or away from the camera. In particular, the situation where a hand is stretched forward and occludes the arm itself is handled well. The second row of images in Figure 6.7 shows cases where one or both hands move in front of the torso. Here, our method relies on the optical flow-based disambiguation approach described in Section 6.3.4. Tracking does not fail, even when more than one limb moves in front of the body. The speed of movements is not a critical parameter to our technique, as long as the positions of primary anatomical landmarks can be matched successfully across subsequent frames.

Figure 6.8 illustrates the pose estimation results on our Kinect-based testing data. Each row shows four depth images overlaid with a projection of the detected anatomical landmarks and the fitted skeleton. Displayed below are 3D views of each pose, where the output of the NITE skeleton tracker is shown for direct comparison. While the poses appear almost identical,

it is worth noting that the poses predicted by our approach tend to look more anatomically plausible. The reason here is the underlying human body model with kinematic constraints and fixed, person-specific limb lengths that is fitted to anatomical landmarks. In contrast, the NITE tracker detects interest points (i.e. joints) and connects them to obtain a skeleton. In our experiments, both methods failed to correctly track the person when turning out of the frontal pose towards the image plane for more than 60 degrees. The NITE tracker can cope with the challenging situations of limb self-occlusions that our method solves by means of optical flow. However, in such situations, the NITE skeleton becomes unstable and joint positions start jittering. Our approach faced problems when two arms or legs crossed each other in front of the torso. In such cases, parts of one limb were cut off by the occluding limb, resulting in inaccurate landmark detections.

## 6.4 Discussion

In this chapter, we have presented a method for tracking human full-body poses from sequences of ToF camera or Kinect depth images. The approach does not require any training data and is able to track arbitrary movements for gesture-based human-computer interaction. Our method takes full advantage of the data provided by typical depth cameras by utilizing both, depth and intensity information. Based on the depth data, we segment the person in front of static background and construct a graph-based representation of the 3D points. This graph allows us to robustly identify anatomical landmarks in each frame by selecting points with a maximal geodesic distance from the body center of mass. In cases where body parts occlude each other, we rely on optical flow, computed on the intensity images, to disconnect occluding limbs from the body part behind. We conclude the chapter with a brief discussion of a few key insights.

**Inherent Assumptions.** While our method builds upon an initialization phase, the user is only required to hold a T-pose for determining the approximate limb lengths and localizing the initial anatomical landmark locations. A similar calibration pose has to be taken by users of the NITE skeleton tracking method. Another assumptions of our method is that persons are facing the camera and do not fully rotate around their vertical axis. We argue that this assumption is reasonable for many application scenarios of gesture-based human-machine interaction. We also assume that the person is always visible from head to toe in the depth images. In other cases, the body part detection and propagation steps would fail. However, short phases of occlusions can be handled, as long as body parts re-appear at similar locations as before the occlusion. Also, the presented method can easily be adapted to consider only upper-body movements, if necessary.

**Comparison of ToF and Kinect-based Tracking.** Using ToF and Kinect data in parallel and running our algorithm on both types of depth information has led us to several observations. The higher resolution of Kinect depth images leads to a higher stability in detected landmark locations, as compared to the ToF data. This increase in resolution, however, also negatively affects runtime performance of our algorithm that depends on graph construction and on the Dijkstra algorithm [28]. A positive aspect of the Kinect data is that regions with

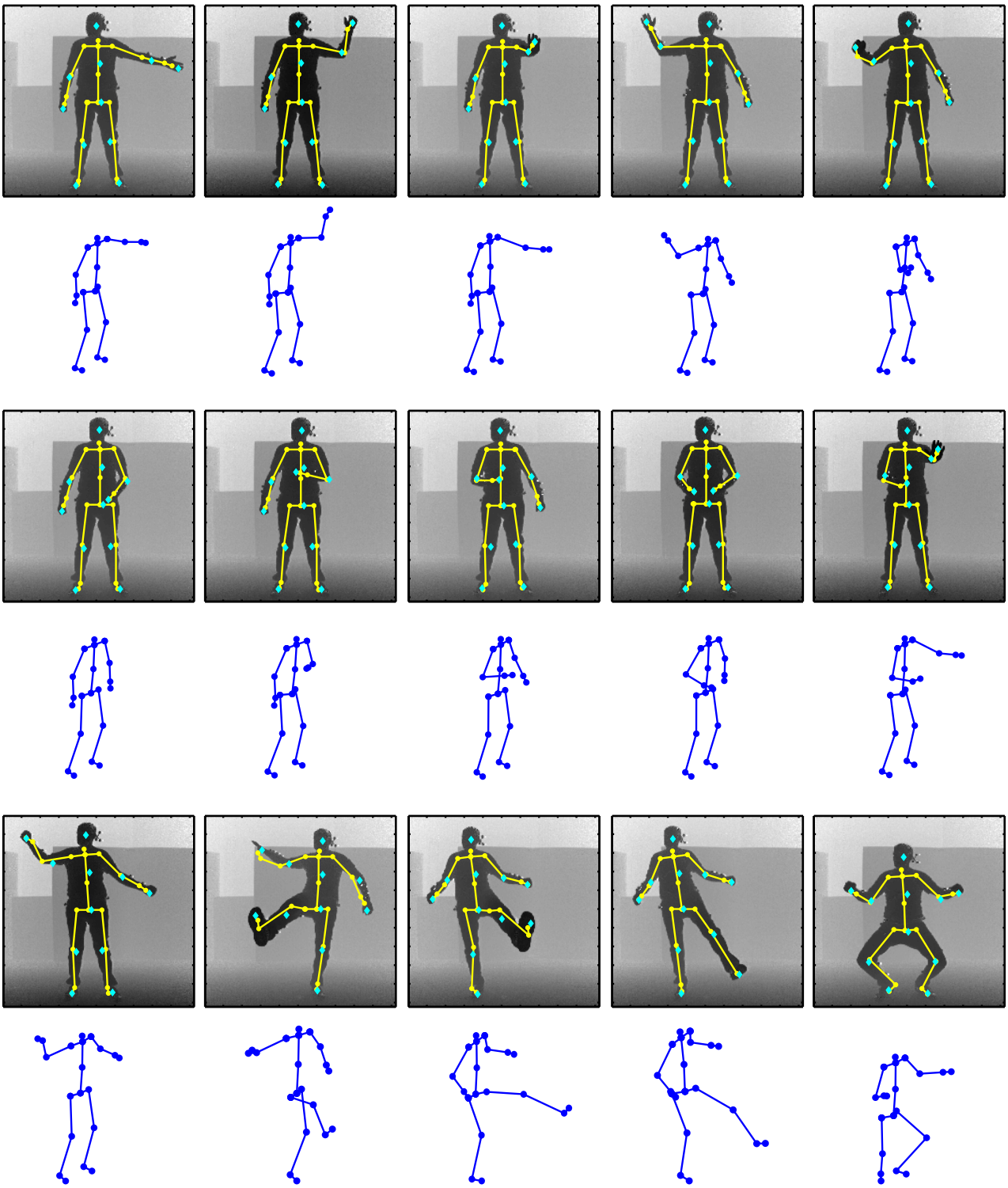


Figure 6.7: Pose estimation results on ToF depth images. In each of the three rows, depth images are overlaid with projections of the estimated skeleton pose (yellow). Blue markers indicate the positions of detected anatomical landmarks that play the role of targets for skeleton fitting. Below each row, perspective views of the corresponding estimated poses are displayed, emphasizing the 3D appearance of the predictions.

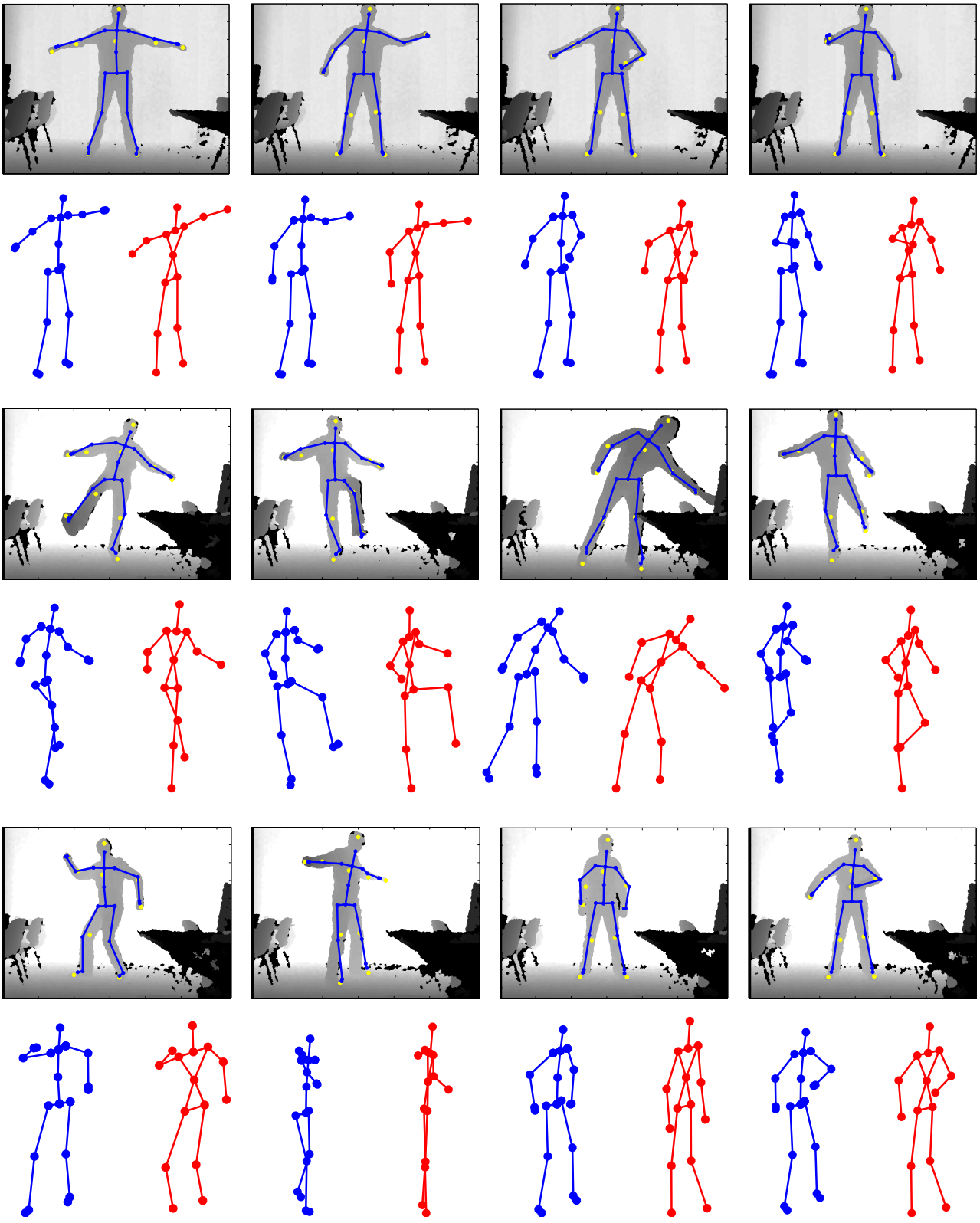


Figure 6.8: Pose estimation results on Kinect depth images. Depth images are overlaid with the detected anatomical landmarks (yellow markers) and the fitted skeleton (blue). Below each image is a 3D view of the predicted skeleton (blue) and of the corresponding NITE skeleton prediction (red).

invalid measurements result in values that are clearly separable from valid depth measurements. Noise in ToF data often appears indistinguishable from regular depth measurements, requiring to make heavier use of filtering.

**Comparison to Machine Learning Approaches.** This chapter was an excursus from the main theme of this work, machine learning methods for human motion analysis. The excursus was motivated by the interesting properties of depth data generated by ToF cameras or the Kinect that give the impression of enabling human pose tracking without the need for machine learning. While the 2.5D data does provide more informative observations than, for instance, a monocular video stream, identifying anatomical landmarks and fitting a skeleton body model remains a non-trivial task. Compared to machine learning methods, such as ours evaluated in Section 5.5, the direct approach is more sensitive to noise and occlusions, as there is no prior information on feasible human poses that can complement missing input data or help in the case of outlier observations. The obvious advantage of a learning-less method is that arbitrary poses can be tracked, and not just the poses corresponding to a set of learned activities. However, recent methods using a large amount of training data [146, 51] show that machine learning techniques can even be suitable for person-independent tracking on various activities. In fact, learning-based methods have the significant advantage of moving computational complexity away from the online application phase to an offline training step. Our methods presented in Chapters 4 and 5 easily achieve real-time performance after training, which is not the case for the learning-less approach described in this chapter.

**Future Work.** While the current implementation based on Matlab does not provide real-time frame rates (we currently achieve 2-6 frames per second), we believe there is the potential for improving computational efficiency, without requiring GPU acceleration. Future work should focus on improving the labeling process for detected anatomical landmarks that currently depends on matching to the locations of anatomical landmarks in previous frames.

# Part III

# Gesture Recognition

Chapter 7

# Gesture-based Interaction in the Operating Room

## 7.1 Introduction

In the previous chapters, we have presented and discussed various approaches for human activity recognition and full-body pose estimation from limited observations, such as inertial sensors or depth camera images. We now turn to a particular application scenario, the clinical operating room (OR), where our focus is on recognizing and tracking human gestures. The objective is to let a surgeon interact with computer-based medical systems without the need to touch a keyboard or a mouse. Our input modality of choice are wireless inertial sensors, as these sensors provide a high movement flexibility inside the operating room, compared to cameras. In this chapter, we make use of several concepts and methods introduced in the previous chapters, but adapted to the gesture recognition scenario.

The OR is a highly complex environment and surgical staff often work under high pressure. For their assistance, various types of computerized medical systems are used in the OR, for instance to visualize volumetric patient data or to monitor vital parameters. However, when interacting with such devices during surgery, surgeons face challenges. Sterility regulations restrict the use of classical mouse-driven or touch-based user interfaces and control terminals are often required to be spatially separated from the main operating site [77]. Also, interventional workflow might not allow the surgeon to leave this site and, while handling medical instrumentation, the surgeon might only have very limited freedom of movement. Typical solutions for this situation include delegating physical control over computerized systems to less-skilled assistants [52, 77]. This, however, increases the amount of verbal communication and raises the chance of misunderstandings. The added level of indirection can also adversely affect the precision of the surgeon and lead to inefficiency [77].

In this chapter, we present a solution for a touch-free user interface that gives surgeons direct control over computerized systems by means of gestures. Our approach learns gestures from samples given by a surgeon in a training phase. The surgeon is not required to adhere to a set of pre-defined gestures but can define gestures that are most suitable in a particular surgical workflow. Gestures are captured using a few wireless inertial sensors on the surgeon's arms that can easily be integrated into garment (Figure 7.1). The sensors make our system independent

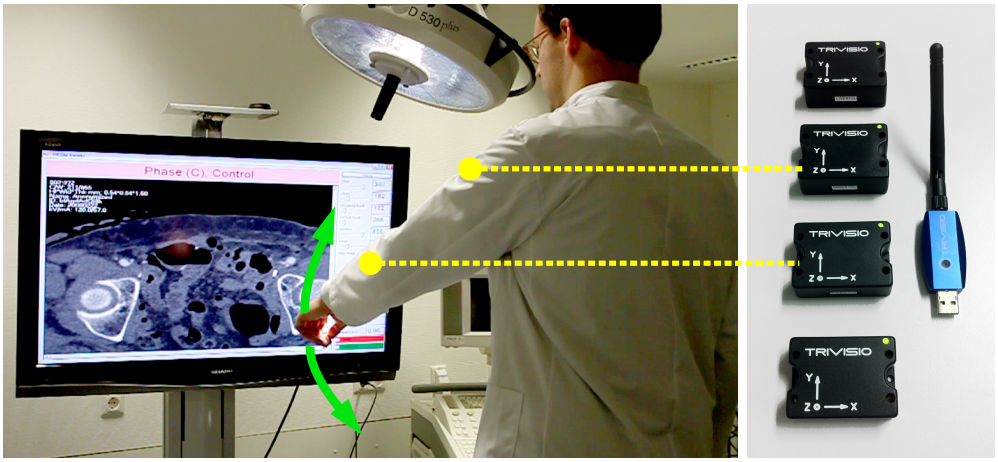



Figure 7.1: Gesture-based control of a medical image viewer application. Four wireless inertial sensors (*right*) are placed on the person's arms underneath the blouse. In this example, a vertical arm gesture is used to zoom the visualization of a medical image viewer application.

of cameras, enabling robust gesture recognition in the presence of critical OR conditions, including difficult lighting, cluttered backgrounds and occlusions by staff and equipment.

