Graz University of Technology

Institute for Computer Graphics and Vision

Dissertation

Human Pose and Shape Estimation from Multi-View Images for Virtual Dressing Rooms

Matthias Straka

Graz, Austria, January 2014

Thesis supervisors

Prof. Dr. Horst Bischof

Prof. Dr. Jürgen Gall

Statutory Declaration

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

Place	Date	Signature

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommene Stellen als solche kenntlich gemacht habe.

Ort	Datum	Unterschrift

ABSTRACT

Estimating the 3D pose and shape of a human body is an essential task for a variety of computer vision applications. A 3D model of a person helps to understand what the person is doing and how the person looks like. Previous work in this area already yields high quality models from a variety of image sources but requires several seconds of processing time. However, fast processing times are essential for interactive applications such as virtual mirrors, virtual dressing rooms and tele-presence systems.

The main focus of this thesis is to generate a 3D model of a moving person while the person is recorded by multiple static cameras. We want to capture and process both the 3D pose and outer surface of a human body at camera frame rate. This task comes with many challenges. For example, how can the system be initialized fully automatically when a new user is present? Furthermore, the estimation of pose and shape must be robust and long-term stable to enable interactive applications.

We propose a novel graph-based pose estimation algorithm that uses multi-view silhouette images of the person as input. This algorithm first estimates a noisy skeleton graph from a volumetric body model. To robustly identify hands and feet in this graph, we apply a novel end-node matching algorithm. Therefore, we assign a body part to each end-node and fit a full skeleton model to the graph. Then, the fitted skeleton is used to initialize a polygonal mesh of a human body. This mesh is adapted to the silhouette outline of the person in the input images using Laplacian mesh editing methods. While state of the art methods cannot take advantage of the skeleton during shape adaptation, we co-optimize shape and skeleton joints to make mesh adaptation more robust. To reduce the overall runtime for shape optimization, we use a novel decoupled constraints solver which operates in a multi-resolution fashion.

Our experiments show that our approach is able to robustly estimate the skeleton pose even in long sequences. Occasional pose errors are mostly caused by ambiguous graph nodes but our algorithm is able to recover the correct pose automatically after a few frames. Subsequent shape and skeleton adaptation shows superior performance to state of the art methods. Especially with a low number of input views, the co-optimized skeleton keeps the mesh adaptation stable and avoids implausible deformations. A run-time evaluation of our algorithms demonstrates that our complete system is able to estimate both human pose and shape at a rate of 15 frames per second.

Due to long processing times, most existing approaches are restricted to estimating the pose and shape of a moving human person in an offline stage. The work presented in this thesis provides a way to perform these computations in real-time. Thus, our work makes it possible to create interactive applications where a user can control his or her own virtual avatar.

Keywords. Human pose estimation, body shape, real-time, virtual dressing room

KURZFASSUNG

Die Schätzung der 3D-Skelettpose und der menschlichen Körperform ist eine wesentliche Aufgabe bei einer Vielzahl von Computer-Vision-Anwendungen. Ein 3D-Modell einer Person hilft zu verstehen, was die Person tut und wie die Person aussieht. Bisherige Arbeiten in diesem Bereich liefern bereits qualitativ hochwertige Modelle aus einer Vielzahl von Bildquellen, erfordern jedoch eine Verarbeitungszeit von mehreren Sekunden. Kurze Verarbeitungszeiten sind für interaktive Anwendungen wie virtuelle Spiegel, virtuelle Umkleidekabinen und Telepräsenz-Systeme allerdings unerlässlich.

Der Schwerpunkt dieser Arbeit ist die Echtzeit-Generierung eines 3D-Modells von einer sich bewegenden Person, welche von mehreren Kameras aufgenommen wird. Daher müssen wir sowohl die 3D-Körperpose als auch die Oberfläche eines menschlichen Körpers mit der Bildrate der Kamera erfassen und verarbeiten. Diese Aufgabe beinhaltet viele Herausforderungen. Beispielsweise stellt sich die Frage, wie dieses System vollständig automatisch initialisiert werden kann, wenn ein neuer Benutzer die Szene betritt. Außerdem muss die Schätzung der Pose und Form robust und langzeitstabil sein, um interaktive Anwendungen zu ermöglichen.

Wir präsentieren einen Graphen-basierten Schätzalgorithmus, welcher Silhouettenbilder der Person aus mehreren Kameras als Eingabe verwendet. Dieser Algorithmus schätzt zunächst einen verrauschten Skelett-Graphen aus einem volumetrischen Körpermodell. Um die Hände und Füße in diesem Graphen robust erkennen zu können, verwenden wir einen neuartigen Algorithmus, der für jeden End-Knoten die Zugehörigkeit zu einem Körperteil bestimmt und anhand des Graphen ein vollständiges Skelett in den Körper eingepasst. Mit diesem Skelett wird ein Polygonnetz eines menschlichen Körpers initialisiert. Das Netz wird im nächsten Verarbeitungsschritt so verformt, dass es optimal mit den Silhouetten der Person übereinstimmt. Im Unterschied zu existierenden Methoden, die während der Formanpassung nicht von dem zuvor bestimmten Skelett profitieren können, erlaubt unsere Methode eine gemeinsame Optimierung der Oberfläche des Netzes und des Skeletts. Dadurch wird die Robustheit gegenüber Bildfehlern erhöht. Um die Gesamtlaufzeit für die Formoptimierung zu reduzieren, verwenden wir einen neuartigen Optimierer, der Kontrollgleichungen einzeln anwenden und auf verschiedenen Netzauflösungen arbeiten kann.

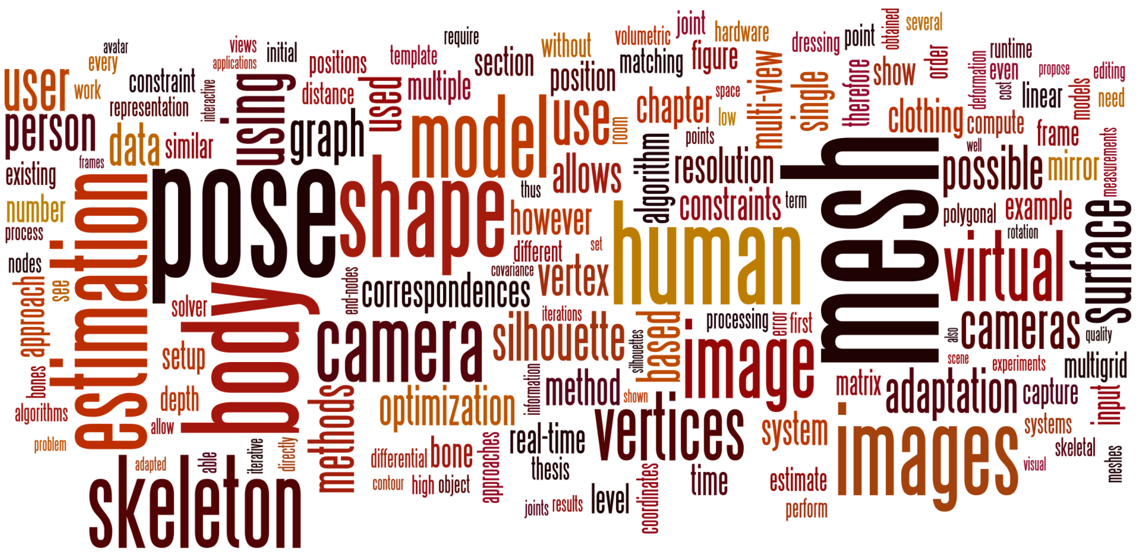
Unsere Experimente zeigen, dass unser Algorithmus das menschliche Skelett auch in langen Sequenzen robust schätzen kann. Gelegentliche Posen-Fehler werden meist durch mehrdeutige Graphenknoten verursacht. Unser Algorithmus ist in der Lage, nach einigen Bildern automatisch zur korrekten Pose zurückzufinden. Die nachfolgende Form- und Skelett-Anpassung zeigt eine überlegene Leistung im Vergleich zu bestehenden Methoden. Vor allem bei einer geringen Anzahl von Kameras ermöglicht die gemeinsame Optimierung von Form und Skelett die Vermeidung von unplausiblen Ergebnissen. Eine Laufzeit-Auswertung unserer Algorithmen zeigt, dass unser System in der Lage ist sowohl die menschliche Pose als auch die Körperform mit einer Rate von 15 Bildern pro Sekunde zu schätzen.

Aufgrund der langen Verarbeitungszeiten beschränken sich die meisten bestehenden Ansätze darauf die Pose und Form einer sich bewegenden Person in einer Offline-Phase zu schätzen. Unser Ansatz bietet eine Möglichkeit diese Berechnungen in Echtzeit durchzuführen. Damit wird eine Reihe von interaktiven Anwendungen ermöglicht, mit denen der Benutzer seinen eigenen virtuellen Avatar steuern kann.

Schlüsselwörter. Menschliche Posenbestimmung, Körperformbestimmung, Echtzeit, Virtuelle Umkleidekabine

THESIS WORD CLOUD

This word cloud shows the 150 most frequently used words in this thesis



Generated using `http://www.wordle.net/create`

ACKNOWLEDGMENTS

I am thankful for the advice and support of my supervisor Prof. Horst Bischof as well as Matthias Rüther. They have given me the freedom to explore possible directions of my research and allowed me to focus on the topics presented in this thesis. In addition, I would like to thank my project team member and co-author Stefan Hauswiesner for the good cooperation and the confluence of Computer Vision and Computer Graphics.

A special 'thank you' goes to the Institute of Vision and Graphics (ICG) and the Graz University of Technology. In particular, I want to mention not only Christian Reinbacher but also all current and former members of the Robot Vision lab. None of this work would have been possible without the funding I received from the Austrian Research Promotion Agency (FFG)[*].

My Graz office was not the only location where I was spending time on this thesis. I want to thank Intel Labs in Santa Clara, CA for a great internship on a topic strongly related to my research area. Altogether, I traveled a distance of about 80% of the earth's circumference on trips related to the writing of this thesis. On these trips, I had the opportunity to interesting discussions with researchers from all over the world.

Last but not least, I want to thank my parents, Margit and Gerhard for their support throughout my educational life. Without my family and friends, this thesis would not have been possible. I am grateful to my fiancé Birgit Hofer for her support and understanding for long office hours, trips and weekends that I spent on my work and not with her. Moreover, she helped me in proof-reading this thesis.

[*]This work was supported by the Austrian Research Promotion Agency (FFG) under the BRIDGE program, project #822702 (NARKISSOS).

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

Contents

1.1 Virtual Garment Try-on

The human body is a fascinating entity that can move freely and has a seemingly infinite variety in shape. People have different gender, age and stature. Moreover, people want to dress in clothing that fits their body shape and looks appealing to them and others. The traditional place for buying apparel is a clothing store. First, a customer browses through the in-store collection and picks clothing that he or she likes. Then, the customer takes the selected pieces of clothing into a dressing room to try them on. Looking into a mirror, the customer decides whether the new clothing fits or not. Online shopping tries to have some share of the market by letting users buy clothing over the internet. While this allows for convenient home shopping, it deprives customers of the ability to try on clothing before buying. Naturally, this leads to customers ordering clothing in different sizes and colors first, but returning the majority of the clothing when they do not fit. These high return rates are one of the major cost factors for online stores today.

Virtual garment try-on is the ideal solution to reduce product return rates. It allows customers to judge beforehand whether clothing will appeal to them without having physical access to the garment. The most realistic virtual try-on (VTO) experience is achieved when a customer sees the selected apparel on his or her own body. Such a virtual representation of oneself is commonly referred to as an *avatar*. In an interactive VTO session, the avatar mimics the pose, shape and texture of the customer (the customer is the user of the system). Existing VTO systems either display non-customized avatars or apply simple techniques to create a custom avatar using camera technologies. The user's avatar is then augmented with virtual clothing. Due to cost and run-time limitations, camera based systems often use a single 2D color camera and/or depth cameras to capture images of the person. However, the lack of complete 3D information limits realism and allows displaying the user's body only from the cameras's point of view. We believe that with today's technology and scientific background it is possible to create a personalized 3D avatar from camera images in real-time.

One goal of this thesis is to improve camera-based techniques for estimating the human pose and shape of the user to create virtual avatars for interactive virtual dressing rooms. Consequently, one requirement for interactive applications is to develop algorithms that allow real-time performance, i.e., it must be possible to estimate the human pose and shape at camera frame rate. Besides virtual try-on, virtual avatars of humans are useful in many situations. The most obvious applications include computer games and the movie industry. For these applications, one wants to capture a real body and use rendering techniques to embed a real person in a virtual environment. A variety of techniques exist that are able to create avatars using camera technologies [116]. It is not sufficient to capture a point cloud of the person but a watertight surface. Only a watertight surface makes rendering the avatars from arbitrary distances possible. Furthermore, it facilitates further processing such as cloth simulation or even 3D printing. The majority of existing approaches for creating virtual avatars adapt an existing body scan to one or multiple camera images. The result is usually a textured 3D polygonal mesh which can be rendered and animated. Many algorithms such as [165] are able to provide movie-grade quality and capture even small details on human bodies. However, such algorithms require up to several minutes of processing time per frame. Therefore, these algorithms are suited only for offline processing.

Fast processing is needed in applications where users interact with their personal avatar in virtual environments. Typical scenarios requiring real-time interaction are 3D telepresence [48, 103], virtual mirrors and virtual dressing rooms. Similar to video conferencing, 3D telepresence allows two or more people to communicate with each other across long distances using both audio and visual images. Usually, there are cameras and monitors at both ends of the connection and data is transferred via a network. While video conferencing merely transfers a 2D video image, 3D telepresence uses partial or full 3D avatars that not only look more realistic but also enable interaction with virtual objects.

A *virtual mirror* [76] is conceptually similar to 3D telepresence but with one significant difference. Instead of communicating with a distant person, a virtual mirror displays a rendered image of the person standing in front of a monitor such that the image looks like an actual mirror image. A virtual mirror no longer requires efficient network transmission across long distances. Instead, it is important to process camera data as fast as possible and to generate visual output on the monitor with minimal delay. The reason for this requirement is that the person in front of the mirror expects the avatar in the mirror image to instantly mimic his or her movements.

Real world mirrors show an image of the person in front of the mirror from a viewpoint that is exactly the mirrored position of the eyes [21]. It is not possible to achieve a realistic virtual mirror effect on a monitor with only one camera because that camera would have to be placed behind the monitor such that the user is occluded. Placing a camera above or on the side of the monitor will show a mirror image from a strange perspective. It will not allow the user to maintain eye contact with his or her mirrored self. When the user wants to look at his or her own eyes on the monitor, an above-monitor camera will inevitably show the user looking down. Thus, the user's mirror image will seem to be avoiding eye contact at all times. In this thesis, we provide a solution that avoids this effect by rendering the avatar from a suitable viewpoint.

As mentioned in the beginning of this section, the most prominent application for a virtual mirror is *virtual try-on* (VTO). VTO is also known as virtual dressing. It is concerned with showing an image of the user and augmenting virtual clothes onto his or her own body. Such a system enables the user to pick items from a catalog of virtual clothing and try them on instantly without the need for undressing first. While it is hard to judge if the clothing will actually fit, the main benefit of virtual dressing is that users are able to quickly try on a large number of clothes and narrow down the selection. This is especially useful in stores where people are queuing for extended times before a traditional dressing room becomes available – which is not unusual in emerging markets such as India and China. Furthermore, a virtual try-on allows for browsing through a large catalog of clothing without physically walking to a store. This possibility to quickly mix and match various garments has the potential to increase sales and reduce return rates.

In Section 1.2, we unfold the conceptional ideas that are essential for understanding the software and hardware requirements of a virtual dressing room. We give a short overview of existing approaches that display clothes on the user's body in Section 1.3. Unfortunately, existing techniques are often limited in terms of visual quality. Some techniques simply augment a single camera image while others do not use personalized avatars at all. We argue that a personalized avatar is essential for the shopping experience because the user sees his or her body wearing the clothes instead of some generic model. In Section 1.4, we analyze the hardware requirements for a system that captures a moving person with cameras. In Section 1.5, we discuss the challenges that arise when one wants to capture a fully personalized avatar that is viewable from all sides. We will outline our contributions for capturing the human shape and pose and elaborate on the their importance for personalizing avatars in real-time. In Section 1.6, we introduce mathematical notations and in Section 1.7, we guide the reader through the remainder of this thesis.

1.2 What is a Virtual Dressing Room?

We explain the term *virtual dressing room* by analyzing the individual words one by one. The word *virtual* implies that parts of the experience inside such a room are not real. In fact, when a *virtual mirror* is used to reflect what is happening inside a virtual dressing room, purely computer generated content is presented to the user. One goal of this thesis is to create the impression that user sees his or her own virtual mirror image. Thus, a virtual avatar should look as similar to the real person as possible. This is in contrast to systems that display a model of a generic person [115, 162].

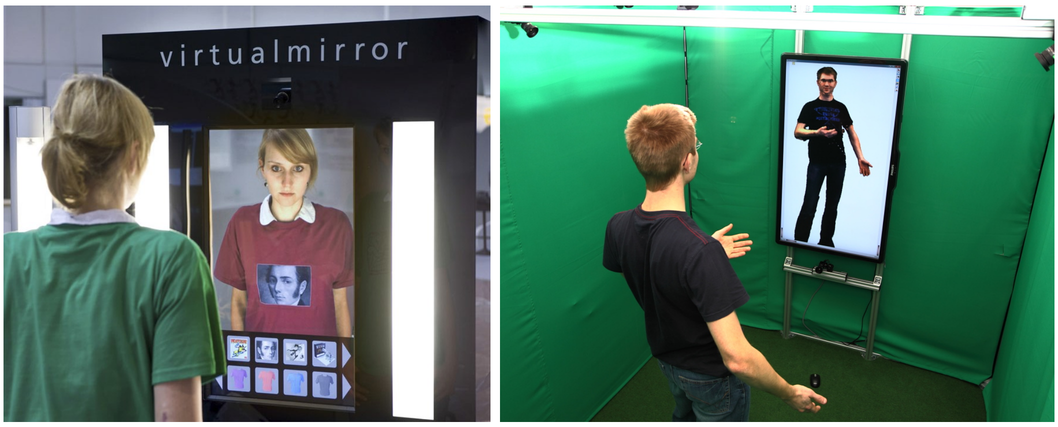
The word *dressing* refers to the ability of a virtual dressing room to showcase clothing worn by the user's avatar. Similar to the avatar, clothing is only virtual and may not even exist in reality. A natural requirement for such a showcase is that the avatar needs to have the possibility to interact with garment. Without interactions, garment would be either floating in mid-air or falling down to the floor. Therefore, the avatar cannot be a simple 2D image of the user. It needs to be a virtual model that can interact with virtual garment and possibly other virtual items. For a more realistic experience, the avatar should be able to move either by scripted motion or allow direct control by the user (*e.g.* mimic the exact pose of the user). Ideally, the shape and pose of the user is known at all times to allow for accurate control of the virtual avatar.

Finally, a virtual dressing room contains the word *room* which signifies some spatial restrictions. This need not be a physical room. For example, the room can be a limited area that is within a camera frustum such that the user can be seen. Typically, a virtual dressing room is constructed such that the space where a user can be recognized by the system is directly in front of the monitor of a virtual mirror. Thus, the user is encouraged to maintain a position in front of the monitor, which facilitates the layout of camera positions. There are virtual dressing rooms where the virtual room coincides with a physical room surrounded by walls. The reasons for being in an enclosed room are either protection of privacy or that the user is placed within a controlled environment (*e.g.* a wall color that facilitates the segmentation of the user and the background).

The term *virtual mirror* is used for a device that mimics an actual mirror. It consists of cameras and a monitor to show a mirrored image of the user (see Fig. 1.1). Ideally, this image coincides with the image that a real mirror would show if mounted instead of the monitor. There is, however, no restriction on the content. Displaying a modified representation of the user is possible and often desirable. The main purpose of a virtual dressing room is to augment the image of the user with virtual clothing. Therefore, the virtual mirror is a natural display concept for a virtual dressing room.

1.3 Current State of Virtual Try-Ons

In recent years, the concept of the virtual dressing room has inspired both the scientific community as well as commercial companies. Two conceptually distinct types of virtual



(a) Fraunhofer Virtualmirror (see [76]) (b) Our virtual mirror solution

Figure 1.1: A virtual mirror is a combination of a monitor and one or multiple cameras that generates an artificial mirror image of the user.

dressing rooms can be identified. First, there exist methods that achieve almost photo realistic quality using purely virtual avatars and clothing. As a consequence, the avatar does not look like the actual user and graphical content is typically generated in an offline stage. Second, there are approaches that augment camera images of a user with virtual clothing. These approaches are capable of running at camera frame-rates and allow displaying a personalized avatar that looks like the user.

The first type of approaches uses an existing virtual avatar and dresses him or her in scanned or designed virtual clothing. The major advantage is the complete control over the resulting output. Since all possible parameters such as viewpoint, dresses and even possible motions are known during design time, complex rendering or pre-computation algorithms can be used to maximize visual quality. Examples for such commercial products include Intel's *Magic Mirror* [162] or the *Virtual.Fitting.Room* [115] by PhiSix fashion labs. The main target market is online marketing on websites or personal computing devices. However, the major disadvantage is that such approaches currently do not support on-the-fly customizations of the avatar. It is difficult to make the avatar look like the user. In fact, many solutions use a fixed set of avatars to display clothing, which is similar to traditional showcasing of clothing using human models. There exist some solutions to customize the avatar. For example, BodyPal [25] allows the user to enter his or her body measurements into the system to create a roughly customized avatar.

In contrast, camera-based virtual try-on systems allow full user customization. Most systems capture a single camera image of the user and overlay it with a 3D apparel model. Photo realism can already be achieved for simple head-wearable gadgets such as eye-wear. For example, *TOMS* [153] provides virtual try-on for eye-glasses that can be executed in a normal web browser. While rigid gadgets are easy to attach to a mostly rigid human head, the simulation of clothing on a moving body is a significantly harder task due to the large space of possible movements.

A simple way to simulate clothing is to visually modify existing clothing through retexturing. For example, Hilsmann et al. [76, 77] present a system which can track a T-shirt worn by the user and modify its color and texture. Retexturing existing clothing eliminates both the need for simulating the physical behavior of clothing and making it fit to a human body. The approach by Hilsmann et al. produces realistic results but does not allow to change the shape and size of the clothing.

In order to completely exchange the clothing worn by a user, virtual clothing is needed. Such clothing has to be adapted to the shape and pose of the user. Representatives in this area are the *EON Interactive Mirror* [60] and *FittingReality* [51]. These systems use a Microsoft® Kinect™ camera to track the body of the users during arbitrary movement and overlay virtual garment. This creates the impression that the user is actually wearing virtual clothing. However, using Microsoft® Kinect™ based pose tracking only is not reliable enough and often deviates from the actual pose. The resulting effect is that clothing seems to float in front of the body in a detached manner. In addition, it is difficult to automatically match the size of clothing with the size of the user. Such effects lead to the conclusion that camera-based virtual try-on can be improved by using a better model of the user. In [167], Yuan et al. compare three variants of a Kinect™ based virtual try-on system: (1) showing virtual clothes on a body-size adapted virtual avatar, (2) augmenting virtual clothes on a camera image of the user and (3) customizing only the face of the avatar to resemble the face of the user. They conducted a user study that reveals that people prefer to see their camera image augmented with clothing instead of a virtual avatar. This leads to the conclusion that users do want to see their own body image in virtual try-on systems.

Body measurements are an important aspect for virtual fitting rooms. In contrast to VTO systems, a fitting room simulation determines whether clothing will fit the user or not. Simple solutions such as *UPcload* [155] allow users to obtain rough body measurements using a simple webcam. This is sufficient for predicting the rough size of clothing. More advanced systems such as $(TC)^2$ [148] allow to obtain accurate body scans and measurements using active scanning methods such as structured light. In recent years, the Microsoft® Kinect™ camera has enabled measuring body shape at the user's home [24, 163]. Such body scanners are capable of creating a customized avatar for a user. The customized avatar can be used to fit virtual clothing or to improve the visual quality of VTO systems. However, specialized body scanning hardware will not allow random movement of the user or enable real-time interaction with a VTO system.

In today's systems, there is a significant gap between high quality visual output, true interactive user personalization and accurate body measurements. Photo realistic methods use advanced rendering and physics simulations but are very limited in terms of user personalization. Virtual try-on systems achieve authentic results when they are limited

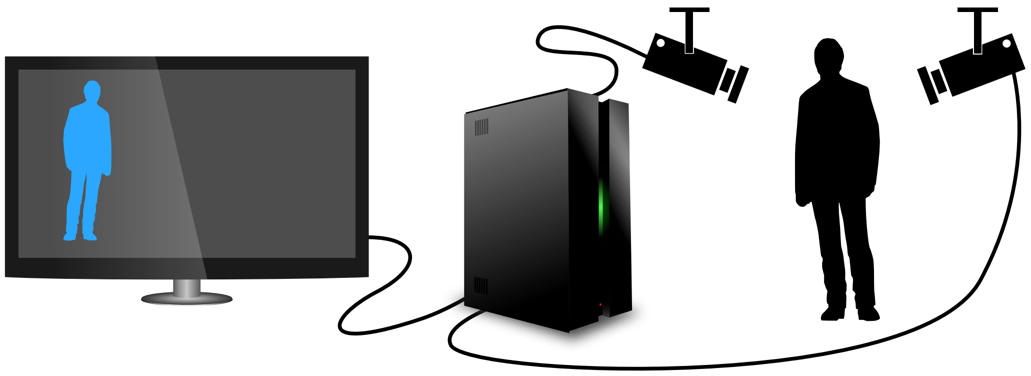


Figure 1.2: Overview of hardware used in a virtual dressing room installation.

to rigid objects such as eye-glasses. Real-time full body VTO has become feasible on cheap hardware in recent years. However, results often look unnatural due to inaccurate body tracking possibilities. In contrast, sophisticated hardware can only be used for body scanning and does not allow interactive virtual try-on. In this thesis, we address some of these issues in order to enable a more realistic interactive VTO experience.

1.4 Hardware Requirements for Virtual Dressing Rooms

In order to build a virtual dressing room which uses a virtual mirror, several hardware components are required. In Fig. 1.2, we show a person captured by multiple cameras, a computer that processes image data and a monitor which shows the mirror image of the user and possibly virtual content. This section discusses hardware requirements for interactive installations.

One goal of this thesis is to generate a 3D model of the user that is viewable from an arbitrary viewpoint. In order to capture such a virtual avatar of the user, we propose to use multiple calibrated color cameras. Existing multi-camera setups often require a dedicated full size room to mount the cameras [57, 130]. A virtual dressing room should require significantly less space with a footprint of only a few square meters. The reason is that the setup will be placed in a store environment where physical space is expensive. This means that cameras will be mounted around a limited space in front of the monitor where the user is allowed to move. In such a setup, it is not possible that every camera sees the whole body of the user at all times. A location closer to the user's body has the advantage that the images have a better resolution of the body even though the number of recorded pixels can be kept relatively low. However, a larger number of cameras are needed to record all parts of the body when the person moves.

The algorithms required for a virtual dressing room include processing of camera images, interpreting the 3D human shape and pose of the user and rendering visual output. Ideally, the computer system needs to be able to execute all of these algorithms in real-time (i.e. at camera frame rate). There are two possible choices of how to connect the cameras to a computer system:

1. A multi-computer architecture consists of several dedicated slave computers and a single master computer [1, 136]. Each slave computer pre-processes images recorded from at most a few cameras in parallel. A master computer collects the pre-processed images from the slaves and combines their results for 3D pose and shape estimation as well as for rendering.

2. Alternatively, a single computer can be used for both recording and pre-processing camera images as well as extracting 3D information [140]. In contrast to systems that use separate capture servers and a centralized rendering server, a single computer system eliminates communication latencies between machines. However, it requires sufficient computational power to process all camera images.

We propose to connect all cameras to a single computer. Thus, we minimize the time between capturing the image of the user and displaying a virtual mirror image. A thorough description of hardware components and implementation details can be found in Section 5.2.

An important design decision is concerned with how users interact with the system. Traditional user inputs include keyboard, mouse or touch interfaces. In a virtual dressing room where a user can move freely, these user input devices are not suitable. Instead, we propose to use a gesture based user interface that uses the existing cameras. By determining the 3D positions of the user's hands, an intuitive user interface can be built. For example, the user can trigger actions such as cloth selection or viewpoint changes by touching virtual objects in space.

1.5 Challenges and Contributions

In Section 1.3, we have discussed the state of the art in virtual dressing and virtual fitting rooms. Currently, there exists no system which supports both real-time virtual try-on and body scanning at the same time. One reason for this shortcoming is that there is an inherent difficulty to perform fast human shape and pose estimation. However, body shape estimation is essential for simulating virtual garment and an accurate body pose is important for both user interaction as well as for aiding shape estimation.

Current systems face a variety of challenges. Accurate shape and pose estimation requires expensive hardware and long processing times. Real-time operation requirements usually steer design choices towards a simple camera setup, which allows only pose tracking and the animation or adaptation of an existing body model. Therefore, hybrid methods often separate the body scanning part (shape estimation) from animation and tracking (pose estimation). For example, Gall et al. [57] propose to scan a person using a laser scanner first. Then, they use a multi-view camera setup to track the human pose and adapt the shape of the scanned model according to the images. However, their pose and shape adaptation requires several seconds per frame, which does not allow for real-time operation.

Another challenge is to make both pose and shape estimation robust towards distractions in images. Long term stability is crucial for an interactive system. This means that pose or shape estimation must not get stuck in an erroneous state but needs to be able to automatically recover.

In this thesis, we tackle the problem of real-time human pose and shape estimation on a single hardware setup. We try to answer the following two questions:

- How can we quickly estimate the unknown skeleton pose of a human body without manual initialization?

- How can we quickly estimate the body shape of a person given an initial pose?

Our goal is to develop efficient methods to capture a moving human person in order to obtain a complete 3D model. In our experimental hardware setup, the person is allowed to move freely inside a small room and should be able to see a virtual mirror image of himself or herself on a screen with minimum delay. Even users without technical knowledge should be able to use our system without supervision. Therefore, we require that all algorithms work without manual initializations and without an initial pose. To fulfill the above mentioned requirements, we have made the following contributions in this thesis:

- We propose a novel **human pose estimation** algorithm that operates on volumetric data. The main idea of the algorithm is to compress dense volumetric data into a graph structure which can be filtered and processed efficiently. Our pose estimation approach is capable of real-time operation and automatic initialization.

- In addition to the user's pose, we **capture the 3D body shape**. Therefore, we adapt an existing model of the human body to the shape of the user. In contrast to related methods that are able to optimize only the surface of the model, we derive a linear model that simultaneously adapts both the shape and skeletal pose. While existing approaches discard skeletal information during shape adaptation, our approach uses the skeletal structure of the body during optimization. This is essential for robustness when adapting the 3D body shape from multiple cameras. In addition, the skeleton can be used to efficiently handle surface rotations during linear shape adaptation, which further improves the quality.

- Finally, we show how to perform **shape estimation at fast frame rates**. Our optimizer shares properties with real-time physics simulations that allow fast processing of large polygonal meshes. Thus, we achieve a high quality mesh adaptation at camera frame rate.

These contributions make it possible to display a textured polygonal model of the user on a screen in real-time. Alternatively, the model can be stored for subsequent offline animation and rendering. Moreover, a polygonal model can be used to improve virtual dressing rooms. As mentioned in Section 1.3, virtual clothing overlaid on images of a human body often appears floating when the true body shape is not known. When a full 3D body model is adapted to camera images, we can accurately place virtual clothing on the body. Moreover, the 3D model allows performing basic body measurements without an additional body scan.

1.6 Mathematical Notations

Throughout this thesis, we use a common mathematical notation. Scalar values are depicted using italic font, *e.g.*, a or x. Vectors are written in bold font and generally interpreted as column vectors, *e.g.*, $\mathbf{b} = [x, y, z]^T$. Similarly, matrices are written in capital bold letters such as $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_2]$. To denote vector spaces, we use blackboard bold characters such as $\mathbb{R}^2$, which denotes the two-dimensional Euclidean space, for example. Unless otherwise noted, we address elements of a matrix using indices in the form $\mathbf{C}_{r,c}$, where r is the row index and c the column index. The r-th row of a matrix is addressed as $\mathbf{C}_r$.

We define the number of elements of a vector or set as $|.|$. For example, let $\mathbf{v} = [x, y, z]^T$ then $|\mathbf{v}| = 3$. This notation shall not be confused with the norm of a vector computed such as $\|\mathbf{v}\|_2$, which computes the Euclidean norm of vector $\mathbf{v}$.

Linear vector functions are written using calligraphic font. For example, a linear function $\mathcal{F}(\mathbf{v})$ that takes a vector $\mathbf{v}$ as input may be defined as $\mathcal{F}(\mathbf{v}) = \langle \mathbf{v}, \mathbf{v} \rangle$. $\langle ., . \rangle$ signifies the vector dot product. A set $\mathscr{S} = \{a, \mathbf{b}, \mathbf{C}\}$ can contain elements of different types such as scalars, vectors and matrices.

1.7 Thesis Outline

This thesis is organized as follows. In Chapter 2, we review existing work of human pose and shape estimation from images and give an overview of suitable camera technologies. We present our novel graph based pose estimation algorithm in Chapter 3. In Chapter 4, we present our method for body shape estimation as well as a real-time capable solver. Both pose and shape estimation methods are evaluated in Chapter 5. Finally, we give a summary of our approaches and conclude the thesis in Chapter 6.

CHAPTER 2 RELATED WORK

Contents

Reconstructing and tracking the human body has been an active research topic for decades. In recent years, there has been significant progress in capturing a human body using computer vision technologies. As a consequence, there exist a variety of algorithms for human pose and shape estimation [106, 116]. This chapter gives an overview of the scientific developments in this area.

We start with a general introduction on existing camera technologies such as multi-view setups and active depth sensing camera hardware. A short overview of the projective geometry explains the essential relationships found in multi-camera setups. After this introduction to cameras and their mathematical relationships, we focus on algorithms for human pose and shape estimation. We discuss existing work in the area of human skeleton pose estimation from different image sources. At the end of this chapter, we present existing approaches to capture the moving surface of the full human body and evaluate their potentials for interactive applications.

2.1 Image Based 3D Sensing

One of the most important considerations for capturing a 3D model of the human body is the choice of the imaging sensor. For example, there exist single and multi-camera setups

(a) SwissRanger 4000 TOF (b) Microsoft® Kinect™ (c) Point Grey® Flea™2

Figure 2.1: A selection of passive and active cameras for capturing a scene.

as well as active cameras. This section gives an overview of existing camera technologies. We explain the main concepts behind each technology and discuss a possible use for capturing a human body. The goal is to identify a camera technology that is capable of capturing a metric 3D representation of a human body in real-time.