Our method uses manifold learning to capture the underlying structure of the movements for each trained gesture. The learned manifold models provide a low-dimensional search space for gesture recognition and tracking. As opposed to the pose estimation methods presented in previous chapters, we do not make use of full-body pose data in the training phase, but only rely on the inertial sensors. We argue that requiring a motion capture system to obtain the full-body pose data, even if it is only in a training phase, is a significant burden for surgeons. In fact, we do not require knowledge of the exact full-body poses for gesture recognition, as we are mainly interested in classifying and tracking gestures from inertial sensor data.

While recognizing gestures, the learned manifold models enable us not only to determine which gesture is performed, but also to infer the particular pose within one gesture. Existing gesture-based user interfaces give users either *categorical* or *spatio-temporal* control [57, 84]. Our method combines both approaches. As a result, surgeons can select discrete functions of the target system, since their gestures are automatically recognized (categorical control). At the same time, their particular arm movements in space are used to precisely adjust continuous parameters related to the selected functions (spatio-temporal control), see Figure 7.2.

Consider the following illustrative application scenario. During minimally-invasive aortic repair procedures, a stent graft is inserted through an artery into the aorta to exclude an aneurysm from blood circulation. Positioning the stent graft is guided by imaging modalities that combine intra-operative X-ray views and tomographic patient scans [36]. For the complex task of inserting the stent graft, a hypothesized surgeon would like to control the visualization himself, instead of giving instructions to an assistant. Knowing that his right hand is occupied during catheter insertion, the surgeon trains the gesture recognition system on three gestures he performs with his left arm. He uses a circular movement for browsing slices in the tomographic scan, a vertical movement for zooming and a horizontal movement for translating the image. During an actual intervention in the OR, where lights are dimmed, the surgeon needs to adjust the visualization while inserting the stent graft. To activate gesture recognition, he

pronounces a voice command and starts tracing a circle with his left arm. The system immediately recognizes the intended browsing functionality. Having found the region of interest, the surgeon keeps his arm still and moves it back and forth for fine adjustment. To increase the zoom factor, he then moves his arm horizontally. The image is scaled in proportion to his arm movement. Another voice command stores the current setting and deactivates gesture control. The surgeon can now continue inserting the stent graft without accidentally modifying the visualization with his movements.

In our experiments, we evaluate our gesture-based interaction approach in the scenario of controlling a medical image viewer application. Quantitative experiments show that the proposed method can simultaneously recognize up to 18 gestures to a high accuracy from as little as four inertial sensors. We also describe the outcome of a qualitative user study that encourages a practical application of our method in the operating room.

## 7.2 Related Work

Using gestures for human-machine interaction has been studied intensively and surveys are available, e.g. in [73, 102]. Gesture-based human-machine interaction has recently received particular attention in the context of consumer devices or entertainment applications [96, 146, 132]. Throughout the literature, different modalities are used to capture gestures, such as pens [176] or sensors, but cameras are the most frequent choice. Several authors have even targeted the operating room scenario, e.g. [85, 52, 149, 171]. For instance, Graetzel et al. [52] use a stereo camera setup for tracking the hand of a surgeon and thereby controlling the mouse pointer on a screen. Kipshagen et al. [85] present a medical image viewer that can be controlled by means of hand gestures that are tracked with two video cameras. Soutschek et al. [149] employ a ToF camera for interaction with the display of an intra-operative imaging device. Guerin et al. [54] describe a gesture-based system that allows controlling a surgical robot. The aforementioned approaches are all vision-based and thus face challenges in realistic OR environments, where optimal lighting and line-of-sight between the surgeon and the cameras cannot always be ensured. We address this issue by capturing gestures with a few body-worn inertial sensors instead of cameras.

Inertial sensors attached to the surgeon's body are a promising alternative that allow circumventing the restrictions of camera-based systems. In addition, wireless sensors enable a surgeon to perform gestures independent of his or her location and orientation within the OR. Previous research on gesture recognition using wearable inertial sensors includes [84, 59, 163, 44]. To our knowledge, inertial sensor-based gesture recognition for the operating room has not been proposed before. Kela et al. [84] measure short hand gestures using accelerometers and evaluate the usefulness of these devices in controlling home entertainment systems. The authors present a gesture recognition approach based on a Hidden Markov Model (HMM). Hand gestures, such as quick up-down movements or circles, are recognized in [59] based on a Dynamic Time Warping (DTW) mechanism. The authors of [163] and [44] recognize the suitability of *orientation* sensors for gesture recognition, as opposed to pure accelerometers. We also employ this particular type of devices.

A similar concept to our combination of *categorical* and *spatio-temporal* control has surfaced before, however, in the context of 2D interaction [123, 55]. Pook et al. [123] present a specific type of menus for graphical user interfaces that allow users to select commands

and adjusting their parameters in one mouse sweep. Parameter tuning is realized by means of scroll bars that are integrated into the individual menu items. Guimbretière et al. [55] discuss an extended approach for interaction in pen-based environments. In a single gesture with the pen, a user can select features, adjust parameters and enter hand-written text. Both techniques do not allow for a user-dependent configuration of gestures and the interaction depends on classical, mouse-based 2D interfaces. The method of Wachs et al. [171] aims at 3D gesture-based interaction in the OR and can process gestures for binary commands and gestures for spatio-temporal control. However, as opposed to our approach, both types of interaction cannot be combined simultaneously.

The contributions of our proposed system for gesture-based interaction can be summarized in three propositions: (1) Gestures can be defined freely according to the requirements of a particular interventional scenario. By demonstrating an example per gesture to the system, surgeons do not have to adhere to a pre-defined set of movements that might interfere with their hand usage during surgery. (2) Gestures are not only recognized for triggering discrete actions, e.g. "click the mouse", but simultaneously allow tracking the movements within a gesture in 3D for fine-tuning continuous parameters, such as the size of a displayed image. (3) Our gesture recognition method creates dimensionality-reduced gesture models from training sensor data and directly predicts the current gesture and the relative pose within the gesture from previously unseen sensor measurements.

## 7.3 Gesture Recognition Based on Manifold Learning

The gesture-based user interaction method introduced in this chapter is based on the generative method for human pose estimation and activity recognition presented in Chapter 5. We assume the reader is familiar with the former method and focus on particular adaptations to the gesture recognition scenario. Overall, the method consists of a training phase, where gesture models are learned from example sensor data for each considered gesture, and a testing phase, where the models are used to recognize gestures from previously unseen sensor data.

### 7.3.1 Method Overview

In our understanding, a gesture is at the same time categorical and spatio-temporal [57]. A user performs a gesture to invoke a particular function of the target system and to control this function in a fine-grained, continuous manner. Our gesture recognition method therefore has two goals: (1) to recognize which gesture a user is performing at each time instant and (2) to identify the relative spatial pose within that particular gesture. To this end, we define a gesture to be an arbitrary movement including one or both arms that has a temporal extent and encompasses many smoothly-varying poses. In the example shown in Figure 7.2, a user chooses a horizontal movement of both arms for browsing slices in a medical volumetric dataset. As soon as he starts raising his hands, the intended gesture (*browse*) is recognized and the browsing function is activated. His exact hand movement is now translated to a relative pose value between 0 and 1. This value is then mapped to the value range of any parameter he desires to adjust. In the example, the parameter controlled by the user is the slice number between 1 and 256 of a medical image viewer application.

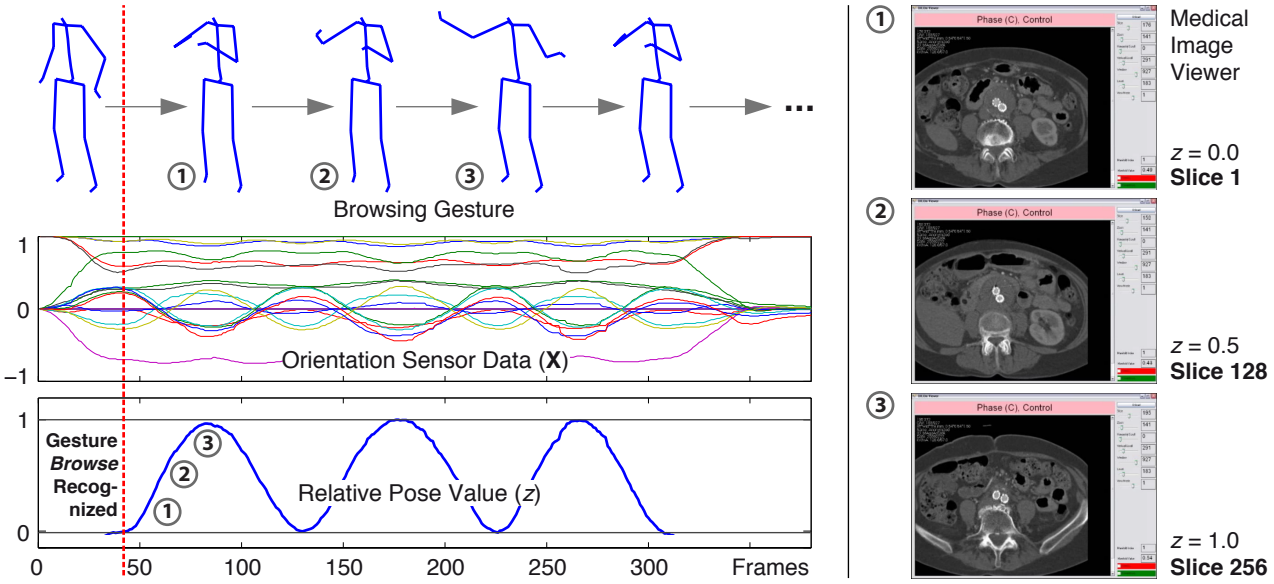



Figure 7.2: Spatio-temporal gesture control of a browsing function in a medical image viewer. Selected body poses (*top*), the corresponding stream of orientation sensor data (*middle*) and the resulting relative pose value between 0 and 1 (*bottom*) are shown. Shortly before ①, the gesture is recognized as browsing. Thereafter and through ② and ③, the relative pose value is computed and mapped to the range of available slices in the medical dataset (*right*).

### 7.3.2 Learning Gesture Manifolds

In the training phase, we learn prior models of gestures from sample sensor data. While our generative method for human pose estimation and activity recognition (Chapter 5) builds motion models from full-body pose training data, we rely only on the sensor information for gesture recognition. This way, we eliminate the need to use a motion capture system in the training phase, while still being able to capture movement characteristics for multiple gestures. As discussed below, the exact pose of the person is not of interest to us in the gesture-recognition scenario. Instead, we derive an abstract, relative pose representation that is sufficient for adjustment of user interface parameters.

Let $M$ be the number of considered gestures and let $\mathcal{X}^\alpha = \{\mathbf{x}_1^\alpha, \ldots, \mathbf{x}_{N_\alpha}^\alpha\}$, $\alpha \in \{1, \ldots, M\}$, be a training dataset of labeled sensor measurements for each gesture. As before, we write the dataset in matrix form as $\mathbf{X}^\alpha = [\mathbf{x}_1^\alpha, \ldots, \mathbf{x}_{N_\alpha}^\alpha]^\top$. Here, $N_\alpha$ is the number of available sensor vectors for gesture $\alpha$. For each gesture, the two sensor vectors $\mathbf{x}_{\min}^\alpha$ and $\mathbf{x}_{\max}^\alpha$ are assumed to be known, e.g. by manual labeling in the training phase. These vectors represent poses at the beginning and end of the actual control part of a gesture and let us separate introductory movements, such as raising an arm. To obtain a compact parameterization of feasible sensor values for each gesture $\alpha$, we use Laplacian Eigenmaps [14] and map the training data $\mathcal{X}^\alpha$ to a low-dimensional representation $\mathcal{Z}^\alpha = \{\mathbf{z}_1^\alpha, \ldots, \mathbf{z}_{N_\alpha}^\alpha\}$, such that $\mathbf{z}_i^\alpha \in \mathbb{R}^{d_z}$ and $d_z \ll d_x$. Here, we select the $k$-nearest neighbors approach for setting up the neighborhood graph and rely on Euclidean distances. This straightforward choice of similarity measure is reasonable as the input data $\mathcal{X}^\alpha$ consisting of sensor measurements itself has few dimensions and does not exhibit any properties that necessitate a more sophisticated metric.

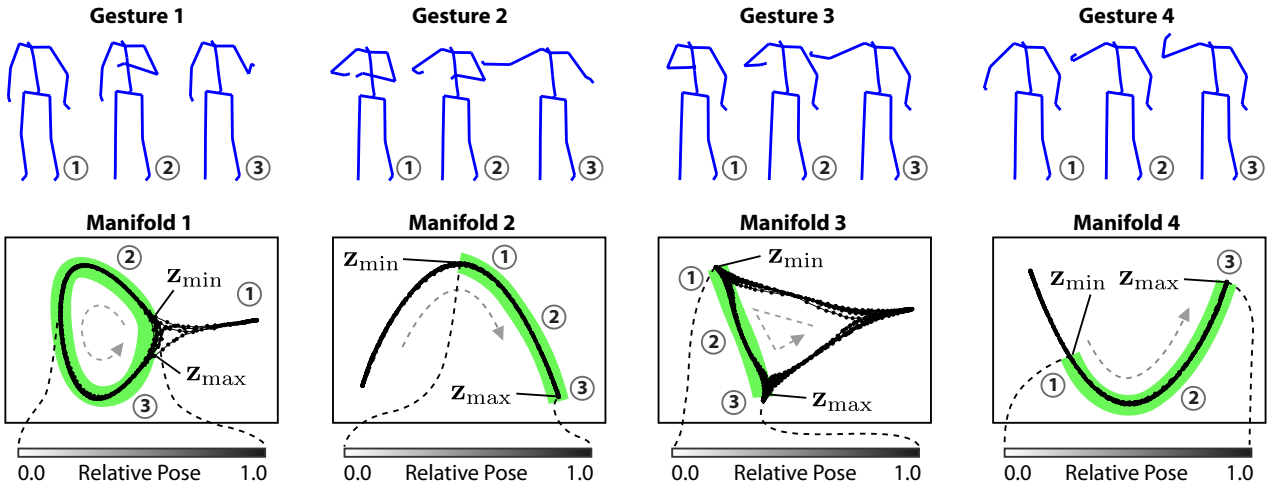



Figure 7.3: a) The proposed method recognizes multiple user-defined gestures (*top*) and tracks the relative pose within a gesture by means of learned gesture manifolds (*middle*). The region on each manifold used for actual control (green) is within $[\mathbf{z}_{\min}, \mathbf{z}_{\max}]$. The relative gesture pose, given by a value in the interval $[0, 1]$, is computed from the relative location of a pose on a manifold embedding and is used for smooth parameter adjustment (*bottom*).

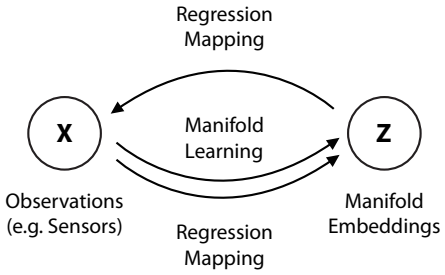



Figure 7.4: Diagram illustrating the relationship between observations ($\mathcal{X}$), manifold embeddings ($\mathcal{Z}$) and the learned regression mappings. Compare also to Figures 5.4 and 4.2.

Figure 7.3 shows exemplary two-dimensional manifold embeddings for four gestures. Note that these embeddings, generated from inertial sensor data, are similar in nature to the embeddings generated from full-body pose data (compare e.g. to Figure 5.3). The crucial property of the manifold embeddings is that the local spatial distribution of vectors in the original representation is preserved. In particular, similar sensor measurements will map to close-by embedding points, even if they occur at different times within a gesture.

We relate the space of sensor measurements and the low-dimensional manifold embeddings using regression mappings according to the Nadaraya-Watson estimator (Section 3.1.1). The mappings allow predicting sensor vectors $\hat{\mathbf{x}}$ from given embedding points $\mathbf{z}$ (reconstruction mapping) and projecting new sensor values $\mathbf{x}$ to points $\hat{\mathbf{z}}$ in embedding space (out-of-sample mapping). Note that in our pose estimation and activity recognition method, we did not define an out-of-sample mapping, but a reconstruction mapping and a projection from manifold embedding space to full-body pose space. The latter mapping is not required in the method presented here, as we do not have any full-body pose training data. Figure 7.4 illustrates

the relationships between the regression mappings, the sensor observations and the manifold embeddings. In analogy to Equation 5.2, we define a reconstruction mapping for a gesture with index $\alpha$ as

$$f^{\alpha}_{z\to x}(\mathbf{z}) = \sum_{i=1}^{N_\alpha} \frac{k_z(\mathbf{z}, \mathbf{z}^{\alpha}_i)}{\sum_{j=1}^{N_\alpha} k_z(\mathbf{z}, \mathbf{z}^{\alpha}_j)} \mathbf{x}^{\alpha}_i. \tag{7.1}$$

A sensor vector can be predicted from any location $\mathbf{z}$ in manifold embedding space as $\hat{\mathbf{x}} = f^{\alpha}_{z\to x}(\mathbf{z})$. The mapping is a weighted average of all training sensor vectors $\mathbf{x}^{\alpha}_i \in \mathcal{X}^{\alpha}$, with the largest weights attributed to sensor vectors that project to a manifold embedding location $\mathbf{z}^{\alpha}_i$ that is close to $\mathbf{z}$. Here, we express distance in terms of a Gaussian kernel $k_z(\mathbf{z}_i, \mathbf{z}_j) \propto \exp(-\frac{1}{2}\|(\mathbf{z}_i - \mathbf{z}_j)/\sigma_z\|^2)$ with a width $\sigma_z$ derived from the standard deviation of the learned manifold embedding points. By interchanging the roles of sensor values and manifold embedding points, we obtain the out-of-sample mapping for gesture $\alpha$,

$$f^{\alpha}_{x\to z}(\mathbf{x}) = \sum_{i=1}^{N_\alpha} \frac{k_x(\mathbf{x}, \mathbf{x}^{\alpha}_i)}{\sum_{j=1}^{N_\alpha} k_x(\mathbf{x}, \mathbf{x}^{\alpha}_j)} \mathbf{z}^{\alpha}_i. \tag{7.2}$$

This mapping allows predicting the location $\hat{\mathbf{z}} = f^{\alpha}_{x\to z}(\mathbf{x})$ in the manifold embedding space of gesture $\alpha$ from a given sensor vector $\mathbf{x}$. The Gaussian kernel $k_x(\cdot,\cdot)$ is defined in analogy to $k_z(\cdot,\cdot)$, with a width $\sigma_x$ derived from the standard deviation of the sensor vectors in the training data. We describe below how we make use of this out-of-sample mapping within our particle filter-based generative tracking approach.

### 7.3.3 Recognizing and Tracking Gestures

In the testing phase, our aim is to estimate the manifold index $\hat{\alpha}_t$ and location $\hat{\mathbf{z}}_t$ that best explain the sensor measurements $\mathbf{x}_t$ at any time $t$. In addition, we wish to compute the relative pose value $\hat{z}_t \in [0, 1]$ that serves to control arbitrary parameters of a target system.

Analogous to our method described in Chapter 5, we employ a particle filter [68] for this purpose that continuously explores the gesture-specific manifold embeddings. Every particle $\mathbf{p}^i_t = (\alpha^i_t, \mathbf{z}^i_t)$, $1 \le i \le n$, represents one gesture pose hypothesis. Initially, all $n$ particles are randomly distributed across the manifold embeddings. In every iteration of the particle filter, we let the particles propagate through the manifold embeddings, ensuring that only positions close to the learned embedding points are sampled (see Figure 7.5). With a certain probability, particles are allowed to switch between different manifold embeddings. We model this probability to be high in embedding space regions that correspond to an idle pose separating gestures. The prior probability and the switching probability are defined in analogy to Equations 5.5 and 5.6, respectively. To evaluate the fitness of the $i$-th particle, we define the observation likelihood

$$p(\mathbf{x}_t|\mathbf{z}^i_t, \alpha^i_t) = k_x(\mathbf{x}_t, f^{\alpha^i_t}_{z\to x}(\mathbf{z}^i_t)) \;\cdot\; k_z(\mathbf{z}^i_t, f^{\alpha^i_t}_{x\to z}(\mathbf{x}_t)) \tag{7.3}$$

Note that this expression simultaneously employs the regression mappings defined in Equations 7.1 and 7.2. In the first term of the product, a sensor vector is predicted from the particle's current position $\mathbf{z}^i_t$ and is compared to the true measurement $\mathbf{x}_t$. The Gaussian kernel $k_x(\cdot,\cdot)$ is has a maximal value when the two quantities are most similar. In the second term, $\mathbf{x}_t$ is

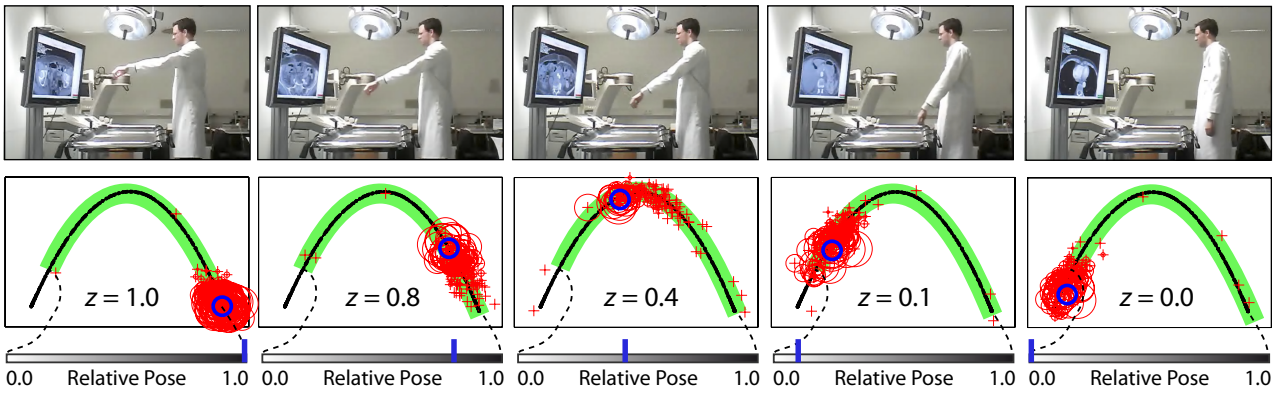



Figure 7.5: *Top row*: Sample images of a test subject performing one of the learned gestures. *Bottom row*: Manifold embedding for the same gesture with distribution of particles, shown in red, for each of the above images. The point $\hat{\mathbf{z}}_t$ is given by a dark circle.

directly mapped to embedding space and the projection is compared to the particle's location $\mathbf{z}_t^i$. Particles that score best in both directions are favored. By combining the contributions of both, the out-of-sample mapping $f^{\alpha}_{x\to z}(\mathbf{x})$ and the reconstruction mapping $f^{\alpha}_{z\to x}(\mathbf{z})$, we reduce the ambiguity that is inherent to each mapping alone and increase the chance to give high weight to particles that are most appropriate, given the sensor observation.

The final estimated gesture index $\hat{\alpha}_t$ is selected as the most frequent index $\alpha_t^i$ among the best-scoring particles. Among these particles, those with $\alpha_t^i = \hat{\alpha}_t$ are used to compute the final position $\hat{\mathbf{z}}_t$ in embedding space. In a last step, we translate this position to a relative pose value $\hat{z}_t \in [0, 1]$. To this end, we transfer the Cartesian coordinate $\hat{\mathbf{z}}_t$ into a polar representation $(r_t, \theta_t)$, such that the pole is at the centroid of the manifold embedding $\mathcal{Z}^{\hat{\alpha}_t}$, and keep the angular component $\theta_t$. Since the angular representation $[\theta^c_{\min}, \theta^c_{\max}]$ of the points $[\mathbf{z}^{\alpha}_{\min}, \mathbf{z}^{\alpha}_{\max}]$ labeled in the training phase is known, we can compute the desired relative pose value as

$$\hat{z}_t = \begin{cases} (\theta_t - \theta^{\hat{\alpha}_t}_{\min})/(\theta^{\hat{\alpha}_t}_{\max} - \theta^{\hat{\alpha}_t}_{\min}) & \text{if} \quad \theta^{\hat{\alpha}_t}_{\min} \le \theta_t \le \theta^{\hat{\alpha}_t}_{\max}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.4}$$

Illustrations of the pose value $\hat{z}_t$ in practice can be found in Figures 7.2 and 7.3.

#### 7.3.3.1 Temporal Gesture Segmentation

Having estimated $\hat{\alpha}_t$, $\hat{\mathbf{z}}_t$ and $\hat{z}_t$, we can determine the likelihood that the current pose is in fact a gesture and not another movement that might be similar. This process is often referred to as gesture *segmentation*. Note that the method will always generate an estimate for the gesture index, the manifold embedding position and the relative pose value, even if the performed movement is totally unrelated to the training data. Detecting such unknown (or non-gesture) movements is analogous to detecting anomalies, as discussed in Section 5.3.5 for our activity recognition and pose estimation method. Consequently, we declare the current pose to be part of an unknown (or non-gesture) movement, when the predictive confidence

$$c_t = \log(k_x(\mathbf{x}_t, f^{\hat{\alpha}_t}_{z\to x}(\hat{\mathbf{z}}_t))) \tag{7.5}$$

falls below a preset threshold. This measure evaluates how closely the estimated state $\hat{\mathbf{z}}_t$, projected to sensor space using the predictive regression mapping, matches the true sensor

| | Gesture |
|---|---|
| 1. | Left arm stretch and retract left of body |
| 2. | Right arm stretch and retract right of body |
| 3. | Left arm forward and backward in front of the body |
| 4. | Right arm forward and backward in front of the body |
| 5. | Left hand left and right in front of the body |
| 6. | Right hand left and right in front of the body |
| 7. | Both hands outwards and back in front of the body |
| 8. | Left hand up and down in front of body |
| 9. | Right hand up and down in front of the body |
| 10. | Both hands up and down in front of the body |
| 11. | Left hand left and right above the head |
| 12. | Right hand left and right above the head |
| 13. | Left hand front and back above the head |
| 14. | Right hand front and back above the head |
| 15. | Left arm circular movement in front of body |
| 16. | Right arm circular movement in front of body |
| 17. | Both hands parallel left and right in front of body |
| 18. | Both hands parallel up and down in front of body |

Table 7.1: Short description of gestures used in the full gesture dataset for evaluation.

observation $\mathbf{x}_t$. During a non-gesture movement, the gesture prediction will likely be incorrect and can be disregarded. Note that the non-gesture movement detection based on Equation 7.5 allows the user to exit gesture recognition at any time by performing movements that were not shown to the system in the training phase.

### 7.3.4 Evaluation

To evaluate our gesture-based interaction method for the OR, we conducted qualitative and quantitative experiments. We used two to six Colibri wireless orientation sensors [161] attached to the arms of our testing persons. The inertial sensor and motion capture dataset described in Chapter 4 was not used in this evaluation, as the movements contained in the dataset are not specifically targeting the gesture recognition scenario. Instead, we defined and recorded gestures for evaluation that included moving an arm horizontally or vertically, tracing out circles, or other movements with one or both arms. The full set of gestures is listed in Table 7.1. Our implementation with Matlab and C++ components runs in realtime and the frame rate is only bounded by the update frequency of the inertial sensors.

#### 7.3.4.1 Gesture Recognition

The gesture recognition accuracy of our method was evaluated systematically on a dataset of 18 different gestures, each recorded four times. We created labeled sequences of multiple gestures in a row, each between 1000 and 5000 frames. Experiments were performed in a cross-validation manner, each time using one of the sequences for training and one of the others

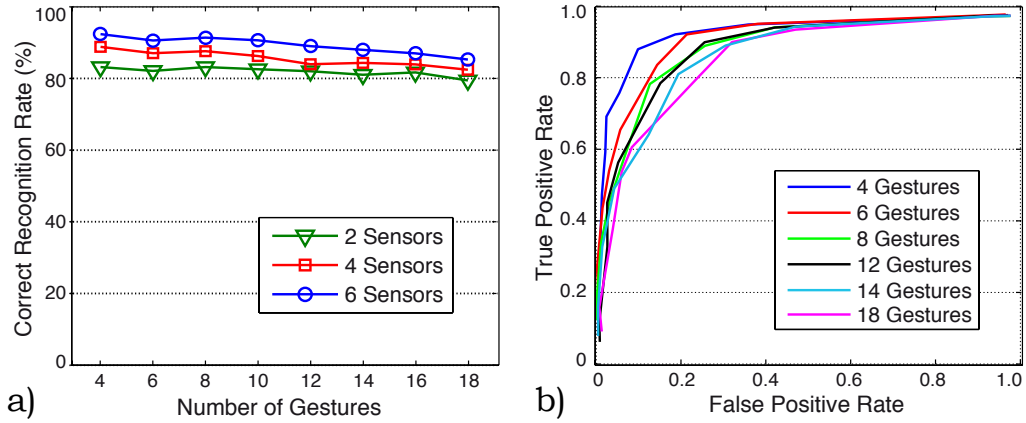



Figure 7.6: a) Correct gesture classification rates for different numbers of inertial sensors and an increasing total number of learned gestures. b) ROC curves for automatic differentiation of learned gestures from non-gesture movements.

for testing. While measuring the percentage of frames with correctly recognized gestures, we varied the number of simultaneously trained gestures and the number of inertial sensors. As shown in Figure 7.6 a), best gesture recognition rates were achieved with six inertial sensors. In this case, 95% of all frames were correctly recognized with a set of four gestures, and 88% when the method was trained on 18 gestures. Although the recognition rates decrease for less than six sensors, even only two sensors (one on each arm) yield correct recognition rates above 80% for all considered numbers of gestures. Our method thus scales well with respect to the number of gestures to be recognized simultaneously. In fact, we expect that less than 10 gestures need to be distinguishable in practical applications.

#### 7.3.4.2 Gesture Segmentation

It is crucial for a gesture-based interaction method to distinguish instances of learned gestures from arbitrary other movements. Using six inertial sensors, we varied the threshold associated with the confidence measure $c_t$ and measured the percentage of frames with gesture movements recognized as such (true positive rate, $r_{\mathrm{TP}}$) and the percentage of non-gesture frames wrongly identified as gestures (false positive rate, $r_{\mathrm{FP}}$). We trained the method on different numbers of gestures and randomly created sequences where only one of multiple gestures was known to the method. Figure 7.6 b) shows the resulting precision-recall (ROC) curves. The best combination of high true positive rates and low false positive rates was achieved in the setting with four gestures. In this case, above 90% of gestures were detected with less than 10% of false positives. Although distinguishing non-gesture movements becomes more difficult when many gestures are learned simultaneously, the gesture segmentation results remain reasonable, even for 18 learned gestures.

#### 7.3.4.3 Influence of Training Dataset Size

As a short training phase is desirable for users [84], in particular in the medical setting, we evaluated the behavior of our method with respect to a varying number of repetitions per

| | Gesture | Feature |
|---|---|---|
| 3. | Left arm forward and backward in front of the body | Browsing slices |
| 6. | Right hand left and right in front of the body | Horizontal scroll |
| 7. | Both hands outwards and back in front of the body | Image zoom |
| 9. | Right hand up and down in front of the body | Vertical scroll |
| 11. | Left hand left and right above the head | Level adjustment |
| 12. | Right hand left and right above the head | Value adjustment |

Table 7.2: Short description of gestures and associated features of the medical image viewer application, as employed in the user study. The image viewer displays a 3D medical tomographic dataset. The gesture numbers refer to the full set of gestures listed in Table 7.1.

gesture in the training data. We recorded a dataset of 8 gestures with an increasing number of repetitions, ranging from 1 to 8. The sequence with 8 repetitions was taken for testing, while training on the other sequences, one by one. Apparently, training on a single repetition per gesture cannot sufficiently account for typical variation when performing a gesture multiple times. However, gesture recognition rates matched those displayed in Figure 7.6 a) for two repetitions per gesture, without additional improvement up to the case of 7 repetitions per gesture. We argue that performing two repetitions per gesture for training is not a significant burden, considering that the system only rarely needs to be re-trained.

#### 7.3.4.4 User Study

We conducted a user study to evaluate the usability of our proposed approach. In our test setup, we used a medical image viewer application as the target system. This application lets the user load and browse medical volumetric image data. We selected a subset of 8 different gestures, each assigned to one of the main features of the image viewer. Table 7.2 lists these gestures and the associated application features. For allowing users to activate and deactivate gesture control in addition to the automatic method based on the confidence value $c_t$, we implemented two exemplary techniques. The users could work with a voice recognition module that reacts to the commands "start" and "stop", or alternatively, with a hand-held wireless switch with two respective buttons.