2.1.1 Monocular Cameras

A monocular camera system can be built using only a single image sensor and an optical lens. Such a camera captures images from a single point of view under perspective projection. In general, a single static camera cannot be used to capture 3D data. The limiting factor is the missing information about scale [133]. A full sized human body can produce the same perspective image as a significantly smaller puppet standing closer to the camera. Given a reference object such as a previously known marker, it is possible to obtain the 3D pose and scale of a plane from a single image. This allows for simple 3D body measurements using a single camera [155] but not for capturing full 3D information. Hilsmann and Eisert [76] use a single camera to augment the camera image with new information without the need for explicit 3D information.

2.1.2 Multi-View Systems

When more than one camera is used to capture the same scene from different view-points, this setup is called a stereo setup for two cameras or a multi-view setup for more than two cameras, respectively. Using more than one view-point has a major advantage over a single camera system: it is possible to derive scale and depth from the scene if the camera geometries and orientation between the cameras are known. This information can be obtained through camera calibration, which needs to be performed only once for any rigid multi-camera setup. Another important requirement for deriving depth data is that all cameras capture the same scene at the same time. This means that the object of interest must not move or deform until each camera has captured an image. Usually, this

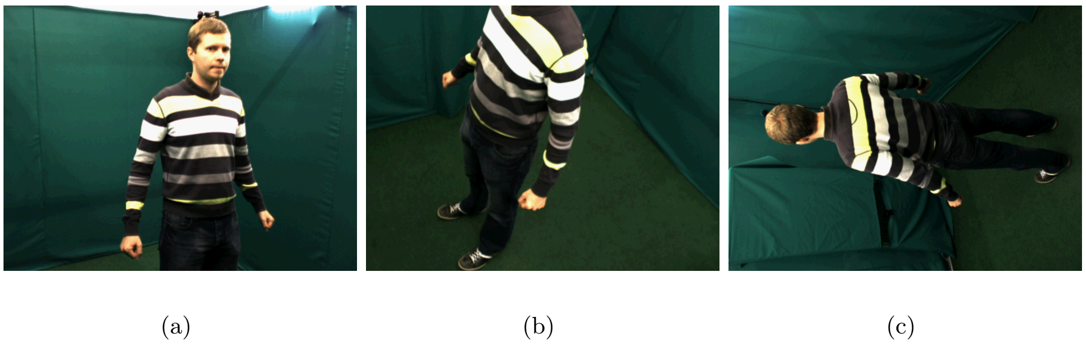


Figure 2.2: Three synchronized images from a multi-view setup showing the same person from multiple angles.

requirement can be enforced by using synchronized cameras or fixed objects. A typical example for multi-view images of a person is shown in Fig. 2.2.

A fast method to obtain 3D data from multi-view images is to process silhouette images. A silhouette is the background segmented binary image of an object. Given a set of multi-view silhouette images, it is possible to compute the 3D visual hull of the object. The visual hull is the closest convex approximation of an object that can be obtained with a volume intersection approach [53, 93]. There exist several approaches to compute the human pose [16, 42, 78] or human shape [4, 57, 158] from silhouette images.

In order to obtain depth information from a multi-view setup, it is possible to exploit epipolar geometry [68]. There exist a variety of algorithms that efficiently compute dense depth maps from multi-view images [122, 124, 151]. Such depth maps can be used to improve a surface estimation of the human body [136, 159]. However, computing dense depth maps for multi-view images is still too slow for real-time applications even when parallel computing platforms are used.

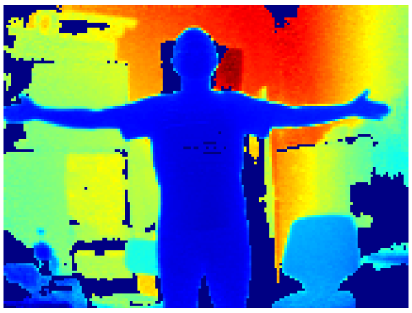
Point clouds are a sparse representation of a 3D object. They describe the surface of the object through a collection of points. Typically, such points are obtained by triangulating the 3D position of key-points from two or more input images [3]. A sparse point cloud can be densified around key points in subsequent steps [54].

2.1.3 Active Depth Cameras

Active cameras are more than just image capturing devices. They use a projector to project visible or invisible light into a scene. A camera captures the pattern and depth information can be derived directly by analyzing how the projected light is altered by the scene. The result is a dense 3D point cloud or a depth map of objects in the scene. However, a point cloud only contains points visible for the camera and not the backside of the object. Examples of images obtained with active depth cameras can be seen in Fig. 2.3.

(a) Microsoft Kinect

(b) SR4000 TOF


(c) SoftKinetic TOF

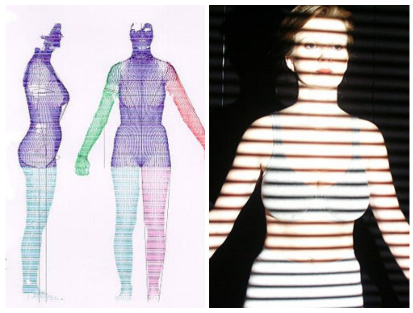

(d) TC2 Structured Light [148] ©Popularmechanics

Figure 2.3: Depth images obtained from different active depth cameras.

Classical structured light systems project multiple patterns in rapid succession. A synchronized camera captures every projected pattern (see Fig. 2.3(d)). By combining these patterns, a code map can be generated which allows stereo matching algorithms to create a depth map without existing image features (such as distinct edges). A laser scanner sweeps one or multiple laser lines across the object while capturing the scene. Depth information can be obtained by analyzing the deformation of the laser line for every sweep step. The laser scanner yields the most accurate measurements but is very slow compared to other active scanning methods. A major disadvantage of sequentially projecting patterns or laser lines is that movement of the object during scanning can produce artifacts or destroy the scan completely.

Modern cameras such as the Microsoft® Kinect™ (see Fig. 2.1(b)) use a single static infra-red projection with a coded dot pattern. Such patterns allow to uniquely identify a neighboring set of dots in a decoding stage, which makes depth computations efficient and robust towards errors. A single pattern has the major advantage that depth information can be obtained from a single camera frame and movement of the object does not influence the scan quality. However, depth image resolution is sacrificed because a single pattern code requires multiple projector pixels. Thus, the obtainable depth image has only a fraction of the resolution of the camera.

A third type of active cameras does not use any structured pattern but measures the time for light to travel from a light source via the object to the sensor. Examples for such cameras are the MESA SwissRanger 4000 (see Fig. 2.1(a)) and the recently released Creative® Senz3D™ camera (based on SoftKinetic). These cameras work with the time-of-flight (TOF) principle which derives depth information from the time related phase shift of a modulated light source [67]. With this method, it is possible to compute a depth value for every camera pixel. Since all depth values are computed at the same time using the same modulated light source, capturing a moving object is possible. However, TOF cameras are either expensive or have a low resolution and produce noisy images. We show some examples for the depth image obtained with TOF cameras in Fig. 2.3 (b) and (c).

One major disadvantage of active cameras is that only one camera can capture the same scene at once. Each camera has to project a pattern into the scene. Consequently, two or more cameras projecting their pattern onto the same object will cause interferences. There exist methods to avoid interference or minimize their effect. For example, time slicing or random shaking of static patterns [32] enables the use of multiple cameras and their projected patterns in the same scene. However, these methods can decrease the quality of the result or increase the capture time. For TOF cameras, it is possible to use several modulation frequencies in order to operate them concurrently.

2.1.4 Summary

A single passive camera has been identified as insufficient for capturing a 3D representation of an object. Active cameras are the ideal choice for capturing 2½D data from a single view-point. However, problems arise when multiple active cameras are used in the same scene which is necessary when a 3D model needs to be captured from all sides simultaneously. Therefore, the most suitable choice for real-time 3D reconstruction of the human body is a combination of multiple passive cameras that are synchronized. In theory, a virtually unlimited number of such cameras can be used in parallel to capture the same scene. The main reason for limiting the number of cameras is the cost and bandwidth factor. In a real-time system, the images of all cameras need to be transferred, processed and combined in a short amount of time. In current literature, it is common to use a number of four [130] to sixteen [1] cameras for real-time 3D reconstruction.

2.2 Projective Geometry

Projective geometry describes the mapping of a three-dimensional scenery to two-dimensional images. This section focuses on the basic notations and the relationship between real-world objects and their perspective images, which are captured by pinhole cameras. For a more detailed description on the concepts behind projective geometry, the reader is referred to [68] and [133].

2.2.1 Homogeneous Coordinate Representation

One of the most important definitions of projective geometry is that of homogeneous coordinates. Homogeneous coordinates are points in a d-dimensional projective space $\mathbb{P}^d$ which are represented by a $(d+1)$ element vector. The projective space is a quotient space where the following equivalence relation holds:

$$\forall \alpha \neq 0 : [\hat{x}_1, \ldots, \hat{x}_{d+1}]^T = [\alpha\hat{x}_1, \ldots, \alpha\hat{x}_{d+1}]^T. \tag{2.1}$$

The homogeneous coordinate $\hat{\mathbf{x}}$ of a non-homogeneous point $\mathbf{x} \in \mathbb{R}^d$ is obtained by a simple mapping from $\mathbb{R}^d$ into $\mathbb{P}^d$:

$$[x_1, \ldots, x_d]^T \to [x_1, \ldots, x_d, 1]^T = [\hat{x}_1, \ldots, \hat{x}_d, \hat{x}_{d+1}]^T. \tag{2.2}$$

A homogeneous point $\hat{\mathbf{x}} \in \mathbb{P}^d$ can be converted into a non-homogenous point by dividing its first d elements by the $(d+1)$-th element:

$$[x_1, \ldots, x_d]^T = \left[\frac{\hat{x}_1}{\hat{x}_{d+1}}, \ldots, \frac{\hat{x}_d}{\hat{x}_{d+1}}\right]^T. \tag{2.3}$$

2.2.2 Perspective Cameras Model

A perspective camera can be used to project a three-dimensional point from object space to the two-dimensional image plane. Using homogeneous coordinates, the projection of a point $\mathbf{x} \in \mathbb{R}^3$ to pixel coordinates $\mathbf{y} \in \mathbb{R}^2$ becomes a linear operation. Only the mapping from homogeneous coordinates to the Euclidean space requires a division. The following equation demonstrates the projection of point $\mathbf{x}$ to pixel $\mathbf{y}$ using the projection matrix $\mathbf{P}$:

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \mathbf{P} \cdot \mathbf{x} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \qquad \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} \hat{y}_1/\hat{y}_3 \\ \hat{y}_2/\hat{y}_3 \end{bmatrix} \tag{2.4}$$

The projection matrix $\mathbf{P}$ encodes the intrinsic and extrinsic parameters of the camera. $\mathbf{P}$ can be computed as:

$$\mathbf{P} = \mathbf{K}[\mathbf{R}| - \mathbf{R}\mathbf{t}] \tag{2.5}$$

where $\mathbf{K} \in \mathbb{R}^{3\times3}$ is an upper triangular matrix containing intrinsic camera parameters, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix which encodes the orientation of the camera ($\mathbf{R}^{-1} = \mathbf{R}^T$ and $|\mathbf{R}| = 1$) and $\mathbf{t} \in \mathbb{R}^3$ is the position of the camera center. The intrinsic camera matrix

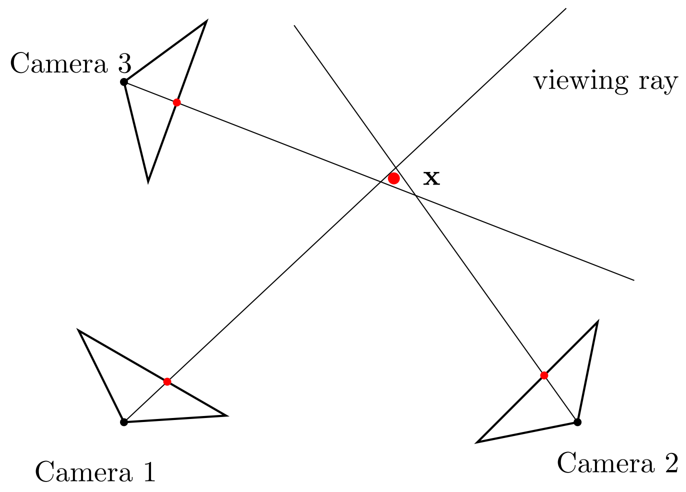


Figure 2.4: Elementary multi-view geometry. The scene point $\mathbf{x}$ is projected into all cameras. It can be fully reconstructed by intersecting the viewing rays in least squares sense.

$\mathbf{K}$ has the following form:

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_x \\ 0 & f_y & u_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

where f_x and f_y represent the focal length of the lens, s is the degree of shear for slanted pixels and u_x and u_y are the projection offsets on the image plane. A real-world optical system consists of a non-optimal lens which produces non-linear distortions in radial and tangential direction. Such distortions can be modeled and corrected using Brown's model [31].

The intrinsic parameters of a camera are estimated using a camera calibration method such as [29]. This method first records multiple images of a known target. Then, it optimizes the intrinsic parameters of the camera such that the reprojection error is minimized in all images.

2.2.3 Multi-View Geometry

From a single camera, it is not possible to reconstruct the 3D position of a projected point because multiple 3D points may project onto the same 2D image location. Multi-view geometry is concerned with methods that extract three dimensional information from two or more cameras.

Given a set of cameras and their projection matrices $\mathbf{P}_\ell$, it is possible to compute the 3D world position of scene points. This process is called triangulation. Triangulation requires known 2D image positions $\mathbf{y}_\ell$ of the 3D world point $\mathbf{x}$ in at least two camera views. The position of such points can be found through searching in image space. When the image position is already known in one camera, the search space in other cameras can be reduced by exploiting the epipolar relationships between two cameras [68]. From each pixel, one can compute a 3D viewing ray which connects the camera center, the pixel on the image plane and the 3D point. Thus, the true 3D location of the unknown point $\mathbf{x}$ must lie somewhere on that ray. Mathematically, this relationship can be expressed as:

$$\begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix} = \hat{y}_3 \begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} = \mathbf{P} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} = \mathbf{P}\hat{\mathbf{x}}. \tag{2.7}$$

This relationship leads to the following two linear equations for each correspondence:

$$\begin{aligned} \mathbf{P}_3\hat{\mathbf{x}}y_1 - \mathbf{P}_1\hat{\mathbf{x}} &= 0 \\ \mathbf{P}_3\hat{\mathbf{x}}y_2 - \mathbf{P}_2\hat{\mathbf{x}} &= 0 \end{aligned} \tag{2.8}$$

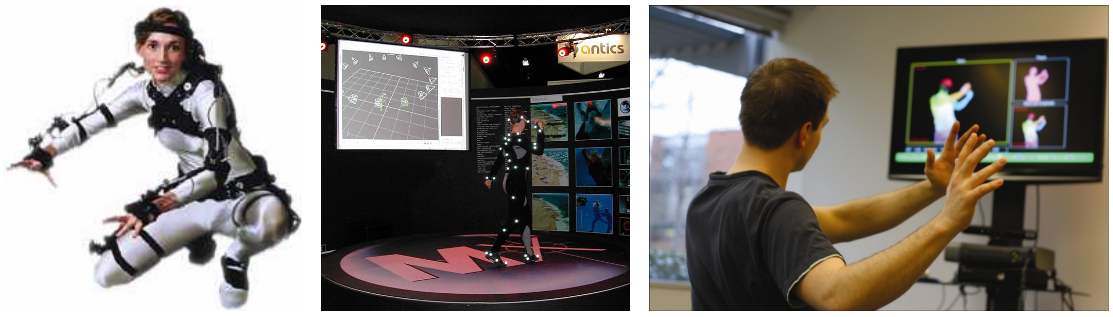
where $\mathbf{P}_n$ denotes the n-th row of the projection matrix.

Due to noisy image observations and noisy projection matrices, the exact location of the point $\mathbf{x}$ cannot be found. Therefore, triangulation of the 3D location can be formulated as a least squares problem to find the location for $\mathbf{x}$ that is closest to all viewing rays (see Fig. 2.4). Through Equation (2.8), each image correspondence $\mathbf{x} \leftrightarrow \mathbf{y}_\ell$ contributes two linear constraints and a solution can be obtained through SVD decomposition.

2.3 Human Pose Estimation

Human pose estimation is concerned with finding the pose parameters of a human body model that fits best to the observations in one or more input images. It is a vivid topic in the current literature [106, 116] due to its wide-spread applications such as motion-capture, telepresence or object manipulation in virtual environments. In addition, when the pose of the user is known, the user can interact with a computer system using only gestures.

While marker-based systems are already available in numerous commercial applications, marker-less pose estimation is still a challenging research topic. There exists a variety of algorithms that solve this task with high accuracy from multiple input images [56, 79] or even a single photograph [65, 75]. Unfortunately, these systems often require manual initialization and cannot process camera images at interactive frame rates. Promising methods for interactive human pose estimation use a volumetric model of the body [78, 102, 154] or utilize a depth camera based on the time-of-flight principle [58] or the Microsoft® Kinect™ camera [127, 128]. This section explains the basic definitions of human pose estimation and gives an overview of existing technologies and algorithms.



(a) Motion Capture Suit ©MetaMotion (b) IR markers ©T-tus (c) Without markers ©E&T Magazine

Figure 2.5: Different technologies for estimating the human pose.

2.3.1 Overview of Human Pose Estimation Methods

There exists a variety of approaches to estimate the pose of a human body [13]. Early approaches use mechanical systems such as magnetic sensors or armatures to measure joint angles [61, 152]. These systems require an expensive hardware setup and the movement of the captured person is limited by cables or limb-mounted hardware (see Fig. 2.5(a)). Modern pose estimation systems are based on optical sensors. Compared to mechanical systems, optical systems either require only small markers attached to the body of the captured person (marker-based systems) or can perform pose estimation without any markers (marker-less systems).

Marker-based systems use either infra red (IR) reflective points (passive) or LED lights (active) as markers [36, 157]. The 3D position of each marker is determined by triangulating its position using two or more cameras. Usually, such cameras run at high frame rates above 100 Hz to facilitate the task of marker tracking. A typical marker-based system can be seen in Fig. 2.5(b). The advantage of marker-based systems is the high reliability of pose estimation. Even at high frame rates, the processing time is low which allows for real-time operation. A major drawback is the time required for placing markers and configuring the system for each subject. In addition, markers alter the body surface and do not allow the subject to wear natural clothing.

Recent research on marker-less optical human pose estimation systems addresses the aforementioned problems [105, 106, 116]. Marker-less pose estimation methods do not rely on markers and can be used by people in natural clothing. This benefit comes at the cost of a higher computational effort and higher error rates due to possible ambiguities in camera images. Marker-less systems can be categorized into pose estimation from monocular 2D images, monocular depth images or multi-view images. Furthermore, they can be categorized into single-shot pose estimation (without prior initialization) and tracking methods that need to know the pose from a previous frame to compute the current pose.

2.3.2 Marker-Less Human Pose Estimation

This thesis is concerned with estimating the shape and pose of people without markers. Therefore, we review existing work in marker-less human pose estimation in more detail. While there are many possible ways to categorize marker-less human pose estimation algorithms, we will categorize them according to the type of input images.

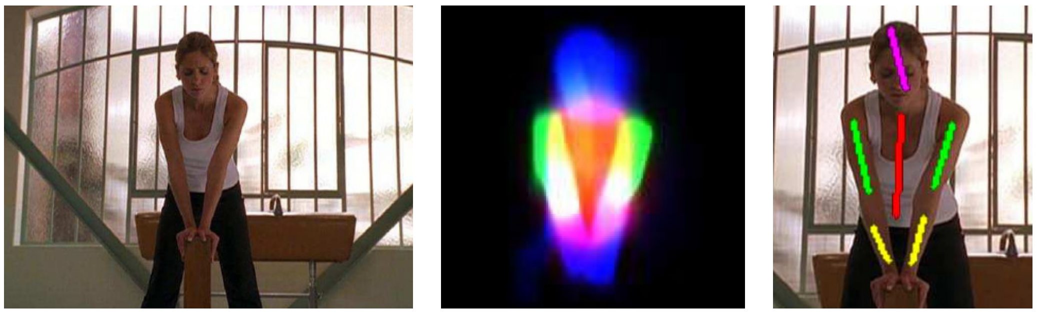
Independently from the type of input images, there are two main schools of thought for human pose estimation [116]: generative (model-based) and estimation-based models. *Generative approaches* use an explicit body model that is driven by pose parameters. Given some pose parameters, a generative model generates a representation of the pose (i.e. rendered body image) and compare this image to the input images [57, 129, 131]. This allows for global or local optimization by minimizing some measured error between the generative model and image. However, optimizing for the correct pose is not trivial due to the high dimensionality of the problem and a good initialization is important. In contrast, *estimation-based approaches* analyze the input image(s) to produce hypotheses about the pose. In other words, they learn a mapping from image observations to the underlying 3D pose [2, 127]. This implies that such methods require training data which contains a good coverage of the possible pose space. The combination of generative and discriminative approaches, a so called *hybrid approach*, tries to improve pose estimation by using the advantages from both schools of thought. First, a discriminative method is used to initialize the pose. Afterwards, a generative method is used to refine the pose until a chosen cost function is optimal [11, 121].

2.3.2.1 Monocular Pose Estimation

Monocular articulated pose estimation is concerned with estimating pose of body and limbs from one single 2D image. Such methods support processing images captured with even the simplest camera. The inherent problem with this task is the ambiguity of 3D pose from monocular image measurements (i.e. the same image can be explained by multiple poses). Nevertheless, impressive algorithms on this topic have been published in recent years [75].

One promising line of research is based on pictorial structures [8, 49, 50], which follow the discriminative approach. For example, Eichner et al. [45] use an object detector that is trained to recognize body parts. They assign to each pixel in the image a probability that encodes the likelihood of that pixel belonging to each body part. These likelihoods form the unary potentials of a probabilistic model. Parts are tied together by pairwise potentials that encode spatial relations (*e.g.* an arm attached to the upper body). By finding the maximum a posteriori probability (MAP) solution in the probabilistic model, the most likely pose of the upper body in the image can be estimated. This procedure is illustrated in Fig. 2.6. A similar method is used for detecting and tracking the full body pose of people in street scenes [9].

The approach by Agarwal and Triggs [2] uses a relevance vector machine (RVM) to



(a) Input color image (b) Part probabilities (c) Pose estimate

Figure 2.6: Ferrari et al. [50] show how to estimate the human pose from a single color image through part detectors.

learn the mapping from monocular silhouettes to pose parameters directly. Their sparse Bayesian nonlinear regression method allows them to distill a large training database into a single compact model with good generalization to unseen examples.

Jaeggli et al. [83] combine a generative approach with statistical learning to estimate the pose of a walking person. Therefore, they train a low dimensional manifold using a database of example images. A pose estimate for a new image always represents a valid pose, even under heavy occlusion. They allow multiple hypotheses about the current activity (*e.g.* running or walking) and perform smooth activity switching to handle a variety of motions.

Pure generative approaches such as by Sidenbladh et al. [129] often use particle filters to generate a discrete set of pose parameters. Using the underlying body model, each particle instance can be rendered and compared to the input image(s). Point-to-point correspondences or edge-to-edge correspondences are commonly used to derive a measure for an error that is then minimized.

Although monocular pose estimation methods have been shown to work in practice, they are often constrained to front-facing poses or limited motion patterns. Furthermore, they are not suited for interactive operation due to their heavy computational burden. However, many concepts from monocular pose estimation can be found in human pose estimation from multiple views or depth maps.

2.3.2.2 Pose Estimation from Multi-View Images

In contrast to monocular images, a set of multi-view images shows the same scene at (usually) the same time from different viewpoints. The major advantage of such a recording setup is that occluded body parts in one view are likely to be visible in another view. In addition, image information is redundant and 3D data can be obtained directly via triangulation of two or more 2D points or similar image features. Alternatively, the human body can be represented by a volumetric model which directly encodes 3D information. The disadvantages of a multi-view setup are higher hardware costs (multiple cameras), camera synchronization and calibration issues. A surprising fact is that the processing time tends to decrease as the number of cameras is increased. The reason for this is that simpler algorithms can be used to estimate the pose. In addition, pre-processing of multiple views can be easily parallelized [92]. More viewpoints allow for fewer ambiguous poses, thus algorithms save time by an early rejection of implausible poses.

One class of multi-view pose estimation algorithms is based on shape-from-silhouette reconstruction. These algorithms require a clean separation between background and foreground (the silhouette of the person). Such input data is easy to obtain when a multi-view camera setup is located in a studio environment that facilitates background segmentation. A multi-view silhouette representation of input data enables the use of efficient volume-based approaches for pose estimation [154]. Volumetric pose estimation methods are often based on a visual hull representation [93] of the human body. Therefore, the human body is discretized into voxels using space carving [91, 92] or converted to a polygonal representation [52].

A volumetric model of the human body allows one to extract the skeleton graph through bottom-up methods. Many approaches use a variant of the medial axis transform introduced by Blum [23] for 2D objects. In 3D, the skeleton of an object is defined as the locus of centers of maximal inscribed balls [41]. This representation can also be referred to as the center-lines of the body. Typical ways to compute the graph of the skeleton are thinning, geometric methods, distance fields or potential fields [41]. Essentially, the skeleton is an undirected acyclic graph with connected nodes that define the topology of the human body.

Some early approaches for graph-based pose estimation use a single 2D silhouette image. Thome et al. [149] use 2D silhouette images to generate a skeletal graph through Delaunay triangulation and assign limb labels to the end-nodes through graph matching. Menier et al. [104] state that using only a single 2D silhouette is prone to occlusion and that a 3D skeleton cannot be mapped to a 2D skeleton graph directly. Instead, they use multiple silhouette images to generate the visual hull of the body, extract medial axis points using Voronoi centers and fit a skeleton model using a probabilistic framework. A hybrid 2D/3D approach is presented by Correa et al. [42] who fit a skeleton in multiple 2D silhouettes and combine them in a 3D fusion step.

Given a volumetric representation of the human body, graph-based approaches allow more efficient and accurate pose estimation. In Sundaresan and Chellappa [145], the volume consisting of thousands of voxels is projected onto a low dimensional manifold with six or less dimensions (see Fig. 2.7). The authors use Laplacian Eigenmaps of the voxel neighborhood graph to transform limbs into smooth 1D curves in the embedded space. This representation resembles the Reeb graph of the human body and limbs can be identified reliably. However, computation times for this type of manifold embedding are not suitable for real-time operations. A parallel algorithm for 3D skeletonization through

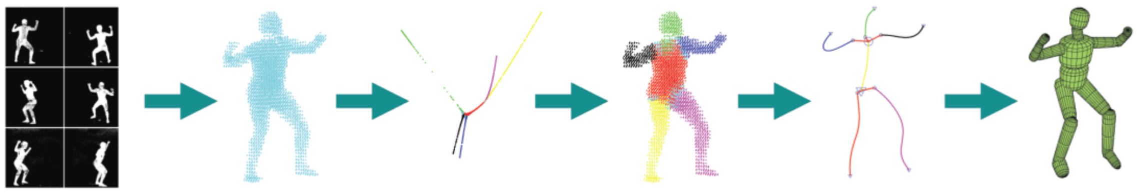


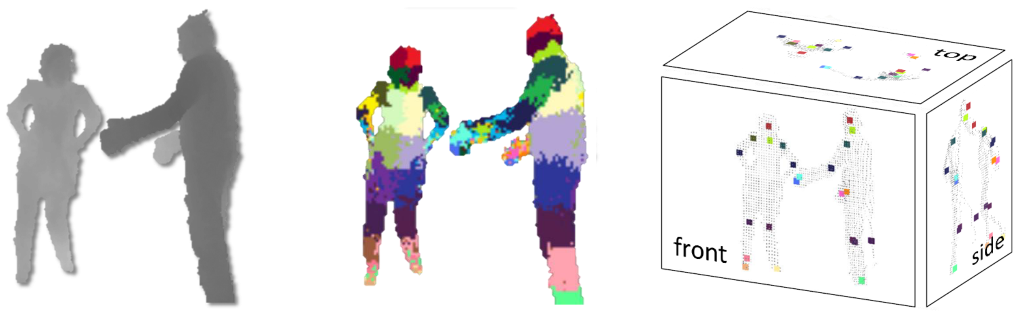
Figure 2.7: Sundaresan and Chellappa [145] use a volumetric approach to extract the human pose from multi-view silhouettes (image taken from [145]).

voxel thinning has been presented by Bakken and Elisassen [14]. It consists of simple execution kernels that fully exploit the GPU processing architecture.

There are natural limitations for approaches based on the visual hull. Situations where limbs are close to the body can lead to a poor pose estimation accuracy because the extracted skeleton becomes degenerate. This problem can be circumvented by using exemplar-based approaches such as in [62, 78, 120]. For example, Hirai et al. [78] have proposed to estimate the 3D skeleton pose in real-time using volumetric regression. Since voxel-based volumetric models generally require a huge amount of data, a naïve regression method would be computationally slow. Therefore, Hirai et al. propose a fast and stable volume tracking method with an efficient volume representation in a low-dimensional dynamical model. Van den Bergh et al. [156] use Haar-lets to recognize poses from a discrete volume. However, the space of possible human poses is large which makes it difficult to train an efficient regression model that captures all possible poses.

Similar to monocular pose estimation, generative approaches can be applied to multi-view silhouette data. One major benefit of using multi-view images over monocular views is that both the local and global optimization of the pose parameters converge faster to the correct pose. For example, Gall et al. [57] use a polygonal model of a person and locally optimize the overlap between silhouette contours and the model. To overcome local optima, all misaligned limbs are then globally optimized using a particle filter [56]. Even though these filters can be optimized in a way that very few particles are needed, computation can take several seconds per frame.

Research on multi-view pose estimation has shown that it is possible to either obtain real-time pose estimates or very accurate results by using more than one camera. However, accurate pose estimation is only possible when the background permits a reliable segmentation of the person. In real-world scenarios, automatic segmentation of persons in front of cluttered background is difficult and obtaining accurate silhouettes of a person may not be possible. For some scenes, it is possible to support an automatic segmentation algorithm with a previously estimated pose. This pose can be used to perform a guided segmentation [65, 69]. Alternatively, there exist methods that estimate the articulated pose by tracking SIFT features [57] or by using a *Sums of Gaussians*-model [139] with simple color information to describe the body of the person. The work of Wu et al. [165] estimates the scene illumination to perform pose estimation under varying illumination.



(a) Input depth image (b) Classified body parts (c) Joint positions

Figure 2.8: Shotton et al. [127] show how to estimate the human pose from depth images through learned body part classification (images taken from [127]).

2.3.2.3 Pose Estimation from Depth Images

In recent years, new camera technologies have been developed that have revolutionized how human pose estimation can be performed using only a single camera. These types of cameras are commonly known as depth-sensing cameras such as time-of-flight cameras and the Microsoft® Kinect$^{\text{TM}}$. These cameras not only capture a scene at fast frame rates but also compute the depth for each pixel, the so called depth map. Using a depth map, the user can be segmented from the background with minimal computational effort. Furthermore, depth information helps to disambiguate between poses that look similar under perspective projection.

Discriminative pose estimation methods are among the most successful techniques for pose estimation at real-time frame rates. For example, Shotton et al. [127] use randomized decision forests to assign a body-part label to every pixel of a depth map of a human body (see Fig. 2.8). Even without temporal constraints, this method estimates the accurate position of skeleton joints at high frame rates. During the training stage, this method requires a massive database of previously labeled depth maps, which contains images of persons of various age and size in different poses. In Sun et al. [144], a similar regression method is combined with a global latent variable that encodes body height or torso orientation. This drastically improves the accuracy of body joint localization.

Other approaches have shown that pose estimation from depth images is possible using generative models [11, 58]. One problem of generative methods is finding the correct correspondences between the image and the model. This problem is solved in an elegant way by Taylor et al. [147] who directly infer correspondences from depth maps or multi-view silhouettes using a regression forest. Using these correspondences, pose estimation can be performed in a single one-shot optimization step.

Method	**Cost**	**Speed**	**Accuracy**	**Summary**
Mechanical	$\geq$ \$10K	$\leq$ 300 Hz	1 mm	wearable hardware and long setup per user
Reflective markers	$\geq$ \$10K	$\leq$ 1000 Hz	1 mm	light markers but long setup per user
Single-View	$<$ \$100	$<$ 1 Hz	50 mm	minimal camera hardware requirements but high computational cost
Multi-View	\$10K	$\leq$ 30 Hz	50 mm	require a static background and camera calibration; usually large input data
Depth sensors	$<$ \$1K	$\leq$ 30 Hz	100 mm	easy use and install but limited to camera facing poses

Table 2.1: Comparison of existing human motion capture approaches. We compare the approximate cost for the required hardware, execution speed in frames per second and the skeleton joint estimation accuracy.

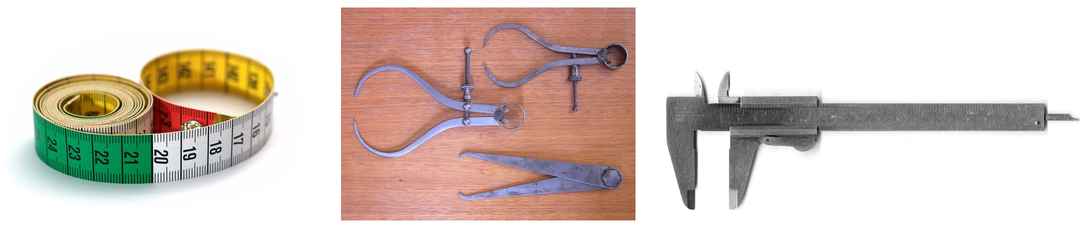
2.3.3 Comparison

To summarize existing methods for human pose estimation and motion capture, we compare the hardware costs, execution speed as well as accuracy of different strategies in Table 2.1. It can be seen that mechanical and marker-based systems have a high accuracy and sample the current pose at up to 1 000 Hz. However, they have high hardware costs and require a substantial setup time per user such as mounting and calibrating markers.

In contrast, marker-less optical systems estimate the human pose without substantial setup times. Depending on the type of input image(s), some algorithms require several seconds to process a single frame but other algorithms achieve real-time performance. Depth sensors are particularly suited for fast pose estimation but have a low accuracy due to low camera resolution and higher noise levels. Typically, depth maps allow pose estimation only when the user faces the camera.

2.4 Human Body Shape Estimation

Shape estimation systems for the human body either aim to capture all relevant measurements that are distinct for the individual subject or create a full 3D model (avatar). Manual body measurements are performed by taking defined measurements on certain body locations. Automated, camera-based shape estimation methods usually perform a dense reconstruction of the body. This section gives an overview of the history of body scanning and measurements, explains related optical technologies and analyzes their potential for creating virtual avatars interactively.



(a) Tape measure (b) Calipers (c) Sliding compass

Figure 2.9: Measurement instruments for manual body measurements (images are released under CC license).

2.4.1 History of Anthropometry

For hundreds of years, measuring the human body has been of great importance to tailors when designing clothing that fits. In the last century, body measurements have gained importance in many other areas such as automobile design, work site ergonomics, equipment design as well as in sizing surveys as an indicator of health status [132].

Traditional methods to obtain anthropometric measurements include non-invasive instruments such as a measuring tape, calipers and the sliding compass. A selection of such tools can be seen in Fig. 2.9. Usually, a human tailor has to locate landmarks such as the hips or waistline on the subject and apply one of the measurement tools to obtain values for length or girth.

One of the biggest problems of anthropometry is the accuracy and repeatability of measurements. When measurements on the same person are performed by multiple observers, the measured values can differ considerably due to imprecisions of landmark locations, subject positioning and measurement instrument application [20]. Even though there exist standards such as ISO 8559:1989 [82] that define how to measure at certain body locations, the true measurement process still allows for much variation. A virtual model of the body helps to perform such measurements automatically.

The first fully automatic anthropometric measurement systems were based on silhouettes that were acquired by taking pictures of a human body against a distinct background [84]. Since then, researchers and companies have developed promising technologies that automatically capture the shape of a human body. However, an optical measurement system faces further challenges. First and foremost, an optical system can only reconstruct and measure the human body including all worn clothing. Even though there exist algorithms that estimate the human shape underneath clothing [69], these methods rely on regularization through learned knowledge and can therefore not accurately reconstruct an unseen person. Therefore, a subject usually is required to wear form-fitting garment that does not alter the real shape of the body.

Existing methods for human shape estimation can be split into two main groups:

1. Model-free approaches reconstruct the shape directly from data available in images. They allow the highest degree of details but do not guarantee that the result has the correct topology of a human body.

2. Model-based approaches require a previously scanned or otherwise generated model of the human body. These approaches adapt this model to image data by modifying its parameters. The adaptation of an existing model ensures that the result will always be a valid human body.

In the remainder of this section, we discuss existing methods that perform human shape estimation with a model-free approach or by using an explicit model.

2.4.2 Model-free Body Shape Estimation

A model-free shape estimation method is able to compute the 3D surface of a human body from multiple images. As the name suggest, such methods require no prior model of a human body. Instead, they reconstruct every detail directly from images. We give a short overview of possible methods to compute the shape of previously unknown persons.

Volumetric Scans: A volumetric body scan classifies a predefined volume into space occupied by the person and background. The most simple volumetric representation is the visual hull [93]. It is the closest approximation of an object that can be obtained with the volume intersection approach. As input, a visual hull method uses multi-view silhouette images of a person. Space carving is a simple method to compute a regular three-dimensional grid of empty and occupied voxels [91]. Alternatively, it is possible to compute a polygonal mesh by intersecting silhouette outlines [52, 53].

If the volumetric description of the body is directly inferred from camera images, special care has to be taken to handle outliers correctly. Artifacts are common with space carving from silhouette images. If the number of views is limited, ambiguities cannot be resolved correctly [125]. This leads to ghosting artifacts, which are structures that perfectly explain the image observations but do not exist on the real person. A possible solution to this problem is to employ photo consistency terms such as in [47, 91]. These methods analyze the color information across multiple cameras to find the correct surface of the object. An additional benefit of a photo consistency term is that concave regions can be reconstructed. This is not possible with silhouettes alone.

It is possible to convert a volumetric representation into a surface model. A discrete voxel representation can be converted into a polygonal surface using the marching cubes algorithm [100]. A volume described by tetrahedra is equivalent to a surface representation if only outer triangle faces are stored [4].

Surface Scans: A point cloud is a simple representation of the surface of an object and consists of a collection of unconnected 3D points. Each point has a 3D location and may have an orientation. Point clouds are a typical output of laser scanners, structured light systems [148] or multi-view stereo systems [55]. A collection of points is not well suited for capturing the shape of a human body because they are unstructured. However, a point cloud can be an intermediate representation for fitting a more advanced model to image data [10, 70].

Instead of point clouds, it is possible to represent the surface through a polygonal mesh. Usually, an initial mesh is obtained through a volumetric method or from a meshed point cloud. Starck and Hilton [136] first compute the visual hull of the body as the upper-bound of the person. Then, they use a dense reconstruction method that fuses silhouette, feature, and stereo cues to reconstruct a smooth surface of the body. Furukawa and Ponce [55] apply the Poisson Surface Reconstruction method [89] on oriented point clouds to obtain a surface mesh. Depth sensing cameras (see Section 2.1.3) allow creating a new mesh by merging depth images from multiple views [96].

Implicit Surfaces: The Microsoft® Kinect™ has not only revolutionized human pose estimation (see Section 2.3.2.3) but opens new possibilities for capturing surfaces of 3D objects. The *KinectFusion* paper by Newcombe et al. [110] has presented an efficient method to integrate multiple depth maps such that the surface of a static object can be reconstructed by a moving Kinect™ camera. The object surface is encoded in an implicit model as the zero-crossing of a signed distance function. A similar method to encode a surface implicitly has been published by Graber et al. [63] who compute depth maps from passive cameras. For human shape estimation, implicit surface models are only partially suitable because they require the scanned person to maintain a static pose during the entire scan. In addition, the cameras must be moved by a second person or robot.

2.4.3 Model-based Body Shape Estimation

A widely used approach to estimate the shape of a human body is to adapt an existing human model to one or multiple images of a person [7, 39]. Therefore, such a model is an essential pre-requisite for most human shape estimation methods. There exist many representations of the human body that are suitable for analyzing the shape of a person. Complex representations have a high degree of freedom and allow accurate and detailed modeling. However, adaptation to images can be hard and over-fitting is not uncommon. Simpler body models do not allow as much variation but can be fit to image data with higher robustness. However, the latter representation can not recover details that are distinct to every person.

Surface Models: A surface representation of a 3D object is a 2D manifold embedded into 3D space. Usually, this manifold is sampled in regular or irregular intervals and represented by a polygonal mesh which consists of vertices and faces. Typically, an existing polygonal mesh can be adapted to image data by moving individual vertices [4, 57, 158]. Usually, a smooth deformation of the surface is ensured through the Laplacian Mesh Editing (LME) framework [28]. The optimal shape can be obtained by optimizing a linear system of equations that constrain some vertices according to image correspondences while the overall mesh is held together by smoothness constraints. However, it may be challenging to find the correct correspondences between vertices and pixels.

There exist two types of model-based surface estimation methods. Those that use an underlying skeleton and those that do not. Aguiar et al. [4] argue that a skeleton limits the application of marker-less motion capture to humanoid models and loose clothing cannot be handled realistically. They propose to adapt a human body mesh to multi-view images by first deforming a low-resolution volumetric mesh to capture the pose. Then, they refine the detailed shape using a coupled high resolution surface mesh. In contrast, Cagniart et al. [34] decompose a mesh into larger surface patches. This increases robustness to noisy data and allows them to handle arbitrary objects.

Using a skeleton allows the explicit encoding of prior information about the deformable object. For example, a skeleton can be used to initialize the mesh prior to deformation. Gall et al. [57] and Vlasic et al. [158] adapt the shape and pose of a mesh in a two step algorithm. First, they determine the skeleton pose without adapting the shape of the surface. Then, they non-rigidly refine the surface of the mesh to fit to image data. While Vlasic et al. [158] propose to optimize the skeleton and polygonal surface independently from each other, Gall et al. [57] use mesh vertices rigged to the skeleton to estimate the pose of the model given the image data. This approach has been extended to work with multiple people if each person can be segmented independently [99]. All above methods use the skeleton for initialization of the mesh but not during shape adaptation itself.

Recent developments in the mesh editing community are motivated by the desire to perform modeling as rigid as possible (ARAP). In [95, 169], a skeleton is embedded in a deformable mesh and surface vertices are connected to evenly sampled points along the bones using a least squares formulation. Such a binding helps to preserve the volume of the mesh while the globally optimal deformation is obtained through solving of a linear system of equations. However, the presented methods can only handle cross-sectional connections between skeleton joints and surface vertices. Thus, artificial joints need to be added to achieve realistic results, which makes these methods only suitable for animation but not for shape adaptation.

Parametric Models: The dimensionality of surface models that consist of individual vertices can be very large. Each vertex has three degrees of freedom for movement. A common technique to reduce the degrees of freedom is to use parametric body models. For example, Alcoverro et al. [5] parametrize each bone of the skeleton with two additional parameters, axial and radial deformation, and optimize the shape using a particle filter. Thus, the number of degrees of freedoms is proportional to the number of bones. However, changing the shape through transformations of an underlying skeleton is not suitable to fully capture shape variations due to muscle deformations, age and gender.

Therefore, many methods compress high-dimensional mesh data into a low dimensional subspace using the principal component analysis (PCA) method [10]. For example, a compact representation could contain dimensions for body size, weight and gender. In order to obtain a model that generalizes well, a large database of aligned body scans is required for training. The database should cover all possible variations in shape, size, weight, age and gender. There exist datasets such as SCAPE [10], the MPI dataset [71] as well as commercial products such as from the Civilian American and European Surface Anthropometry Resource (CAESAR) project [33]. Very often, a parametric body model is used for reconstructing a human body from image data [65, 69, 87, 163] or even reshaping of bodies [118, 170].

Weiss et al. [163] describe how to build a body scanner for home use using a single Kinect$^{\text{TM}}$ camera. In order to obtain a scan from all sides of the body, the user is required to record multiple images from different viewpoints. Using a parametric model, Weiss et al. are able to optimize a single shape under varying pose.

There exist methods to estimate the human shape from a single image. Given an initial skeleton estimate, Guan et al. [65] iteratively fit a parametric body model to image contours of a human person. Due to unknown scale, the height and gender need to be supplied by an operator. The problem of scale has been addressed by a company called UPcload [155]. They process images taken by an ordinary webcam and relate all measurements to the image of a compact disk (CD), which has a known size. The most important benefit of a monocular system is easy setup and calibration. However, there are several disadvantages such as lack of scale and missing direct 3D data. Therefore, researchers and developers of commercial systems hardly ever rely on measurements from a single 2D image.

2.4.4 Comparison

In Table 2.2, we compare different shape models and capturing technologies for their suitability to capture a human body. There is no single preferred choice for an acquisition technique nor for the most suitable model. For example, parametric models are robust towards outliers in input data and even allow reconstructing the skin of clothed people [70]. On the other hand, building a parametric model that generalizes to all possible bodies is nearly impossible. Therefore, the choice of the right data representation strongly depends on the application.

In a similar way, the capturing hardware either allows for fast capture times (*e.g.* with a multi-view setup) or requires several seconds during which the user is supposed to limit body movements to a minimum (*e.g.* structured light or moving camera). Processing time may be an issue for scanning a body. Even if the complete body is captured within a second, it may take minutes before a useful scan is available.

Method	**Cost**	**Time**	**Accuracy**	**Summary**
Volumetric	low	$< 1\,\text{s}$	low - med.	limited resolution but easy and fast to compute
Model-free surface	low	seconds	high	usually highly accurate but slow to compute
Implicit surface	low	seconds	medium	multiple views of a moving camera are combined into one model; require static subject
Model-based surface	low	seconds	med. - high	adapt an existing surface to images but may overfit; finding correct correspondences can be difficult
Parametric	high	seconds	high	few parameters to adapt but require a large training database

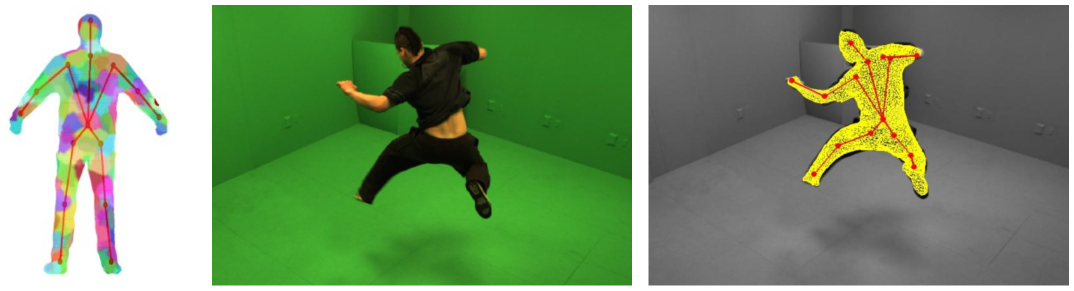
Table 2.2: Comparison of the approximate cost, scanning time and the accuracy of existing human shape estimation approaches.

In an unsupervised setting such as a virtual dressing room, the focus of human shape estimation should be on maximum robustness towards outliers. Therefore, we propose to use a model-based approach which adapts an existing model of a person to image data. An existing model further facilitates anthropometric measurements after adaptation because measurement points need to be defined for the model only once.

2.5 Combined Human Pose and Shape Estimation

The majority of the previously mentioned approaches for human shape estimation require that the rough pose of the person is known before the shape can be estimated. The reason for this is that an existing model of the human body needs to be initialized before adaptation. However, after the initialization of the shape model, the pose is often ignored during the adaptation process [57, 158]. Optimizing pose and shape jointly has several advantages. For example, a simultaneous optimization better handles outliers and erroneous observations which are frequently found in multi-view data [81]. In addition, pose estimation can be made more robust if the shape fits to the actual body. In this section, we analyze approaches that are able to optimize both shape and pose of a human body simultaneously.

Droeschel and Behnke [44] propose a method to adapt both the pose and some parametric shape parameters of an adaptive body model to depth images using an iterative closest point (ICP) algorithm. Hofmann and Gavrila [79] attempt to optimize both pose and shape of a human model by batch processing a set of automatically selected multi-view frames. This offline method requires initial pose estimates before starting the shape estimation framework. The optimized shape is then used to refine the pose. Taylor et al. [147]



(a) Patched mesh and skeleton

(b) Input view with colored background

(c) Estimated body pose and shape

Figure 2.10: Multi-View systems such as shown in Huang et al. [81] use a colored background to segment the user and obtain silhouettes. The pose and shape of the user can be estimated by fitting a model onto the data such that the overlap between model and silhouettes is maximized in all views (images taken from [81]).

improve the commonly needed iterative correspondence search between mesh vertices and image data. They train a regression function that can predict correspondence between image data and vertices directly. This allows them to perform one-shot pose estimation and facilitates the adaptation of existing models to the data.

In Huang et al. [81], probabilistic surface deformation [34] is combined with the bone binding energy presented in this thesis. While probabilistic surface deformation makes soft assignments between the model and the observations, the bone binding energy guides the skeleton fitting. The benefit is that unreliable observations in multi-view data can be handled more robustly. We show a typical result of their work in Fig. 2.10.

2.6 Conclusion

The previous sections have given an overview of the wide range of methods for image-based human pose and shape estimation. However, none of the previously mentioned approaches is able to estimate the pose and shape of the human body at interactive frame rates. There has been significant progress towards real-time pose estimation using either multi-view image data [15] or depth images [127]. But shape estimation still requires several seconds per frame.

Our method is closely related to [57] and [158] as we follow their two-stage approach with separate pose estimation and shape adaptation based on the LME framework. The major difference compared to previous methods is that we do not discard the skeleton during shape adaptation. Instead, we propose a method to jointly optimize both shape and skeleton pose in a common framework. Our approach does not require intermediate skeleton joints such as [95, 169]. We use an unmodified skeleton definition for both pose and shape estimation.

In addition, we employ a real-time capable solver for optimizing the deformable shape. Our method is inspired by position-based physics simulations [109] which are able to compute realistic interactions between soft bodies in real-time. The key to real-time operation is to apply decoupled constraints on individual vertices of a deformable mesh and to optimize for a stable shape using an iterative multi-grid method. We show that this decoupled optimization is suitable for mesh deformation guided by image-space correspondences such that the final mesh resembles the content in the input images.

CHAPTER 3 GRAPH-BASED HUMAN BODY POSE ESTIMATION

Contents

3.1 Introduction

Human pose estimation is concerned with finding the 3D pose parameters of a human body model such that the model fits to the observations in one or more input images. There exists a variety of algorithms that solve this task with high accuracy from multiple input images [56, 79] or even a single photograph [65, 75]. Promising methods for interactive human pose estimation use a volumetric model of the body [78, 102, 154] or utilize a depth camera based on the time-of-flight principle [58] or the newly released Microsoft® Kinect™ camera [127, 147].

In this chapter, we address the problem of real-time human pose estimation from multi-view silhouette images. We propose a volumetric approach which first computes a discretized 3D representation of the actor and then estimates the skeletal pose using an efficient graph-based method. The skeletal graph is a tree that mimics the topology of the human body (i.e. it comprises arms, legs and torso). It can be efficiently estimated from a volumetric body representation using center-line tracing algorithms [119]. As the center-line extraction produces spurious branches, we have developed a novel pose-independent graph matching algorithm. This algorithm robustly classifies end-nodes of the graph into head, hands and feet while detecting end-nodes that do not correspond to any valid limb. Using these correspondences, we obtain a good initialization for fitting a human skeleton

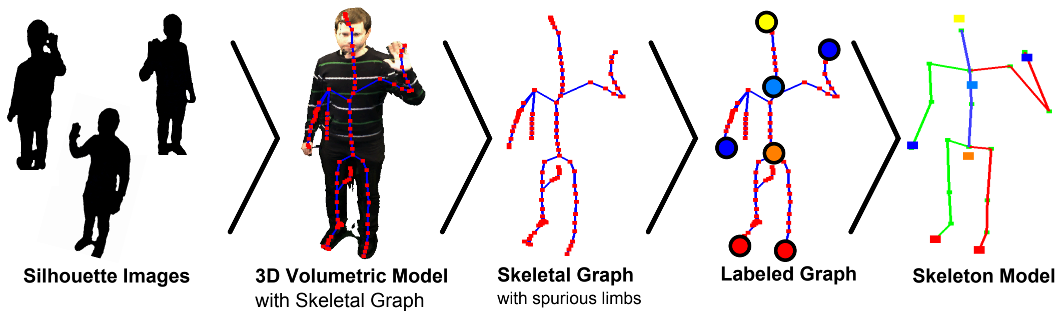


Figure 3.1: Extracting the 3D human skeleton pose from silhouettes.

model onto the graph. Finally, we optimize all skeleton joint positions using a fast local optimization similar to [18] while enforcing that joints lie on the skeletal graph and bones maintain their lengths. A graphical summary of our algorithm is shown in Fig. 3.1.

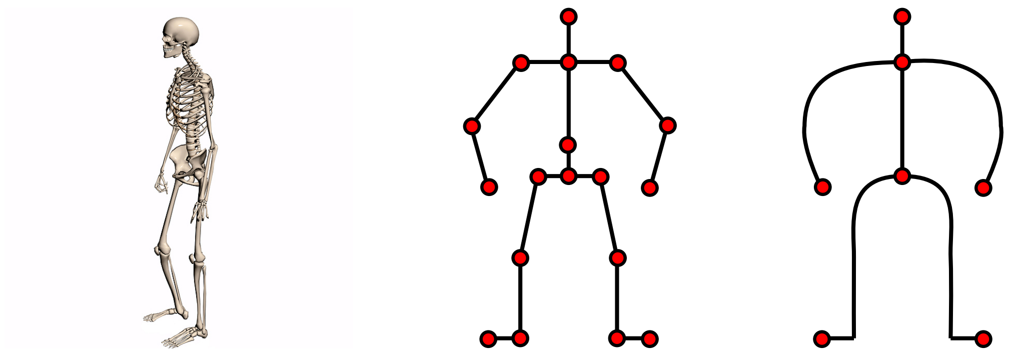
The key benefits of our method are the robustness of limb-labeling and its ability to perform single frame pose estimation. The computational cost is kept low due to an early reduction of the input data from several thousand volumetric elements to a compact graph with a few hundred nodes. Our method does not require any learning phase nor a database with training images, which makes it particularly easy to implement. An additional benefit of single-frame processing over pose tracking approaches is that our method cannot get stuck in an erroneous pose but recovers automatically. Moreover, we do not require an initial pose or manual initialization.

Such properties are particularly important in an interactive setting with visual feedback. Typically, the user will step in front of the monitor and expect that pose estimation starts instantly. In order to use the system without interruptions, pose estimation should work for an extended time of several minutes and must not get stuck in difficult poses. Our system is capable of such operation due to its real-time, single-frame operation.

3.2 Skeleton Models

A skeleton consists of multiple bones that are connected via joints (see Fig. 3.2(a)). Every skeleton can be represented as an acyclic graph with a single root node. This graph is commonly referred to as the hierarchy of bones. Even though the human skeleton is well defined through its 206 bones [66], there exists a variety of models with different levels of detail [18, 57, 114, 127]. In the literature, most skeleton models contain bones for torso, upper and lower limbs. Depending on the level of detail, bones that model hands and feet can be missing from the model. Even the spine can be modeled using a single bone or multiple bones to model bending poses. Similarly, the root of the skeleton is commonly defined either as the pelvis or the neck.

A typical result of pose estimation is either a list of skeleton joint angles [57] or the 3D positions of each joint [114, 127]. The position-based representation defines the skeleton



(a) Anatomical skeleton (b) Joints and bones (c) Reeb graph

Figure 3.2: Different representations of the human skeleton.

similar to a 3D stickman, which consists of lines (bones) between points (joints). The joint angle representation requires more knowledge about the skeleton. It denotes the angle of each joint relative to a rest pose (*e.g.* a person standing with the arms in a T-pose). For the rest pose, the lengths of all bones, rotation axes for each joint and possible rotation limits must be known in advance. Typically, this information is manually designed and adapted to a specific person. When defining the pose through joint angles, the lengths of bones typically do not change during pose estimation. In contrast, when using the position-based representation, it is very easy to adapt the lengths of bones as they are implicitly encoded as the Euclidean distance between two connected joints. Flexible bone lengths have the advantage that they can adapt to unknown persons. Since the skeleton model used for pose estimation will always be incomplete, flexible bone lengths can be used to model movements that are not supported by the skeleton model (*e.g.* bending of the spine).

Throughout this thesis, we use a joint-position-based skeleton model. Our skeleton hierarchy can be seen in Fig. 3.2(b). It models the arms without separate joints for hands. For legs, there is a separate foot bone. The root joint of our skeleton is defined as the middle-joint of the spine above the pelvis. The hierarchy of bones extends naturally from this joint.

For graph matching, we require a representation of the human skeleton which represents only the topology of the skeleton. The minimal skeleton graph is a *Reeb graph* [22] where there are only nodes at critical points (see Fig. 3.2(c)). The Reeb graph contains only branching nodes of the skeleton graph where limbs are attached to the body. The topology of the human body is independent of the skeleton pose. Therefore, a correctly estimated skeleton graph will be the same for every person in any pose.

3.3 Pose Estimation from Skeleton Graphs

We estimate the unknown human skeleton pose by extracting a graph from a visual hull representation of the body. In order to obtain the visual hull from input silhouette images, we perform space carving on a discrete voxel grid [91]. Processing of voxel data can be computationally inefficient when using a high resolution voxel space. The main idea of our method is to reduce the amount of data from roughly 10^6 voxels to a skeletal graph which consists of only about 10^2 connected nodes (see Fig. 3.1 for an example). Due to space carving artifacts, it is possible that the resulting graph contains spurious branches. These branches need to be robustly detected and removed in order to find those branches that represent arms and legs. We call this process end-node labeling. Once the label of every end-node of each branch is known, the skeletal graph can be used to initialize and optimize a full skeleton model such that it represents the pose of the body in the original images.

3.3.1 Generating the Skeletal Graph

We extract the skeletal graph G from a volumetric human body representation using *voxel scooping* by Rodriguez et al. [119]. This algorithm efficiently approximates the center-line from tubular volumetric data. The algorithm was originally intended for tracing center-lines of neurons in medical data. We show that it can be easily applied to volumetric body scans.

Starting with an initial node at a seeding point, each node in the graph spherically expands in voxel space with a locally adaptive radius (hence the name scooping). Based on this expansion, a new node is created and the process is repeated until there are no more voxels left to scoop. When the expanded voxels of a single node no longer form a connected component, a branching node is created and each branch is processed independently. For each node, we store the information about its neighboring nodes as well the 3D position of the expansion sphere. The result of voxel scooping is a graph similar to the human skeleton but with many spurious nodes and branches (such as in Fig. 3.1). An early pruning step is used to remove short branches with only a few nodes. Using a novel graph-matching algorithm which is described in the next section, we classify the remaining branches as outliers or valid hands and feet. In Fig. 3.3, we demonstrate the scooping process on a single 2D silhouette image. Scooping pixels in 2D is similar to scooping voxels in a volumetric body scan.

Voxel scooping requires a known seeding point. Therefore, we require that the seeding point corresponds to the top of the head. We assume that the person is standing upright such that the highest voxel in our volumetric representation is on the head. This assumption no longer holds when the person is allowed to raise the hands above the head. In this case, it is possible to track the head after initialization or to determine the head position by a face detection algorithm.

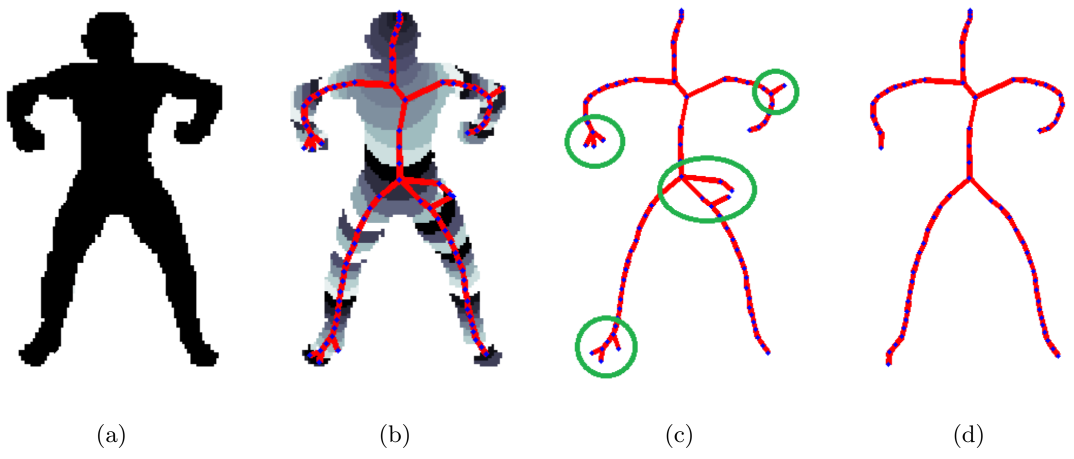


Figure 3.3: Voxel scooping explained on a single 2D silhouette image. Fig. (a) shows the input image. Fig. (b) shows the scoops alongside the generated graph. This graph contain some spurious nodes (c) which are removed in Fig. (d).

3.3.2 Skeleton Graph Matching using End-Nodes

We provide a template skeleton G' of the human body with approximately the same bone lengths as the user. For this template, we provide labels for the end-nodes of arms and legs in order to derive the position of hands and feet. An end-node is a node which has only one sibling and is located at the end of a branch. By robustly matching end-nodes of the skeletal graph G with this template, we propagate limb-labels and detect spurious branches. The matching algorithm needs to be robust to pose changes and must not rely on the position of branching nodes (hips and neck). The position of branching nodes depends on clothing or the current body pose. We use a graph matching method inspired by Bai and Latecki [12]. They perform shape recognition from 2D silhouette images based on skeletal graph matching. Their main idea is to match two graphs by comparing geodesic paths between end-nodes of the skeletal representation of the objects. Hence, matching is performed independent of the graph topology and the articulation of the object. Furthermore, there is no dependence on the global pose of the object in the input images.

We adopt this idea for finding correspondences between a 3D skeletal graph G and a template graph G'. However, the method of Bai and Latecki [12] cannot be directly applied to a 3D graph. Their algorithm requires the end-nodes to be ordered along the outer contour of a silhouette image of the object. In a volumetric model, such ordering along a one dimensional line is not possible as the volume is bounded by a 2D surface which does not allow for a deterministic sorting algorithm. Therefore, we propose a different strategy that takes advantage of the known head position as the root of the skeletal graph: we sort the end-nodes in ascending order by the length of the geodesic path to the head-node. This ordering preserves robustness in presence of articulated body movements and is independent from the graph topology.

3.3.2.1 Constructing Pairwise Distance Matrices

Bai and Latecki [12] compute a descriptor for the graph by sampling a distance field along the geodesic path between each two end-nodes. In contrast, we describe a graph solely by geodesic distances between end-nodes. For the skeletal graph G with N end-nodes $\Omega = \{\omega_1, \omega_2, \ldots, \omega_N\}$ ordered by geodesic distances to the root-node (i.e. the head-node), we define a pair-wise distance matrix

$$\mathbf{D}_G = \begin{bmatrix} gd(\omega_1,\omega_1) & gd(\omega_1,\omega_2) & \ldots & gd(\omega_1,\omega_N) \\ \vdots & \vdots & \ddots & \vdots \\ gd(\omega_N,\omega_1) & gd(\omega_N,\omega_2) & \ldots & gd(\omega_N,\omega_N) \end{bmatrix} \tag{3.1}$$

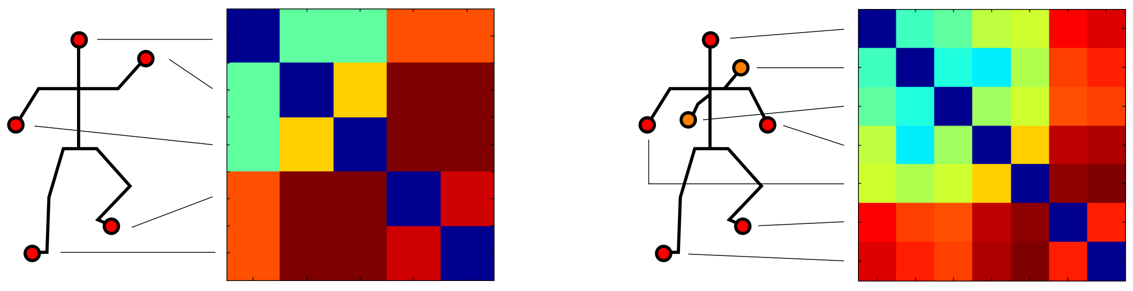
where $gd(\omega_i, \omega_j)$ is the length of the geodesic path between end-nodes ω_i and ω_j of graph G. The geodesic distance between two end-nodes are the accumulated Euclidean distances between all nodes along the path. These distances can be efficiently determined using Dijkstra's shortest path algorithm. Each row of the symmetric matrix $\mathbf{D}_G$ can be seen as a descriptor for an end-node which contains distances to all other end-nodes of the same graph (including spurious end-nodes).

In Fig. 3.4, we show two descriptor matrices where the values are color coded. Fig. 3.4(a) shows the descriptor matrix for the template skeleton. The first row corresponds to the head node, the second and third row to the hands and the remaining two rows describe the feet. In the first column, the geodesic distance values increase from the first to the last row. This signifies that feet have a larger geodesic distance from the head than hands.

Fig. 3.4(b) shows the descriptor matrix for a query skeleton extracted from a volumetric scan. It has seven end-nodes because it contains two spurious nodes (described by row two and three). In order to label the nodes of this skeleton correctly, we propose a novel matching algorithm which can cope with such spurious nodes.

3.3.2.2 Robust Matching of Distance Matrices

In order to assign a label to each end-node of the skeletal graph G with the descriptor matrix $\mathbf{D}_G$, we need to find correspondences to end-nodes of a template skeleton graph G' with a descriptor matrix $\mathbf{D}_{G'}$. This is done by comparing every end-node of graph G to all end-nodes of graph G', or more precisely calculating the matching cost for every pair-wise combination of row-vectors from both distance matrices $\mathbf{D}_G$ and $\mathbf{D}_{G'}$. Generally, the query graph G will not contain the same number of end-nodes N as the template graph G' ($N' = 5$) because some end-nodes are missing or spurious. Consequently, the sizes of $\mathbf{D}_G$ and $\mathbf{D}_{G'}$ will be different and standard comparison methods cannot be used.



(a) Template descriptor with five limbs

(b) Descriptor of a query skeleton with five limbs and two spurious end-nodes

Figure 3.4: Color-coded end-node descriptor matrices for the template skeleton and a query skeleton. Blue denotes a geodesic distance of zero and an increasing redness denotes higher distances.

Therefore, we need an algorithm to calculate the matching cost of feature vectors with different lengths.

Dynamic time warping (DTW) can be applied to various problems that consist of matching sequences of different lengths [117]. It makes use of dynamic programming (DP) in order to efficiently calculate the minimal matching cost of a sequence $a = \{\alpha_1, \alpha_2, \ldots, \alpha_A\}$ of length A by matching, skipping or deleting elements of another sequence $b = \{\beta_1, \beta_2, \ldots, \beta_B\}$. DTW iteratively populates an $(A+1) \times (B+1)$ warping matrix $\mathbf{W}$ (note that we use a 0-based indexing for this matrix). First, the matrix is initialized along the border:

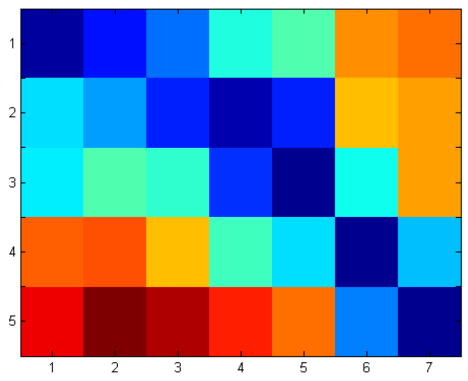
$$\mathbf{W}(0,j) = \infty\big|_{j=1\ldots B} \qquad \mathbf{W}(i,0) = \infty\big|_{i=1\ldots A} \qquad \mathbf{W}(0,0) = 0 \tag{3.2}$$

By applying DP, one can calculate its elements using the following procedure:

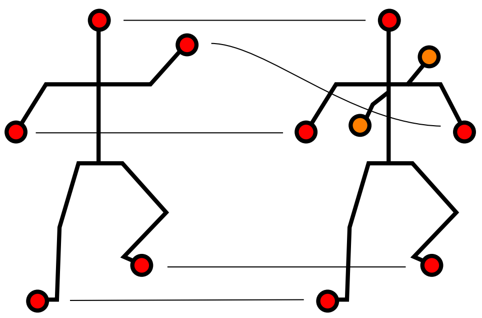
$$\mathbf{W}(i,j) = c(\alpha_i, \beta_j) + \min\left\{\mathbf{W}(i-1,j-1), \mathbf{W}(i-1,j), \mathbf{W}(i,j-1)\right\} \tag{3.3}$$

where $c(\alpha_i, \beta_j)$ denotes a cost function which compares α_i to β_j. In the simplest case, this is the absolute difference $c = |\alpha_i - \beta_j|$. The minimal matching cost for both series can be determined by evaluating the warping matrix at $\mathbf{W}(A, B)$.