Ten test subjects participated in our study with an average age of 28 years, three of them were female. After providing basic instructions to each participant, we recorded training data for every gesture with three repetitions. After 15 minutes of free exploration of the system by each user, we asked them to perform a well-defined test task, consisting of locating an aortic stent in a loaded CT dataset and navigating to its bifurcation. For evaluation, we also provided the users with desired target values for all parameters. To reach the target values, the users had to use the trained gestures (Table 7.2). The procedure was performed twice per participant, once using voice recognition and once the hand-held switch to activate and deactivate gesture control. We measured the time the users needed to complete the task for each of the methods and computed the deviation of each of the adjusted parameters from the target values. At the end, the users were asked to fill a questionnaire. As illustrated in Figure 7.7, the user feedback was generally positive. The results show that the sensors are not

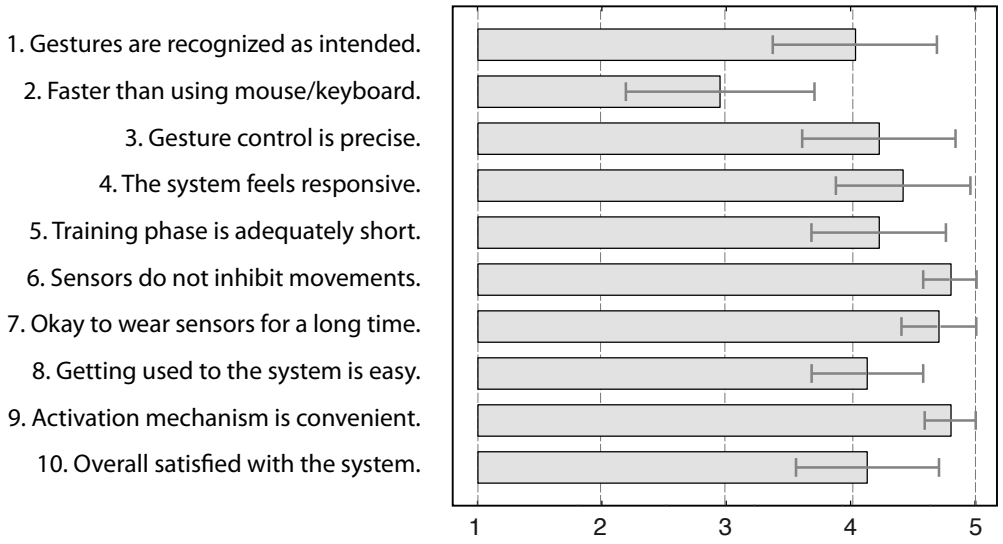



Figure 7.7: Average results of the usability survey for 10 testing subjects. The values indicate subjective consent to the phrases given on the left, on a scale from 1 (*strongly disagree*) to 5 (*strongly agree*). Standard deviations for each item are shown.

inhibiting the usual movement and can be worn for extended periods of time. Furthermore, most of the users agree on the responsiveness and precision of the system. The more skeptical response to the speed comparison with respect to using keyboard and mouse (question 2) can be explained with the fact that the participants have a bias towards using familiar, classical user interfaces. During our study, we noticed that users especially liked being able to move freely within the room. Regarding the activation mechanism, eight of the ten participants preferred the hand-held wireless switch over the voice recognition system. This might be due to the fact that users feel faster response and more direct control over the system when using a switch, as compared to voice commands. In both approaches, the participants achieved an acceptable accuracy in setting the target parameter values. Using the voice command method, users reached the expected parameter values with a deviation of 2% of the respective parameter range. For the hand-held switch, precision was within 4% of the parameter range. This gave enough freedom to the users for accurately controlling the target application. Overall, the study encourages an applicability of the gesture-based interaction method in practice.

## 7.4 Discussion

We have presented a gesture-based interaction solution for the operating room based on wireless inertial sensors. Our approach allows surgeons to define a personalized set of gestures in a training phase that best fits particular requirements. Using manifold learning, we parameterize the inertial sensor training data for each considered gesture in a low-dimensional representation. Apart from automatically recognizing performed gestures from the sensor data, our method simultaneously tracks the relative pose within each gesture, allowing the surgeon to control continuous, smoothly varying parameters. This corresponds to a combination of categorical and spatio-temporal interaction paradigms in one approach. In our quantitative

experiments, we demonstrate that gestures are recognized with high correct classification rates, even when 18 gestures are learned simultaneously. Practical application scenarios would rather require 4-8 different gestures. We believe that our gesture-based interaction approach can also be applied to other collaborative workspace domains, such as industrial assembly lines, with minor modifications.

Using inertial sensors for gesture-based interaction in the operating room allows us to avoid shortcomings of camera-based solutions. Lighting conditions are very challenging in an operating room and there are lots of occlusions due to personnel and medical equipment, such that cameras can hardly be used to capture gestures. In addition, surgeons typically interact with multiple computer screens in the operating room, such that their orientation towards each individual screen is different and not fronto-parallel. Inertial sensors do not depend on any line-of-sight and on the orientation of the surgeon. Moreover, the freedom of moving within the operating room is not restricted at all. In [15], we have presented an adaptation of our gesture-based interaction method that uses depth features extracted with the Kinect as an input. Results in terms of gesture recognition accuracy are comparable to the sensor-based approach. Users initially like the fact that no sensors need to be attached to their arms, however, they quickly realize that their freedom of movement is restricted, as they have to face the Kinect for interaction. An additional advantage of inertial sensors is that each sensor has a unique identification code. This way, we can easily include multiple persons in gesture-based interaction by distributing the sensors.

In our quantitative evaluation in Chapter 5.4, we highlighted the robustness of our manifold-based pose tracking method to changes in sensor placement. Due to the methodological similarity, we argue that the conclusion is equally valid for our gesture-based interaction method. Surgeons can take the sensors off and on again, without requiring new training. The sensor placement only needs to be in a reasonable vicinity of their location during training. Surgeons typically have to completely undress before a surgery and to wear specific sterile operating room clothing. If integrated into this clothing, and keeping in mind continued miniaturization, the inertial sensors would not impose any additional burden to the surgeons.

Our evaluation shows promising results of using the method in an experimental setup. We also received initial positive feedback from surgeons that see our approach as a viable user interaction strategy for the operating room. The surgeons we presented the method to were a minimally invasive abdominal surgeon ($>$ 20 years experience), a vascular surgeon ($>$ 20 years experience), a neuro-surgeon (1 year experience) and a trauma surgeon (3 years experience). While the automatic differentiation between learned gestures and other movements achieves true positive rates above 90%, the surgeons suggested to rely on an additional tool (e.g. a pedal that is frequently used in operating rooms) to activate and deactivate automatic gesture-based control within critical workflow stages.

# Part IV

# Conclusion

Chapter 8

# Discussion

Habe nun, ach! Philosophie,
Juristerey und Medicin,
Und leider auch Theologie!
Durchaus studirt, mit heißem Bemühn.
Da steh' ich nun, ich armer Thor!
Und bin so klug als wie zuvor;[1]

J. W. von Goethe, Faust I

## 8.1 Summary

In this thesis, we have investigated various approaches for human motion analysis, focusing on pose estimation, activity recognition and gesture recognition. There are numerous applications for such methods, ranging from entertainment through industrial monitoring to medical motion analysis. Based on our guiding principle to provide solutions for scenarios where classical camera-based methods face significant challenges, we employed inertial sensors and depth imaging devices for capturing human motion. In order to cope with the noisy and often ambiguous input data, the central idea was to infer additional information on human motion in a training phase where supplementary observations are available. We used an optical marker-based motion capture system as a source of complete and precise human full-body pose data. Machine learning techniques were the core of the presented methods and served the purpose of modeling mappings between observations and full-body poses and creating models of human motion. After training, these models allowed recognizing activities and gestures and predicting poses, given only the sensor or depth data.

We started by directly modeling the mapping from observations, i.e. inertial sensor or depth camera data, to poses. To model this non-functional inverse mapping, we used Gaussian process regression, learning the mapping separately for each considered activity. In this discriminative setting, we introduced two options for combining the activity-specific pose predictions into one final pose estimate: one is using the probabilistic nature of Gaussian process regression and the other uses an SVM classifier. As a result, we derived a simultaneous activity

[1] An English translation of this excerpt can be found in [168].

classification scheme. The method was evaluated on inertial sensor data and achieved results that are comparable to the state-of-art in discriminative human pose estimation. However, this method focuses entirely on the non-functional relationship between the human appearance (here, as measured by inertial sensors) and the underlying body poses. Consequently, the method is sensitive to outliers in the incoming observations. Also, predictions for each consecutive frame are completely independent and no temporal coherence is enforced, as there is no underlying state model of human motion.

In the following, we presented a generative framework for human activity recognition and pose estimation with an integrated prior model of human poses that are likely in a particular application scenario. We used full-body pose training data to extract a low-dimensional search space for human poses based on a manifold representation specific to each considered activity. Regression served the purpose of linking pose hypotheses in the low-dimensional manifold embedding space to observations and to full-body poses, thus avoiding to model the inverse observation-to-pose mapping. We employed an adapted version of the particle filter algorithm to continuously track multiple pose hypotheses in the manifold embedding space. A mechanism was introduced that lets particles switch between activity-specific motion models and allows determining the pose and activity that best match incoming observations. An anomaly detection mechanism enabled us to automatically identify human poses that the system had not been trained on to prevent attributing such poses to a known activity. We evaluated the method on inertial sensor data and depth data from a ToF camera. As expected, the generative method achieved superior results, as compared to the discriminative approach presented before. Moreover, the depth features showed to provide slightly better hints for pose estimation and activity recognition than inertial sensor data.

As an excursus from our main theme, we then proposed an alternative human pose estimation method for depth data that is not based on machine learning. Instead, the idea was to leverage as much as possible the characteristics of depth cameras that provide more than the depth data. For this purpose, our approach was to represent the depth information as a graph, allowing us to estimate geodesic distances between arbitrary points on the human body surface. As opposed to standard Euclidean distances, geodesic distances are nearly invariant to articulated pose changes and thus help us identify anatomical landmarks, e.g. hands and feet, from assumed constant geodesic distances to a reference point on the body surface. To deal with body self-occlusions, where geodesic distances cannot be estimated correctly, we proposed to compute assistive optical flow fields on the intensity images that are provided by current depth cameras. We evaluated this method using ToF cameras and the Kinect and compared the pose estimation performance to the popular NITE skeleton tracker. While providing competitive precision without the need for any training, the method also has disadvantages, compared to our learning-based approaches. First, several restrictive assumptions are made, such as that the person is continuously facing the depth camera and that the person is entirely visible. Second, simultaneous activity recognition is not available.

Based on our generative framework for activity recognition and pose estimation, we finally presented a solution specialized for a particular application scenario. Switching away from recognizing activities in general, we introduced a system that lets surgical staff define arbitrary gestures for touch-less interaction with computerized devices in the operating room. Inertial sensors were used as an input modality, providing complete freedom of movement within the operating room, independent of lighting and occlusions. As the requirement of precise full-

body pose estimation is alleviated for gesture recognition, we removed the need to capture full-body pose data in a training phase. Instead, we used the inertial sensor data directly to learn the low-dimensional manifold representation of feasible poses. We transformed the pose estimation capability of our generative framework to the ability of predicting a relative, one-dimensional pose value. This pose value served the purpose of adjusting arbitrary user interface parameters with fine-grained gesture movements. From a user interaction perspective, this approach enables a simultaneous use of categorical and spatio-temporal gestures, as the system automatically recognizes which gesture is performed and what the relative pose value is at each time instant. We evaluated our method with the exemplary application of controlling several properties of a medical image viewer. Quantitative experiments indicated promising gesture recognition rates and initial feedback from experienced surgeons was highly encouraging.

## 8.2 Learnings

From a high-level perspective, the learnings from this work can be summarized as follows:

- **Inertial orientation sensors** are an interesting alternative input modality for human motion analysis. Compared to accelerometers, the orientation information provides distinctive cues for pose estimation, even when only a few sensors are placed on the body of a person. Similar to accelerometers, orientation sensors are small and lightweight and available in wireless versions. Future technological developments will lead to sensors that are integrated into flexible cloth, making them barely noticeable for their users.

- **Depth cameras**, such as ToF cameras or the Kinect, provide 2.5D scans of a scene in real-time and are thus attractive input modalities for human motion analysis. While segmentation of a person from background can be significantly facilitated with depth data, as compared to classical camera images, pose estimation still requires a sophisticated algorithmic approach. Challenges are mainly related to the noise in the depth measurements and the currently limited resolution of depth cameras. In addition, ToF cameras and the Kinect operate based on infrared light and are thus hardly usable outdoors in direct sunlight. However, depth imaging technology is still in an early stage and future cameras will solve several of the current shortcomings.

- **Machine learning** techniques can help learning general characteristics of human motion, such as which human poses are feasible. Instead of creating an explicit physiological and dynamical human body model, only a training dataset of recorded example movements is required. Machine learning can also be used to learn mappings between human poses in different representations, including vectors of joint angles, inertial sensor measurements or appearance descriptors in image or depth data. A crucial issue with any machine learning method are its generalization properties. When the amount of training data is too small in relation to the complexity of the behavior or mapping to be learned, over-fitting can occur. This means that the training data itself is approximated well but generalization to other, previously unseen data is limited. In the case of human motion analysis, training data that is sufficient for individual types of movements does not necessarily allow to generalize to arbitrary movements. Similarly, training data for one person is often not sufficient to allow tracking poses of another person. Recent work

has shown that vast training datasets can help machine learning methods to track poses for arbitrary activities and persons [146, 51]. A significant advantage of learning-based pose estimation approaches over direct methods without learning is that computational complexity is shifted away from the online application phase to an offline training phase. Additionally, machine learning methods for pose estimation often have implicit activity recognition capabilities.

- **Manifold learning** methods are suitable for generating low-dimensional representations of human motion data, e.g. from motion capture or inertial sensor measurements. In general, human motion data fulfills the main assumption of manifold learning, that the original points (i.e. poses) lie on or close to a manifold embedded in a high-dimensional space (i.e. full-body pose space). Local neighborhood properties between human poses are preserved by manifold learning, which allows efficiently parameterizing human motion data. Meaningful results can be obtained when considering each type of human motion separately, as opposed to combining motion data for multiple activities. An important prerequisite for using manifold learning is to define a similarity measure that fits the properties of a given dataset, e.g. human motion data.

- **Regression** methods, and in particular non-parametric approaches such as kernel regression or Gaussian process regression (GPR), are useful tools for learning mappings between human poses and their appearance as measured by different modalities, such as inertial sensors or depth cameras. While kernel regression in its typical form of the Nadaraya-Watson estimator is a closed-form expression without any training, GPR is based on a probabilistic framework and requires iterative optimization in a training phase. The advantage of GPR is that not only a prediction of a functional value (e.g. a human pose) can be obtained, but also a measure of predictive certainty. This certainty value can be used for combining multiple regression models into one joint predictor. Relevance Vector Machine (RVM) regression also requires training, but can beat other methods in terms of computational performance. Only a subset of the training data, given by the *relevance* vectors that are determined during training, are used for predictions in the application phase.

- **Generative** and **discriminative** models of human motion can both be designed for human pose estimation and activity classification, each with specific advantages and disadvantages. Discriminative models focus on predicting body poses or activities, given some observations of human motion, such as sensor or depth data. Regression is often used to learn this predictive observation-to-pose mapping from training data. While computationally very efficient, such discriminative methods are sensitive to outlier observations, as no internal state and no model of feasible human poses is available. Generative models have an explicit state representation that can *generate* simulated observations and body poses, e.g. based on learned regression mappings. Pose estimation and activity recognition is achieved by searching the state space to find a pose that best matches current incoming observations. Generative models introduce temporal coherence and can better deal with ambiguous input data, as this input is not directly used for pose prediction. However, generative models require techniques for state estimation, e.g. a particle filter, which can be costly from a computational point of view. This disadvantage can be alleviated by employing a dimensionality-reduced state space representation.

Chapter 9

# Perspectives

Having summarized the contributions of this thesis and the key learnings from the associated work, we conclude by pointing out potential future research directions. In this outlook, we separately consider several aspects related to the thesis.

## 9.1 Human Pose Estimation

Research has focused on classical cameras for human pose estimation for decades, until depth cameras appeared a few years ago. This technology simplified formerly challenging computer vision problems right out of the box, such as background segmentation. Pose estimation itself has also benefited from the added third dimension, without the need for complex multi-view reconstruction systems. When the Kinect appeared, its body tracking capabilities received particular attention [146]. For a moment, quite a few believed that all problems of human pose estimation have been solved. Needless to say that the Kinect tracker does have limitations, for instance when tracking subtle movements, particularly when body parts cross or occlude each other. A general question that remains, after having outlined the advantages and limitations of machine learning-based methods for pose estimation, is whether or not such approaches can be robust and flexible enough for practical applications.

In fact, the Kinect body tracker is a machine learning algorithm [146, 51], although typical ingredients of such an approach, e.g. a training phase, seem to be missing. The creators of the algorithm have rather demonstrated the potential of machine learning. Their method uses smart depth features in a vast, partly synthetically generated training dataset of arbitrary human poses. Exceptional computation facilities allowed the authors to learn a random forest classifier that is robust, generalizes well and offers high runtime performance. With a slightly different focus, we have also proposed a machine learning-based approach for human pose estimation from depth data. Our objective was not to track general movements of arbitrary persons to an acceptable degree, but to specialize on poses for application-specific movements, such as the activities of assembly line workers in an industrial environment or the movements of a surgeon in the operating room. While the general tracking approach obviously requires a very large amount of training data, our approach can cope with small, personalized training datasets for a series of considered activities. Machine learning can enable tracking and recognizing these activities that are important for a particular scenario, given incomplete and noisy input data.