We make use of dynamic time warping in order to match sequences of ordered end-nodes with different lengths. In contrast to other applications of DTW such as time-series processing, we are not interested in the optimal alignment of two series but rather in the minimal matching cost as a measure of their similarity. We define a cost matrix $\mathbf{C}$ that



(a)



(b)

Figure 3.5: (a) Matching cost matrix for the descriptor matrices in Fig. 3.4. Cool colors signify a low matching cost. Each row corresponds to the head or a limb of the template skeleton and each column to an end-node of the query skeleton. (b) Correspondences between skeletons.

contains a matching cost for every pair of end-nodes of graphs G and G':

$$\mathbf{C}(\mathbf{D},\mathbf{D}') = \begin{bmatrix} mc(\mathbf{D}_1,\mathbf{D'}_1) & mc(\mathbf{D}_1,\mathbf{D'}_2) & \dots & mc(\mathbf{D}_1,\mathbf{D'}_{N'}) \\ \vdots & \vdots & \ddots & \vdots \\ mc(\mathbf{D}_N,\mathbf{D'}_1) & mc(\mathbf{D}_N,\mathbf{D'}_2) & \dots & mc(\mathbf{D}_N,\mathbf{D'}_{N'}) \end{bmatrix} \tag{3.4}$$

where $mc(\mathbf{D}_i,\mathbf{D'}_j)$ denotes the minimal matching cost of the i-th row of matrix $\mathbf{D}_G$ and the j-th row of matrix $\mathbf{D}_{G'}$. The minimal matching cost is obtained by evaluating the DTW matrix at $\mathbf{W}(N,N')$ when using both row vectors as input sequences α and β. The optimal correspondence of end-nodes of the skeletal graph G to graph G' can be found using bipartite graph-matching based on the cost matrix $\mathbf{C}$. This task can be efficiently performed by the Hungarian algorithm [90], for example.

In Fig. 3.5(a), we show the resulting matching cost matrix $\mathbf{C}(\mathbf{D}_G,\mathbf{D}_{G'})$ after matching the two descriptor matrices shown in Fig. 3.4. For example, the low cost in $\mathbf{C}_{2,4}$ signifies that node 2 (a hand) of the template skeleton matches node 4 in the query skeleton. On the other hand, there is no low matching cost in columns two and three. This means that the corresponding end-nodes in the query skeleton graph are most likely spurious nodes and do not represent limbs. The obtained correspondences are visualized in Fig. 3.5(b).

Note that it is not possible to distinguish left and right limbs due to the body symmetry. This is not visible from Fig. 3.5(a) where one would expect that there is a similar matching cost for the left and right foot, for example. The reason for different matching costs lies is the initial sorting according to geodesic distances from limbs to head. This sorting can cause an arbitrary initial assignment to either side. We determine the correct side of each limb in a post processing step by keeping track of a vector that points in forward direction relative to the body. Initially, this vector points towards the camera as the user is assumed to face the camera. In subsequent frames, this vector is updated by analyzing the shoulder positions. Estimating the front vector from shoulder positions has the advantages that a person generally cannot move the shoulders as fast as the hands and it is not possible to swap left and right sides. Once the direction of the shoulders is known, we propagate the left/right information to each limb.

3.3.3 Using Tracking Information to Enhance Graph Matching

In the previous section, we have described a method to extract the skeleton from a volumetric voxel model and label its end-nodes. Our method is able to obtain a correct labeling without tracking or multi-frame processing. Thus, given a single input frame, the correct assignment of end-nodes to limbs is computed without knowledge of the previous pose or manual intervention. In our experiments, however, we found that there are situations where spurious nodes have descriptors that are very similar to actual limbs. These ambiguities can cause confusion between the correct node and a spurious one. For example, this can result in a spurious node on the body to be mislabeled as a hand.

Therefore, we propose to extend our end-node labeling algorithm by keeping track of the 3D position of the detected end-nodes. We use a linear Kalman filter [86] to track end-nodes detected in previous frames. This filter allows us to predict the position of end-nodes in the current frame. We use these predictions to analyze end-nodes during graph matching based on the cost matrix in Equation (3.4). If there are multiple nodes with a similar label matching cost, we assign a smaller matching cost to the end-node closer to the predicted position of a previous end-node. For computing the Euclidean distance between two nodes, we use their 3D positions obtained during voxel scooping (see Section 3.3.1).

This modification of matching costs effectively prohibits a false assignment of limbs due to similar descriptors. However, if a spurious node appears spatially close to the actual end-node, both nodes can have a similar descriptor. Therefore, it is possible that for a few frames there is a wrong assignment which cannot be detected automatically. However, when the person moves either body or limbs, the spurious node is likely to disappear and the correct labeling can be restored. Care has to be taken that a tracking filter does not delay the detection of the correct node due to motion smoothing. Alternatively, it is possible to reset tracking information periodically and execute our standard single-frame pose estimation algorithm without modified costs.

3.3.4 Skeleton Fitting

In addition to the end-nodes of the graph, we need to determine two additional interior nodes (pelvis and neck) in order to initialize our skeleton model (see Fig. 3.6(b)). We find these nodes by reusing the idea of the graph distance in order to define a discriminative

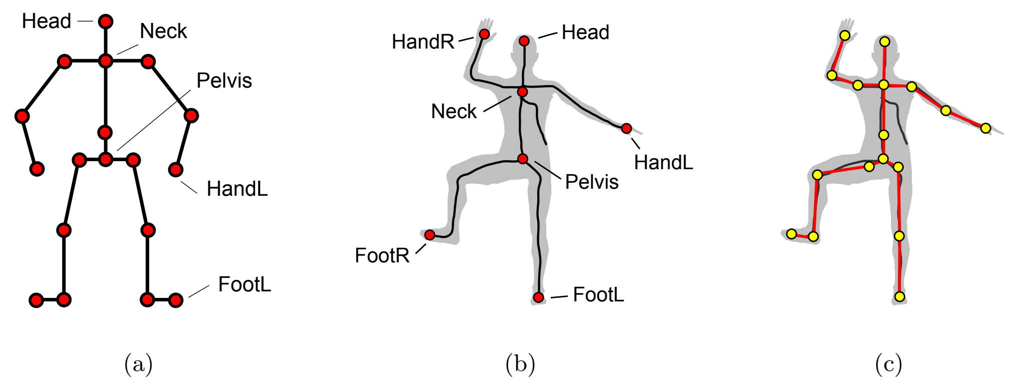


Figure 3.6: Template skeleton model (a). Skeletal graph with labeled end-nodes and nodes for neck and pelvis (b). Optimized skeleton model that fits to the skeletal graph (c).

interior-node descriptor:

$$F_G(v) = \left[gd_G(v, \text{Head}), gd_G(v, \text{HandL}), gd_G(v, \text{HandR}), gd_G(v, \text{FootL}), gd_G(v, \text{FootR})\right]^T \tag{3.5}$$

where v denotes a node of graph G and $gd_G(v, \omega)$ the geodesic distance between node v and an end-node ω of this graph. We use names for the nodes ω in order to make the procedure more comprehensible. For the pelvis and neck node, we calculate a descriptor $F_{G'}(\text{Pelvis})$ and $F_{G'}(\text{Neck})$ from the template skeleton. We evaluate $F_G(v)$ for each node v in the skeletal graph G in order to find the best match for the pelvis and neck:

$$v_{\text{pelvis}} = \arg\min_{v \in G} \|F_G(v) - F_{G'}(\text{Pelvis})\|^2 \qquad v_{\text{neck}} = \arg\min_{v \in G} \|F_G(v) - F_{G'}(\text{Neck})\|^2 \tag{3.6}$$

As a result, the skeletal graph has seven labeled nodes which correspond to the human skeleton as in Fig. 3.6(b). We initialize our skeleton model with the positions of the nodes for head, limbs and inner joints. It is then possible to use any local optimization method to refine the skeleton model until it fits to the graph. We propose to use a method similar to Baran et al. [18], which locally optimizes joint positions until they fit nicely inside the body. In addition, we ensure that bones have the same lengths as in the template skeleton G'. In contrast to Baran et al., our implementation attracts bones towards the skeletal graph instead of the center of the medial body surface.

3.4 Summary and Discussion

In this chapter, we have shown a novel method to estimate the human skeleton pose from volumetric body representations. We follow a bottom-up approach where we first approximate the center-line from the discrete visual hull. This step compresses several thousand voxels into a few hundred skeletal graph nodes. A robust and pose independent end-node labeling algorithm is able to classify hands and feet while ignoring spurious nodes. After the initial node labeling, a full skeleton model is fitted to the graph using local optimization.

The basic algorithm described in this thesis is able to perform pose estimation without knowledge of the pose in a previous frame. This property is essential for automatic initialization of the pose. In addition, a single-frame pose estimation algorithm cannot get stuck in erroneous poses from previous frames, which is a common problem for tracking-based algorithms. Nevertheless, we have presented a possibility to include tracking data in the end-node labeling process to facilitate the disambiguation of end-nodes with similar descriptors. To maintain the advantage of single-frame pose estimation, this tracking data should only be used for disambiguation between two or three similar end-nodes and not for end-node labeling in general.

Our approach is able to achieve state of the art pose estimation results at real-time frame rates (see experiments in Section 5.3). Even though related approaches such as Bakken et al. [15] use a highly parallel GPU for estimating the skeletal graph, our single CPU center-line extraction algorithm based on [119] shows superior performance. Our novel graph matching algorithm successfully copes with artifacts in the skeletal graph in more than 97 % of tested frames. Even if the pose estimation contains errors in a few frames, our algorithm is able to recover automatically.

There is a number of limitations that come with our approach. So far, we have assumed a template skeleton with roughly the same bone lengths as the person we want to estimate the pose for. This assumption does not hold for most persons using the system. A possible solution is to estimate the lengths of arms and legs from the height of the person since there is a high correlation between body height and limb lengths.

One essential problem with our approach is that we cannot handle cases where the arms are held close to the body or occluding objects. In this scenario, arms are no longer separable from the body and a useful skeletal graph cannot be extracted. This can mean that there is no branch for an arm, for example. Without a skeletal graph that contains detectable limbs, our pose estimation will fail. This problem is shared with all methods that use a bottom-up approach for human pose estimation from volumetric data. Our algorithm is designed to estimate the pose of upright standing persons. The location of the head must be easy to identify without prior pose information. Thus, we cannot handle scenes where those assumptions are not fulfilled. In a virtual dressing room environment, however, our pose estimation algorithm can be applied successfully because the person is standing and the head is usually easy to detect.

CHAPTER 4 3D HUMAN MODELS FROM MULTI-VIEW IMAGES

Contents

4.1 Introduction and Problem Statement

4.1.1 Motivation

In the previous chapter, we have shown how to estimate the pose of a human body from multi-view images. However, it is possible to obtain more information from such images than the skeletal pose. We capture the shape of a human body in order to build a 3D model of the person. The shape of the body is the visible outer surface which consists of the skin, hair, clothing and any gadgets the person is wearing.

In this chapter, we address how to capture the temporal evolution of this shape in every frame of a multi-camera image sequence. We use a polygonal template model of the human body and adapt it to images of a multi-camera setup. The main idea of our process is shown in Fig. 4.1. Our approach improves the robustness of existing model-based surface estimation methods by co-optimizing the surface and the skeleton of the human model. In addition, we present a novel optimization scheme that enables low processing times

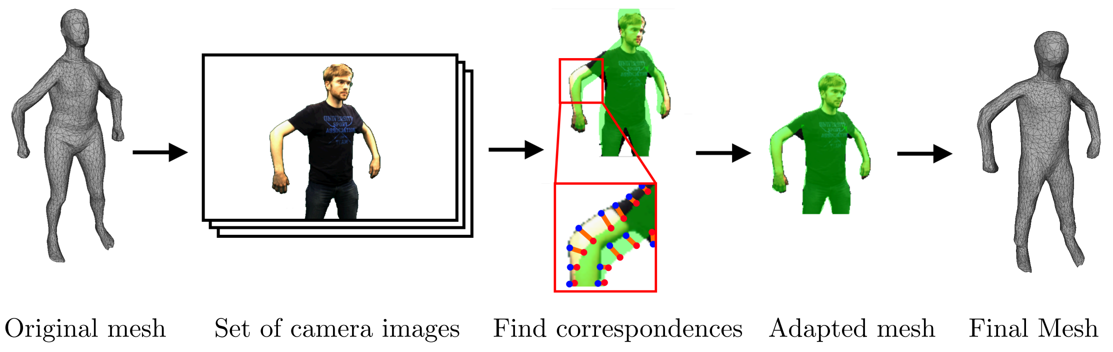


Figure 4.1: Shape adaptation is concerned with deforming an existing polygonal model of the human body to images by minimizing the distance between corresponding mesh vertices and pixels.

without noticeable loss in quality. Thus, our method makes it possible to estimate the shape of a human body at camera frame rate.

There are many applications for capturing the body surface at high frame rates. For example, computer graphics algorithms can be used to render the surface of the person from arbitrary viewpoints. The resulting virtual images look realistic and can be used in a virtual mirror setting. If the surface representation is sufficiently compact to be transported over a network, remote rendering algorithms can be used to create a tele-presence system. Producing a virtual rendering of a person is not the only feasible application when a captured surface is available. The surface information can be used to fit virtual garment to the body of the person or to derive anthropological measurements from the person without contact. Moreover, an exact surface can be used to accurately interact with virtual objects in a scene.

4.1.2 Challenges of Multi-View Shape Estimation

There are a number of challenges associated with capturing the body surface from multi-view data at high frame rates. A multi-camera setup has only a limited number of views. Usually, a multi-camera system that is used for capturing a person consists of 8 to 10 cameras [57, 130, 135]. When eight cameras are arranged around a person, a 360 degree view is possible when each camera has a viewing angle of approximately 45 degrees relative to its neighbors. Such high differences in viewing angles make computations of depth maps or photo consistency measures [124] challenging. A higher number of cameras would allow to form stereo pairs, which is beneficial for such computations. However, more camera images would need to be transferred and processed, which limits the feasibility of real-time performance. In addition, a higher number of cameras increases hardware costs and requires a more complicated setup.

One efficient way to capture the shape of the human body with a limited number of cameras is by analyzing the silhouettes from each camera. A silhouette is the binary image of an object segmented from its background. Given a set of calibrated cameras and a method to compute the silhouettes of the person from multiple views, it is possible to quickly obtain the visual hull of the person [52, 93]. It is possible to use the visual hull directly for rendering [73] or for computing a polygonal model [1, 52].

However, computing the visual hull directly from silhouette images can yield artifacts. These artifact are similar to what caused spurious nodes in volumetric pose estimation (see Chapter 3). Typical artifacts include additional limb-like structures. Therefore, we need a robust way to capture the surface of the human body.

4.1.3 Human Shape Estimation Through Mesh Deformation

In recent years, there has been a strong trend towards model-based approaches for human shape estimation. Model-based approaches often make use of a polygonal mesh which already looks similar to the person or at least has a human shape. An adaptation algorithm deforms this mesh in such a way that its reprojection to input images optimally explains the silhouettes of the person. Prior to adaptation, it is common to pre-deform the mesh such that it has a similar pose as the person in the images [57, 158]. Therefore, the mesh can be automatically rigged to a skeleton [18] such that by transforming the skeleton, the rigged mesh is transformed accordingly. The pose of the skeleton can be determined by a pose estimation algorithm such as described in Chapter 3.

Adapting a model of the human body to camera images has several advantages over bottom-up methods that reconstruct the body shape directly from input data. Compared to bottom-up methods, the number of visible artifacts due to ambiguities in input images is greatly reduced with model-based approaches as the model is a strong prior. Thus, object segmentation errors or isolated wrong correspondences are unlikely to have a visible influence on the result. If the model is a highly detailed laser scan of the person, details such as clothing wrinkles are baked into the model. Adapting such a model to images preserves these details even though they are not visible from the input silhouettes [57, 138]. Performing anthropological measurements directly on images is difficult because it is hard to find the correct measurement points on the body. Such measurement points can be defined once for an adaptable model. After adapting such a model to image data, predefined measurement points can guide body measurements and make the overall measurement process more robust.

The most common way of deforming a mesh such that it fits to images is to apply the Laplacian Mesh Editing (LME) framework [134]. The original application of this framework is to deform existing polygonal meshes by manually editing selected vertices. This framework is based on a least squares energy minimization which can be computed efficiently for reasonably sized meshes. It features a data term which pulls mesh vertices to certain 3D positions in space. Moreover, the LME framework provides a regularization term that preserves surface details in the mesh while allowing smooth deformations. This smoothing information is computed by analyzing the local neighborhood of each mesh vertex and computing differential coordinates. Tuning the weight of data and regularization terms influences the stiffness or flexibility of the deformation. More details about this framework will be given in Section 4.3.

Previous methods which apply the LME framework for adapting a human model to image data mainly focus on the efficiency of the data term. For example, such methods align mesh vertices with silhouette contours or image feature points [57, 158]. The regularization term is usually the same as in classical LME.

The standard framework does not make use of an existing skeleton structure during surface deformation. While the skeleton can be used to initialize the pose of the mesh prior to adaptation, it has to be discarded during shape optimization. Indeed, high quality results can be obtained without skeletons when each camera has a good view of the person in a multi-camera setup. However, our experiments have shown that in non-optimal camera layouts the result of mesh deformation can degrade substantially. For example, this is the case when not every camera is able to capture the complete body or when there are a low number of cameras. Non-optimal camera setups are usually the result of spatial restrictions. The recording setup used in this thesis must be small enough to be operated in a clothing store. It is not acceptable to sacrifice a whole room just for recording a single person. This leads to limitations where cameras can be placed and often cameras are too close to the human subject.

4.1.4 Contributions to Human Shape Estimation

We extend the LME framework by a novel skeleton-based regularization term to address the aforementioned problems. We base this decision on the observation that deformations of the human skin are predominantly caused by movement of the underlying skeleton. Our skeleton based regularization term combines bone movement and surface optimization. Consequently, large deformations of the surface are only possible if the connected bones support this movement. This can help in situations where input images are ambiguous or silhouettes cannot be extracted reliably.

Differential coordinates as defined for the LME framework support only translational movements. Possible surface rotations must be handled either by implicit optimization or explicit coordinate corrections. We show how to compute rotational corrections of differential coordinates efficiently by using the co-optimized skeleton joint positions.

In order to align mesh vertices with silhouette contours, point correspondences are required. Usually, such correspondences are found by projecting vertices into the image and taking the nearest contour point or searching in direction of surface normals [4, 158]. In an iterative fashion, this search for correspondences is alternated with surface optimization until no further improvement is possible. This process is illustrated in Fig. 4.2. False correspondences between mesh vertices and silhouette contours often cause problems and lead to artifacts. We propose to solve this problem by an automatic weighting scheme which takes into consideration the surface orientation of each mesh vertex as well as the

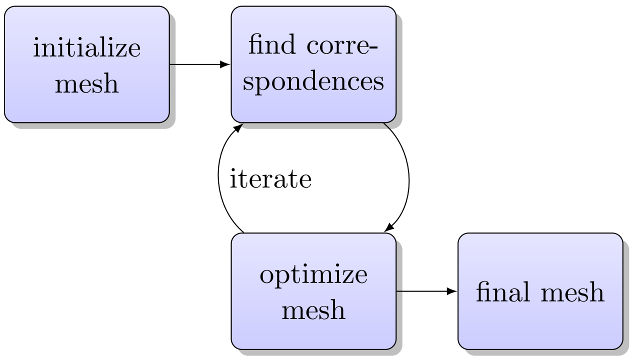


Figure 4.2: Image-based Laplacian mesh editing is an iterative process.

orientation of the silhouette contours.

In Straka et al. [142], we proposed the first real-time capable mesh adaptation system in the literature. We achieved a fast computation time through iterative minimization of decoupled deformation constraints. In this thesis, we extend our previous work and provide a novel framework for efficient shape estimation using a multi-resolution mesh structure and a novel iterative optimization algorithm. At low resolutions, we perform computationally demanding computations at a fraction of the run-time of a full resolution mesh. First, we ensure that a mesh is coarsely aligned with the images without emphasis on a detailed surface reconstruction. After increasing the resolution of the mesh, we adapt the mesh to details in the input images.

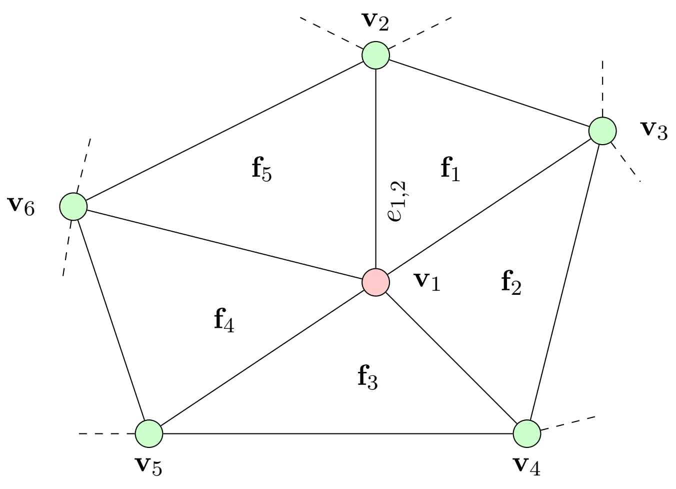
4.2 Polygonal Meshes

4.2.1 Elements of a Polygonal Mesh

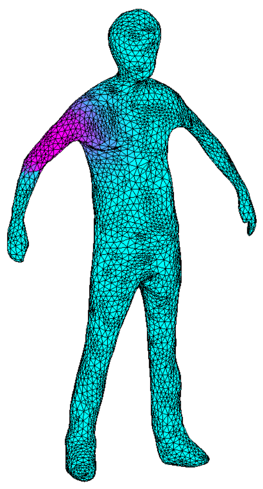
In this thesis, we use polygonal meshes to model the shape of the human body. This section gives a short overview of how to represent such meshes and explains basic definitions. We define a mesh $\mathscr{M} = \{\mathbf{V}, \mathbf{F}\}$ with several vertices $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_n]^T$ with $\mathbf{v}_i \in \mathbb{R}^3$ and multiple triangular faces $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_m]^T$ with $\mathbf{f}_i \in \mathbb{N}^3$. Each face references three vertices by their index. We define that vertex indices must be counter-clockwise oriented when the face is front-facing. An edge e_{ij} is defined as the direct connection between two vertices $\mathbf{v}_i$ and $\mathbf{v}_j$.

In Fig. 4.3(a), we show a small subset of a triangular mesh facing the reader. For example, face 1 is defined as $\mathbf{f}_1 = [1, 3, 2]^T$. We define the set $\mathscr{N}$ as the 1-ring vertex neighborhood for each vertex. For example, the 1-ring of vertex $\mathbf{v}_1$ is the set of all vertices directly connected to $\mathbf{v}_1$ via an edge: $\mathscr{N}_1 = [2, 3, 4, 5, 6]^T$ (i.e. all green vertices).

Another important concept of polygonal meshes deals with face and vertex normals. A normal describes the orientation of the mesh surface at a particular point. Normals are used for lighting computations in the rendering stage and when searching for mesh



(a)



(b)

Figure 4.3: (a) Elements of a polygonal mesh. Circles denote vertices and lines represent edges. (b) Linear skinning weights for the right upper arm as vertex colors.

correspondences. A face normal is a unit-length vector that points orthogonal to the surface spanned by the vertices of the face. The face normal $\mathbf{n}_i \in \mathbb{R}^3$ can be directly computed from the vertices of the face. The following equation computes the normalized face normal for face 1:

$$\mathbf{n}_{\mathbf{f}_1} = \frac{(\mathbf{v}_3 - \mathbf{v}_1) \times (\mathbf{v}_2 - \mathbf{v}_1)}{\|(\mathbf{v}_3 - \mathbf{v}_1) \times (\mathbf{v}_2 - \mathbf{v}_1)\|_2}. \tag{4.1}$$

All meshes used in this thesis are considered smooth (i.e. they have the same normal on both sides of an edge). Thus, we can compute vertex normals as the normalized sum of all adjacent face normals. For example, the vertex normal for vertex $\mathbf{v}_1$ is computed as

$$\mathbf{n}_{\mathbf{v}_1} = \frac{\sum_{i \in \mathscr{N}_1} \mathbf{n}_{\mathbf{f}_i}}{\left\|\sum_{i \in \mathscr{N}_1} \mathbf{n}_{\mathbf{f}_i}\right\|_2}. \tag{4.2}$$

All polygonal meshes used in this thesis are watertight. This means that there is no boundary edge and every edge has exactly two adjacent faces. This implies that a watertight mesh has no holes.

4.2.2 Skeletal Subspace Deformation

A polygonal mesh can consist of thousands of vertices. This number is too large for efficient modifications such as changing the non-rigid pose of the mesh. Similar to the skin of a human body, a mesh can be deformed in a more efficient way through linear blend skinning. Therefore, a skeleton is embedded into the mesh and surface vertices are attached to one or multiple bones. This attachment can be expressed through linear skinning weights $0 \leq \rho_{ij} \leq 1$ that connect a vertex $\mathbf{v}_i$ to a bone b_j. Each bone b_j has a linear affine transformation $\mathbf{T}_j \in \mathbb{R}^{3\times 4}$ that can modify all attached vertices. Mathematically, the overall deformation of the mesh can be expressed as the weighted sum of transformed vertices:

$$\tilde{\mathbf{v}}_i = \sum_j \rho_{ij} \mathbf{T}_j \begin{bmatrix} \mathbf{v}_i \\ 1 \end{bmatrix} \qquad \text{with} \quad \sum_j \rho_{ij} = 1 \quad \forall i \tag{4.3}$$

The skinning weights ρ_{ij} can be automatically obtained through a rigging algorithm [18]. In Fig. 4.3(b), we show the computed skinning weights for the attachment between all mesh vertices and bone of the right upper arm as a colored surface. Purple colors signify a strong attachment with $\rho_{ij} = 1$ while cyan colors represent $\rho_{ij} = 0$. In the vicinity of joints such as elbow and shoulder there is a smooth transition of weights. When a joint is bent, skin areas near joints will be transformed smoothly (*e.g.* partly by the bones of the upper and lower arm).

4.3 Laplacian Mesh Editing

Mesh editing is the process of deforming the surface of an existing polygonal mesh by modifying the position of a few selected vertices [28, 98, 134]. In contrast to skeletal subspace deformation, mesh editing involves no skeleton that defines surface transformations. Instead, the user defines constraints directly for individual vertices of the mesh. These constraints are commonly referred to as target handles and define the position and rotation of selected vertices. The position of remaining vertices is optimized such that the resulting deformation is smooth and structural details of the mesh are preserved. Therefore, each vertex uses a coordinate relative to its neighboring vertices. These relative coordinates are computed using the Laplace operator. Hence, the overall algorithm is called Laplace mesh editing. The main application of mesh editing is 3D modeling and animation. In contrast to alternative modeling techniques, the Laplacian mesh editing approach allows non-rigid deformations to be modeled intuitively. An example for editing of an existing mesh with this method is shown in Fig. 4.4.

Besides applications in the modeling domain, Laplacian mesh editing has proven to be a versatile method for automatic estimation of body shape from images. It is especially useful in the area of human performance capture from multi-view video [4, 57, 158]. The main difference to manual mesh editing is that editing constraints are no longer user defined but are automatically created from image correspondences. In the general case, it is difficult to compute these correspondences directly from images. Therefore, the process of finding image correspondences is iterated with a surface optimization until convergence (see Fig. 4.2).

In this section, we provide a deeper insight into mesh adaptation using Laplacian mesh editing techniques. First, we explain how differential coordinates can be used to describe

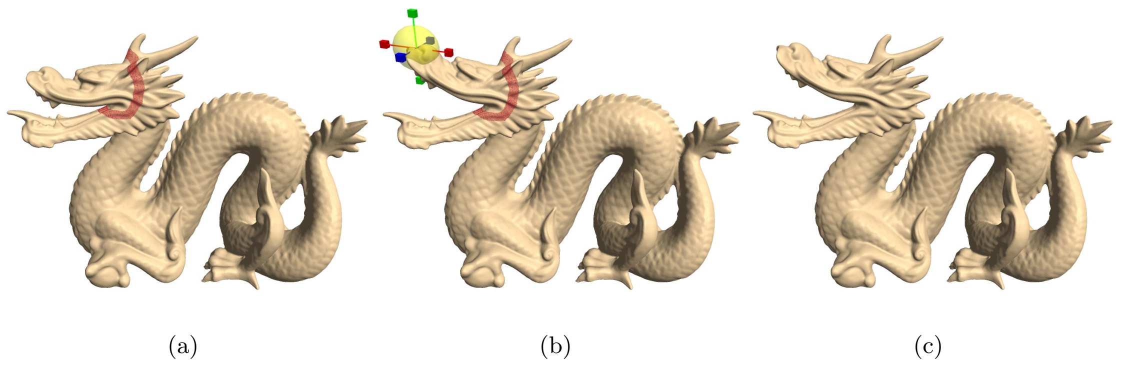


Figure 4.4: Laplacian Mesh Editing: Sorkine et al. [134] (a) select a region of interest and (b) modify the translation and rotation of the region in order to (c) achieve a detail preserving deformation (images taken from [134]).

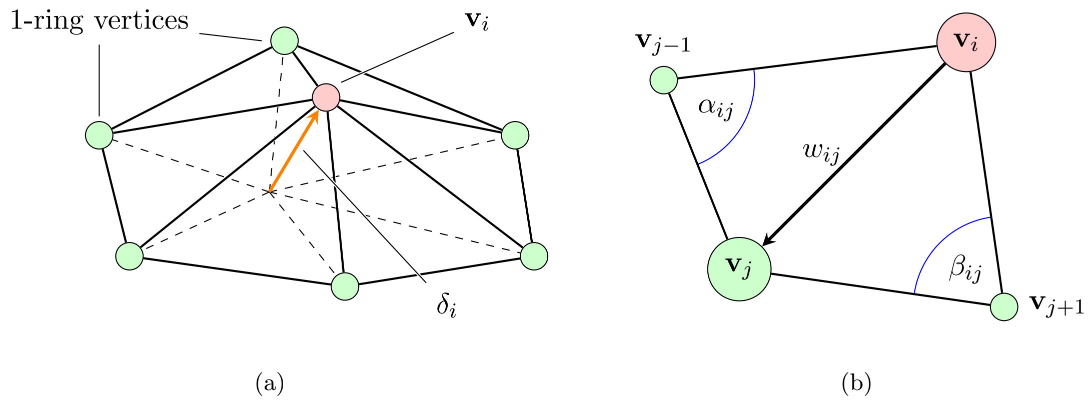


Figure 4.5: (a) Computation of delta coordinates using the 1-ring of neighboring vertices. (b) Definition of angles for the co-tangent weighting scheme [28] for edge $(\mathbf{v}_i, \mathbf{v}_j)$.

the coordinate of each vertex depending on its neighbors. Then, we show how these coordinates enable a smooth mesh deformation in a linear optimization scheme. Finally, we present methods to handle rotation and scaling issues that arise when working with differential coordinates and linear optimization.

4.3.1 Differential Coordinate Representation

Differential coordinates can be used to describe the position of a vertex relative to its neighbors. This is essential for mesh editing as the absolute position of each vertex can change during the editing process. By keeping relative coordinates similar to the unmodified shape, the local structure of the mesh can be preserved during editing.

For a vertex $\mathbf{v}_i$, the differential coordinate $\delta_i \in \mathbb{R}^3$ is computed as the vertex position minus the weighted sum of positions of its 1-ring neighbors $\mathscr{N}_i$ (see Fig. 4.5(a)). This is a discrete approximation of the Laplace-Beltrami operator $\mathcal{L}(.)$:

$$\mathcal{L}_i(\mathbf{V}) = \delta_i = w_i \mathbf{v}_i - \sum_{j \in \mathscr{N}_i} w_{ij} \mathbf{v}_j. \tag{4.4}$$

The weights $w_{ij} \geq 0$ are obtained using the co-tangent weighting scheme [28]:

$$w_i = \frac{1}{A_i} \qquad w_{ij} = \frac{1}{2} \left(\cot \alpha_{ij} + \cot \beta_{ij}\right) \tag{4.5}$$

where A_i is the Voronoi area of vertex $\mathbf{v}_i$ and α_{ij} and β_{ij} are the two angles opposite to the edge $(\mathbf{v}_i, \mathbf{v}_j)$ (see Fig. 4.5(b)).

There are certain properties of differential coordinates that are computed with the discrete Laplace-Beltrami operator. The operator is a linear operator which means that constraints containing the Laplace operator can be used in a linear optimization framework. A differential coordinate is independent of global translation but not on scaling and rotation. Therefore, special treatment is required when rotations and scaling need to be supported during mesh editing (see Section 4.3.3).

4.3.2 Least Squares Mesh Deformation

Deformation of the overall mesh can be expressed as a set of constraints. An optimally deformed mesh is obtained when all constraints are satisfied. This can be achieved by jointly minimizing the error of all constraint. There are two types of constraints involved in Laplace mesh editing: (1) position constraints modify the position of selected vertices (data term) and (2) smoothness constraints keep differential coordinates of the vertices close to their initial value (smoothness term). Note that smoothness does not mean that the resulting mesh is smooth but rather details of the original mesh are preserved. The deformed vertex positions of the overall mesh $\tilde{\mathbf{V}}$ can be obtained via linear least squares minimization:

$$\tilde{\mathbf{V}} = \arg\min_{\mathbf{V}} \underbrace{\sum_{i=1}^{M} w_i \|\mathbf{v}_i - \mathbf{t}_i\|^2}_{\text{data term}} + \underbrace{\lambda \sum_{i=1}^{N} \|\delta_i - \mathcal{L}_i(\mathbf{V})\|^2}_{\text{smoothness term}} \tag{4.6}$$

where $\mathbf{t}_i$ denotes the target position of handles and w_i a weight for each target. λ is a global weight factor that balances the influence of the data term and the smoothness of the result. The set of equations in (4.6) can be expressed in matrix notation:

$$\tilde{\mathbf{V}} = \arg\min_{\mathbf{V}} \left\| \begin{bmatrix} \mathbf{S} \\ \lambda \mathbf{L} \end{bmatrix} \mathbf{V} - \begin{bmatrix} \mathbf{T} \\ \lambda \mathbf{\Delta}_L \end{bmatrix} \right\|^2 \tag{4.7}$$

where $\mathbf{T} = [\mathbf{t}_1, \dots, \mathbf{t}_M]^T$ is the set of target positions and $\mathbf{S}$ is a matrix that selects the handle vertices. $\mathbf{L}$ denotes the discrete Laplacian matrix for the mesh and the initial differential coordinates of the mesh are $\mathbf{\Delta}_L = [\delta_1, \dots, \delta_N]^T$. The initial δ-coordinates of the mesh are computed by applying the Laplacian operator on the initial vertex positions $\mathbf{\Delta}_L = \mathbf{L} \cdot \mathbf{V}_0$. Since both $\mathbf{L}$ and $\mathbf{S}$ are sparse matrices, this system of linear equations can be efficiently solved in closed-form using sparse Cholesky decomposition [38].

4.3.3 Handling Rotations

As mentioned in Section 4.3.1, Laplacian mesh editing cannot accurately handle 3D rotations of the mesh using linear optimization. The problem is that differential coordinates as defined in Equation (4.4) support only the translation of vertices. Thus, any induced rotation due to stretching or bending of the object result in shearing of vertices.

Sorkine et al. [134] present a possible solution to this problem. They implicitly optimize for a transformation $T_i(\mathbf{V})$ for every vertex. This transformation rotates and scales initial differential coordinates δ_i during shape optimization. The modified optimization problem can be written as:

$$\tilde{\mathbf{V}} = \arg\min_{\mathbf{V}} \sum_{i=1}^{M} w_i \|\mathbf{v}_i - \mathbf{t}_i\|^2 + \lambda \sum_{i=1}^{N} \|T_i(\mathbf{V})\delta_i - \mathcal{L}_i(\mathbf{V})\|^2. \tag{4.8}$$

For two-dimensional meshes, the transformation T_i can be fully estimated using linear equations. Thus, a deformed mesh can still be optimized using a linear solver. However, for three dimensional meshes, only a linear approximation of the transformation is possible when linearity is to be maintained. This approximation works for small rotation angles only.