We thus believe that further research in human pose estimation and motion tracking should focus on machine learning methods. Solutions can be developed for various types of requirements, including general motion tracking where person-specific training is not desired, to customizable approaches in situations where a series of workflow-related activities are of main interest. The use of depth imaging for human tracking will probably increase in the near future, along with the quality of depth cameras. Nonetheless, there are applications where the line-of-sight requirement of depth cameras and the missing full 3D data (only 2.5D data is provided) will have a prohibitive effect. In these cases, hybrid approaches integrating any combination of regular cameras, depth cameras and body-worn sensors can be worthwhile. An example of such a hybrid approach is shown in [122, 11], where inertial sensors on the body of a person help disambiguating poses in a video-based system. Applications where cameras can hardly be used at all for human tracking include medical long-term motion analysis or future user interfaces for the operating room.

## 9.2 Performance-based Character Animation

The presented discriminative method for human pose estimation and activity recognition using body-worn inertial sensors can be the basis for a performance-based animation system. In animation industry, human motion capture is a common tool for creating character animations by transferring the movements of a real person to a virtual character. Animators typically have to edit the motion capture data to match the constraints of a particular animation, especially in terms of timing [157, 39]. Natural human movements are characterized by highly intertwined spatial and temporal parameters and modifying either of these can significantly alter the appearance and expression of a given movement sequence. Thus, retiming a human motion sequence is often tedious and requires complex expert tools [65].

Based on our method, one professional person, perhaps a dancer, could perform a series of movements, completely independent of the timing constraints of a particular animation. An animator could then transfer the motion-captured professional movements to virtual characters, however, not by relying on keyboard and mouse. Instead, the animator could use his or her own body to directly and intuitively control the sequence of recorded movements. Consequently, the animator would not have to master the desired movements in their full precision but could concentrate on particular timing constraints. Moreover, the animator would not be bound to a motion capture system when creating animations from the recorded professional movements. By using inertial sensors to track his or her body, the animator would be able to move freely without caring about optical markers and camera visibility.

In a training phase, the animator, equipped with inertial sensors, records a control movement for each professional motion sequence. These control movements can vaguely resemble the professional sequences up to the capabilities of the animator, but can also be very simplified proxy movements that match the sequences only in terms of temporal structure. The method presented in this work only needs to be extended with a component for synchronization of the control movements in the training data to the professional motion sequences. Once this synchronization is achieved, a discriminative mapping from inertial sensor data of the animator to full-body poses can be defined, as before. To synchronize the inertial sensor sequences in the training data and the full-body pose data, we propose to use Canonical Time Warping [180], a variant of Dynamic Time Warping that allows for multi-modal input sequences.

After training, the animator can use the control movements to *act out* the desired temporal arrangement of the professional motion sequences. As the details of the movement, e.g. a dance, are encoded in the full-body motion capture data, the animator does not have to concentrate on the precision of the movement itself, but only on its timing, its repetitions or its direction of playback. As opposed to animation editing using keyboard and mouse, the animator can take advantage of the expressiveness of his or her own movements. The activity recognition capabilities of our method allow the animator to select multiple different motion sequences just by performing the corresponding control movements in the desired order. This way, the animator can simultaneously arrange multiple professional motion sequences in the animation time line and intuitively specify their temporal properties.

## 9.3 Medical Long-term Motion Analysis

The approaches for human activity recognition and pose estimation based on inertial sensor data presented in this thesis can also be the basis for applied research in medical long-term motion analysis. While existing systems for ambulatory activity monitoring, e.g. in elderly care, mainly aim at an overall quantification of patient mobility [116, 83, 131] or at an automatic detection of dangerous events [103, 78], a machine learning approach could be used for more patient-specific motion analysis. Learning to recognize a variety of everyday activities, performed in an everyday surrounding, is a problem that is far away from a practical solution. However, particular types of movements that are relevant to a given medical condition could be trained for each patient and occurrences of these selected activities during everyday life could be isolated and analyzed. The training setup, where full-body poses are recorded with a motion capture system synchronized to the inertial sensors, could be established in a clinical setting. Each patient would go through the training procedure once, before being able to leave for everyday life, equipped only with wireless inertial sensors.

Such a system would be suitable for diagnosis and monitoring of various medical conditions. Diseases with motion disorders, such as Multiple Sclerosis, Parkinson's or Epilepsy, are of particular interest. Multiple Sclerosis patients regularly have to go through a mobility assessment for an evaluation of their disease progress. Standardized, pre-defined protocols of assessment movements exist [116] but an occasional performance in front of a physician is influenced by nervousness of the patient and subjectivity. Using inertial sensors, the protocol movements could be trained for each patient at the clinical setup and then measured as performed over longer periods of time under everyday life circumstances. Clinical studies indicate that treatment quality could benefit from such a quantitative, long-term mobility assessment as the inter-observer reliability would increase [116]. In Epilepsy treatment, a significant problem in determining the patient-specific disease causes is that patients themselves do not notice many seizures [31, 110]. Consequently, physicians base their judgement only on the fraction of seizures observed by personnel or remembered by the patients. Automatically detecting seizures using sensors in a stream of random everyday-life movements is very challenging, as seizure patterns are highly patient-specific. Our sensor-based system could be used to learn these particular seizure patterns for a more reliable identification of seizures.

Further work directed towards a practical implementation of such an application should focus on several aspects. First and foremost, suitable commercially available sensors need to be identified that provide orientational measurements, that are small and lightweight and

that have an appropriate battery lifetime. Given current technological developments, such sensors will be available in the upcoming couple of years. Then, a suitable setup needs to be developed such that patients can easily attach and wear the sensors on their own. Our prototypical implementation currently directly transmits the sensor readings to a computer for storage and on-line evaluation. For the long-term monitoring scenario, a small device needs to be distributed together with the sensors that communicates with the sensors and stores the readings locally for a later transmission to an evaluation computer. This would save computational power and battery lifetime for the wearable setup. A stationary evaluation module needs to be developed that displays the reconstructed series of activities and poses to the physician, possibly together with further disease-specific assessments. From an algorithmic perspective, an application with patients in uncontrolled environments also requires additional work on the robustness of sensor-based activity recognition.

## 9.4 User Interfaces for the Operating Room

Potential for future work also exists in the direction of gesture-based user interfaces for the operating room. Recent consumer electronics devices, such as the iPad or iPhone, have set standards in terms of intuitiveness of interaction. Touch-based interfaces are flexible and the capabilities of the underlying system are often clearly exposed without requiring users to study manuals. In fact, highly complex professional systems, such as medical computerized devices, lag behind when it comes to their user interfaces. Complaints of medical professionals that the user interfaces of medical equipment are too cumbersome are increasing [77]. The gesture-based interaction method discussed in this thesis can be seen as one step in the direction of a modern user interface for the operating room that adapts itself to surgical workflow and to preferences of surgical staff. As touch interfaces known from consumer electronics are hardly suitable for the operating room due to sterility requirements, our gesture-based method is a viable alternative that also alleviates significant disadvantages of camera-based methods. Most importantly, surgical staff that is equipped with a few inertial sensors does not have to face cameras and free movement within the operating room is possible.

Based on our current prototypical implementation, further work should mainly focus on usability of the system in a practical setting. First and foremost, a consistent user interface encapsulating the training and application phase needs to be developed. Currently, a skilled expert is in charge of controlling the training recordings and creating the link to the runtime system. Obviously, these steps should be accomplishable by medical staff. Depending on which intra-operative target devices need to be controlled using our gesture-based interface, software interfaces need to be implemented that allow accessing the devices. The critical issue here is that interaction with third-party device vendors is required and access to programming interfaces needs to be granted by these vendors. Furthermore, a user interface has to be implemented that allows connecting the gesture recognition output of our method to specific functions of the target devices. Such a user interface could be based on a graphical representation with components for sensors and functions of target devices that can be arranged using the mouse. We have developed an initial prototype in this direction [16].

Longer-term work could focus on automatically changing the bindings of gestures to functions of target devices based on the input of an external workflow recognition and management system. Based on the current workflow phase, a learned gesture could, thus, be used for con-

trolling different functionalities. An extensible software framework architecture would allow connections between input devices, learned gestures and functions of target devices to be changed dynamically at run-time. Automatic recognition of workflow stages is an active and challenging field of research [114, 115]. While in general incorrect workflow phase recognitions can lead to very undesirable effects, a workflow-aware user interface system would not necessarily pose a threat on patient safety. For instance, when the workflow recognition module is unsure about the current phase, the gesture-to-function bindings could remain in their last secure state. Once a certain estimate of the workflow phase is available (or a short intervention by surgical staff), the interface would react and change its behavior to match the new phase.

# Part V

# Appendix

Appendix A

# Human Body Model

In this work, we frequently use a simple, skeletal human body model. The body model determines the information that is recorded as ground truth and training data and it determines what exactly is predicted with the presented pose estimation algorithms.

## A.1 Degrees of Freedom

Given the lengths of all skeletal segments of the body model, we can uniquely describe a body pose by means of a vector $\mathbf{y} \in \mathbb{R}^{d_y}$ of joint angles. We describe in Appendix B how to determine the person-specific limb lengths in a calibration procedure. In our representation, we simplify a real human skeleton both, in terms of the number of joints, and in terms of the flexibility of each joint. The joints considered in our body model are illustrated in Figure A.1 b) and listed in Table A.1. We model ball joints with three degrees of freedom and hinge joints with either one or two degrees of freedom. In total, our model has 22 joints and $d_y = 33$ degrees of freedom. The model stores internally for each joint a vector that determines the spatial axis of rotation for each degree of freedom. Additionally, each joint is associated with a lower and an upper angular bound to prohibit unfeasible body poses. The model is parameterized such that a vector of zeros $\mathbf{y} = [0, \ldots, 0]^\top \in \mathbb{R}^{d_y}$ results in an idle standing pose.

## A.2 Kinematic Chains

As illustrated in Figure A.1, a kinematic chain is a concept originating from robotics. Here, the movement capabilities of a robotic arm are parameterized by the angles of its joints. The joint locations of the robotic arms are referred to as *effectors*, and there is typically one end-effector that performs some type of work. When the angles are given, the effector locations can be determined easily, as the lengths of all robotic segments are known and also the rotational axes for each joint. This type of computation is called *forward kinematics*. When the desired targets for one or more effectors are given, the angles are sought such that the distance between the effectors and the targets becomes minimal. This more complicated problem is called *inverse kinematics*. We model the human skeleton as a structure with five kinematic chains (two legs, two arms, torso). The root of the torso and the leg chains is in the pelvis, the root of the arm chains is in the clavicle. Hands and feet can be considered as end-effectors.

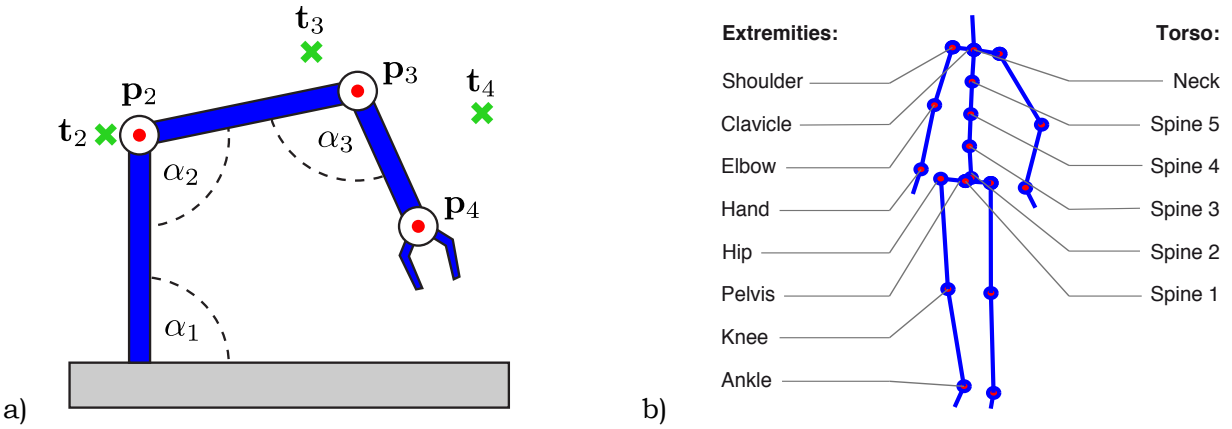



Figure A.1: a) Illustration of a kinematic chain as it is common in robotics. The angles $\alpha_1, \alpha_2$ and $\alpha_3$ determine the spatial location of the effectors $\mathbf{p}_2, \mathbf{p}_3$ and the end-effector $\mathbf{p}_4$. Desired target locations for the effectors are denoted as $\mathbf{t}_2, \mathbf{t}_3$ and $\mathbf{t}_4$. b) Skeletal human body modeled with five kinematic chains (torso and four extremities). The root is at the pelvis.

| **Chain** | **Joint** | **Type** | **Count** | **d.o.f.** |
|---|---|---|---|---|
| Torso | Spine 1 | Hinge | 1 | 1 |
| | Spine 2 | Hinge | 1 | 2 |
| | Spine 3 | Hinge | 1 | 2 |
| | Spine 4 | Hinge | 1 | 2 |
| | Spine 5 | Hinge | 1 | 2 |
| | Neck | Hinge | 1 | 2 |
| Arms | Clavicle | Hinge | 2 | 1 |
| | Shoulder | Ball | 2 | 3 |
| | Elbow | Hinge | 2 | 1 |
| | Hand | Hinge | 2 | 1 |
| Legs | Pelvis | Hinge | 2 | 1 |
| | Hip | Ball | 2 | 3 |
| | Knee | Hinge | 2 | 1 |
| | Ankle | Hinge | 2 | 1 |
| **Total:** | | | **22** | **33** |

Table A.1: Joints in our human body model with their distribution over the kinematic chains, their type and their respective number of degrees of freedom (d.o.f.).

Appendix B

# Marker-based Motion Capture

In this appendix, we illustrate the marker-based motion capture system that is used throughout this work for acquisition of human motion data. The motion captured data serves for training the machine learning algorithms and as a ground truth in our evaluations. We developed our motion capture system based on a commercial infrared marker-based tracking system [2] that can precisely track the spatial location of reflective markers on a person's body. The precision of the tracking system is in the order of millimeters. Our work creates the link between these marker locations and the human body model described in Appendix A, allowing us to record sequences of human full-body poses represented as sets of joint angles (Figure B.1).

While other marker-based motion capture systems exist in the market, the approach developed in this work makes particular use of the capabilities of the ART Dtrack tracking system [2] and allows for a quick and automatic calibration of body parameters for each new user. Typical commercial motion capture systems require a manual specification of limb lengths before tracking can start. In our approach, the limb lengths are computed from marker trajectories in a sequence of arbitrary body movements. In the following sections, we first outline the hardware and software setup that is the basis for our motion capture system. Then, we turn to the person-specific skeleton calibration approach. Finally, we discuss how we fit the skeleton body model to the marker locations provided by the tracking system.

## B.1 System Overview

The Dtrack tracking system consists of infrared cameras, a central computer and a set of $T$ motion capture targets (Figure B.1). Each target consists of several reflective marker spheres that are rigidly arranged in a characteristic pattern. The central computer triggers the cameras at each time instant $t$ and each camera extracts the location of the markers in 2D image coordinates. This information is used on the central computer for 3D triangulation. As every target can be identified by its unique marker pattern, the tracking system returns the location $\mathbf{t}_k^t \in \mathbb{R}^3$, $k \in \{1, \ldots, T\}$ for each target as a whole. The tracking system can also determine the orientation of each target that is returned as a rotation matrix $\mathbf{R}_k^t \in \mathbb{R}^{3\times3}$.

For human motion capturing, we use $T = 15$ targets distributed over the body. Figure B.1 b) illustrates the targets when worn by a person. The target locations do not necessarily coincide with any anatomical landmarks, such as joints. Our goal is to compute, at each time

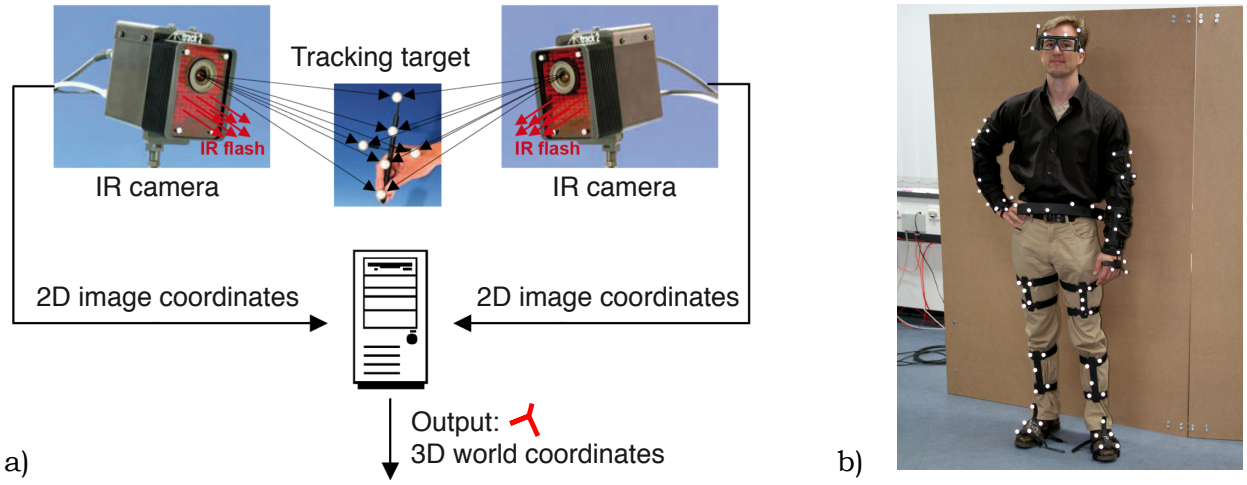



Figure B.1: Overview of the marker tracking process. a) The tracking system computes the 3D locations and orientations for every tracking target worn by a person (picture adapted from [186]). b) Person with 14 tracking targets for motion capture.

instant, the locations of all $K$ skeleton joints, denoted as $\{\mathbf{s}_i\}_{i=1}^{K}$, where $K$ is the total number of joints in the body model. For this purpose, we first obtain preliminary joint locations $\{\tilde{\mathbf{s}}_i\}_{i=1}^{\tilde{K}}$ that are directly coupled to the tracking targets and use data from the calibration procedure. The number of preliminary joint locations $\tilde{K}$ is less than $K$, as there are joints that are not associated to any tracking target. Given the preliminary joint locations, we fit the skeleton body model using an inverse kinematics approach. To prevent losing track of targets due to body self-occlusions, we use six to eight tracking cameras. The cameras are positioned around a cubic tracking volume with a side length of approximately three meters.

## B.2 Person-specific Skeleton Calibration

Our motion capture algorithm fits the human skeleton body model to the locations of the targets as returned by the tracking system. Before tracking, the person-specific proportions of this skeleton model need to be determined for being able to accomodate the target locations. Typical marker-based motion capture systems require the user to manually specify these proportions by supplying rough estimates of the lengths of arms, legs and other limbs of each new user. Inspired by the work of O'Brien et al. [109] and Hornung et al. [64], we avoid this cumbersome step and provide an automatic calibration procedure. In this approach, the user is equipped with motion capture targets and performs a sequence of calibration movements. These movements are arbitrary, with the requirement that the degrees of freedom of all body joints are exerted. The trajectories of the markers throughout the calibration sequence then allow computing the locations of the joints and the lengths of the limb segments.

Consider for the following discussion the schematic in Figure B.2. The rotations $\mathbf{R}_k^t$ and translations $\mathbf{t}_k^t$ returned by the tracking system at each time $t$ are expressed in the world coordinate system. In order to compute the length of the limb segments, we need to determine the rotation centers between all pairs of tracking targets on adjacent limb segments. For every target, there is an offset vector, expressed in the target's local coordinate system, pointing

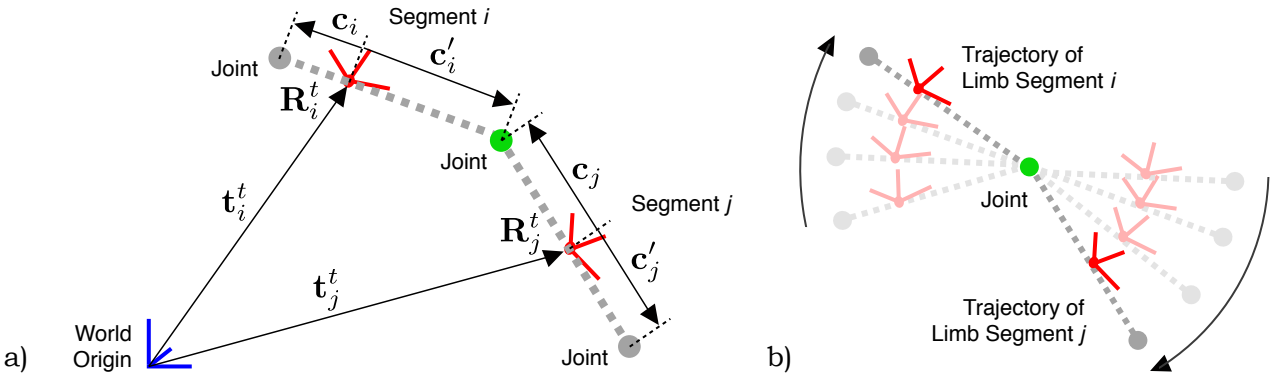



Figure B.2: Estimation of a joint position from tracking targets on two adjacent limb segments $i$ and $j$. a) At each frame $t$, rotations and translations in world coordinates for the two targets (red) are given. The local offsets $\mathbf{c}_i, \mathbf{c}'_i$ and $\mathbf{c}_j, \mathbf{c}'_j$ from each target to its adjacent joints are constant throughout all frames. Transformed to the world coordinate system, the offsets surrounding one joint ideally meet in the same point (green). b) Using this constraint, the offsets are estimated from the trajectories of the limb segments over several frames.

to the inward joint and one offset vector pointing to the outward joint (Figure B.2). Let us identify the targets on two adjacent limb segments with the indices $i$ and $j$. We will denote the inward and outward offsets for the first target as $\mathbf{c}_i, \mathbf{c}'_i \in \mathbb{R}^3$ and $\mathbf{c}_j, \mathbf{c}'_j$ for the second target. Expressed in the global coordinate frame, the two offsets $\mathbf{c}'_i$ and $\mathbf{c}_j$ that point towards each other should actually coincide,

$$\mathbf{R}_i^t \mathbf{c}'_i + \mathbf{t}_i^t = \mathbf{R}_j^t \mathbf{c}_j + \mathbf{t}_j^t. \tag{B.1}$$

This gives us a constraint we can use to compute the rotation center (i.e. the joint location) for the pair of limb segments, given the rotations and translations of the corresponding motion capture targets. Rearrangement of the terms in Equation B.1 to isolate the unknown offsets yields

$$\mathbf{R}_i^t \mathbf{c}'_i - \mathbf{R}_j^t \mathbf{c}_j = \mathbf{t}_j^t - \mathbf{t}_i^t, \tag{B.2}$$

and this equation can be rewritten as a matrix-vector product

$$\begin{bmatrix} \mathbf{R}_i^t & -\mathbf{R}_j^t \end{bmatrix} \begin{bmatrix} \mathbf{c}'_i \\ \mathbf{c}_j \end{bmatrix} = \begin{bmatrix} \mathbf{t}_j^t - \mathbf{t}_i^t \end{bmatrix}. \tag{B.3}$$

Solving this equation directly, we would only take into account the constraint for one single frame $t$. Instead, to obtain a robust estimate of $\mathbf{c}'_i$ and $\mathbf{c}_j$, we stack up Equation B.3 for a sequence of $s$ frames, $t \in \{1, \ldots, s\}$ and solve the resulting linear system of equations

$$\begin{bmatrix} \mathbf{R}_i^1 & -\mathbf{R}_j^1 \\ \vdots & \vdots \\ \mathbf{R}_i^s & -\mathbf{R}_j^s \end{bmatrix} \begin{bmatrix} \mathbf{c}'_i \\ \mathbf{c}_j \end{bmatrix} = \begin{bmatrix} \mathbf{t}_j^1 - \mathbf{t}_i^1 \\ \vdots \\ \mathbf{t}_j^s - \mathbf{t}_i^s \end{bmatrix} \tag{B.4}$$

in the least squares sense [109]. We actually set up one linear system as in Equation B.4 for each pair of adjacent limb segments $i$ and $j$ until the offsets to all joints are known. In practice, we

```
>> [jointPos, limbLen] = mocap();
Listening to port: 5000
--- timeout while waiting for udp data
Using 3098 frames to compute joints.
Upper arm left:   296.94, Lower arm left:   227.77
Upper arm right:  292.43, Lower arm right:  237.48
Thigh left:       387.37, Lower leg left:   372.06
Thigh right:      394.27, Lower leg right:  360.65
Shoulders:        211.97, Hips:             175.16
Torso:            448.55
```

Figure B.3: Exemplary output of the person-specific body calibration algorithm. 3098 frames are used for computation of the indicated limb segment lengths (in millimeters).

use calibration sequences of roughly 30 seconds, corresponding to $s = 1800$ frames, given that the tracking system operates at a frequency of 60 Hz. As a result of the calibration procedure, we store for each tracking target the two offset vectors $\mathbf{c}_i$ and $\mathbf{c}'_i$ pointing to its inward and outward joints. The limb lengths are determined from the computed world coordinates of all joints locations, averaged over the entire calibration sequence. An example calibration output from our Matlab implementation is shown in Figure B.3.

## B.3 Inverse Kinematic Skeleton Fitting

After the person-specific calibration procedure, we initialize a human skeleton model with the computed limb lengths. Before applying a skeleton fitting technique, we first transform the target locations to preliminary joint locations $\{\tilde{\mathbf{s}}_i^t\}_{i=1}^{\tilde{K}}$. To achieve this, we use the offsets for every target that we obtain as described above. Let $i$ and $j$ be the indices of the targets surrounding the $k$-th joint. We set the corresponding preliminary joint location to

$$\tilde{\mathbf{s}}_k^t = \frac{1}{2}(\mathbf{R}_i\mathbf{c}'_i + \mathbf{t}_i + \mathbf{R}_j\mathbf{c}_j + \mathbf{t}_j). \tag{B.5}$$

As this equation contains the outward pointing offset $\mathbf{c}'_i$ from target $i$ and the inward pointing offset $\mathbf{c}_j$ from target $j$, we basically average the joint location as seen from the two targets. For terminal joints, i.e. joints that only have one incident limb segment, we simply use the preliminary joint location from one target. Figure B.4 b) shows preliminary joint locations, together with the locations of the tracking targets.

Having determined the $\tilde{K}$ preliminary joint locations, we proceed by fitting the skeleton model to these locations. We follow an inverse kinematics approach, considering every limb of the skeleton model as a kinematic chain. In inverse kinematics, the locations for one or more goals[1] are given that should be reached with the effectors on the kinematic chain. The output of inverse kinematics is a set of joint angles for the kinematic chain that allow the effectors to reach the goals. In robotics, there is often just one end-effector and one goal. In our case, we define several effectors per kinematic chain and use the preliminary joint locations as goals. Let $\kappa(i)$ be the index of the body joint (here: the effector) that should be fitted to the $i$-th

[1]We use the term *goal* instead of the term *target* to avoid confusion with the motion capture targets.

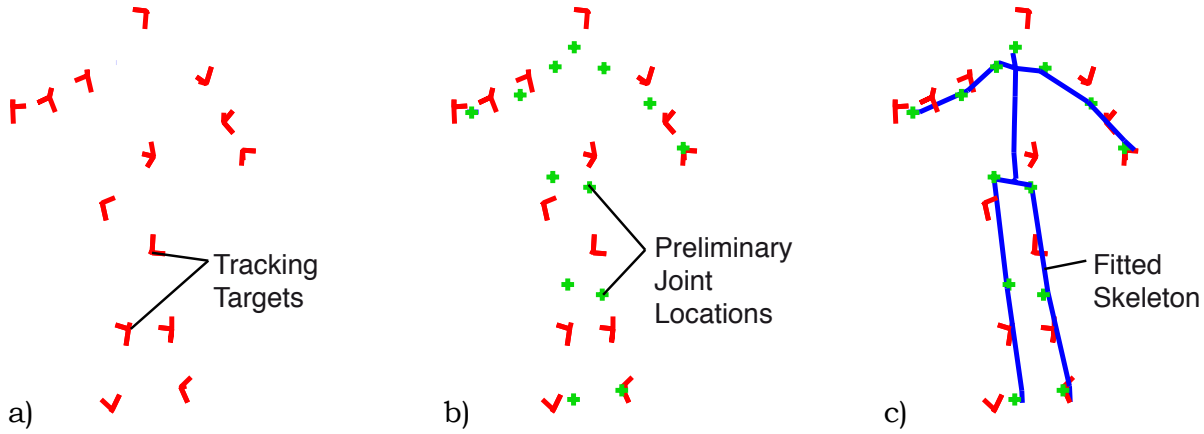



Figure B.4: Overview of the skeleton fitting process. a) Target locations and orientations as returned by the tracking system (red). b) Preliminary joint locations (green) estimated using pre-computed information from the skeleton calibration process. c) Final result after fitting a body model (blue) to the joint locations.

preliminary joint location (here: the goal). Our formal objective is then to find the optimal joint angle configuration $\mathbf{y} \in \mathbb{R}^{d_y}$, such that the residual error

$$\mathcal{E}(\mathbf{y}, t) = \sum_{i=1}^{\tilde{K}} \left\| \tilde{\mathbf{s}}_i^t - f_{\text{kin}}^{\kappa(i)}(\mathbf{y}) \right\| + c(\mathbf{y}) \tag{B.6}$$

is minimized. Here, $f_{\text{kin}}^{(i)} : \mathbb{R}^{d_y} \rightarrow \mathbb{R}^3$ is a forward kinematic function that computes the 3D position of the $i$-th skeleton model joint, given a vector $\mathbf{y}$ of joint angles. The term $c(\mathbf{y})$ penalizes joint angle configurations that violate the joint angle bounds specified in the human body model. This term increases polynomially when any of the joint angles approaches its lower or upper limits.

To determine the optimal joint angle configuration $\mathbf{y}_t$ at each time step $t$, we employ an iterative Gauss-Newton optimization approach that, starting with an initial value $\mathbf{y}_t^0$, computes updates $\Delta_\mathbf{y}$ such that $\mathbf{y}_t^{i+1} = \mathbf{y}_t^i + \Delta_\mathbf{y}$, until convergence. In each frame, we use the joint angles of the previous frame as an initial value, $\mathbf{y}_t^0 = \mathbf{y}_{t-1}$. Assuming incremental body movement between subsequent frames, this increases convergence rates and decreases the probability of hitting local minima. In addition to the joint angles $\mathbf{y}_t$, the skeleton fitting step also gives us the corresponding 3D locations $\{\mathbf{s}_i^t\}_{i=1}^K$ of all skeleton joints. Note that these locations will be close to the preliminary joint locations that we obtained directly from the motion capture targets, but skeleton fitting will ensure that the final joint locations agree with the proportions of the body model. Figure B.4 c) shows the result of skeleton fitting and illustrates that the skeleton has more joints than there are preliminary joints locations.

The skeleton fitting procedure introduces favorable robustness properties against motion capture targets that get lost temporarily. As each preliminary joint location is computed from the position and orientation of two neighboring motion capture targets (Equation B.5), we can estimate the preliminary joint location even when one of the targets is lost. Consider as an example the wrist joint that is computed from the hand target and the lower arm target. When either the hand or the arm target is lost, the preliminary location of the wrist joint remains

intact and the skeleton can be fitted. In the less likely case that both targets surrounding one joint get lost, the preliminary joint location will also become invalid, but the properties of the skeleton model will prevent body tracking from failing. If an interior joint, such as the elbow, is lost for this reason, skeleton fitting will find a plausible location for the joint, even in absence of an explicit inverse kinematics target. If an outer joint is lost, such as the wrist (i.e. both, the hand and lower arm targets are lost), the corresponding part of the body model will keep the angular values from the last valid measurement. This will not necessarily match the true pose of the limb at this moment, but the returned pose will be a feasible human pose.

Appendix C

# Authored and Co-authored Publications

### Journal Publications

- L A Schwarz, A Mkhitaryan, D Mateus, and N Navab. Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing*, 30(3):217–226, 2012.
- L A Schwarz, D Mateus, and N Navab. Recognizing multiple human activities and tracking full-body pose in unconstrained environments. *Pattern Recognition*, 45(1):11–23, 2012.