In case of larger rotations, a possible solution is given by Lipman et al. [98]. First, they compute a rough deformation of the mesh using Equation (4.6). Then, for each vertex $\mathbf{v}_i$, a local rotation matrix $\mathbf{R}_i$ is estimated by analyzing the 1-ring neighborhood of the vertex. Finally, the initial differential coordinates are corrected $\tilde{\delta_i} = \mathbf{R}_i \cdot \delta_i$ and the updated system in Equation (4.6) is solved again.

4.4 Mesh Editing based on Silhouette Images

In the previous section, we have given an introduction to Laplacian Mesh Editing (LME) with no specific focus on how to apply this framework to multi-view images. In this section, we will explain how LME can be used to iteratively deform a polygonal mesh based on image observations. More specifically, we focus on methods that operate on multi-view silhouette images of a person. These methods are based on finding correspondences between rim vertices and silhouette contours. In the optimization step, these correspondences pull those vertices towards the silhouette contour which maximizes the overlap of the projected mesh with the silhouettes (see Fig. 4.6). This procedure is iterated until

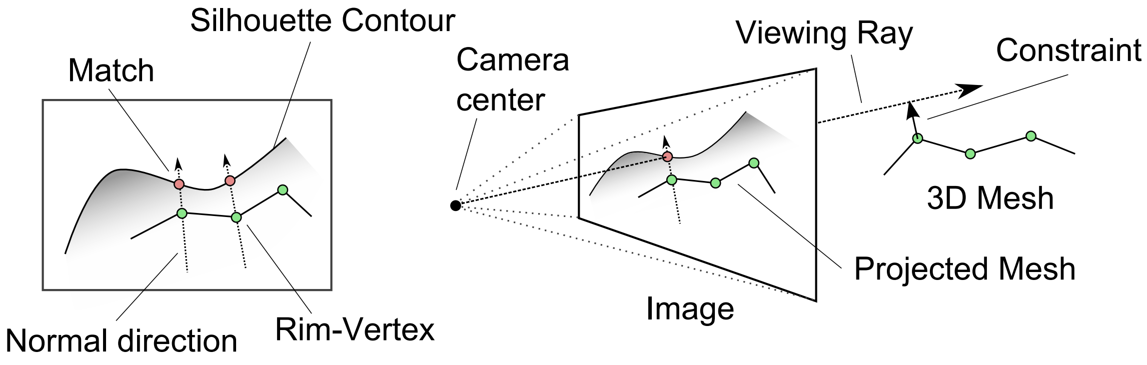


Figure 4.6: Silhouette constraints pull rim-vertices towards the silhouette contour in every camera image.

convergence. The result is equivalent to fitting the mesh to the visual hull. However, the visual hull does not need to be computed explicitly.

Rim vertices are vertices that are located on the border of the mesh when projected onto a camera image. In order to find rim vertices, we project vertices $\mathbf{v}_i$ of mesh $\mathscr{M}$ into all camera views using the corresponding 3×4 projection matrices $\mathbf{P}_\ell = \mathbf{K}_\ell[\mathbf{R}_\ell | -\mathbf{R}_\ell \mathbf{t}_\ell]$ and rotate the corresponding vertex normals $\mathbf{n}_i$ onto the image plane using rotation matrix $\mathbf{R}_\ell \in \mathbb{R}^{3\times 3}$:

$$\mathbf{v}_i^\ell = \frac{\begin{bmatrix}\mathbf{P}_\ell^1 \\ \mathbf{P}_\ell^2\end{bmatrix} \cdot \begin{bmatrix}\mathbf{v}_i \\ 1\end{bmatrix}}{\mathbf{P}_\ell^3 \cdot \begin{bmatrix}\mathbf{v}_i \\ 1\end{bmatrix}} \qquad \mathbf{n}_i^\ell = \mathbf{R}_\ell \cdot \mathbf{n}_i \tag{4.9}$$

where $\mathbf{P}_\ell^r$ denotes the r$^{\text{th}}$ row of the projection matrix of camera ℓ. We compute vertex normals $\mathbf{n}_i$ as the normalized mean of face normals adjacent to the vertex $\mathbf{v}_i$. A rim vertex in image $\mathcal{I}_\ell$ is a vertex with a normal almost parallel to the image plane of camera ℓ. The absolute value of the z-component of $\mathbf{n}_i^\ell$ is small. We find rim-vertices by comparing the absolute z-component with a threshold.

Note that this threshold is dependent on the resolution of the mesh. In a high resolution mesh, faces adjacent to vertices have a similar orientation and vertex normals are accurate. However, when the mesh resolution is low, only a few vertices model the cross section of a body part (*e.g.* an arm) and the mean orientation of adjacent faces may not represent a good vertex orientation. Therefore, the threshold must be increased when the mesh resolution is low. We use a threshold of $|z| < 0.1$ for a mesh with more than 2 000 vertices and gradually increase this threshold to $|z| < 0.3$ as the mesh resolution decreases to 500 vertices or lower.

Next, we need to find correspondences between rim vertices and silhouette contour pixels. For all rim vertices, we sample pixels from $\mathcal{I}_\ell$ along a 2D line

$$l(t) = \mathbf{v}_i^\ell + t \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{n}_i^\ell \tag{4.10}$$

for intersections with the silhouette contour where $-\tau \leq t \leq \tau$ defines the search region in pixels. Note that it is important that only intersections with a contour gradient similar to the normal direction $\mathbf{n}_i^\ell$ are considered a match $\mathbf{p}_k^\ell \in \mathbb{R}^2$. To speed up the search for correspondences, it is possible to use a binary search along the line. Each successfully matched rim-vertex/contour pair $(\mathbf{v}_i^\ell, \mathbf{p}_k^\ell)$ yields a 2D correspondence in image space.

4.4.1 Constraints based on Point Correspondences

There are several possibilities to translate a 2D correspondence into a 3D constraint. Each pixel of a 2D image captured by a perspective camera can be represented by a viewing ray. A viewing ray is a 3D line that connects the camera center with a pixel on the projective plane. More importantly, the part of the object that is projected onto a specific pixel lies on the corresponding ray. However, the distance of that object from the camera cannot be known from a single image alone.

The standard LME framework which is presented in Section 4.3 supports only point-to-point correspondences. A simple solution to attract a 3D vertex towards a viewing ray is to compute the point on the ray with the shortest Euclidean distance to the vertex. Then, a constraint is generated that minimizes the distance between this point and the vertex during mesh deformation. This method already yields good results when the mesh has been properly initialized before finding correspondences.

A more advanced correspondence scheme is presented by Gall et al. [57], who define the correspondence between a three dimensional vertex and the camera ray itself. They minimize the distance between a 3D vertex $\mathbf{v}_i \in \mathbb{R}^3$ and a viewing ray corresponding to an image pixel $\mathbf{p}_k \in \mathbb{R}^2$. Therefore, the corresponding constraint has one degree of freedom where the vertex is allowed to freely move along the ray. For each correspondence between a vertex and a ray, Gall et al. define two linear constraints:

$$\begin{aligned} \left(\mathbf{N}_\ell^x - \mathbf{p}_k^x \mathbf{N}_\ell^z\right) \mathbf{v}_i + \left(T_\ell^x - \mathbf{p}_k^x T_\ell^z\right) &= 0 \\ \left(\mathbf{N}_\ell^y - \mathbf{p}_k^y \mathbf{N}_\ell^z\right) \mathbf{v}_i + \left(T_\ell^y - \mathbf{p}_k^y T_\ell^z\right) &= 0 \end{aligned} \tag{4.11}$$

where the 3×3 rotation matrix $\mathbf{N}_\ell = \mathbf{K}_\ell \mathbf{R}_\ell$ is computed from the corresponding projection matrix $\mathbf{P}_\ell$ and $\mathbf{T}_\ell = -\mathbf{K}_\ell \mathbf{R}_\ell \mathbf{t}_\ell$ denotes the translation.

When determining correspondences between vertices and target points automatically, it is possible that a vertex matches with an incompatible contour pixel. Therefore, a reliable rejection or weighting of such matches is required. A feasible way to implement such weighting is to analyze the surface normal of the vertex and compare it to the orientation of the contour pixel [158]. The weight should be high for similar orientations and low if the vertex has a different orientation than the contour. A simple computation is based on the dot product of the contour normal $\mathbf{n}_k \in \mathbb{R}^2$ and the projected vertex normal $\mathbf{n}_i \in \mathbb{R}^3$:

$$w_{i,k} = \left\langle \mathbf{n}_k, \mathbf{N}_\ell^{xy} \cdot \mathbf{n}_i \right\rangle \tag{4.12}$$

where $\mathbf{N}_\ell^{xy}$ denotes the first two rows of the projective rotation matrix of camera ℓ. We do not normalize the rotated and truncated normal $\mathbf{N}_\ell^{xy} \cdot \mathbf{n}_i$, which contains only the x and y axis components. This strategy down-weights rotated normals with a higher z-component. In practice, this ensures that only true rim vertices have a high weight $w_{i,k}$.

4.4.2 Constraints based on Surface Correspondences

Since the matching of vertices and silhouette contours is based on an iterative closest point scheme, it is likely that these point correspondences will change across iterations. This leads to the question: why should we use point correspondences in the first place?

In fact, it makes more sense to align the mesh surface with the silhouette contour instead of aligning individual vertices with contour points. Therefore, we propose a novel covariance-based weighting scheme based on the Generalized Iterative Closest Point (GICP) algorithm presented in Segal et al. [123]. Segal et al. propose a better way to align three dimensional point clouds than by simply reducing the Euclidean distance between two corresponding points. Instead, they minimize the distance between corresponding planes. They make the assumption that every point $\mathbf{v}_i$ lies on a small, local plane which can be described using an anisotropic covariance matrix $\mathbf{\Sigma}_i$. This matrix models a small variance in the normal direction of the surface but a high variance on the local surface plane. Each point $\mathbf{v}_i$ is therefore described not only by its position but also by a covariance matrix which is computed by analyzing the neighboring points.

The plane-to-plane distance between two point/covariance tuples $(\mathbf{v}_i, \mathbf{\Sigma}_i)$ and $(\mathbf{v}_j, \mathbf{\Sigma}_j)$ can be expressed using the squared Bhattacharyya distance:

$$(\mathbf{v}_i - \mathbf{v}_j)^T \underbrace{\left(\frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2}\right)}_{\mathbf{H}_{ij}}^{-1} (\mathbf{v}_i - \mathbf{v}_j) = (\mathbf{v}_i - \mathbf{v}_j)^T \mathbf{H}_{ij}{}^{-1} (\mathbf{v}_i - \mathbf{v}_j) = \|\mathbf{v}_i - \mathbf{v}_j\|^2_{\mathbf{H}_{ij}} . \tag{4.13}$$

The Bhattacharyya distance measure computes the combined covariance matrix

$$\mathbf{H}_{ij} = \frac{\mathbf{\Sigma}_i + \mathbf{\Sigma}_j}{2} \tag{4.14}$$

as the average of the two covariance matrices. This covariance matrix has two key advantages when minimizing the distance between two planes: First, an anisotropic covariance matrix allows movement of points along the surface plane but constrains corresponding points to have a low distance in normal direction. Second, the mean of two similarly oriented covariance matrices conserves a high weight in normal direction while differently

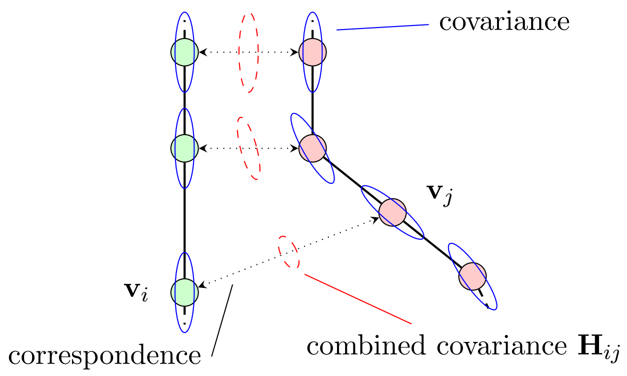


Figure 4.7: Covariances describe the orientation of the surface around a vertex $\mathbf{v}_i$. The mean of two corresponding covariances yields an automatic weighting $\mathbf{H}_{ij}$.

oriented covariances automatically decrease the strength of the correspondence. This behavior is demonstrated in Fig. 4.7. Since covariance matrices are symmetric, surface points with an opposing normal direction would wrongly yield a strong correspondence. Therefore, such correspondences need to be discarded. Note that if $\mathbf{H}_{ij}$ is the identity matrix, the distance computed in Equation (4.13) is equal to the standard squared Euclidean distance.

In this thesis, we apply the concept of covariance-weighted point correspondences presented in [123] to correspondences found in image-based Laplacian mesh editing. There are two problems that need to be addressed:

- The original GICP algorithm computes plane covariance matrices by sampling dense points clouds [123]. The mesh used in mesh editing is not as dense but we can exploit the polygonal structure of the mesh to compute plane covariances.
- Initial correspondences are given between three dimensional vertices and two dimensional pixels. Thus, we need to rotate the plane covariance into the image plane before we can compute a combined covariance matrix $\mathbf{H}_{ij}$.

For three dimensional meshes, we compute the surface covariance $\mathbf{\Sigma}_i$ as the weighted average of the covariances of the faces adjacent to vertex $\mathbf{v}_i$. First, we assign a covariance $\mathbf{\Sigma}_{f_j}$ to each face:

$$\mathbf{\Sigma}_{f_j} = \mathbf{R}_{f_j} \begin{bmatrix} \varepsilon_{\text{cov}} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R}_{f_j}{}^T. \tag{4.15}$$

The rotation matrix $\mathbf{R}_{f_j}$ rotates the normal of face f_j onto the x-axis. $\varepsilon_{\text{cov}} \ll 1$ defines the variance in normal direction. As the value of ε_{cov} becomes smaller, the resulting covariance ellipsoid $\mathbf{\Sigma}_{f_j}$ is flattened. The covariance $\mathbf{\Sigma}_i$ for vertex $\mathbf{v}_i$ is obtained as the weighted sum of inverse-covariances of its neighboring faces:

$$\mathbf{\Sigma}_i = \left(\sum_{j \in \mathrm{NF}(\mathbf{v}_i)} \alpha_{ij} \left(\mathbf{\Sigma}_{f_j}\right)^{-1} \right)^{-1} \qquad \text{with} \qquad \sum_{j \in \mathrm{NF}(\mathbf{v}_i)} \alpha_{ij} = 1 \tag{4.16}$$

where α_{ij} is a weight proportional to the area of face f_j and $\mathrm{NF}(\mathbf{v}_i)$ the list of faces adjacent to vertex $\mathbf{v}_i$. For 2D meshes or silhouette contours, we obtain $\mathbf{\Sigma}_i$ via edge covariances weighted by edge lengths.

The 3×3 covariance matrix $\mathbf{\Sigma}_i^v$ corresponding to vertex $\mathbf{v}_i$ cannot be averaged with the 2×2 contour covariance $\mathbf{\Sigma}_k^p$ of a corresponding silhouette contour pixel $\mathbf{p}_k$. Thus, we cannot compute $\mathbf{H}_{ik}$ from Equation (4.14) directly. The solution is to first rotate $\mathbf{\Sigma}_i^v$ into the image coordinate system of camera ℓ and then drop the depth component (z axis). This allows the additive combination of surface covariances with the 2D contour covariance of pixel $\mathbf{p}_k$:

$$\tilde{\mathbf{H}}_{ik} = \frac{\tilde{\mathbf{\Sigma}}_i^v + \mathbf{\Sigma}_k^p}{2} \qquad \text{with} \qquad \tilde{\mathbf{\Sigma}}_i^v = \begin{bmatrix} \mathbf{R}_\ell^1 \\ \mathbf{R}_\ell^2 \end{bmatrix} \mathbf{\Sigma}_i^v \begin{bmatrix} \mathbf{R}_\ell^1 \\ \mathbf{R}_\ell^2 \end{bmatrix}^T \tag{4.17}$$

where $\tilde{\mathbf{H}}_{ik}$ is the combined correspondence covariance. By combining Equations (4.11), (4.13) and (4.17), we obtain the squared covariance-weighted distance constraint for 2D/3D correspondences of vertex $\mathbf{v}_i$ and pixel $\mathbf{p}_k$:

$$\left\| \begin{bmatrix} \mathbf{N}_\ell^x - \mathbf{p}_k^x \mathbf{N}_\ell^z \\ \mathbf{N}_\ell^y - \mathbf{p}_k^y \mathbf{N}_\ell^z \end{bmatrix} \mathbf{v}_i + \begin{bmatrix} \mathbf{T}_\ell^x - \mathbf{p}_k^x \mathbf{T}_\ell^z \\ \mathbf{T}_\ell^y - \mathbf{p}_k^y \mathbf{T}_\ell^z \end{bmatrix} \right\|_{\tilde{\mathbf{H}}_{ik}}^2 = 0 \tag{4.18}$$

This distance constraint can be directly embedded into the linear least squares solver for mesh deformation.

4.5 Skeleton Supported Robust Mesh Adaptation

So far, we have presented a method to deform the surface of a mesh using automatically obtained correspondences between mesh vertices and silhouette contours. We initialize a mesh with roughly the same pose as the user in the images and then match rim vertices with the closest silhouette contour. There is an inherent problem with matching closest contour points: the contour may be occluded by other body parts. Thus, when matching with the closest visible contour, the correspondence may be wrong and there is no way to detect this from silhouette images alone. A typical example of this case is illustrated in Fig. 4.8.

Laplacian surface constraints help to keep surface deformations consistent and minimize the effect of single outliers in the data term. However, in the example shown in Fig. 4.8(c) all correspondences along the upper leg yield a consistent but wrong corre-

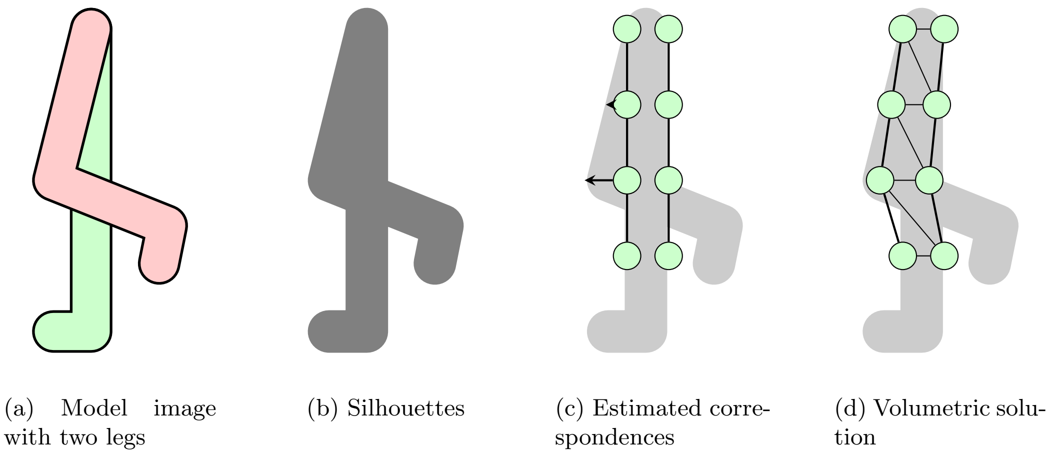


Figure 4.8: Demonstration of how wrong correspondences caused by overlapping silhouettes can cause unnatural deformations.

spondence. Thus, surface optimizations with these correspondences will pull the front of the right (green) upper leg towards the image of the left (red) leg. The vertices on the backside of the leg will find correct correspondences. However, there is no direct connection between vertices in the front and back.

Aguiar et al. [4] use a volumetric mesh consisting of tetrahedra in order to create connections between one side of the surface with the opposite side. A volumetric mesh preserves the volume of limbs. They can, however, not prevent an arbitrary bending of limbs. In the worst case, vertices from the backside of the left leg will get pulled forward as shown in Fig. 4.8(d).

Our solution tries to model the anatomy of the human body. The skin is not directly connected to the backside of a limb or torso but attached to the bones of the skeleton. It is a hybrid method between skeletal subspace deformation and Laplacian mesh editing. While recent work on linearly connecting a mesh surface to a skeleton [95, 169] is limited to handle short bone segments only, our approach is able to handle arbitrary bone lengths. We propose to use the anatomical skeleton of the human body and introduce differential bone coordinates to attach skin vertices to bones. Ideally, a bending of the mesh should only be allowed near skeleton joints, thus allowing only a piecewise rigid deformation.

4.5.1 Differential Bone Coordinates

We introduce differential bone coordinates to connect mesh vertices to a skeleton. A skeleton of the human body model is either provided with the model or can be estimated automatically [18]. The skeleton lies inside the body and consists of joints $\mathbf{g}_j \in \mathbb{R}^3$ and bones b. Each bone connects two joints. Following the hierarchy of bones in the human body, we can assign each bone b_j to exactly one joint $\mathbf{g}_j$ and define that this bone connects to the joint $\mathbf{g}_{prev(j)}$. By definition, $\mathbf{g}_{prev(j)}$ is the bone hierarchically closer to the root of the skeleton, which is the body center. For example, the forearm bone is connected to joint $\mathbf{g}_{\text{wrist}}$ and the previous joint is $\mathbf{g}_{prev(\text{wrist})} = \mathbf{g}_{\text{elbow}}$.

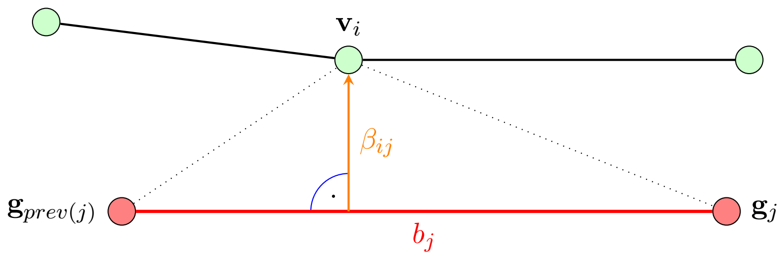


Figure 4.9: Differential bone coordinates β_{ij} for vertex $\mathbf{v}_i$.

Our differential bone coordinates β are similar to δ-coordinates of Equation (4.4). They encode the position of vertex $\mathbf{v}_i$ relative to its connected bones. First, we define the differential bone coordinate β_{ij} for vertex $\mathbf{v}_i$ connected a single bone b_j:

$$\beta_{ij} = \mathbf{v}_i - \left(\gamma_{ij}\,\mathbf{g}_j + (1-\gamma_{ij})\,\mathbf{g}_{prev(j)}\right). \tag{4.19}$$

Each γ_{ij} is chosen such that β_{ij} is orthogonal to bone b_j (see Fig. 4.9):

$$\gamma_{ij} = \frac{1}{2} - \frac{\|\mathbf{v}_i - \mathbf{g}_j\|^2 - \|\mathbf{v}_i - \mathbf{g}_{prev(j)}\|^2}{2\,\|\mathbf{g}_j - \mathbf{g}_{prev(j)}\|^2}. \tag{4.20}$$

Each surface vertex $\mathbf{v}_i$ can be connected to one or multiple bones b_j as shown in Fig. 4.10. This is similar to mesh animation with linear blend skinning where a rigging algorithm such as [18] computes multiple skinning weights ρ_{ij} for each vertex $\mathbf{v}_i$. These weights describe the influence of bone b_j on vertex $\mathbf{v}_i$ ($\sum_j \rho_{ij} = 1$). We use the same weights ρ_{ij} to connect a vertex to multiple bones. A multi-bone assignment is common near joints to support smooth bending. In case of multiple bones, Equation (4.20) is extended to:

$$\mathcal{B}_i(\mathbf{V},\mathbf{G}) = \beta_i = \mathbf{v}_i - \sum_{j=1}^{|\mathbf{G}|} \rho_{ij}\left(\gamma_{ij}\,\mathbf{g}_j + (1-\gamma_{i,j})\,\mathbf{g}_{prev(j)}\right) \tag{4.21}$$

where $|\mathbf{G}|$ is the total number of joints in the skeleton.

4.5.2 Skeleton Constraints for Mesh Deformation

Similar to differential Laplace coordinates in mesh editing, we can enforce that surface vertices maintain a relative position to their connected bones. Therefore, we compute β_i values from an undeformed model and penalize any deviation during shape optimization.

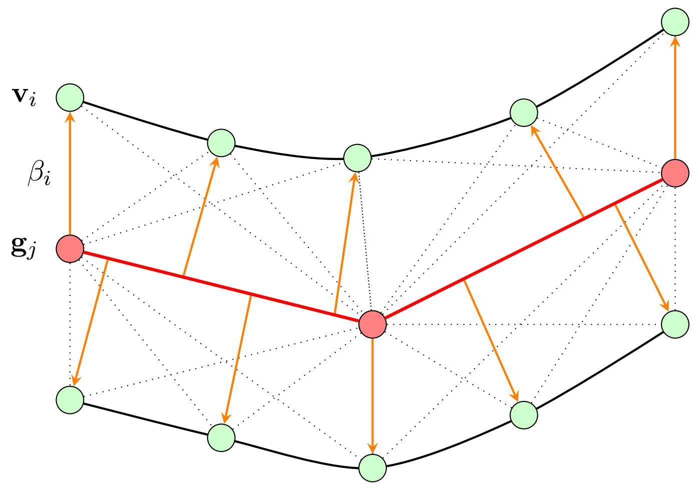


Figure 4.10: Implicit skinning through multi-bone connection of vertices.

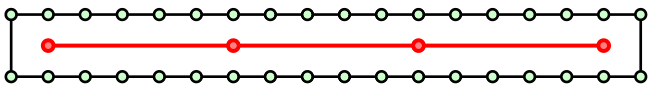
Our skeleton consistency term can be easily integrated into Equation (4.6):

$$\tilde{\mathbf{V}} = \arg\min_{\mathbf{V},\mathbf{G}} \underbrace{\sum_{i=1}^{M} w_i \|\mathbf{v}_i - \mathbf{t}_i\|^2}_{\text{data term}} + \underbrace{\lambda_{\text{skin}} \sum_{i=1}^{N} \|\delta_i - \mathcal{L}_i(\mathbf{V})\|^2}_{\text{skin term}} + \underbrace{\lambda_{\text{bone}} \sum_{i=1}^{N} \kappa_i \|\beta_i - \mathcal{B}_i(\mathbf{V},\mathbf{G})\|^2}_{\text{bone term}} . \tag{4.22}$$

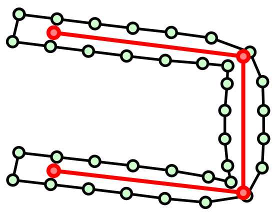
Now, both mesh vertices $\mathbf{V}$ and joint positions $\mathbf{G}$ are optimized together. For example, joint positions are automatically updated when surface vertices are deformed and vice versa. Similar to differential Laplace coordinates, bone coordinates are not rotation and scale invariant. However, we can apply the method of Sorkine et al. [134] to implicitly approximate a transformation matrix for bone coordinates or correct the rotation of bone coordinates after each iteration.

$\kappa_i \geq 0$ are weights that adjust the strength of the binding between surface and skeleton for each vertex individually. This allows for some regions of the mesh to be more rigid than others. A typical application is to optimize a model with partly loose clothing (such as a skirt). There, certain vertices should be able to move independently from the skeleton and have a low weight κ. Stoll et al. [138] show how to learn such weights automatically.

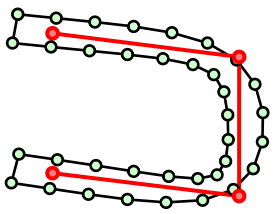
Equation (4.22) can be used for 2D and 3D meshes and allows natural deformations near joints where vertices are affected by more than one bone. In Fig. 4.11, we show some examples for deforming a 2D mesh. In these examples, we compute the initial configuration of δs and βs from the default pose in Fig. 4.11(a). Then, we create some constraints to set the positions of the bones and optimize for the vertex positions. This experiment also demonstrates the effect of rotations and scale when using implicitly optimized transformations of differential coordinates [134]. In contrast to 3D meshes, it is possible to linearly estimate rotation and scale corrections for 2D meshes accurately.



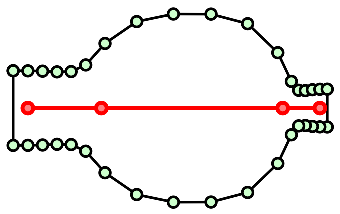
(a) Default pose



(b) $\lambda_{\text{bone}} = 2$



(c) $\lambda_{\text{bone}} = 0.1$



(d) Changing bone lengths

Figure 4.11: Effects of bending and stretching the skeleton on a 2D mesh. The undeformed mesh is shown in Fig. (a). In (b), we show that high values for λ_{bone} cause a stiff surface when bending. (c) A small λ_{bone} produces smoother results due to the skin term. (d) Changing bone lengths leads to scaling of the attached surface.

4.5.3 Efficient Updates of 3D Differential Coordinates with Skeletons

As mentioned before, 3D meshes cause problems when the optimization introduces rotations or scale changes of body parts. The problem is that differential coordinates can model only translations but cannot handle rotations and scale in a linear optimization framework. In Section 4.3.3, we have presented two existing methods to correct differential coordinates. These methods either implicitly or explicitly analyze the deformed surface in order to update the scale and rotation of differential coordinates. Implicit linear methods that estimate a coordinate transformation are well suited for 2D meshes. As three dimensional rotations cannot be fully represented with linear terms, implicit methods can only be used to approximate small rotations for three dimensional meshes [134]. When larger rotations are to be expected, it is necessary to explicitly compute a transformation that corrects for rotations of the surface.

Explicitly computing a transformation can be computationally complex. For example, Lipman et al. [98] estimate a local rotation matrix $\mathbf{R}_i$ for each vertex $\mathbf{v}_i$ by analyzing the 1-ring neighborhood of the vertex. In addition, they require that the rotation matrices of two neighboring vertices $\mathbf{v}_i$ and $\mathbf{v}_j$ are similar ($\|\mathbf{R}_i - \mathbf{R}_j\| \approx 0$). Finally, they apply each rotation $\mathbf{R}_i$ to the corresponding differential Laplace coordinate δ_i in order to update its orientation. A new surface optimization using the updated differential coordinates produces more naturally looking results at rotated surface areas.

Our joint surface and skeleton pose optimization computes updates skeleton joint positions during each iteration. Using updated joint positions, we can efficiently compute

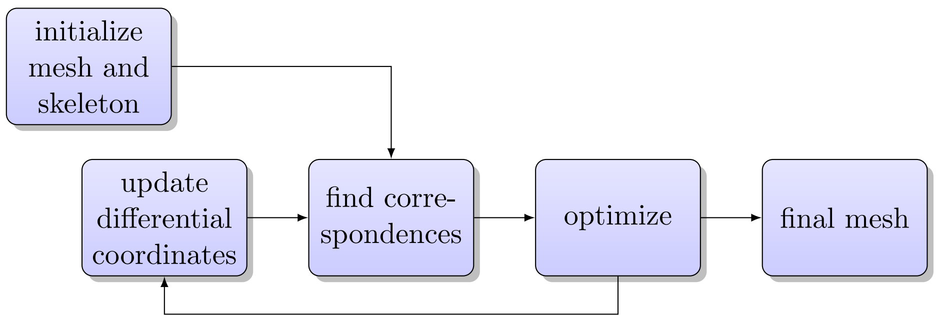


Figure 4.12: We use skeleton supported mesh adaptation to efficiently update differential coordinates.

a rotation matrix for each vertex as well as ensure that neighboring vertices have similar rotation matrices. Our approach is based on the observation that the surface rotation of body parts is always caused by the underlying bones. Therefore, it is not possible that two surface vertices attached to the same bone have different surface rotations when the bone is rotated. In fact, the same principle is used to animate a polygonal mesh using linear blend skinning where only a single transformation matrix per bone is used to animate the complete surface (see Section 4.2.2).

First, we estimate a rotation matrix for every bone. Then, we apply this rotation to the differential coordinates of all connected vertices. As a vertex can be attached to multiple bones, we use linear skinning weights ρ_{ij} to blend together multiple rotations. This allows for smooth changes in rotations near skeleton joints. Our approach performs time consuming computations only once per bone. Since there are significantly less bones than surface vertices in a polygonal model of a human body, this allows for a significant speedup compared to estimating and smoothing rotations for every vertex.

In image-based surface adaptation, we can integrate explicit differential coordinate updates into the optimization process. Fig. 4.12 shows that after every surface optimization iteration, we update differential coordinates. In order to estimate rotation matrices for bones, we need to analyze the joint positions of the skeleton before and after a optimization. Each bone b_j has a joint position $\mathbf{g}_j$ and a previous joint position $\mathbf{g}_{prev(j)}$ (see Fig. 4.9). Thus, we compute a direction vector for each bone as:

$$\xi_j = \frac{\mathbf{g}_j - \mathbf{g}_{prev(j)}}{\|\mathbf{g}_j - \mathbf{g}_{prev(j)}\|}. \tag{4.23}$$

We compute the rotation $\mathbf{q}_j$ that rotates the initial bone direction ξ_j^0 onto the current bone direction $\tilde{\xi}_j$. As we want to blend together multiple rotations near joints, we choose the quaternion representation because multiple rotations can be linearly combined [88]. A quaternion can be efficiently computed using the axis-angle notation:

$$\mathbf{a}_j = \xi_j^0 \times \tilde{\xi}_j \qquad \varphi_j = \arctan\left(\frac{\left\langle \xi_j^0, \tilde{\xi}_j \right\rangle}{\|\mathbf{a}_j\|}\right) \qquad \mathbf{q}_j = \left[\sin\left(\frac{\varphi_j}{2}\right) \cdot \frac{\mathbf{a}_j^T}{\|\mathbf{a}_j\|}, \cos\left(\frac{\varphi_j}{2}\right)\right]^T \tag{4.24}$$

Finally, we compute a rotation $\mathbf{q}_i$ for each vertex as the weighted sum of rotations of each bone $\mathbf{q}_j$:

$$\mathbf{q}_i = \sum_{j=1}^{|\mathbf{G}|} \rho_{ij} \mathbf{q}_j \tag{4.25}$$

where ρ_{ij} denote the linear skinning weights associated with vertex $\mathbf{v}_i$. There is no need to normalize the quaternion $\mathbf{q}_i$ as $\sum_{j=1}^{|\mathbf{G}|} \rho_{ij} = 1$. We apply the rotation quaternion $\mathbf{q}_i$ to both the differential Laplacian surface coordinates δ_i as well as differential bone coordinates β_i in order to update their rotations. In addition, we use these rotations to update surface covariances which are needed for weighted surface correspondence (see Section 4.4.2).

4.5.4 Preserving Skeleton Symmetries

The quadratic energy minimization from Equation (4.22) can be efficiently solved using efficient linear system solvers [27]. The optimization problem can be re-written as:

$$\underset{\mathbf{V},\mathbf{G}}{\arg\min} \left\| \begin{bmatrix} \mathbf{S} \\ \lambda_{\text{skin}} \mathbf{L} \\ \lambda_{\text{bone}} \mathbf{B} \end{bmatrix} \cdot \begin{bmatrix} \mathbf{V} \\ \mathbf{G} \end{bmatrix} - \begin{bmatrix} \mathbf{T} \\ \lambda_{\text{skin}} \mathbf{\Delta}_L \\ \lambda_{\text{bone}} \mathbf{\Delta}_B \end{bmatrix} \right\|^2 = \underset{\mathbf{V},\mathbf{G}}{\arg\min} \left\| \mathbf{C} \cdot \mathbf{x} - \mathbf{d} \right\|^2 \tag{4.26}$$

where $\mathbf{S}$, $\mathbf{L}$ and $\mathbf{B}$ correspond to the sparse matrices that implement the linear constraints. $\mathbf{T}$, $\mathbf{\Delta}_L$ and $\mathbf{\Delta}_B$ signify the target positions, initial differential Laplace coordinates and initial differential bone coordinates.

However, there is no constraint that controls the lengths of bones (i.e. $\|\mathbf{g}_j - \mathbf{g}_{prev(j)}\|$). Thus, the skeleton can change in size if joint positions $\mathbf{G}$ move relative to each other. In situations when there is noisy or occluded image data, the stability of mesh deformation can be increased when the lengths of bones remain constant. It is possible to use quadratic equality constraints during optimization which keep the lengths of bones constant or enforce symmetric bones during shape adaptation:

$$\begin{aligned} &\underset{\mathbf{V},\mathbf{G}}{\arg\min} && \frac{1}{2}\mathbf{x}^T(\mathbf{C}^T\mathbf{C})\mathbf{x} + (\mathbf{C}^T\mathbf{d})\mathbf{x} \qquad \text{with} \quad \mathbf{x} = \begin{bmatrix} \mathbf{V} \\ \mathbf{G} \end{bmatrix} \\ &\text{subject to} && \frac{1}{2}\mathbf{x}^T\mathbf{E}_k\mathbf{x} = e_k \qquad \text{for} \quad k = 1 \dots K \end{aligned} \tag{4.27}$$

where $\mathbf{C}$ and $\mathbf{d}$ are the combined matrices from Equation (4.26).

$$\mathbf{x} = [v_1^x, v_1^y, v_1^z, v_2^x, \ldots, v_{|\mathbf{V}|}^z, g_1^x, g_1^y, \ldots, g_{|\mathbf{G}|}^z]^T \tag{4.28}$$

contains vertex and joint positions. For example, the length of a single bone can be fixed to e_k using an equality constraint

$$\|\mathbf{g}_j - \mathbf{g}_{prev(j)}\|^2 = e_k \tag{4.29}$$

where the squared Euclidean distance operator can be expressed as a symmetric matrix $\mathbf{E}_k$. Similarly, body symmetry (*e.g.* bones with equal length) can be expressed as

$$\|\mathbf{g}_a - \mathbf{g}_{prev(a)}\|^2 = \|\mathbf{g}_b - \mathbf{g}_{prev(b)}\|^2. \tag{4.30}$$

A quadratically constrained quadratic problem (4.27) cannot be solved using a linear solver directly. Thus, we use the iterative Sequential Quadratic Programming (SQP) algorithm [113]. This algorithm iteratively solves a (sparse) symmetric linear system of equations and therefore increases the time required for solving (4.27) only by a linear factor.

4.6 Multigrid Optimization for Mesh Adaptation

4.6.1 Introduction

Mesh adaptation from image correspondences is an iterative process which alternates searching for correspondences and mesh optimization (see Fig. 4.2). So far, we have considered working with the same resolution of the mesh at every iteration. This means that we optimize a highly detailed surface even though for early iterations it is unlikely that correspondences are correct. Therefore, we propose to use a coarse to fine approach for shape adaptation which first optimizes a mesh with only a few hundred vertices and then gradually increases the number of vertices.

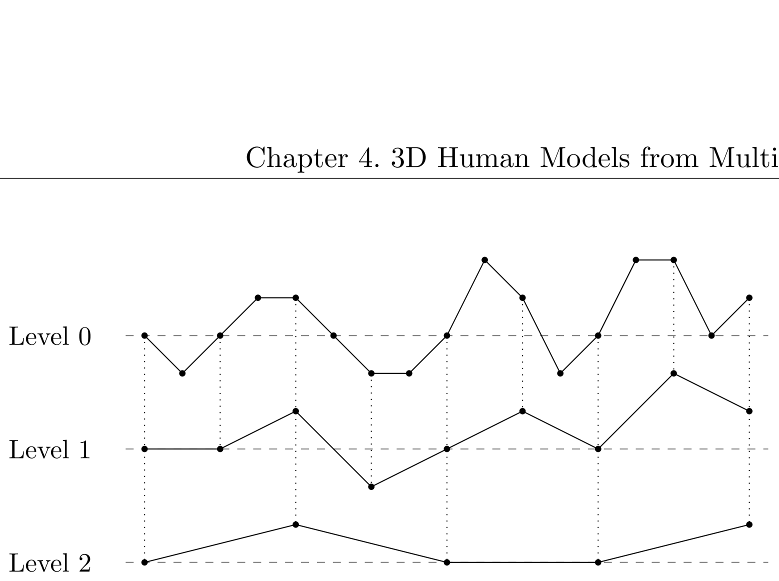
Coarse to fine approaches are well known in a variety of computer vision problems. Examples for multi-resolution analysis are scale space theory [97], feature tracking [101] and real-time optical flow computations [168]. Instead of changing the resolution of input images, we change the resolution of our model mesh. There exist methods to compute an arbitrary low resolution representation of meshes [59, 80]. Usually, such methods generate a data structure that allows fast down and up sampling at run-time. This makes them perfectly suited for adaptive rendering where each mesh is rendered at the optimal resolution for a given distance to the camera. Multi-resolution meshes can also help to speed up the process of mesh deformation [126].

One might wonder why a multi resolution approach helps when the globally optimal deformed mesh can be obtained by a single call of the optimizer. Highly optimized linear least squares solvers such as supernodal Sparse Cholesky factorization [38] can easily optimize more than 100 000 vertices. But even with such sophisticated solvers, the solving time becomes too slow for interactive applications. A possible solution is to use iterative solvers based on the Gauss-Seidel or conjugate gradient algorithm [27]. However, due to the sparse connectivity between vertices, convergence can be rather slow.

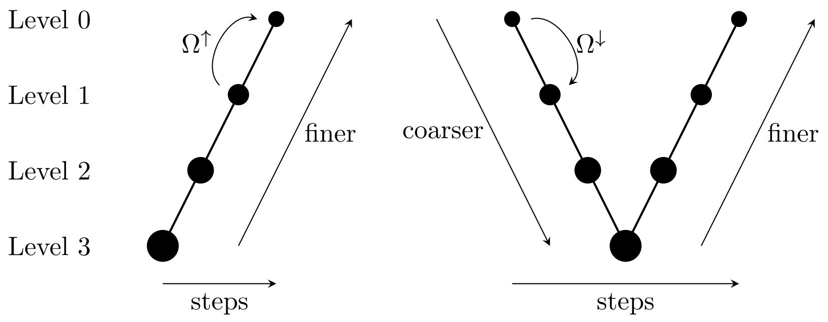
Multigrid methods are a group of algorithms that use iterative optimization techniques efficiently through multi-resolution analysis [30]. These methods are based on the assumption that the residual value of an error function can be decomposed into low frequency and high frequency errors. Many iterative solvers reduce high frequency errors quickly while overall convergence is slow. The main idea to speed up convergence is to reduce low frequency residuals by performing optimization on a lower resolution representation of the model. High frequency errors are eliminated at a more detailed resolution level, afterwards. The resolution of the model can be changed using restriction and prolongation operators to decrease and increase the resolution of a grid, respectively. Multigrid methods can operate on arbitrary grids of interconnected nodes. Although highly irregular, polygonal meshes are a grid structure that is well suited to be processed by multigrid methods [27].

There exist two types of optimization schemes commonly found in multigrid methods (see Fig. 4.13): A *cascading multigrid* is initialized with the coarsest representation of the grid. This implies that it must be possible to generate an initial coarse solution without knowing the highest resolution. An optimizer solves the problem at the coarsest level of resolution and therefore reduces the low frequency parts of the overall error. After an intermediate solution is obtained, the next level of resolution is initialized through the prolongation operator $\Omega^{\uparrow}$. The optimizer is called again at this finer grid resolution to further improve the solution. Usually, the number of solver iterations at each level can be decreased as the resolution gets finer [26]. At the finest level, the original problem is solved on the full grid. In a *V-cycle* scheme, optimization starts on the full grid at level 0. A restriction operator $\Omega^{\downarrow}$ first decreases the resolution of the grid until the coarsest resolution in reached. At each intermediate level, the optimizer can refine the solution. After reaching the coarsest level of resolution, optimization is the same as for the cascading multigrid. A full multigrid contains multiple V-cycles.

In this thesis, we propose to use a cascading multi-grid approach to speed up the process of adapting a polygonal mesh $\mathscr{M}$ to silhouette images of a human body. Similar to single resolution mesh adaptation in Section 4.4, multigrid mesh adaptation consists of four phases: (1) finding correspondences between the mesh and the input images, (2) setting up the constraints, (3) preparing the optimizer and (4) the actual shape optimization. We initially optimize the mesh at the coarsest resolution level. This optimization is followed by upsampling the resulting mesh to the next finer level using the prolongation operator $\Omega^{\uparrow}$. At the finer level, shape refinement is continued. We iteratively increase the mesh resolution while refining the deformed result until we reach the full resolution. A coarse resolution in the beginning mainly helps to pull a mesh surface to the right place



(a) Three levels of grid resolution



(b) Cascading multigrid (c) V-cycle multigrid

Figure 4.13: Multigrid methods can be used to optimize a grid at multiple levels of resolution (a). A cascading multigrid (b) starts optimization at the coarsest level, refines the solution and then increases the resolution of the model. In contrast, a V-cycle scheme (c) is initialized with a full resolution grid. First the resolution is decreased before it is increased again.

(*e.g.* pull the arms to the correct location according to input images). At finer resolutions, we can adapt the mesh to details such as wrinkles in clothing.

4.6.2 Changing the Resolution of the Mesh

In order to apply multigrid methods, the operators for coarsening and refining a mesh need to be defined. A restriction operator $\Omega_n^{\downarrow}$ reduces the resolution from a fine-grained mesh $\mathscr{M}_n$ to a coarser mesh $\mathscr{M}_{n+1}$. The index n denotes the current level of the mesh and $\mathscr{M}_0$ denotes the initial level at full resolution. The restriction operator selects some vertices from the fine mesh $\mathscr{M}_n$ that can be discarded such that the coarser mesh $\mathscr{M}_{n+1}$ contains only a subset of all vertices. The prolongation operator $\Omega_n^{\uparrow}$ is the inverse of the restriction operator. It takes a coarse mesh $\mathscr{M}_{n+1}$ and computes the finer grained mesh $\mathscr{M}_n$ from it. Therefore, it needs to interpolate the positions of vertices that are not present at level $n+1$.

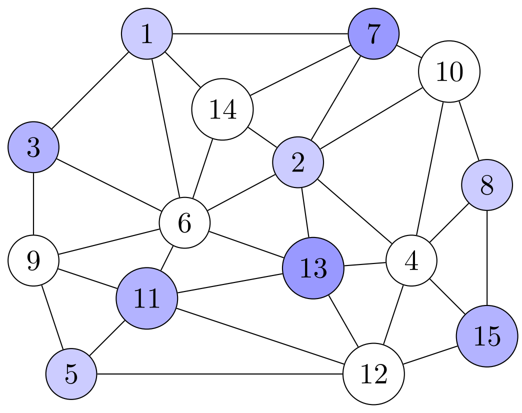
Our definition of the restriction operator $\Omega_n^{\downarrow}$ and the prolongation operator $\Omega_n^{\uparrow}$ is based on Shi et al. [126]. Given a mesh $\mathscr{M}_n$ consisting of vertices $\mathbf{V}^n$ and a set of edges $\mathbf{E}^n$, we select coarse vertices based on Algorithm 1. This algorithm iteratively marks vertices as coarse until each unmarked vertex has at least a number of $\eta_{\min}$ coarse 1-ring neighbors. From this selection, the operator $\Omega_n^{\downarrow}$ produces a set of coarse vertices $\mathbf{V}^{n+1}$ for the next level. These vertices will be present in mesh $\mathscr{M}_{n+1}$. All vertices that will not be present in level $n+1$ are guaranteed to have at least $\eta_{\min}$ coarse neighbors at level n. Typically, we require a minimum of $\eta_{\min} = 3$ neighbors such that the prolongation operator can perform interpolation. We show an example for this selection algorithm in Fig. 4.14(a). Note that there is no preferred ordering in which the vertices are accessed. We use the order given by the vertex index. In addition, it is possible to ensure that certain vertices never get removed in a coarser mesh by setting the corresponding *marked*-flag to *true* before selecting discardable vertices.

Algorithm 1 Selecting vertices for the next level.

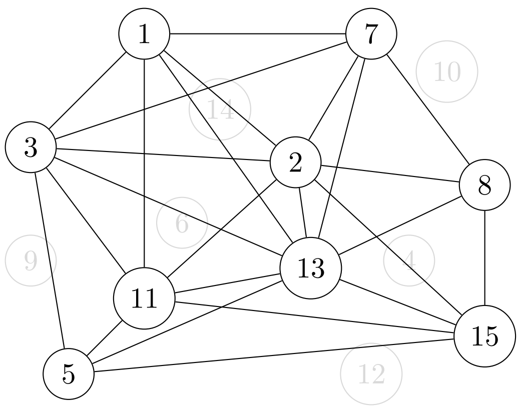
```
1:  for all vertices $v$ do
2:      marked($v$) = false
3:  end for
4:  for $t = 1$ to $\eta_{\min}$ do
5:      for all vertices $v$ do
6:          if not marked($v$) then
7:              $\mathbf{r}$ = GetNeighbors($v$)
8:              if count(marked($\mathbf{r}$)) $< t$ then
9:                  marked($v$) = true
10:             end if
11:         end if
12:     end for
13: end for
14: return marked
```

In order to maintain connectivity of the mesh at level $n+1$, we need to create a new set of edges $\mathbf{E}^{n+1}$. For each vertex marked as coarse at level n, we create an edge to every coarse vertex within its 1- and 2-neighborhood. This remeshing is demonstrated in Fig. 4.14(b). Note that the resulting connectivity is neither a triangular mesh nor a planar graph. This will have an effect on how to compute Laplacian mesh coordinates.

The prolongation operator $\Omega_n^{\uparrow}$ restores the vertices $\mathbf{V}^n$ from a set of coarser vertices $\mathbf{V}^{n+1}$. Therefore, the positions of all vertices at level $n+1$ are copied to their corresponding vertices in $\mathbf{V}^n$. The positions of vertices that have been removed by the restriction operator $\Omega_n^{\downarrow}$ need to be interpolated. Similar to [126], we use a linear interpolation function to compute the new position for each vertex as the weighted sum of their 1-ring neighbors.



(a) Level 0



(b) Level 1

Figure 4.14: Demonstration of the restriction operator applied on the mesh in (a). Blue nodes are selected for the next level and white nodes have a minimum number of $\eta_{\min} = 3$ neighbors. In (b), the next level is displayed with its new edges.

$$\mathbf{V}^{n+1} = \Omega_n^{\downarrow}\mathbf{V}^n \qquad \mathbf{V}^n = \Omega_n^{\uparrow}\mathbf{V}^{n+1} \tag{4.31}$$

Restriction and prolongation operators are both linear operators which can be represented as sparse matrices. Thus, changing the level of a mesh is a simple sparse matrix-vector multiplication, which has a low computational cost. As long as the topology of the original mesh is not altered, there is no need to change these operators for subsequent input frames. However, the interpolation weights of the prolongation operator need to be updated if the vertex positions of the mesh at level 0 are changed. This is the case when the mesh $\mathscr{M}^0$ is pre-deformed using pose estimation results, for example.

4.6.3 Laplacian Mesh Adaptation with Multigrid Meshes

Mesh adaptation with multigrid meshes is similar to traditional mesh adaptation at a single resolution. However, the number of vertices and the connectivity changes at every resolution level. To compute differential vertex coordinates at every level, a different Laplace operator $\mathcal{L}^n$ is required. In addition, there is one important modification required for coarsened meshes $\mathscr{M}^{n>0}$: It is no longer possible to compute co-tangent Laplacian weights using Equation (4.5) because coarsened meshes do not consist of triangular faces anymore. Instead, each vertex has a set of connected neighbors $\mathscr{N}_i$ defined via the set of edges $\mathbf{E}^n$. Instead of co-tangent weights, we use a uniform weighting scheme [134] for all levels $n > 0$:

$$\mathcal{L}_i^n(\mathbf{V}) = \mathbf{v}_i - \frac{1}{|\mathscr{N}_i|}\sum_{j\in\mathscr{N}_i}\mathbf{v}_j. \tag{4.32}$$

We propose to perform a cascading multigrid approach for image-based mesh adaptation. First, the mesh has to be initialized at the coarsest level and correspondences need to be found between vertices and image features. Then, the LME framework (see Section 4.3) can be used to optimize for new vertex positions at this level. Afterwards, we use the prolongation operator $\Omega_n^{\uparrow}$ to increase the resolution and interpolate the positions of remaining vertices. We repeat the process of finding correspondences and shape optimization at the finer level. At level $n = 0$, it is possible to use the co-tangent Laplace weights from Equation (4.5) again and perform the final surface optimization.

One key advantage of multi-grid methods is that the prolonged surface from a coarser level $n + 1$ is a good initialization for iterative solvers such as conjugate gradient [19, 27]. For large meshes, using iterative solvers can be more efficient than to perform sparse matrix decompositions. At intermediate levels, iterative solvers do not need to run until convergence as optimization is continued at a finer level. Thus, the number of overall iterations can be kept low.

4.6.4 Skeleton Supported Multigrid Meshes

In Section 4.5, we have introduced skeleton constraints for mesh adaptation. They are based on differential bone coordinates that connect every vertex to one or multiple bones. Since each bone constraint references only a single vertex, skeleton constraints can be used at every level of multigrid meshes without modifications. For every resolution level, we can use the same skeleton with the same joints. Therefore, there is no need to define separate skeleton restriction and prolongation operators. However, care has to be taken that if a vertex is not present at the current level, the corresponding skeleton constraint is not applied.

There is a hidden problem with shape adaptation that arises when using our multigrid method: It is no longer possible to compute vertex normals by averaging face normals as described in Section 4.4 because the triangular face structure is lost at coarser levels. This impacts both the ability to determine rim vertices and finding silhouette correspondences along normal directions. When using a skeleton supported multigrid mesh, we propose to initialize vertex normals at the full resolution mesh. Then, we use rotation quaternions $\mathbf{q}_i$ defined in Section 4.5.3 to correct the normal directions after each iteration of surface optimization. This method allows us to compute vertex normals for every resolution level even though the mesh does not consist of triangular faces.

4.6.5 Discussion

The multi-grid approach presented in this section enables a significant reduction of optimization time. Only a reduced number of vertices need to be optimized in early iterations of the shape adaptation process. Global solvers such as sparse Cholesky decomposition can be inefficient for large meshes. Therefore, they are only suited for optimizing the low resolution levels of our multi-grid approach.

Iterative methods such as the conjugate gradient method [19] allow a significant speedup when the number of iterations is low. However, they require a decent initialization for fast convergence. In our multi-grid framework, we provide such an initialization by optimizing the shape at lower resolutions first. Then, we interpolate initial vertex positions for iterative optimization of a finer level.

4.7 Decoupled Constraint Solving for Mesh Adaptation

Even though multigrid methods promise a significant speedup compared to single-resolution methods, shape optimization of full-resolution meshes still poses a performance bottleneck. However, fast mesh adaptation is essential for real-time operation. Therefore, we propose an iterative mesh optimization method that has a low computational workload, yet provides high quality results.

Our method is inspired by position-based physics simulations [108, 109]. Such simulations can be used to compute realistic interactions between soft bodies in real-time (*e.g.* cloth simulation). A deformable model of the human body is very similar to a soft body. Instead of simulating physical forces that act on individual vertices (*e.g.* gravity), we apply silhouette constraints on selected vertices to align them with corresponding silhouette contours that are found in input images.

The key to real-time operation is to apply such constraints in a decoupled manner and optimize one constraint at a time. For example, one optimization step pulls a single vertex towards a corresponding silhouette contour while another step enforces a single smoothness constraint. This optimization is similar to the Gauss-Seidel (GS) optimization scheme and stochastic gradient descent (SGD) algorithms. It is a trade-off between quality and speed and allows for extremely fast computation of vertex updates. This optimization scheme is able to decrease the initial error quickly, but shows a slower overall convergence. We show that decoupled optimization is suitable for mesh deformation guided by image-space correspondences and that the results are qualitatively comparable to traditional solvers such as direct Cholesky decomposition and conjugate gradient.

4.7.1 Constraint-based Mesh Deformation

Similar to the previous sections, we consider the problem of deforming a polygonal mesh $\mathscr{M} = \{\mathbf{V}, \mathbf{F}\}$ consisting of vertices $\mathbf{V}$ by using the Laplace mesh editing framework and image correspondences. We control the deformation and shape consistency through constraints:

$$C_j(\mathbf{V}|\Phi_j) = 0 \qquad 1 \leq j \leq M. \tag{4.33}$$

Each constraint is a function $C_j : \mathbb{R}^{3\times V} \rightarrow \mathbb{R}$ with a set of parameters Φ_j that encodes a relationship between selected vertices with other vertices of $\mathscr{M}$ or the scene. For example, a constraint can be responsible for aligning the mesh with image data. We use the parameters Φ_j for storing constraint properties such as initial curvature or correspondences with images. Usually, these parameters are initialized before optimization. The vertex positions of the deformed mesh can be obtained by minimizing over all constraints:

$$\tilde{\mathbf{V}} = \underset{\mathbf{V}}{\operatorname{argmin}} \sum_{j=1}^{M} C_j(\mathbf{V}|\Phi_j). \tag{4.34}$$

Note that such constraints need not be linear but have to be differentiable.

Inspired by the GS optimization scheme for linear systems of equations [19], we do not minimize Equation (4.34) as a whole. Instead, we break it down into individual constraints and project each C_j onto the vertices independently. We use a first-order Taylor series expansion to find a position-correction term $\Delta\mathbf{V}_j$ such that

$$C_j(\mathbf{V} + \Delta\mathbf{V}_j) \approx C_j(\mathbf{V}) + \nabla_{\mathbf{V}} C_j(\mathbf{V}) \cdot \Delta\mathbf{V}_j = 0 \tag{4.35}$$

where $\nabla_{\mathbf{V}} C_j$ denotes the gradient of constraint j. Solving for $\Delta\mathbf{V}_j$ yields the step for the iterative minimization

$$\Delta\mathbf{V} = -\frac{C_j(\mathbf{V})}{\|\nabla_{\mathbf{V}} C_j(\mathbf{V})\|^2} \cdot \nabla_{\mathbf{V}} C_j(\mathbf{V}) \tag{4.36}$$

which is the standard Newton-Raphson update step. We use Equation (4.36) to perform a weighted correction of the current vertex positions $\mathbf{V} \leftarrow \mathbf{V} + k_j \Delta\mathbf{V}_j$ for every constraint C_j. $k_j \in [0, 1]$ is the step size for the gradient descent. Following Müller et al. [109], we use a modified step size $k_j{}' = 1 - (1 - k_j)^{1/N^i}$ which allows projecting constraints with linear dependence on the number of iterations N^i.

Analog to the GS scheme, we use updated values of $\mathbf{V}$ for subsequent calculations as soon as available. This requires less memory and allows the solution to converge faster while keeping time complexity linear in the number of constraints. By iterating constraint projection multiple times, we allow the effect of constraints to propagate along the surface of the mesh until all vertices of the deformed mesh reach a stable position.

4.7.2 Constraints for Skeleton-Based Mesh Adaptation

The presented solver is capable of handling nonlinear constraints of any type. We propose to use the constrains described in this chapter to perform decoupled constraint-based mesh adaptation. Silhouette constraints C^{sil} align rim vertices of a template mesh with silhouette contours in the images. The surface constraint C^{skin} and the skeleton constraint C^{bone} act as a regularization term. This allows Equation (4.34) to be rewritten as

$$\tilde{\mathbf{V}}, \tilde{\mathbf{G}} = \underset{\mathbf{V},\mathbf{G}}{\operatorname{argmin}} \sum_{j=1}^{M} C_j^{\text{sil}}(\mathbf{V}) + \lambda_{\text{bone}} \sum_{j=1}^{N} C_j^{\text{bone}}(\mathbf{V}, \mathbf{G}) + \lambda_{\text{skin}} \sum_{j=1}^{N} C_j^{\text{skin}}(\mathbf{V}). \tag{4.37}$$

Similar to the previous sections, we define the constraints as quadratic errors such that the derivatives are easy to compute. More specifically, the silhouette consistency

Algorithm 2 Constraint projection algorithm.

```
Require: $\mathbf{V} = \{\mathbf{v}_i \dots \mathbf{v}_{|V|}\}$
1: $\{\Phi_1 \dots \Phi_M\} \leftarrow$ initialize($\mathbf{V}$)
2: for number of outer iterations $N^o$ do
3:   $\{\Phi_1 \dots \Phi_M\} \leftarrow$ update($\mathbf{V}$, $\Phi_1 \dots \Phi_M$)
4:   for number of inner iterations $N^i$ do
5:     for $j = 1 \dots M$ do
6:       $\mathbf{V} \leftarrow \mathbf{V} - k_j{}' \frac{C_j(\mathbf{V}|\Phi_j)}{\|\nabla_{\mathbf{V}} C_j(\mathbf{V}|\Phi_j)\|^2} \cdot \nabla_{\mathbf{V}} C_j(\mathbf{V}|\Phi_j)$
7:     end for
8:   end for
9: end for
```

constraint C_j^{sil} is defined similar to Equation (4.11) as:

$$C_i^{\text{sil}}(\mathbf{V}|\mathbf{p}_i) = \left\| \begin{bmatrix} \left(\mathbf{N}_\ell^1 - \mathbf{p}_i^x \mathbf{N}_\ell^3\right) \mathbf{v}_i + \left(T_\ell^1 - \mathbf{p}_k^x T_\ell^3\right) \\ \left(\mathbf{N}_\ell^2 - \mathbf{p}_i^y \mathbf{N}_\ell^3\right) \mathbf{v}_i + \left(T_\ell^2 - \mathbf{p}_k^y T_\ell^3\right) \end{bmatrix} \right\|^2 . \tag{4.38}$$

The surface constraint is equivalent to the constraint defined in Section 4.3 and minimizes the deviation of the surface from its initial value (δ_i):

$$C_i^{\text{skin}}(\mathbf{V}|\delta_i) = \left\| \left(w_i \mathbf{v}_i - \sum_{j \in \mathcal{N} i} w_{ij} \mathbf{v} j \right) - \delta_i \right\|^2 . \tag{4.39}$$

Finally, a bone constraint ensures that the skeleton does not get detached from the skin surface during optimization. Therefore, we use the skeleton constraints defined in Section 4.5.

$$C_i^{\text{bone}}(\mathbf{V}, \mathbf{G}|\beta_i) = \left\| \left(\mathbf{v}_i - \sum_{j=1}^{|\mathbf{G}|} \rho_{ij} \left(\gamma_{ij}\, \mathbf{g}_j + (1 - \gamma_{i,j})\ \mathbf{g}_{prev(j)} \right) \right) - \beta_i \right\|^2 \tag{4.40}$$

The main idea of the proposed solver is to minimize each constraint in a sequential manner. To achieve smooth results, we suggest to process silhouette constraints (they represent the data term) before skin and bone constraints (the regularization constraints). Moving selected vertices closer to silhouette contours first creates bumps and spikes on the surface. The subsequent skin and bone constraints affect neighbors of these vertices and ensure that the overall surface becomes smooth again.

4.7.3 The Iterative Solver

Our iterative solver for initializing and updating constraint parameters $\Phi_1 \dots \Phi_M$ and projecting constraints $C_1 \dots C_M$ is outlined in Algorithm 2. In Line 1, we set up all constraints using the initial vertex positions estimates (i.e. we calculate δ_i and β_i). The solver contains two loops: the outer loop (Line 2) is executed N^o times and controls how often constraint parameters are updated (i.e. matching of rim-vertices with the silhouette contour) while the inner loop in Line 4 projects the constraints. Since constraints are projected independently of each other, the number of inner iterations N^i influences how far the effect of each constraint can propagate along the surface of the mesh.

The constraint projection in Line 6 prohibits parallelization because each calculation depends on the updated values $\mathbf{V}$ of the previous projection. When a parallel processing architecture such as a GPU is available, it is possible to compute the update step $\Delta\mathbf{V}$ from the same vertex positions $\mathbf{V}$ for all constraints in parallel. However, care has to be taken that a vertex position is not updated simultaneously by multiple threads [112]. In addition, the number of inner iterations N^i needs to be increased since the convergence rate is slower compared to the Gauss-Seidel type solver. For our evaluation in Section 5.4.2, we did not implement a parallel solver. Therefore, parallel decoupled constraint solving remains future work.

4.8 Summary and Discussion

We have presented a novel method to adapt the surface of an articulated mesh to capture geometric details from silhouette images. The mesh is attached to a skeleton using linear skinning weights that are obtained through rigging methods such as [18]. In contrast to existing approaches [57, 158], our formulation has the key advantage that bones and vertices can be refined simultaneously. Our constraint formulation uses purely linear optimization methods, which allows efficient deformation even of large meshes.

In Section 5.4, we show that the skeleton constraint increases robustness towards outliers and occlusions as it only allows physically plausible deformations. The strength of our method is especially prominent when the number of cameras is reduced. Traditional approaches lead to unnatural deformations when there are not sufficient input views. However, our approach still yields realistic results in the same situation. One might wonder why we want to reduce cameras? A lower number of cameras saves hardware costs and reduces processing times. More important to our goal of building a virtual dressing room is the fact that the recording room cannot be arbitrary large and cameras are usually very close to the person. Thus, cameras are usually focused on a small part of the person and not every camera will see the entire body.

Mesh adaptation is an iterative procedure. It alternates between projecting the current mesh into images to find correspondences, and optimization using the found correspondences. Our novel multi-grid solver minimizes costly shape optimization times by reducing the amount of mesh vertices that need to be processed in early iterations of adaptation. Especially in early iterations, there is no need for a high resolution adaptation as correspondences are not yet accurate. Thus, there is no loss in quality compared to optimization with a full-resolution mesh at every iteration.

By using our decoupled constraint solver, the processing time can be further reduced. In Section 5.4.2, we compare the processing times of several state-of-the-art linear solvers with our method. The outcome of our experiments is that a polygonal mesh can be adapted to camera images at camera frame rate. This allows us to use our algorithm for interactive applications. For example, one can use the adapted mesh for rendering or as a collision object in physical simulations.

Even though we improve over state-of-the-art methods in terms of robustness and adaptation speed, there are some limitations to our approach. We rely on a fairly accurate initialization of the skeleton joints to create an initial mesh. Small displacements of joints can be handled without loss of quality since the mesh automatically gets pulled towards the silhouette contour. However, if the displacement is too large or completely wrong, the search for silhouette contours will fail and no silhouette constraints can be generated for affected vertices.

Another problem with the contour-based approach is that an occluded contour does not allow for a valid match. Similar to pose estimation, our shape adaptation method cannot accurately handle cases where limbs are held tightly near the body. While the skeleton constraint is able to prevent unnatural deformations, the adapted mesh may not represent the actual surface in occluded areas. A possible solution is to use color information to detect feature points on the texture of the body surface [57]. However, tracking and matching of such points has a negative impact on runtime performance. The intersection of multi-view silhouettes produces an estimate of the convex visual hull of an object [53]. Our shape adaptation is essentially similar to visual hull modeling. This means that we cannot recover concave regions from silhouette images alone. Concavities that are found in the resulting mesh have been baked into the template mesh beforehand.

Our approach cannot adapt the body shape if the user wears substantially different clothing than the template mesh (*e.g.* a skirt). This is a limitation shared by all model-based shape adaptation methods. In this case, a specialized template with similar clothing can be used. In a virtual dressing room scenario, we require that the person wears tight fitting clothing. Thus, using a single template mesh is usually sufficient. In the next chapter, we show several examples where we successfully adapt a single template mesh to a variety of users.

CHAPTER 5 EXPERIMENTS

Contents

5.1 Introduction

In the previous chapters, we have described our algorithms and methods to estimate the human pose and shape from multi-view images. Our goal is to execute these algorithms in real-time on a single computer. This means that capturing and preprocessing of camera images, pose estimation and shape adaptation must be finished before a new set of images is recorded. Only when these time limits are adhered to, we can use our methods in an interactive setting such as a virtual dressing room. We not only provide a runtime analysis but also show that our approach matches or improves existing approaches in terms of quality and robustness.

First, we present our hardware setup and computing platform in Section 5.2. This setup allows us to record a synchronized video stream from ten color cameras on a single computer. We give a short overview of our calibration procedure that quickly and fully automatically configures the positions of each camera relative to a static calibration target.

Then, we evaluate the novel pose and shape estimation methods presented in this thesis. Our graph-based human pose estimation algorithm is a promising method for fast and reliable estimation of the human pose from multi-view images. Reliable and automatic initialization is an important aspect for interactive settings. Therefore, we conduct several experiments to show how we can process long sequences of multi-view videos without losing track of the moving person. In addition, we perform experiments to estimate the accuracy of our results compared to ground truth data.

Human shape estimation goes further than pose estimation as it models the visible outer surface of a person with a polygonal mesh. We estimate the surface from multi-view silhouettes of a person using a novel skeleton constraint and an efficient multi-grid solver. A natural measure for the quality of an adapted mesh is how well it agrees with the silhouettes in all input images. The silhouette overlap measure counts all input silhouette pixels that are different from the reprojected adapted model image. We evaluate how our skeleton constraints improve the quality of the model. Existing shape estimation methods have not been developed with real-time performance in mind. In an interactive setting, we require fast processing times. Therefore, we evaluate the runtime of our multi-grid solver and show its feasibility for real-time operation.

Finally, we evaluate the complete pose and shape estimation process. We present example applications where real-time mesh adaptation is beneficial. A special emphasis is put on applications where a user can interact with his personal avatar in a 3D environment.

5.2 Experimental Setup

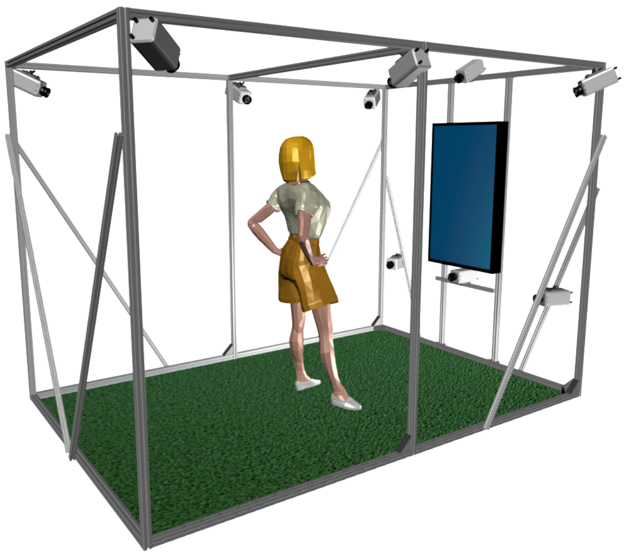
In this section, we describe the prototype hardware system that we use for recording multi-view images of a person. The setup consists of a camera setup connected to a single computer. Moreover, we provide details for our camera calibration and image pre-processing methods.