### Peer-Reviewed Conference Publications

- L A Schwarz, A Bigdelou, and N Navab. An adaptive solution for intra-operative gesture-based human-machine interaction. *ACM Conference on Intelligent User Interfaces (IUI)*, 2012.
- A Bigdelou, T Benz, L A Schwarz, and N Navab. Simultaneous categorical and spatio-temporal 3D gestures using Kinect. *IEEE Symposium on 3D User Interfaces (3DUI)*, 2012.
- L A Schwarz, A Bigdelou, and N Navab. Learning gestures for customizable human-computer interaction in the operating room. *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 129–136, 2011.
- L A Schwarz, A Mkhitaryan, D Mateus, and N Navab. Estimating human 3d pose from time-of-flight images based on geodesic distances and optical flow. *IEEE Conference on Automatic Face and Gesture Recognition* - ***Outstanding Paper Award***, 2011.
- L A Schwarz, D Mateus, V Castaneda, and N Navab. Manifold learning for ToF-based human body tracking and activity recognition. *British Machine Vision Conference (BMVC)* - ***Best Supplementary Material Prize***, 2010.
- L A Schwarz, D Mateus, and N Navab. Multiple-activity human body tracking in unconstrained environments. *International Conference on Articulated Motion and Deformable Objects (AMDO)* - ***Best Paper Award***, pages 192–202, 2010.

### Workshop Publications

- A Bigdelou, L A Schwarz, T Benz, and N Navab. A flexible platform for developing context-aware 3d gesture-based interfaces. *ACM Conference on Intelligent User Interfaces (IUI)*, 2012.
- L A Schwarz, D Mateus, J Lallemand, and N Navab. Tracking planes with time of flight cameras and J-linkage. *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 664–671, 2011.
- L A Schwarz, D Mateus, and N Navab. Discriminative human full-body pose estimation from wearable inertial sensor data. *Modelling the Physiological Human (3DPH)*, pages 159–172, 2009.

### Book Chapters

- D Mateus, C Wachinger, S Atasoy, L Schwarz, and N Navab. *Learning manifolds: design analysis for medical applications. In: Suzuki Kenji (Ed.): Machine Learning in Computer-Aided Diagnosis: Medical Imaging Intelligence and Analysis.* IGI Global, 2011.
- B Becker, S Cui, M Huber, P Keitler, G Klinker, S Lieberknecht, P Lükewille, D Pustka, R Scheibe, L A Schwarz, B Schwerdtfeger, C Wächter, A Weiss, and K Zürl. *Lokalisation und Tracking. In: Schreiber W. and Zimmermann, P. (Eds.): Virtuelle Techniken im industriellen Umfeld: Das AVILUS-Projekt - Technologien und Anwendungen.* Springer Berlin/Heidelberg, 2011.

# Bibliography


[1] S Abe. *Support Vector Machines for Pattern Classification*. Springer Berlin/Heidelberg, 2005.

[2] Advanced Realtime Tracking GmbH. DTrack 2. `http://www.ar-tracking.de`, 2011.

[3] A Agarwal and B Triggs. Learning to track 3d human motion from silhouettes. *International Conference on Machine Learning (ICML)*, 2004.

[4] A Agarwal, B Triggs, and F Montbonnot. Recovering 3d human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(1):44–58, 2006.

[5] S A Ahmadi, N Padoy, S M Heining, H Feussner, M Daumer, and N Navab. Introducing wearable accelerometers in the surgery room for activity detection. *Jahrestagung der Deutschen Gesellschaft für Computer-und Roboter-Assistierte Chirurgie (CURAC)*, 2008.

[6] S A Ahmadi, N Padoy, K Rybachuk, H Feussner, S M Heining, and N Navab. Motif discovery in OR sensor data with application to surgical workflow analysis and activity detection. *MICCAI Workshop on Modeling and Monitoring of Computer Assisted Interventions*, 2009.

[7] J Alon, V Athitsos, Q Yuan, and S Sclaroff. A unified framework for gesture recognition and spatiotemporal gesture segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(9):1685–1699, 2009.

[8] K Altun, B Barshan, and O Tunçel. Comparative study on classifying human activities with miniature inertial and magnetic sensors. *Pattern Recognition*, 43(10):3605–3620, 2010.

[9] K Aminian, E De Andres, K Rezakhanlou, C Fritsch, and P Robert. Motion analysis in clinical practice using ambulatory accelerometry. *Modelling and Motion Capture Techniques for Virtual Environments*, 1998.

[10] S Atasoy, D Mateus, J Lallemand, A Meining, G Z Yang, and N Navab. Endoscopic video manifolds. *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 437–445, 2010.

[11] A Baak, T Helten, M Müller, G Pons-Moll, B Rosenhahn, and H P Seidel. Analyzing and evaluating markerless motion tracking using inertial sensors. *European Conference on Computer Vision (ECCV) Workshops*, 2010.

[12] J Bandouch, F Engstler, and M Beetz. Accurate human motion capture using an ergonomics-based anthropometric human model. *Articulated Motion and Deformable Objects (AMDO)*, 2008.

[13] B Becker, S Cui, M Huber, P Keitler, G Klinker, S Lieberknecht, P Lükewille, D Pustka, R Scheibe, L A Schwarz, B Schwerdtfeger, C Wächter, A Weiss, and K Zürl. *Lokalisation und Tracking. In: Schreiber W. and Zimmermann, P. (Eds.): Virtuelle Techniken im industriellen Umfeld: Das AVILUS-Projekt - Technologien und Anwendungen*. Springer Berlin/Heidelberg, 2011.

[14] M Belkin and P Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373 – 1396, 2003.

[15] A Bigdelou, T Benz, L A Schwarz, and N Navab. Simultaneous categorical and spatio-temporal 3D gestures using Kinect. *IEEE Symposium on 3D User Interfaces (3DUI)*, 2012.

[16] A Bigdelou, L A Schwarz, T Benz, and N Navab. A flexible platform for developing context-aware 3d gesture-based interfaces. *ACM Conference on Intelligent User Interfaces (IUI)*, 2012.

[17] C M Bishop. *Pattern Recognition and Machine Learning*. Springer Berlin/Heidelberg, 2006.

[18] A Bissacco, M H Yang, and S Soatto. Fast human pose estimation using appearance and motion via multi-dimensional boosting regression. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[19] J Blackburn and E Ribeiro. Human motion recognition using Isomap and Dynamic Time Warping. *Conference on Human Motion: Understanding, modeling, capture and animation*, 2007.

[20] A Bleiweiss and EEG Kutliroff. Markerless motion capture using a single depth sensor. *ACM SIGGRAPH ASIA Sketches*, 2009.

[21] R Boulic, J Varona, L Unzueta, M Peinado, A Suescun, and F Perales. Evaluation of on-line analytic and numeric inverse kinematics approaches driven by partial vision input. *Virtual Reality*, 10(1):48–61, 2006.

[22] H Bubb, F Engstler, F Fritzsche, C Mergl, O Sabbah, P Schaefer, and I Zacher. The development of RAMSIS in past and future as an example for the cooperation between industry and university. *International Journal of Human Factors Modelling and Simulation*, 2006.

[23] C Cagniart, E Boyer, and S Ilic. Free-form mesh tracking: a patch-based approach. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[24] C Cagniart, E Boyer, and S Ilic. Probabilistic deformable surface tracking from multiple videos. *European Conference on Computer Vision (ECCV)*, pages 326–339, 2010.

[25] M A Carreira-Perpinán and Z Lu. The Laplacian eigenmaps latent variable model. *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[26] I Chang and S-Y Lin. 3d human motion tracking based on a progressive particle filter. *Pattern Recognition*, 43(10):3621–3635, 2010.

[27] T Chen and J Ren. Bagging for Gaussian process regression. *Neurocomputing*, 72(7-9):1605–1610, 2009.

[28] T Cormen, C E Leiserson, R Rivest, and C Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, 2001.

[29] J Costa, A Girotra, and A Hero. Estimating local intrinsic dimension with k-nearest neighbor graphs. *IEEE Workshop on Statistical Signal Processing*, pages 417–422, 2005.

[30] J Costa and A Hero. Manifold learning with geodesic minimal spanning trees. *Arxiv preprint cs/0307038*, 2003.

[31] J P S Cunha, C Vollmar, Z Li, J Fernandes, B Feddersen, and S Noachtar. Movement quantification during epileptic seizures: a new technical contribution to the evaluation of seizure semiology. *25th Annual International Conference of the IEEE EMBS*, 2003.

[32] J Darby, B Li, and N Costen. Behaviour based particle filtering for human articulated motion tracking. *International Conference on Pattern Recognition (ICPR)*, 2008.

[33] J Darby, B Li, and N Costen. Tracking a walking person using activity-guided annealed particle filtering. *IEEE International Conference on Automatic Face and Gesture Recognition*, 2008.

[34] J Darby, B Li, and N Costen. Tracking human pose with multiple activity models. *Pattern Recognition*, 43(9):3042–3058, 2010.

[35] A Datta, Y Sheikh, and T Kanade. Modeling the product manifold of posture and motion. *Workshop on Tracking Humans for the Evaluation of their Motion in Image Sequences (THEMIS)*, Aug 2009.

[36] S Demirci, F Manstad-Hulaas, and N Navab. Quantification of aortic deformation after EVAR. *SPIE Medical Imaging*, 2009.

[37] S Denman, V Chandran, and S Sridharan. An adaptive optical flow technique for person tracking systems. *Pattern recognition letters*, 28(10):1232–1239, 2007.

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

[39] M Dontcheva, G Yngve, and Z Popovic. Layered acting for character animation. *ACM Transactions on Graphics (TOG)*, 22(3):409—416, 2003.

[40] R D Edwards. The topology of manifolds and cell-like maps. *International Congress of Mathematicians*, pages 111–127, 1978.

[41] C H Ek, P H S Torr, and N D Lawrence. Gaussian process latent variable models for human pose estimation. *Machine Learning for Multimodal Interaction*, pages 132–143, 2008.

[42] A Elgammal and C Lee. The role of manifold learning in human motion analysis. *Human Motion Understanding, Modeling, Capture and Animation*, pages 1–29, 2008.

[43] H Feldwisch-Drentrup, B Schelter, M Jachan, J Nawrath, J Timmer, and A Schulze-Bonhage. Joining the benefits: Combining epileptic seizure prediction methods. *Epilepsia*, 51(8):1598–1606, 2010.

[44] A Ferscha, S Resmerita, C Holzmann, and M Reichör. Orientation sensing for gesture-based interaction with smart artifacts. *Computer communications*, 28(13):1552–1563, 2005.

[45] A Fossati, M Salzmann, and P Fua. Observable subspaces for 3d human motion recovery. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[46] J Gall, B Rosenhahn, T Brox, and H P Seidel. Optimization and filtering for human motion capture. *International Journal of Computer Vision*, 87(1):75–92, 2010.

[47] J Gall, C Stoll, E De Aguiar, C Theobalt, B Rosenhahn, and H P Seidel. Motion capture using joint skeleton tracking and surface estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1753, 2009.

[48] J Gall, A Yao, and L Van Gool. 2d action recognition serves 3d human pose estimation. *European Conference on Computer Vision (ECCV)*, pages 425–438, 2010.

[49] V Ganapathi, C Plagemann, D Koller, and S Thrun. Real time motion capture using a single time-of-flight camera. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[50] M Georg, R Souvenir, A Hope, and R Pless. Manifold learning for 4D CT reconstruction of the lung. *Computer Vision and Pattern Recognition Workshops*, 2008.

[51] R Girshick, J Shotton, P Kohli, A Criminisi, and A Fitzgibbon. Efficient regression of general-activity human poses from depth images. *IEEE International Conference on Computer Vision (ICCV)*, 2011.

[52] C Graetzel, T Fong, S Grange, and C Baur. A non-contact mouse for surgeon-computer interaction. *Technology and Health Care*, 12(3):245–257, 2004.

[53] K Grochow, S Martin, A Hertzmann, and Z Popović. Style-based inverse kinematics. *ACM Transactions on Graphics*, 23(3):522–531, 2004.

[54] K Guerin, B Vagvolgyi, A Deguet, C Chen, D Yuh, and R Kumar. ReachIN: A modular vision based interface for teleoperation. *MICCAI SACAI Workshop*, 2010.

[55] F Guimbretiere and T Winograd. FlowMenu: combining command, text, and data entry. *Symposium on User Interface Software and Technology (UIST)*, pages 213–216, 2000.

[56] A Gupta, T Chen, F Chen, D Kimber, and L S Davis. Context and observation driven latent variable model for human pose estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[57] S Gustafson, D Bierwirth, and P Baudisch. Imaginary interfaces: spatial interaction with empty hands and without visual feedback. *Symposium on User Interface Software and Technology (UIST)*, pages 3–12, 2010.

[58] M Haker, M Böhme, M Martinetz, and E Barth. Scale-invariant range features for time-of-flight camera applications. *Time-of-Flight Camera Based Computer Vision*, 2008.

[59] B Hartmann and N Link. Gesture recognition with inertial sensors and optimized DTW prototypes. *IEEE Conference on Systems Man and Cybernetics*, 2010.

[60] X He and P Niyogi. Locality preserving projections. *Neural Information Processing Systems (NIPS)*, 16:153, 2004.

[61] M B Holte, T B Moeslund, and P Fihl. Fusion of range and intensity information for view invariant gesture recognition. *Computer Vision and Pattern Recognition Workshops*, 2008.

[62] R Horaud, M Niskanen, G Dewaele, and E Boyer. Human motion tracking by registering an articulated surface to 3d points and normals. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 158–163, 2008.

[63] B K P Horn and B G Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[64] A Hornung, S Sar-Dessai, and L Kobbelt. Self-calibrating optical motion tracking for articulated bodies. *Virtual Reality*, Jan 2005.

[65] E Hsu, M da Silva, and J Popović. Guided time warping for motion editing. *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 45–52, 2007.

[66] R Hu, A Hartfiel, J Tung, A Fakih, J Hoey, and P Poupart. 3d pose tracking of walker users' lower limb with a structured-light camera on a moving platform. *Computer Vision and Pattern Recognition Workshops*, 2011.

[67] Warner Bros. Entertainment Inc. The polar express. `http://polarexpressmovie.warnerbros.com/`, 2004.

[68] M Isard and A Blake. Condensation – conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[69] M Isard and A Blake. A mixed-state condensation tracker with automatic model-switching. *IEEE International Conference on Computer Vision (ICCV)*, pages 107–112, 1998.

[70] G Isguder, G Unal, M Groher, N Navab, A Kalkan, M Degertekin, H Hetterich, and J Rieber. Manifold learning for image-based gating of intravascular ultrasound (IVUS) pullback sequences. *Medical Imaging and Augmented Reality*, pages 139–148, 2010.

[71] S Ishigaki, T White, VB Zordan, and CK Liu. Performance-based control interface for character animation. *ACM Transactions on Graphics (TOG)*, 28(3):1–8, 2009.

[72] T Jaeggli, E Koller-Meier, and L van Gool. Learning generative models for multi-activity body pose estimation. *International Journal of Computer Vision*, 83(2):121–134, 2009.

[73] A Jaimes and N Sebe. Multimodal human-computer interaction: A survey. *Computer Vision and Image Understanding*, 108(1-2):116–134, 2007.

[74] R Jensen, R Paulsen, and R Larsen. Analyzing gait using a time-of-flight camera. *Scandinavian Conference on Image Analysis*, pages 21–30, 2009.

[75] V John, E Trucco, and S Ivekovic. Markerless human articulated tracking using hierarchical particle swarm optimisation. *Image and Vision Computing*, 28(11):1530–1547, 2010.

[76] A E Johnson and M Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 1999.

[77] R Johnson, K O'Hara, A Sellen, C Cousins, and A Criminisi. Exploring the potential for touchless interaction in image-guided interventional radiology. *ACM Conference on Human Factors in Computing Systems*, 2011.

[78] E Jovanov, A Milenkovic, C Otto, and P C de Groen. A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, 2(1):6, 2005.

[79] D Achacon Jr, D Carlos, M Puyaoan, C T Clarin, and P C Naval Jr. REALISM: Real-time hand gesture interface for surgeons and medical experts. *Citeseer*, 2009.

[80] H Junker, O Amft, P Lukowicz, and G Tröster. Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition*, 41(6):2010–2024, 2008.

[81] A Kanaujia, C Sminchisescu, and D Metaxas. Spectral latent variable models for perceptual inference. *IEEE International Conference on Computer Vision (ICCV)*, 2007.

[82] R Kehl and L van Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 107(2):190–209, 2006.

[83] N L W Keijsers, M Horstink, and S Gielen. Ambulatory motor assessment in Parkinson's disease. *Movement Disorders*, 21(1):34–44, 2006.

[84] J Kela, P Korpipää, J Mäntyjärvi, S Kallio, G Savino, L Jozzo, and S D Marca. Accelerometer-based gesture control for a design environment. *Personal and Ubiquitous Computing*, 10(5):285–299, 2006.

[85] T Kipshagen, M Graw, V Tronnier, M Bonsanto, and U G Hofmann. Touch-and marker-free interaction with medical software. *World Congress on Medical Physics and Biomedical Engineering 2009*, pages 75–78, 2009.

[86] S Knoop, S Vacek, and R Dillmann. Modeling joint constraints for an articulated 3d human body model with artificial correspondences in ICP. *Humanoid Robots*, 2005.