5.2.1 The Recording Studio

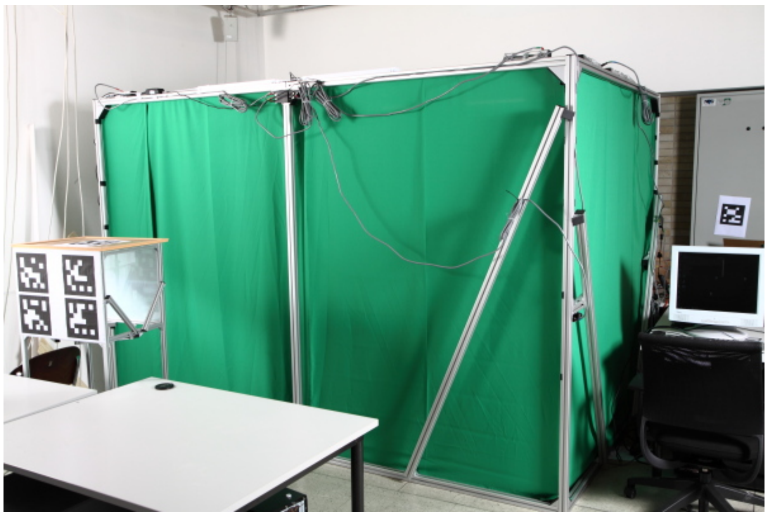
One goal of this thesis is to build a virtual dressing room that can be placed in an ordinary store. Such an installment should be small and must comply with existing features of the room it will be placed in. This means that our hardware system has some spatial restrictions that need to be adhered to. Ideally, the complete setup should not be larger than a regular dressing room. In addition, it must provide its own background and lighting.

A model of our setup is shown in Fig. 5.1. It consists of a 2×3 meters aluminum enclosure with cameras mounted on all sides and a TV monitor in the front. The enclosure is surrounded by green walls. Colored walls facilitate the segmentation of the user as long as the user is wearing garments with a color contrast to the background. As a display, we use a standard 47 inch Full-HD television set mounted in portrait mode. All devices are connected to a single computer.

We use ten Point Grey® Flea™2 cameras (see Fig. 5.2). Each camera is equipped with a 5 mm C-mount lens. The exact location of each camera has been determined experimentally. We have placed cameras in each top corner, in the middle of the enclosure



(a) Model of the recording setup with camera positions (without walls)



(b) Real-World view from the outside of the setup

Figure 5.1: Our hardware setup comprises an aluminum enclosure with green walls, a monitor and ten cameras.

and below the monitor in order to get a sufficient variety in viewpoints. Lower camera positions were not chosen such that users can not accidentally touch the cameras while moving. The viewing frustums of all cameras are focused on an area approximately 1.5 meters in front of the monitor. This means that the user is only allowed to move within a small area in order to stay in an optimal position. When the person is moving, some body parts will always be outside the view frustum in some cameras. These spatial restrictions and limited viewpoints make shape and pose estimation more challenging than when using larger recording setups such as [57, 135, 136].

Each camera records 640×480 pixel images at 15 frames per second. Color images are encoded in the `YUV422` format (2 byte/pixel). In order to transfer the data to a single computer in real-time, we connect them via three separate FireWire 800 buses. The FireWire buses not only transmit data but supply the cameras with power and enable synchronization. Therefore, only a single cable needs to be connected to each camera.

5.2.2 A Computing Platform for Real-Time Image Processing

Fast computing platforms allow an ever increasing number of computer vision tasks to be executed in real-time. Several years ago, there has been a global trend to build CPUs with increasing core frequency. However, this increase was slowed down by a natural barrier: faster CPUs became less efficient in terms of power consumption [146]. This stagnation led to the development of parallel and even massively parallel computing architectures that distributed the load from one processor to a multitude [37, 111].

Our system features a 3.4 GHz Intel® Core™i7 processor with 8 cores and a CUDA enabled Nvidia® GTX 480 GPU. This GPU consists of 15 SIMD processing units that can execute 32 threads in parallel, each. In order to achieve optimal performance, care has to be taken that algorithms are executed on the suitable processor type [94]. While the CPU can quickly execute single threaded algorithms such as graph processing, the GPU can be used for efficient parallel processing tasks. For example, pixel-wise image operations can be performed extremely fast on a GPU. However, memory transfers between CPU and GPU need to be minimized as they can easily become a performance bottleneck.

The cameras of our hardware system gather data at a rate of about 100 MB/s. We exploit the computational power of the graphics card to perform basic image manipulations such as undistortion and background segmentation. Furthermore, the visual hull can be efficiently computed on a GPU when each voxel of a discrete volumetric grid is processed independently. Our real-time capable system therefore performs image and volume processing on the GPU. This data is then compressed and transferred to CPU memory. The CPU further processes the data in order to estimate the pose and shape of a human body.

5.2.3 Calibration

In order to reconstruct an object with a multi-view setup, the cameras need to be fully calibrated. Camera calibration is a two step procedure. First, the intrinsic parameters

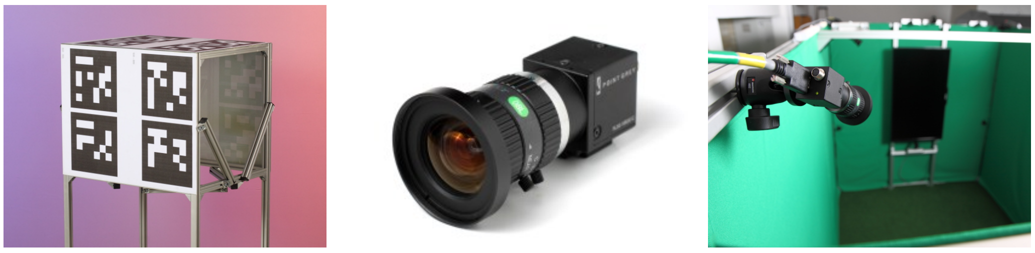

(a) Calibration Target (b) PtGrey camera with lens (c) Mounted camera

Figure 5.2: Calibration target and cameras used in our hardware setup.

of each camera are calibrated individually. Therefore, we use a modified version of the Bouguet calibration toolbox [29] which requires several images of a planar calibration target to be recorded from different viewpoints. From these images, we estimate the intrinsic camera matrix $\mathbf{K}_\ell$ and undistortion parameters for each camera. Note that we perform camera calibration at the maximum possible camera resolution of 1280×960 pixels.

Once every camera is calibrated, we mount the cameras on our aluminum frame. In order to use all cameras in a single coordinate framework, we need to determine their position relative to a defined origin. Our extrinsic camera calibration method utilizes a 3D target with four augmented reality (AR) Toolkit markers [160] attached on three sides (see Fig. 5.2(a)). The total number of markers is twelve. We place this calibration target in a central position inside our setup to perform extrinsic calibration.

For a single planar AR marker, it is possible to compute its exact position and rotation in space and thus define the relative camera orientation. Each marker on the target is accurately referenced with respect to a common coordinate system which we define as the location of the first marker. Even if a camera can see only one marker, the locations of all other markers can be computed. This is an important property as our cameras will see the calibration target from different sides. No single marker can be seen simultaneously from all cameras.

In Fig. 5.3, we show a typical situation where the calibration target is placed inside the enclosure. Each camera sees at least two markers. Thus, we can compute the position of the coordinate origin more accurately. In order to maximize calibration accuracy, it is important to place the calibration target in the position where the human person is supposed to stand during normal operations. Due to calibration from a single target position, our procedure determines the extrinsic calibration ($\mathbf{R}_\ell$, $\mathbf{t}_\ell$) of all ten cameras at a sufficient accuracy within seconds. However, this calibration method limits the choice of possible camera orientations in our setup (*e.g.* it is not possible to have one camera focus exclusively on the head of the person).

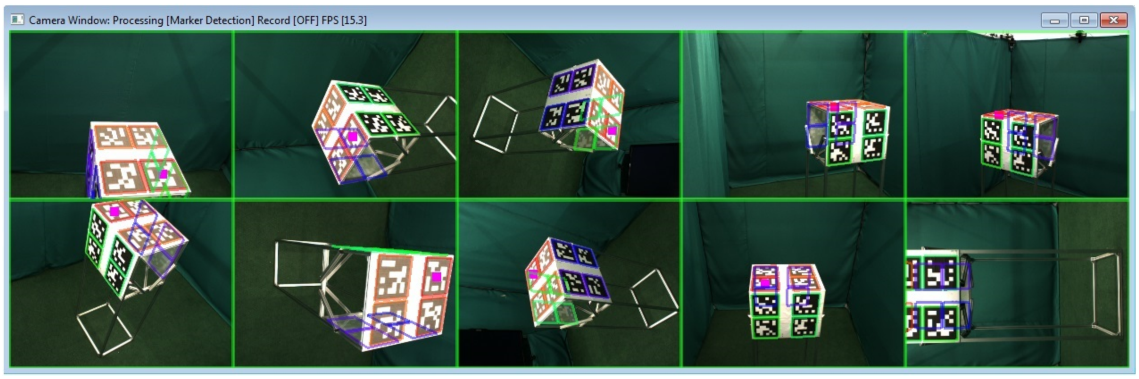

Camera Window: Processing [Marker Detection] Record [OFF] FPS [15.3]

Figure 5.3: Our calibration target and calibration GUI. The colored rectangles on the target are augmented onto the camera image to verify that the camera has been registered correctly. The purple dot on one marker signifies the coordinate origin.

5.2.4 Image Recording and Pre-Processing

In order to perform shape and pose estimation from silhouette images, we need a processing method that computes silhouettes of the human body from the camera images. Our method to compute silhouettes is simple, straight forward and is implemented on fast GPU hardware.

First, we record synchronized images from all cameras and transfer them to the computer. These raw images are copied to the GPU memory immediately. Then, the GPU performs simple image processing tasks such as color conversions (`YUV422` to `RGB`) and image undistortion. In order to compute silhouette images, we perform background subtraction. This method requires an image of the room to be recorded without a person, first. Once the person is inside the room, an illumination insensitive background subtraction algorithm can detect regions in the image where the person is present but ignore shadows caused by the person on the floors and walls. Our background subtraction method is based on normalized `RGB` colors. Finally, we perform simple filtering to remove small segmentation errors. The total pre-processing time for all camera images is 6.5 ms.

5.3 Real-Time Human Pose Estimation

In this section, we show results of the graph-based human pose estimation that we described in Chapter 3. We perform experiments on our own recordings, public datasets as well as on synthetic scenes with ground-truth data. The main focus of our evaluation is to show that we can provide long-term pose estimation with automatic recovery in case of an error. Long-term stability is essential for interactive systems where no user corrections of the pose are possible. In addition, we show that our algorithms are sufficiently fast to be used at camera frame rate.

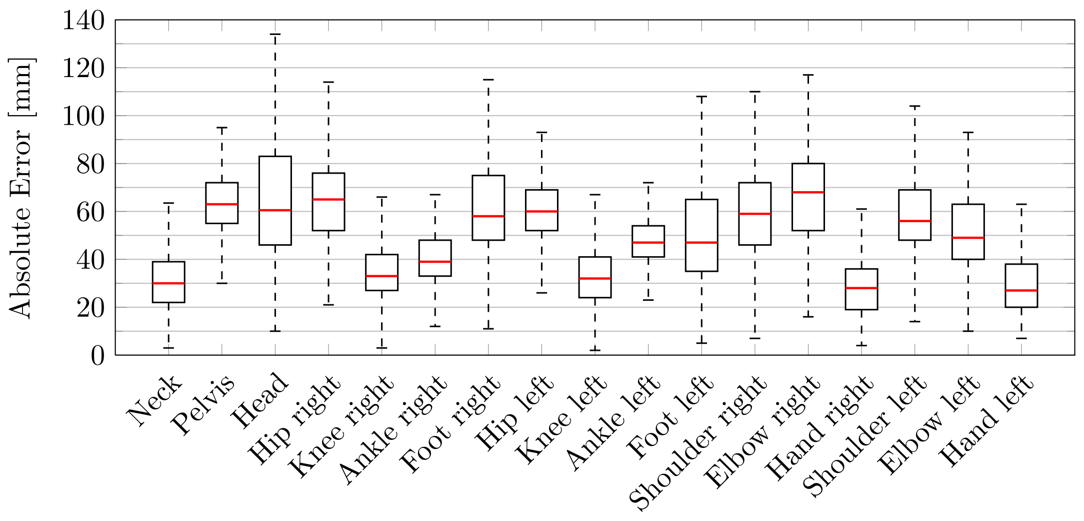


Figure 5.4: Estimation error for each joint. The error is the Euclidean distance between estimated joint positions and ground truth data.

5.3.1 Data Acquisition

Our method requires a volumetric scan of the human body in order to extract a skeletal graph. We perform GPU accelerated space carving from multi-view silhouette images to compute the visual hull in a dense voxel grid. Unless otherwise specified, we use a $96 \times 96 \times 128$ voxel grid with a resolution of about 15 mm per voxel for this evaluation. For qualitative evaluations, we use our own hardware setup consisting of 10 synchronized color cameras (see Section 5.2) and publicly available datasets [57]. To obtain ground truth joint position data for our quantitative evaluation, we use motion capture data from the CMU motion capture database [35] to animate a human polygon model and render it from multiple views. Such images are equivalent to the images recorded with our camera setup. In addition, the position of each skeleton joint is known from the motion capture data. This allows us to compare our estimations to groundtruth data.

5.3.2 Quantitative Results

We quantify the joint position estimation accuracy as well as the robustness of end-node labeling. To measure the estimation accuracy, we calculate the Euclidean distances of our joint pose estimates to the corresponding ground truth position given by the motion capture data. We perform this evaluation for every frame in several sequences of the CMU motion capture database [35]. In total, we have evaluated our algorithm on almost twenty thousand frames of sequences containing a variety of movements.

In Fig. 5.4, we show that the median of the distances stays below 80 mm for all joints while the distance for end-joints such as hands and feet is even smaller. Note that the 25th

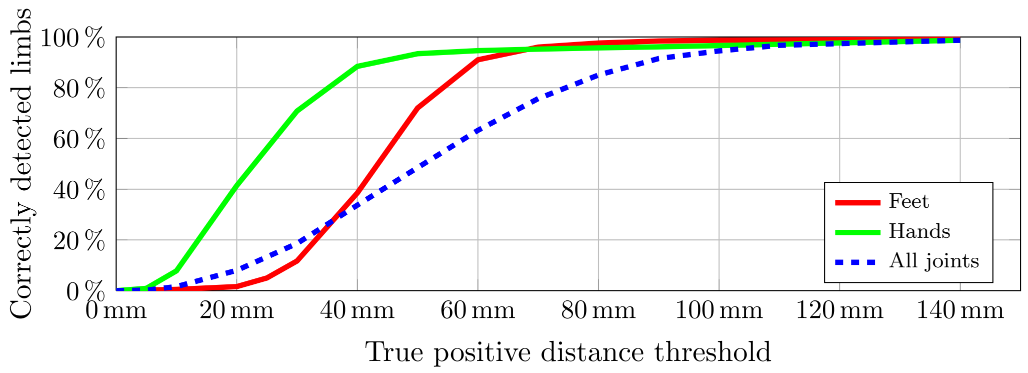


Figure 5.5: Percentage of correctly classified joints dependent on the confidence threshold.

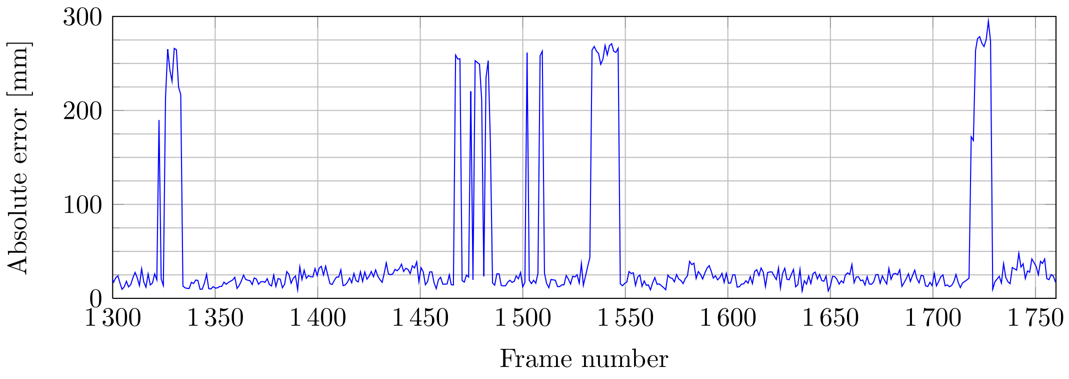


Figure 5.6: Distance between the left hand and its ground truth position in an interval taken from the sequence 13_18 of [35].

and 75th percentiles in the boxplot are less than twice the voxel size apart. This suggests that most errors are systematic due to structural differences between our template skeleton model and the skeleton used in the motion capture database.

In the next experiment, we determine how far joints deviate from their ground truth position on average. In Fig. 5.5, we gradually increase the classification radius for each joint. For example, hands are within a 25 mm radius from the ground truth position in more than 50 % of all frames. On average, all joints are within 50 mm of their ground truth position in more than half of all frames of our test sequences. When we increase this threshold to 100 mm, we estimate all joints correctly in more than 95 % of all frames. Again, the performance for hands and feet is superior to other joints as their position is determined directly by the graph matching step (see Section 3.3.2.2), while other joints depend on the position of hands and feet.

A major benefit of our algorithm is single-frame recovery. In Fig. 5.6, we show the estimation error for the left hand over the period of some frames of a motion capture sequence. We deliberately have chosen a time interval with many ambiguous poses that

Sequence	**# Frames**	**FootR**	**FootL**	**HandR**	**HandL**
01_01	2 750	0 (0.0 %)	0 (0.0 %)	51 (1.9 %)	36 (1.3 %)
02_01	342	0 (0.0 %)	0 (0.0 %)	10 (2.9 %)	47 (13.7 %)
02_05	1 854	1 (0.1 %)	0 (0.0 %)	5 (0.3 %)	85 (4.6 %)
03_01	431	0 (0.0 %)	0 (0.0 %)	11 (2.6 %)	5 (1.2 %)
05_02	1 122	8 (0.7 %)	5 (0.4 %)	5 (0.4 %)	7 (0.6 %)
06_04	395	1 (0.3 %)	0 (0.0 %)	6 (1.5 %)	10 (2.5 %)
13_17	4 839	29 (0.6 %)	28 (0.6 %)	119 (2.5 %)	48 (1.0 %)
13_18	2 999	37 (1.2 %)	41 (1.4 %)	162 (5.4 %)	129 (4.3 %)
13_29	4 591	75 (1.6 %)	96 (2.1 %)	172 (3.7 %)	90 (2.0 %)
16_11	533	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)	0 (0.0 %)
Total	**19 856**	**151 (0.8 %)**	**170 (0.9 %)**	**541 (2.7 %)**	**457 (2.3 %)**

Table 5.1: Evaluation of the foot/hand classification errors on some sequences of the CMU motion capture database [35]. Each limb that is not within a 100 mm radius of the ground truth joint position is counted as an error.

cause a jump of the hand position in the resulting skeleton. Even though the position cannot be determined accurately in some frames, our single-frame recovery prohibits that the hand gets stuck in the erroneous position for more than a few frames. These findings are also supported by Table 5.1, which gives a detailed count of how often each limb is away more than 100 mm from its ground truth position in several motion capture sequences. Even on long sequences, the error rate stays below 3 % on average.

5.3.3 Qualitative Results

In Fig. 5.7, we show qualitative results of our skeleton fitting algorithm. The input images are taken from our own multi-camera hardware setup and publicly available datasets [57]. Colored squares mark the detected joint positions in the graph matching step that are used for initialization of the skeleton model. Note that even long spurious branches do not affect our end-node classification in most cases.

There are cases where pose estimation fails (such as in Fig. 5.7(e)). Such incorrect poses are a result when end-nodes are classified incorrectly. This can happen when a spurious node has a descriptor similar to an actual limb and thus gets a good matching score. Usually, such spurious nodes disappear when the person slightly rotates the body or keeps moving the arms. In Fig. 5.8, we show examples where a person is interacting with objects such as a chair and a table. In (b) and (c), the person is touching these objects, which causes the skeletal graph to grow into these objects. When the graph no longer has the structure of a human body, end-node labeling fails (*e.g.* the leg of the table gets classified as a human foot) and the skeleton pose becomes invalid. Our algorithm is not able to cope with such occlusions. A possible way to detect wrong poses is to check if the estimated pose is physically plausible.

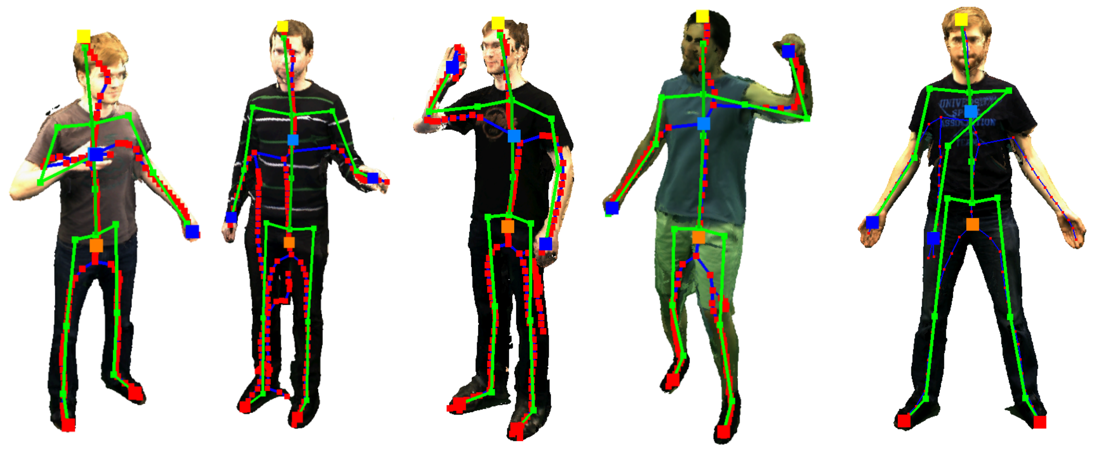


Figure 5.7: Skeleton model automatically fitted to the human body. We used [73] for rendering the 3D model. Besides the skeleton (green), we show the skeletal graph (red/blue) as well as labeled nodes. (a) to (d) show successful estimations while (e) contains a wrongly labeled hand node.

5.3.4 Runtime Evaluation

We performed our experiments on a state of the art PC system equipped with a Nvidia® GTX 480 graphics card and an Intel® Core™ i7 processor. It is possible to estimate a human skeleton model in real-time at 15 frames per second using our algorithm, limited by the frame rate of our camera setup. Given silhouette images of the body from multiple views, we were able to generate a voxel model in 5 ms on the GPU and then use a single CPU thread to extract a skeletal graph within 15 ms. Labeling end-nodes and fitting a skeleton to the graph takes less than 1 ms. This allows for skeleton estimation at more than 30 frames per second.

The main reason for this efficiency is the reduction of data early on: we compress the voxel model by expressing its structure with a graph consisting of at most a few hundred nodes. Compared to other center-line estimation algorithms such as Wang et al. [161] who are able to process only around 6 000 voxels per second, our implementation of the algorithm by Rodriguez et al. [119] is able to generate a skeletal graph by processing up to 3 million voxels per second on a single CPU.

5.3.5 Comparison to Related Approaches

Top-performers in the field of human pose estimation from multi-view images achieve average joint position errors of around 50–100 mm [56, 79] at the cost of a processing time of more than one second per frame. Even then, such methods rely on tracking information

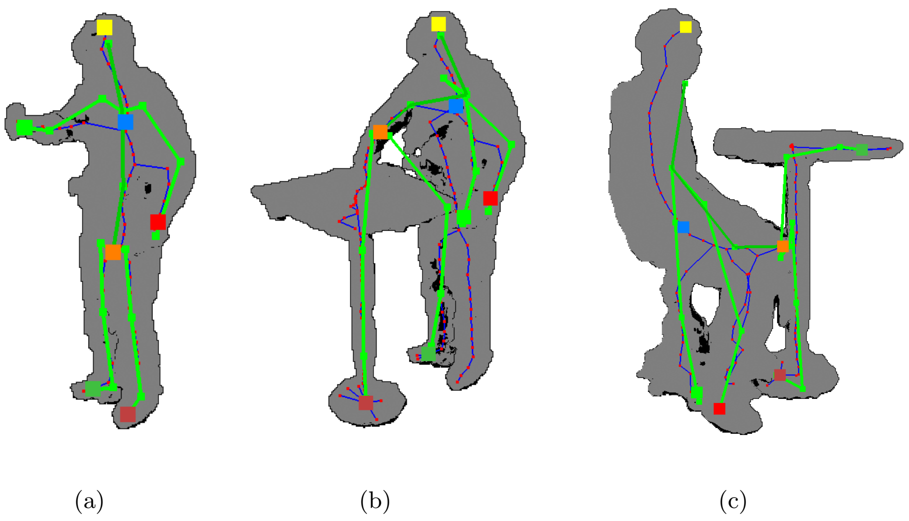


Figure 5.8: Graph-based pose estimation has intrinsic problems with occluding objects. In (a), the pose is estimated correctly, while the same person is touching a table in (b), which causes pose estimation to fail. (c) A person sitting on a chair infront of a table leads to a totally wrong pose.

and can get stuck in local minima if tracking information is wrong. Our system does not necessarily depend on temporal information and is capable of providing the same error rates at up to 30 frames per second.

Other methods such as [58, 127] rely less on tracking data. They provide real-time performance at similar error rates but incur a substantial training effort for part detectors or require a database of exemplar images [156]. We do not require any training data but require only a skeleton model with known dimensions. Previous skeletal-graph-based methods that work on 2D images [42, 149] or 3D data [104, 145, 158] can either not operate at interactive frame rates or work only if users directly face the camera. Similar to our method, the method by Bakken et al. [15] is evaluated on synthetic ground truth data generated using the CMU dataset [35]. They achieve similar pose estimation accuracy around 60 mm and limb misclassification rates of three percent but at higher computational costs.

We present a comparison of existing approaches in Table 5.2. In particular, we compare features relevant for interactive applications such as real-time capabilities and required user interaction or initialization. When available in the corresponding paper, we provide mean joint position accuracies and error rates. However, it is not possible to compare these numbers directly as there is no common evaluation benchmark.

		Method	Requires temporal tracking	Requires initial pose	Requires training data	Real-time capable (faster than 5 frames/second)	User interaction required	Average joint position error	Average joint misclassification rate
Single	Correa et al. [42]	2D medial axis	O	✗	✗	✓	✗	-	8 %
	Eichner et al. [45]	per-pixel label	✗	✗	✓	✗	✗	-	20 %
	Thome et al. [149]	skeleton graph	✓	✗	✗	-	✗	-	-
Depth	Ganapathi et al. [58]	part detectors	✓	✗	✓	✓	✗	20 cm	-
	Shotton et al. [127]	per-pixel label	✗	✗	✓	✓	✗	10 cm	21 %
	Sun et al. [144]	cond. per-pixel label	✗	✗	✓	-	✗	-	17 %
Multi-View	Bakken et al. [15]	skeleton graph	✗	✗	✗	✓	✗	6 cm	3 %
	Menier et al. [104]	3D medial axis	✓	-	✗	✗	✗	-	2 %
	Liu et al. [99]	local/global PF	✓	-	✗	✓	✗	2.8 cm	-
	Vlasic et al. [158]	visual hull	✓	✗	✗	✗	✓	-	-
	Stoll et al. [139]	color blobs	✓	✓	✗	✓	✗	4.5 cm	-
	This work	skeleton graph	O	✗	✗	✓	✗	10 cm	3 %

Table 5.2: Comparison of human pose estimation methods. Legend: (✓) yes, (✗) no, (O) optional, (-) unknown.

5.4 Human Shape Estimation

In this section, we evaluate skeleton-based mesh adaptation and the multigrid solver that we described in Chapter 4. We evaluate our approach on public datasets, artificially generated data as well as on recordings made with our own hardware setup. If not provided by a dataset, we use a mesh from the SCAPE database [10] as the template mesh. The main focus of the evaluation lies on the quality of the obtained results as well as the runtime. A fast execution time of the described algorithms is essential for real-time performance which is required for interactive use of the adapted mesh.

We compare skeleton-based mesh adaptation to existing methods that do not contain a skeleton constraint. In this evaluation, we provide our algorithm with challenging scenes that have only a limited number of views of a person. A limited view is a common problem when cameras are too close to the person to capture the full body in every camera image.

In order to reduce the runtime required for optimizing the adapted shape, we have presented a cascading multigrid solver in Section 4.6. We show the quality of the generated low resolution meshes and give insights into how the initial mesh converges to the final solution. In this thesis, we claim that iterative solvers can improve runtime performance significantly when they are initialized with a mesh close to the optimal solution. Therefore, we compare direct and indirect solvers with respect to their runtime and their output quality.

Another set of experiments evaluates the robustness of our shape adaptation approach. As our approach requires a good initial pose estimate, we simulate a noisy pose estimation result with a number of different noise levels. Such noisy poses produce an inaccurate initial mesh which will degrade the performance of shape adaptation.

Finally, we show qualitative examples of full body modeling and performance capture using our approach. We discuss the benefits and limitations and compare our approach to related work.

5.4.1 Skeleton Supported Human Shape Adaptation

We provide a qualitative and quantitative evaluation of our approach for adapting a 3D human body model to multiple synchronized silhouette images using a skeleton constraint. Similar to [4, 57, 158], we require a template mesh of the person in roughly the same pose. Typically, such a mesh is obtained by a laser scanner or via image-based methods. A rough pose can be obtained through pose estimation directly from silhouette images using the method described in Chapter 3. By using linear blend skinning, we transform the template mesh such that it has the same pose as the person in the images. This initial mesh is then deformed such that its reprojection onto the input images has a maximum overlap with the silhouettes of the human body. Any deviation from an optimal overlap is measured using the *silhouette overlap error* [4, 17, 34]. Therefore, we count the number of pixels that are different in the reprojection of the deformed mesh and the input segmentations.

5.4.1.1 Influence of the Skeleton Term

We evaluate our pose and mesh adaptation on a public dataset which contains high quality silhouettes of multiple actors recorded by eight one-megapixel cameras [57]. In every frame, we initialize the actor specific template model using the 3D skeleton pose information provided in this dataset. In the first set of evaluations, we make use of our linear skeleton binding energy for shape adaptation but do not use bone length preserving constraints nor a multigrid solver. The configuration used in this section is similar to the method of Gall et al. [57].

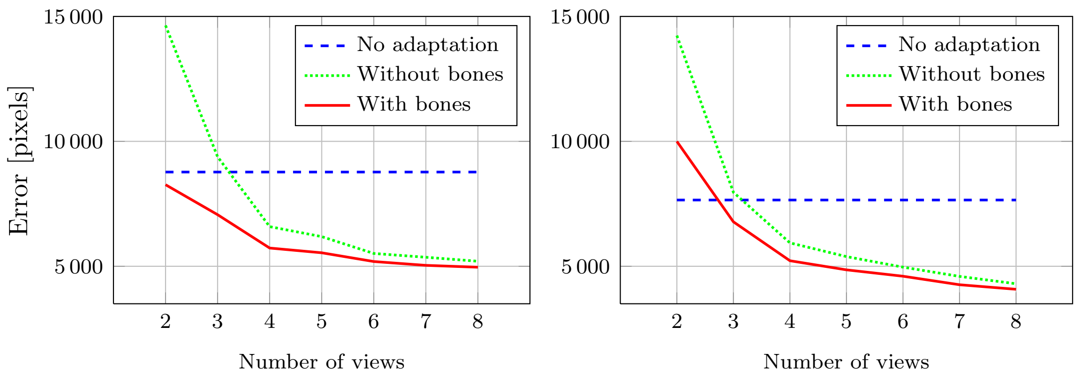
In Table 5.3, we evaluate the influence of our skeleton term and covariance-based correspondence weighting. Almost all scenes benefit from an additional skeleton term, which decreases the silhouette overlap error by 200 pixels on average compared to mesh adaptation without skeleton and covariance weighting. The average number of pixels the human actor occupies per camera image is between 50 000 and 80 000.

There are larger errors in configurations where a bone term is used without covariance weighting for correspondences ($\varepsilon_{\text{cov}} = 1$). The surface covariance weighting scheme is described in Section 4.4.2. The reason for larger errors without covariance weighting is as follows: the bone energy competes against the deformation energy to maintain a natural

Sequence	**# frames**	**not adapted**	$\varepsilon_{\text{cov}} = 1$		$\varepsilon_{\text{cov}} = 0.01$	
			$\lambda_{\text{bone}} = 0$	$\lambda_{\text{bone}} = 0.1$	$\lambda_{\text{bone}} = 0$	$\lambda_{\text{bone}} = 0.1$
Dance	574	7 600	4 400	4 500	4 300	4 100
Skirt	721	6 900	4 100	4 300	4 300	4 100
Handstand	401	8 800	5 100	5 200	5 200	4 900
Wheel	281	7 200	4 400	4 600	4 300	4 300
Dog	60	4 700	3 300	3 100	3 100	3 100

Table 5.3: Effect of covariance weighting ε_{cov} and bone energy when adapting a mesh to multi-view silhouette images. Reported values are the mean silhouette overlap error over all cameras for the given sequence from [57].

distribution of vertices along the mesh surface while the deformation pulls vertices to their closest silhouette contour. Covariance-based weighting enables both energies to be optimized with minimal interference.

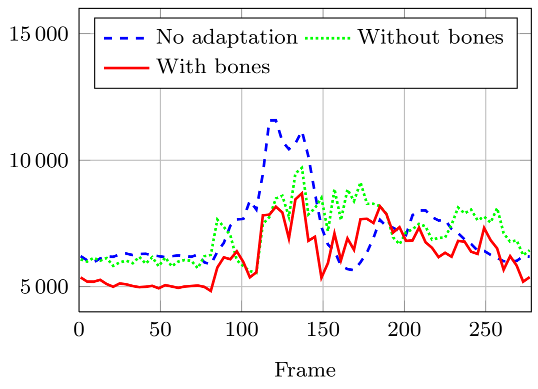


(a) Handstand (401 frames) (b) Dance (574 frames)

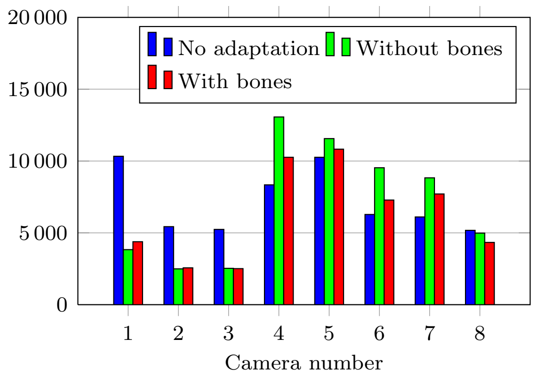
Figure 5.9: Mean silhouette overlap error in pixels (evaluated on all views) when the mesh is adapted only to the first n views of the given sequence.