[87] R Koch, I Schiller, B Bartczak, F Kellner, and K Koser. MixIn3D: 3D mixed reality with ToF-camera. *Dynamic 3D Imaging*, pages 126–141, 2009.

[88] A Kolb, E Barth, R Koch, and R Larsen. Time-of-flight sensors in computer graphics. *Eurographics*, pages 119–134, 2009.

[89] E Kollorz, J Penne, J Hornegger, and A Barke. Gesture recognition with a time-of-flight camera. *International Journal of Intelligent Systems Technologies and Applications*, 5(3):334–343, 2008.

[90] T Komura, B Lam, R Lau, and H Leung. e-Learning Martial Arts. *Advances in Web Based Learning (ICWL)*, 2006.

[91] D Y Kwon and M Gross. Combining body sensors and visual sensors for motion training. *ACM International Conference on Advances in computer entertainment technology*, pages 94–101, 2005.

[92] N D Lawrence and J Quiñonero-Candela. Local distance preservation in the GP-LVM through back constraints. *Proceedings of the 23rd international conference on Machine learning*, pages 513–520, 2006.

[93] C Lee and A Elgammal. Modeling view and posture manifolds for tracking. *IEEE International Conference on Computer Vision (ICCV)*, Jan 2007.

[94] J M Lee. *Introduction to Topological Manifolds, Second Edition*. Springer Berlin/Heidelberg, 2011.

[95] E Levina and P J Bickel. Maximum likelihood estimation of intrinsic dimension. *Neural Information Processing Systems (NIPS)*, 48109:1092, 2005.

[96] J Liu, L Zhong, J Wickramasuriya, and V Vasudevan. uWave: Accelerometer-based personalized gesture recognition and its applications. *Pervasive and Mobile Computing*, 5(6):657–675, 2009.

[97] M Losch, S Schmidt-Rohr, S Knoop, and S Vacek. Feature set selection and optimal classifier for human activity recognition. *Robot and Human interactive Communication*, 2007.

[98] Z Lu, M Carreira-Perpinán, and C Sminchisescu. People tracking with the Laplacian eigenmaps latent variable model. *Neural Information Processing Systems (NIPS)*, 2007.

[99] S Malassiotis and M G Strintzis. Real-time hand posture recognition using range data. *Image and Vision Computing*, 26(7):1027–1037, 2008.

[100] D Mateus, C Wachinger, S Atasoy, L Schwarz, and N Navab. *Learning manifolds: design analysis for medical applications. In: Suzuki Kenji (Ed.): Machine Learning in Computer-Aided Diagnosis: Medical Imaging Intelligence and Analysis.* IGI Global, 2011.

[101] Microsoft Corporation. XBox Kinect. `www.xbox.com/kinect`, 2010.

[102] S Mitra and T Acharya. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324, 2007.

[103] S T Moore, H G MacDougall, J M Gracies, H S Cohen, and W G Ondo. Long-term monitoring of gait in Parkinson's disease. *Gait & posture*, 26(2):200–207, 2007.

[104] M Mortara, G Patane, and M Spagnuolo. From geometric to semantic human body models. *Computers and Graphics*, 30:185–196, 2006.

[105] E A Nadaraya. On estimating regression. *Theory of Probability and its Applications*, 9(1):141–142, 1964.

[106] B Najafi, K Aminian, and A Paraschiv-Ionescu. Ambulatory system for human motion analysis using a kinematic sensor: monitoring of daily physical activity in the elderly. *IEEE Transactions on Biomedical Engineering*, 2003.

[107] K Niazmand, C Jehle, L D'Angelo, and T Lueth. A new washable low-cost garment for everyday fall detection. *International IEEE EMBS Conference*, 2010.

[108] H Ning, T Tan, L Wang, and W Hu. Kinematics-based tracking of human walking in monocular video sequences. *Image and Vision Computing*, 22(5):429–441, 2004.

[109] J F O'Brien, B Bodenheimer, G Brostow, and J Hodgins. Automatic joint parameter estimation from magnetic motion capture data. *Graphics Interface*, 2000.

[110] R O'Dwyer, J P S Cunha, C Vollmar, C Mauerer, B Feddersen, R C Burgess, A Ebner, and S Noachtar. Lateralizing significance of quantitative analysis of head movements before secondary generalization of seizures of patients with temporal lobe epilepsy. *Epilepsia*, 48(3):524–530, 2007.

[111] G Ogris, T Stiefmeier, H Junker, P Lukowicz, and G Troster. Using ultrasonic hand tracking to augment motion analysis based recognition of manipulative gestures. *IEEE International Symposium on Wearable Computers*, pages 152–159, 2005.

[112] R Okada, Y Shirai, and J Miura. Tracking a person with 3D motion by integrating optical flow and depth. *IEEE International Conference on Automatic Face and Gesture Recognition (FG)*, 2000.

[113] R Okada and S Soatto. Relevant feature selection for human pose estimation and localization in cluttered images. *European Conference on Computer Vision (ECCV)*, pages 434–445, 2008.

[114] N Padoy, T Blum, A Ahmadi, H Feussner, M O Berger, and N Navab. Statistical modeling and recognition of surgical workflow. *Medical Image Analysis*, 2010.

[115] N Padoy, D Mateus, D Weinland, M O Berger, and N Navab. Workflow monitoring based on 3d motion features. *IEEE International Conference on Computer Vision Workshops*, pages 585–592, 2009.

[116] O R Pearson, M E Busse, R W van Deursen, and C M Wiles. Quantification of walking mobility in neurological disorders. *Quarterly Journal of Medicine*, 97(8):463, 2004.

[117] C Plagemann, V Ganapathi, and D Koller. Real-time identification and localization of body parts from depth images. *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[118] R Pless. Image spaces and video trajectories: Using isomap to explore video sequences. *IEEE International Conference on Computer Vision (ICCV)*, pages 1433–1440, 2003.

[119] R Pless and I Simon. Using thousands of images of an object. *Computer Vision, Pattern Recognition and Image Processing*, 1:684–687, 2002.

[120] R Pless and R Souvenir. A survey of manifold learning for images. *IPSJ Transactions on Computer Vision and Applications*, 1:83–94, 2009.

[121] PMDTec GmbH. CamCube. `http://www.pmdtec.com`, 2011.

[122] G Pons-Moll, A Baak, T Helten, M Müller, H P Seidel, and B Rosenhahn. Multisensor-fusion for 3d full-body human motion capture. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[123] S Pook, E Lecolinet, G Vaysseix, and E Barillot. Control menus: Execution and control in a single interactor. *ACM Conference on Human Computer Interaction (CHI)*, pages 263–264, 2000.

[124] PrimeSense, Inc. NITE Middleware. `http://www.primesense.com/Nite/`, 2010.

[125] C E Rasmussen and C K I Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

[126] M Rehm, N Bee, and E André. Wave like an egyptian: accelerometer based gesture recognition for culture specific interactions. *Conference on HCI: People and Computers XXII: Culture, Creativity, Interaction*, pages 13–22, 2008.

[127] L Ren, G Shakhnarovich, J Hodgins, H Pfister, and P Viola. Learning silhouette features for control of human motion. *ACM Transactions on Graphics (TOG)*, 24(2):129, 2005.

[128] I Rius, J Gonzalez, J Varona, and F X Roca. Action-specific motion prior for efficient bayesian 3d human body tracking. *Pattern Recognition*, 42(11):2907–2921, 2009.

[129] D Roetenberg, P J Slycke, and P H Veltink. Ambulatory position and orientation tracking fusing magnetic and inertial sensing. *IEEE Transactions on Biomedical Engineering*, 54(4):883–890, 2007.

[130] S T Roweis and L K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(22):2323–2326, 2000.

[131] H M Schepers, E Asseldonk, and P Veltink. Ambulatory estimation of center of mass displacement during walking. *IEEE Transactions on Biomedical Engineering*, 2008.

[132] T Schlömer, B Poppinga, N Henze, and S Boll. Gesture recognition with a Wii controller. *International Conference on Tangible and Embedded interaction*, pages 11–14, 2008.

[133] L A Schwarz. Inertial Sensor and Mocap Dataset. `http://campar.in.tum.de/Chair/ProjectSensorsHumanMotion`, 2011.

[134] L A Schwarz, A Bigdelou, and N Navab. Learning gestures for customizable human-computer interaction in the operating room. *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 129–136, 2011.

[135] L A Schwarz, A Bigdelou, and N Navab. An adaptive solution for intra-operative gesture-based human-machine interaction. *ACM Conference on Intelligent User Interfaces (IUI)*, 2012.

[136] L A Schwarz, D Mateus, V Castaneda, and N Navab. Manifold learning for ToF-based human body tracking and activity recognition. *British Machine Vision Conference (BMVC)* - ***Best Supplementary Material Prize***, 2010.

[137] L A Schwarz, D Mateus, J Lallemand, and N Navab. Tracking planes with time of flight cameras and J-linkage. *IEEE Workshop on Applications of Computer Vision (WACV)*, pages 664–671, 2011.

[138] L A Schwarz, D Mateus, and N Navab. Discriminative human full-body pose estimation from wearable inertial sensor data. *Modelling the Physiological Human (3DPH)*, pages 159–172, 2009.

[139] L A Schwarz, D Mateus, and N Navab. Multiple-activity human body tracking in unconstrained environments. *International Conference on Articulated Motion and Deformable Objects (AMDO)* - ***Best Paper Award***, pages 192–202, 2010.

[140] L A Schwarz, D Mateus, and N Navab. Recognizing multiple human activities and tracking full-body pose in unconstrained environments. *Pattern Recognition*, 45(1):11–23, 2012.

[141] L A Schwarz, A Mkhitaryan, D Mateus, and N Navab. Estimating human 3d pose from time-of-flight images based on geodesic distances and optical flow. *IEEE Conference on Automatic Face and Gesture Recognition* - ***Outstanding Paper Award***, 2011.

[142] L A Schwarz, A Mkhitaryan, D Mateus, and N Navab. Human skeleton tracking from depth data using geodesic distances and optical flow. *Image and Vision Computing*, 30(3):217–226, 2012.

[143] Y Shen and H Foroosh. View-invariant action recognition from point triplets. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1898–1905, 2009.

[144] T Shiratori and J K Hodgins. Accelerometer-based user interfaces for the control of a physically simulated character. *ACM Transactions on Graphics (TOG)*, 27(5):123, 2008.

[145] A Shon, K Grochow, A Hertzmann, and R Rao. Learning shared latent structure for image synthesis and robotic imitation. *Neural Information Processing Systems (NIPS)*, 18:1233, 2006.

[146] J Shotton, A Fitzgibbon, M Cook, T Sharp, M Finocchio, R Moore, A Kipman, and A Blake. Real-time human pose recognition in parts from single depth images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[147] R Slyper and J K Hodgins. Action capture with accelerometers. *ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 193–199, 2008.

[148] C Sminchisescu, A Kanaujia, Z Li, and D Metaxas. Discriminative density propagation for 3d human motion estimation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[149] S Soutschek, J Penne, J Hornegger, and J Kornhuber. 3-d gesture-based scene navigation in medical imaging applications using time-of-flight cameras. *Computer Vision and Pattern Recognition Workshops*, 2008.

[150] R Souvenir and R Pless. Image distance functions for manifold learning. *Image and Vision Computing*, 25(3):365–373, 2007.

[151] J Starck and A Hilton. Free-viewpoint video for interactive character animation. *Symposium on Intelligent Media Integration for Social Information Infrastructure*, 2006.

[152] J Starck and A Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, pages 21–31, 2007.

[153] T Stiefmeier, G Ogris, H Junker, P Lukowicz, and G Troster. Combining motion sensors and ultrasonic hands tracking for continuous activity recognition in a maintenance scenario. *IEEE International Symposium on Wearable Computers*, 2006.

[154] T Stiefmeier, D Roggen, G Ogris, and P Lukowicz. Wearable activity tracking in car manufacturing. *Pervasive Computing*, 7:42–50, 2008.

[155] Y Sun, M Bray, A Thayananthan, B Yuan, and P Torr. Regression-based human motion capture from voxel data. *British Machine Vision Conference (BMVC)*, 2006.

[156] J B Tenenbaum, V de Silva, and J C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(22):2319–2323, 2000.

[157] S C L Terra and R A Metoyer. A performance-based technique for timing keyframe animations. *Graphical Models*, 69(2):89–105, 2007.

[158] A Thayananthan, R Navaratnam, B Stenger, P H S Torr, and R Cipolla. Pose estimation and tracking using multivariate regression. *Pattern Recognition Letters*, 29(9):1302–1310, 2008.

[159] C Theobalt, M Magnor, P Schueler, and H P Seidel. Multi-layer skeleton fitting for online human motion capture. *International Fall Workshop on Vision*, 2002.

[160] M E Tipping. The relevance vector machine. *Neural Information Processing Systems (NIPS)*, 12:652–658, 2000.

[161] Trivisio Prototyping GmbH. Colibri Wireless Sensors. `http://www.trivisio.de`, 2011.

[162] P Turaga, R Chellappa, V Subrahmanian, and O Udrea. Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology*, 2008.

[163] M Urban, P Bajcsy, R Kooper, and J C Lementec. Recognition of arm gestures using multiple orientation sensors: Repeatability assessment. *Intelligent Transportation Systems*, pages 553–558, 2004.

[164] R Urtasun and T Darrell. Sparse probabilistic regression for activity-independent human pose inference. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[165] R Urtasun, D J Fleet, and P Fua. 3d people tracking with gaussian process dynamical models. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[166] R Urtasun, D J Fleet, A Hertzmann, and P Fua. Priors for people tracking from small training sets. *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[167] D Vlasic, R Adelsberger, G Vannucci, and J Barnwell. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics (TOG)*, 2007.

[168] J W von Goethe. Faust I, English translation. `http://www.monologuearchive.com/g/goethe_001.html`, 2011.

[169] C Wachinger and N Navab. Manifold learning for multi-modal image registration. *British Machine Vision Conference (BMVC)*, 2010.

[170] C Wachinger, M Yigitsoy, and N Navab. Manifold learning for image-based breathing gating with application to 4d ultrasound. *Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pages 26–33, 2010.

[171] J P Wachs, H Stern, Y Edan, M Gillam, C Feied, M Smith, and J Handler. A real-time hand gesture interface for medical visualization applications. *Applications of Soft Computing*, pages 153–162, 2006.

[172] J P Wachs, H Stern, Y Edan, M Gillam, C Feied, M Smith, and J Handler. Real-time hand gesture interface for browsing medical images. *International Journal of Intelligent Computing in Medical Sciences and Image Processing*, 1(3):175–185, 2007.

[173] J P Wachs, H Stern, Y Edan, M Gillam, J Handler, C Feied, and M Smith. A gesture-based tool for sterile browsing of radiology images. *Journal of the American Medical Informatics Association*, 15(3):321, 2008.

[174] J M Wang, D J Fleet, and A Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 30(2):283–298, 2008.

[175] C K I Williams. On a connection between kernel PCA and Metric Multidimensional Scaling. *Machine Learning*, 46(3):11–19, 2002.

[176] J O Wobbrock, A D Wilson, and Y Li. Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *Symposium on User Interface Software and Technology (UIST)*, pages 159–168, 2007.

[177] Z Zhang, J Wang, and H Zha. Adaptive manifold learning. *Pattern Analysis and Machine Intelligence (PAMI)*, 34(2):253—265, 2012.

[178] X Zhao, Y Fu, H Ning, Y Liu, and T S Huang. Human pose estimation with regression by fusing multi-view visual information. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(7):957–966, 2010.

[179] X Zhao, H Ning, Y Liu, and T Huang. Discriminative estimation of 3d human pose using Gaussian processes. *International Conference on Pattern Recognition (ICPR)*, 2008.

[180] F Zhou and F de la Torre. Canonical time warping for alignment of human behavior. *Neural Information Processing Systems (NIPS)*, 2009.

[181] H Zhou and H Hu. Human motion tracking for rehabilitation – a survey. *Biomedical Signal Processing and Control*, 3(1), 2008.

[182] H Zhou, T Stone, H Hu, and N Harris. Use of multiple wearable inertial sensors in upper limb motion tracking. *Medical Engineering and Physics*, 30:123–133, 2008.

[183] R Zhu and Z Zhou. A real-time articulated human motion tracking using tri-axis inertial/magnetic sensors package. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(2):295–302, 2004.

[184] Y Zhu, B Dariush, and K Fujimura. Controlled human pose estimation from depth image streams. *Computer Vision and Pattern Recognition Workshops*, 2008.

[185] Y Zhu and K Fujimura. A Bayesian framework for human body pose tracking from depth image sequences. *Sensors*, 10(5):5280–5293, 2010.

[186] K Zürl. ARTtrack & DTrack Benutzerhandbuch. pages 1–78, Jul 2005.