In the original experiment configuration, the improvement of our method is rather small compared to [57] because the number of camera views (eight) is sufficient for a good adaptation with surface-only regularization. The real benefit of the skeleton constraint becomes apparent when fewer input silhouette images are available. In Fig. 5.9, we analyze the silhouette overlap error depending on the number of input views and compare mesh adaptation with and without a skeleton term. For reference, we plot the initial error of the not yet adapted mesh, which is independent on the number of views. This mesh is obtained through linear blend skinning of the template mesh using the skeleton pose estimate provided in the database.

When all eight cameras are used, our skeleton term does not significantly increase the



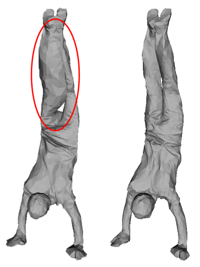
(a) Mean error for each frame (in pixels)



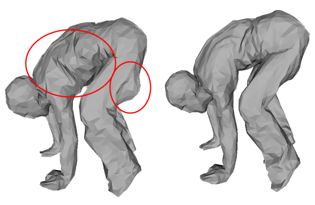
(b) Mean error for each camera (in pixels)

Figure 5.10: Evaluation of the Wheel sequence [57] when only cameras with number 1–3 are used for mesh adaptation. The silhouette overlap error (y-axis) is computed from all views (number of erroneous pixels).

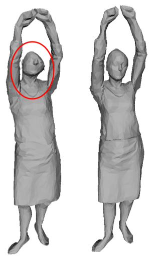
performance. However, our bone energy term yields a significantly lower error when only a few views are available. In Fig. 5.10, we take a closer look at the reason for these results. By means of the *Wheel* sequence adapted to the first three camera views, we analyze the mean silhouette overlap at each frame and camera individually. It can be seen that our bone energy consequently yields a lower error in all frames (Fig. 5.10(a)). While the skeleton term effectively minimizes the errors in views used for adaptation, its preference for plausible deformations is honored by a lower error in the remaining views. Adaptation without an underlying skeleton simply overfits to the given views and causes unnatural effects visible in remaining views. Such effects can be seen in a qualitative analysis in Fig. 5.11.



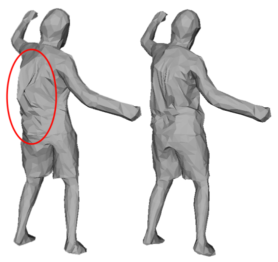
(a) Handstand



(b) Handstand



(c) Skirt



(d) Dance

Figure 5.11: Our bone energy term reduces unnatural deformations even when only a few camera views are available (here: 4 views). Deformation without (left) and with skeleton (right).

5.4.1.2 Runtime Evaluation

We analyze the runtime of our approach when adapting a mesh with 2 500 vertices and 18 skeleton joints. The overall adaptation of the mesh to a single frame in an eight camera setup takes 4 seconds in an unoptimized MATLAB implementation on an Intel® Core™i7 CPU. This measurement includes the time for eight iterations of matching rim vertices and silhouette contours, computing vertex covariances and solving the equation system using a sparse Cholesky decomposition. A single minimization of the objective function accounts for about 60 ms when skeleton information is not used. By jointly optimizing for joint positions, this time increases to 75 ms. This increase is negligible since shape optimization requires only a fraction of the overall runtime. When solving the quadratically constrained version of Equation (4.27) with 6 symmetry constraints, the time for a single optimization increases to 500 ms.

5.4.2 Multigrid Shape Adaptation

The experiments in the previous section have been performed using a direct linear least squares solver for computing a deformed mesh. In this thesis, we have proposed a multigrid approach to perform mesh adaptation on multiple levels of resolution. We argue that the overall runtime can be reduced by using iterative solvers at each level of resolution instead of using a direct sparse Cholesky decomposition. For example, the conjugate gradient (CG) algorithm [19] can be used to minimize a linear least squares system in a computationally efficient manner when the initial solution is already close to the optimum. At each level of resolution, a few iterations of optimization are typically sufficient since the overall mesh is further optimized at a subsequent finer resolution. Similar to Cholesky decomposition, the CG method requires the symmetric positive definite (SPD) squared system matrix to be computed, which can take a relatively long time for larger systems of linear equations. In Section 4.7, we have presented an iterative solver that directly operates on individual constraints and does not require the computation of the SPD system matrix.

In this section, we compare the results of optimizing mesh deformation constraints in a cascading multigrid using the direct Cholesky solver, the iterative conjugate gradient method and our decoupled constraints solver. We evaluate these methods using a mesh consisting of 2 502 vertices and 18 skeleton joints. The evaluation is performed on several multi-view sequences of moving persons recorded with our own multi-camera setup (see Section 5.2).

5.4.2.1 Generating the Multigrid

We analyze the result of the restriction operator on the input mesh by showing the vertices and edges at different resolutions. In Fig. 5.12, we show the four levels of resolution we generate for this mesh using the restriction operator defined in Section 4.6.2. From left to right, the mesh consists of 171, 487, 1 233 and 2 502 vertices. These meshes have color coded vertices that display the maximum level of each vertex. For example, red vertices are only present at level 0 while purple vertices exist at all levels throughout level 3.

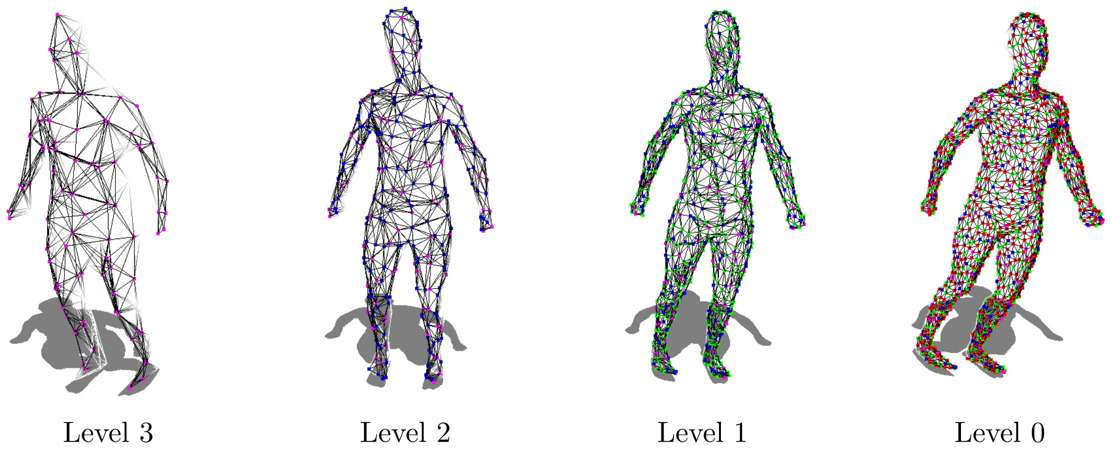
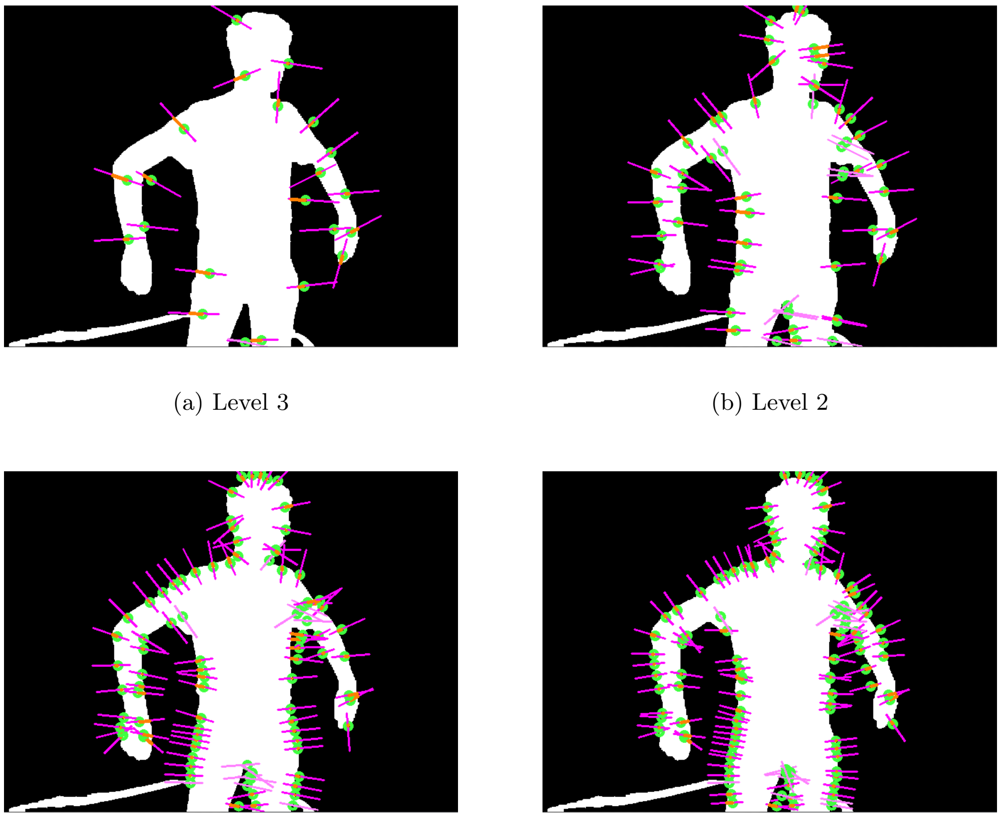


Figure 5.12: Several resolution levels of a multigrid hierarchy starting with the coarsest mesh $\mathscr{M}^3$ on the left side. The mesh at its full resolution $\mathscr{M}^0$ is shown on the right side. The vertex color signifies the maximum level of each vertex.

5.4.2.2 Image Correspondences in a Multigrid Framework

In Fig. 5.13, we demonstrate the advantage of the multigrid approach in terms of matching vertices with silhouette contours. We initialize our mesh at level 3 with the pose obtained from a pose estimation algorithm. Initially, the skeleton pose is not sufficiently accurate. This inaccuracy is clearly visible in Fig. 5.13(a), where rim vertices of the arms are far away from the actual silhouette contour. In addition, the shape of the model can significantly differ from the shape of the actual human body in the images. During initial iterations of shape adaptation, the main goal is to correct the pose and perform a rough shape alignment. Obviously, a high resolution mesh has no advantage over a coarse version in these early iterations. A low resolution mesh with only a few hundred vertices makes matching vertices with silhouette contours computationally efficient. This means that more time can be spent for processing each vertex (*e.g.* searching for correspondences in a wider range). In subsequent iterations of mesh adaptation, we increase the resolution of the mesh such that finer details can be adapted to the input images. When analyzing the progression of the optimization in Fig. 5.13 from (a) to (d), we can see that rim vertices become better aligned with the silhouette contour at finer levels. Thus, we can reduce the range where we search for correspondences. A narrower search range minimizes the risk of wrong correspondences and allows faster matching.

In Fig. 5.14(a), we analyze how many outer iterations are needed until the contour correspondences stabilize. At 100 %, all vertices have converged to a stable position and do not move after another iteration of finding correspondences and optimization. Depending



(a) Level 3

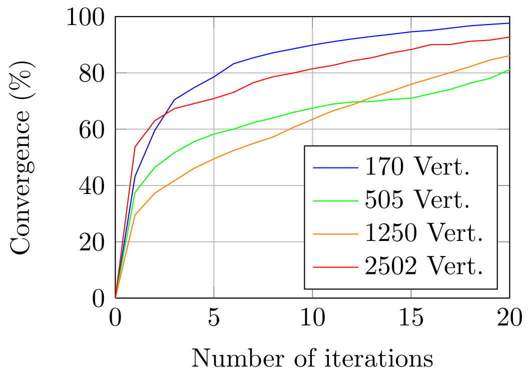
(b) Level 2

(c) Level 1

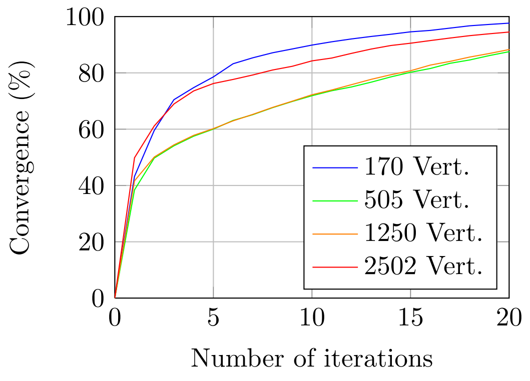
(d) Level 0

Figure 5.13: Silhouette correspondences of rim vertices at different levels of resolution. Optimization starts at a low resolution (a) and the resolution is gradually increased throughout images (b) to (d). Green dots represent the rim vertices, purple lines mark the search range and orange lines denote a found correspondence.

on the resolution of the mesh, the convergence rate is different. In Fig. 5.14(a), we use the same initialization for the mesh at each level. This is in contrast to the multigrid cascade, where the results of a lower resolution can be used to initialize a higher resolution level. Therefore, this figure demonstrates the convergence our solver would yield without a multigrid approach. In Fig. 5.14(b), we show the benefits of propagating the optimized result from one level to the next level. An intermediate optimization provides a good initial mesh for a finer level. This helps to find better silhouette correspondences and leads to a faster convergence rate, especially at early iterations. In both experiments, the highest resolution (2 502 vertices) shows the best convergence rate. The reason for this is that we can use the co-tangent weighting scheme for skin constraints, which requires a triangular face structure. The co-tangent weighting scheme on triangular faces has superior performance compared to the uniform weighting scheme on a non-triangular grid



(a) Independent convergence



(b) Multigrid convergence

Figure 5.14: Convergence rates of contour matching and shape optimization for several levels of optimization. We have used a Cholesky solver to optimize the shape.

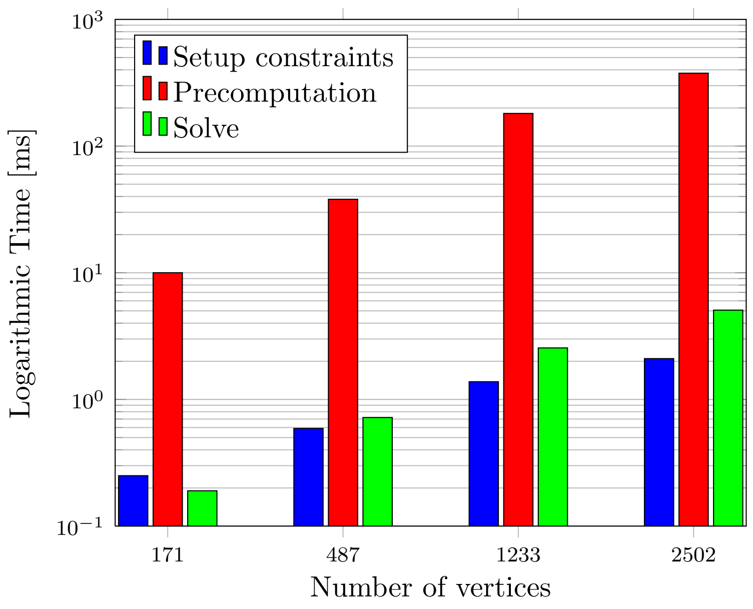
structure of sub-sampled meshes.

The optimization of a coarse resolution is significantly faster than the time for optimizing a full resolution mesh. Therefore, we propose to perform multiple outer iterations of finding correspondences and shape optimization at coarse resolutions. In the following experiments, we use two outer iterations each for the coarser levels 3 and 2. For the finer levels 1 and 0, we only use one outer iteration at each level to limit the runtime requirements.

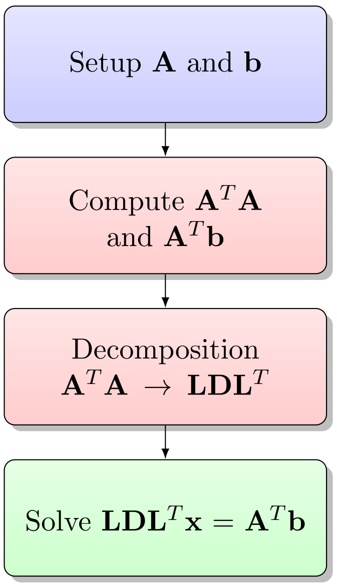
5.4.2.3 Runtime Evaluation

In the following experiments, we compare the runtime of different solvers for mesh optimization using the multigrid cascade. We measure the runtime of each phase of the algorithm separately in order to better show the differences between the solvers. We do not include the time for upsampling the mesh from a coarser level to a finer level as the time is generally less than 1 ms. For each level, we measure the time only for one outer iteration of finding correspondences and optimization. The full resolution mesh has a number of 2 502 vertices. We use the symbols $\mathbf{A}$ and $\mathbf{b}$ to denote the system of linear equations that are solved in least squares sense ($\min \|\mathbf{Ax} - \mathbf{b}\|^2$).

In Fig. 5.15, we show the runtime for the optimization phases at all four levels when using the sparse Cholesky solver provided by the Eigen matrix library [46]. The total runtime for a complete cascading multigrid optimization with four levels is about 700 ms. Note that the most time-consuming phase of the optimization is the preparation of the SPD system matrices. It consists of squaring the system of linear constraints $\mathbf{A} \rightarrow \mathbf{A}^T\mathbf{A}$ and Cholesky decomposition itself. As the number of vertices increases, this preparation phase prohibits any fast optimization. This problem is addressed in the mesh editing



(a) Runtime evaluation



Setup $\mathbf{A}$ and $\mathbf{b}$
Compute $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$
Decomposition $\mathbf{A}^T\mathbf{A} \rightarrow \mathbf{LDL}^T$
Solve $\mathbf{LDL}^T\mathbf{x} = \mathbf{A}^T\mathbf{b}$

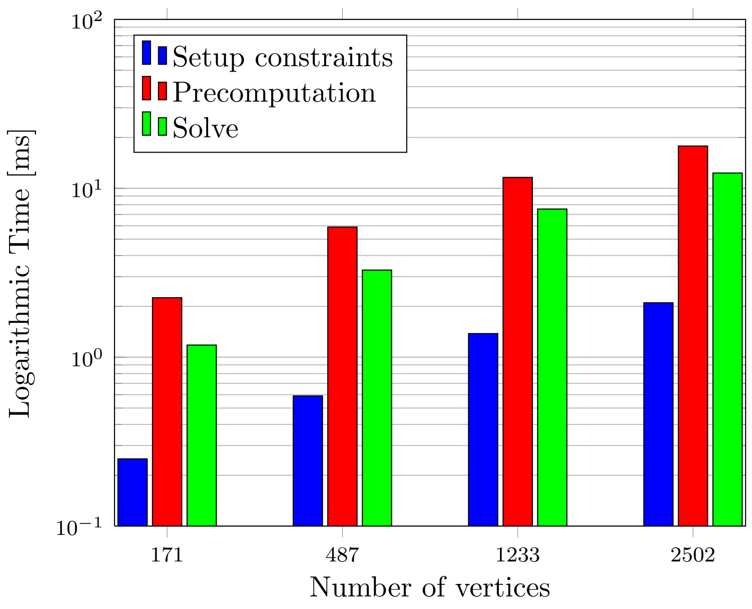
(b) Algorithmic blocks

Figure 5.15: Runtime for different phases and levels of the multigrid approach using a direct Cholesky solver with symmetric permutation.

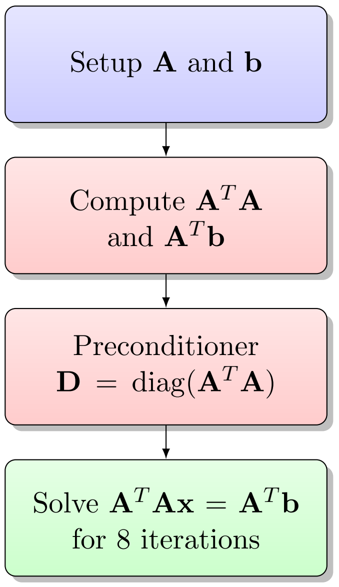
literature [27, 28] where a user manually modifies the positions of a few selected vertices. The solution to reduce long precomputation times is to compute the Cholesky factorization once and reuse it when the user applies a change to these vertices. In automatic mesh adaptation, the correspondences between vertices and silhouette contours change at every iteration. The reason for changing correspondences is that a vertex can gain or loose its rim status and not all rim vertices have a matching silhouette contour. Thus, the number of constraints constantly changes and we cannot reuse the symbolic results of a previous Cholesky decomposition.

One advantage of multigrid methods is that the upsampled result of a coarser level provides an excellent initialization for optimization at a finer level. In Fig. 5.16, we show an optimization scheme where we use an iterative conjugate gradient algorithm with a diagonal preconditioner. This reduces the total runtime for shape optimization to about 90 ms. Note that the time required for finding image space correspondences and setting up the linear constraints is independent of the solver in use. Nevertheless, a CG solver still requires the computation of a SPD system matrix which is responsible for the high precomputation times.

We can avoid the computation of a squared matrix $\mathbf{A}^T\mathbf{A}$ using our decoupled constraint solver, which optimizes each constraint independently. In Fig. 5.17, we show the runtime required for optimizing the complete multigrid cascade with the decoupled constraint



(a) Runtime evaluation



Setup $\mathbf{A}$ and $\mathbf{b}$
Compute $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$
Preconditioner $\mathbf{D} = \text{diag}(\mathbf{A}^T\mathbf{A})$
Solve $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$ for 8 iterations

(b) Algorithmic blocks

Figure 5.16: Runtime for different phases and levels of the multigrid approach using an iterative CG solver with 8 iterations.

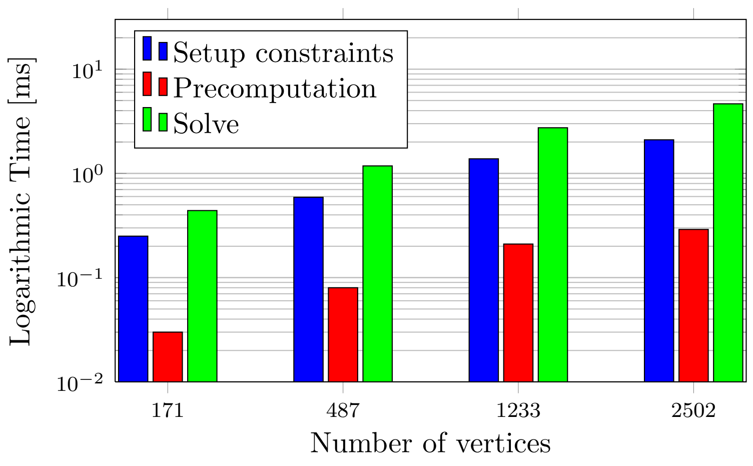
solver. It uses simpler precomputations (*e.g.* precomputation of gradient magnitudes) and is able to optimize all constraints directly. This further reduces the total runtime to less than 30 ms, which makes real-time operation feasible even without parallel hardware such as a GPU.

A summary and comparison for all three solver types is shown in Fig. 5.18. This figure clearly shows that the decoupled constraint solver outperforms both the direct Cholesky solver and the iterative conjugate gradient solver.

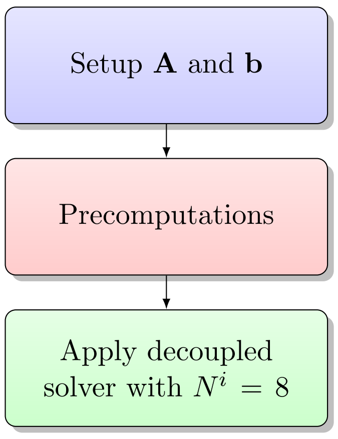
5.4.3 Robustness Towards Initialization Errors

Our joint shape and pose estimation approach requires an initial pose estimate to initialize the template model. Usually, such a pose can be estimated automatically using a human pose estimation algorithm. However, pose estimation is prone to errors and thus can influence shape estimation. For example, one consequence of an inaccurate initial pose is that closest-point silhouette contour correspondences will be wrong. Small errors of the initial pose have only a minor effect on the performance of our algorithm. The question is, how much deviation from the correct pose can be tolerated?

We quantitatively evaluate the robustness of simultaneous shape and pose estimation by performing an experiment on a dataset of artificial images. We use a skinned mesh and



(a) Runtime evaluation



(b) Algorithmic blocks

Figure 5.17: Runtime for different phases and levels of the multigrid approach using our iterative solver with 8 iterations.

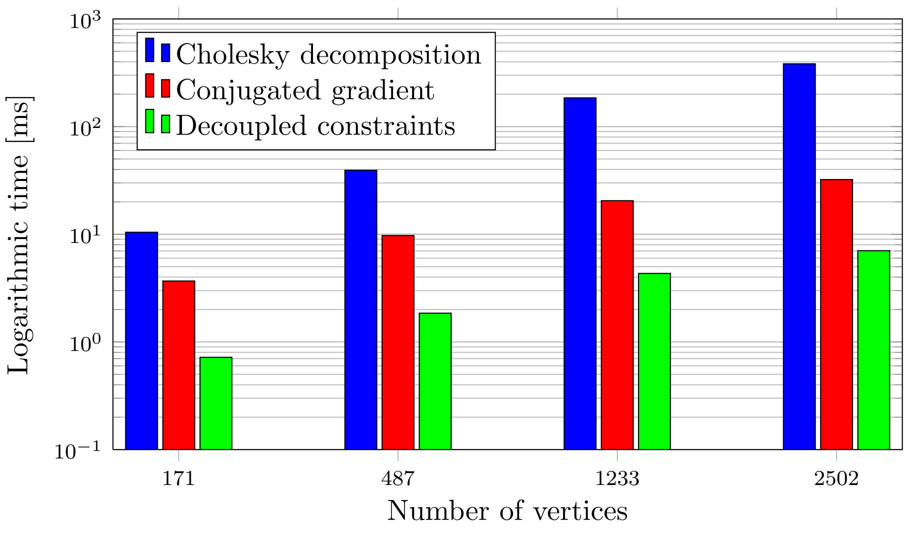


Figure 5.18: Total runtime for Cholesky decomposition, iterative CG and our decoupled constraint solver.

animate it through motion capture data from the CMU motion capture database [35]. We simulate the multi-camera setup from Section 5.2 by rendering the animated mesh from ten viewpoints to generate silhouette images. These artificial images are then used as input to our pose and shape estimation algorithm. The initial pose is directly computed from motion capture data. This dataset of artificial images allows us to obtain ground truth data for both the pose and the shape of the human body. The mesh that we use for

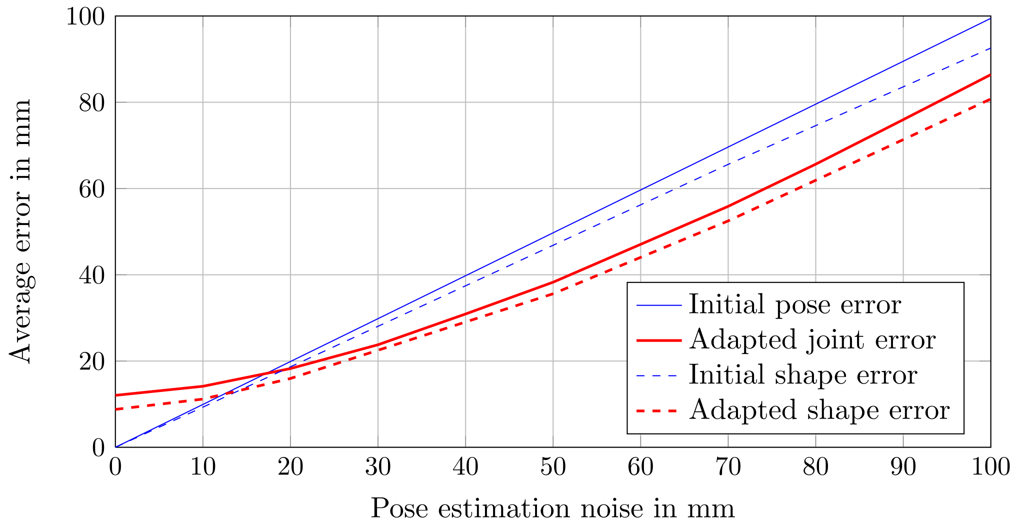


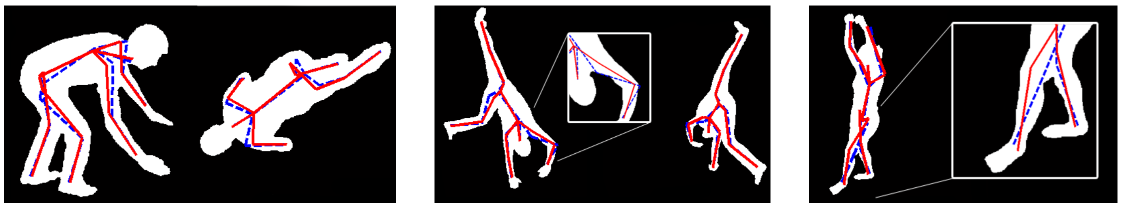
Figure 5.19: Influence of random noise on estimated joint positions on the resulting estimated shape and pose.

rendering the input images is used as the template mesh for shape estimation. In theory, this should allow perfect alignment between the silhouettes and the template mesh.

In Fig. 5.19, we evaluate the reconstruction error when pose estimates are noisy. Therefore, we add a random offset vector to motion captured joint positions. This simulates inaccuracies of the pose estimation algorithm. We increase the length of this vector in order to simulate an increasing level of noise. We plot both the errors of the unadapted mesh as well as the adapted version using our multigrid approach. The unadapted mesh is generated directly by applying the noisy pose to the template mesh. The pose error is computed as the average Euclidean distance between reported joint positions and motion captured joint positions. The shape error is the average Euclidean distance between vertices of the model and the ground truth mesh that was used to generate the input images.

Without noise on the initial joint positions, our adaptation algorithm produces an average error of approximately 10 mm. This error is mainly caused by the limited number of views and small errors in rim vertex matching. Up until a random noise of about 40 mm, the figure shows sub-linear increase of the error of both the adapted shape and pose. This means that our adaptation algorithm can compensate for most inaccurate joint estimates. For higher noise levels, our adaptation algorithm is able to reduce the initial error by about 15 mm on average. Note that the random noise in this experiment is added equally to every joint of the skeleton. Thus, the experimental setup is more challenging than the expected error distribution reported in Section 5.3. The error of pose and shape is strongly correlated because the skeleton term links the shape to the pose.

We have performed additional experiments concerning automatic corrections of an



(a) Handstand #38 (b) Wheel #120 (c) Skirt #297

Figure 5.20: Our simultaneous shape and pose adaptation corrects an inaccurate initial pose estimate (dashed). Solid lines represent our optimized skeleton.

inaccurate initial pose on a public dataset [57]. This dataset is provided with an initial pose estimate for each frame of an eight view setup. We use this pose estimate and the provided template mesh and refine shape and pose using our algorithm. In Fig. 5.20, we show qualitative results on some frames with visible benefits. In these images, the given pose (dashed) is clearly inaccurate and our approach is able to improve the locations of skeleton joints (solid) to reasonable positions.

5.4.4 Qualitative Results

Our decoupled constraint solver and the conjugate gradient method require multiple iterations until a satisfying mesh deformation is obtained. In Fig. 5.21, we compare the quality of the resulting mesh (2 500 vertices) after two and eight inner solver iterations N^i while keeping the rim-vertex/contour matches constant. The decoupled constraint approach produces smooth results after two iterations already whereas the conjugate gradient solver yields a noisy mesh after the same number of iterations. After eight iterations both approaches yield similar results, which are comparable to the mesh obtained by directly solving the constraint system via Cholesky decomposition in Fig. 5.21(c). The reason for smooth results at early iterations is the fact that silhouette constraints are evaluated before smoothing constraints. Thus, even a low number of iterations produces pleasing results. The CG solver does not distinguish between data and smoothing constraints and aims for a high reduction of the error, which can be achieved by pulling rim vertices to their corresponding contour.

In Fig. 5.22, we show some results achieved with our approach. The first two columns show one out of ten input images that were used during mesh adaptation (the image is mirrored). The three-dimensional model shown in the right most column is the adapted template mesh colored with simple projective texture mapping. The examples in the first three rows show a good match between the model and the image across the whole body. In row four, however, we can see that the hand is modeled incorrectly because it is touching the body. This is a common problem with all silhouette-based approaches because the silhouette outline of the hand is indistinguishable from the silhouette of the body.

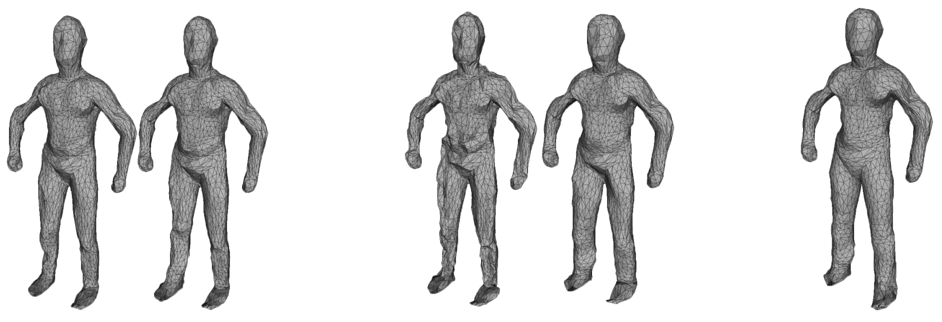


Figure 5.21: Quality of the deformed mesh after two and eight inner iterations of the decoupled constraints solver (a) and a conjugate gradient solver (b). The mesh obtained by solving the linear constraints using a Cholesky decomposition is shown in (c).

Related methods often present the deformation of a subject specific laser scan, which includes details such as the face and wrinkles of garment [4, 57, 165]. In contrast, we show that it is possible to deform one template mesh to multi-view silhouette images of a variety of people (we use the SCAPE mesh [10]). Consequently, this means that the mesh will only adapt to details that are visible in silhouette contours. For example, the template model only supports hands that are formed to a fist and contains no individual fingers. Our recordings contain many scenes where fingers of the users are visible. Therefore, the hand cannot be reconstructed correctly. However, we can recover such details in a rendering stage where individual fingers are visible in the texture. The advantage of using a generic mesh is that we can estimate the body shape of previously unknown people without additional 3D scanning. Note that the quality of feet in our results is comparatively low as the majority of our cameras are pointed towards the upper body and feet are not covered sufficiently in the view frustums. This is a limitation of our hardware setup, which was designed to achieve a good quality for the upper body.

5.4.5 Comparison to Related Approaches

In the current literature, there is no focus on real-time adaptation of a polygonal mesh to multi-view image data. Usually, related approaches focus only on a high quality reconstruction of the human body. Therefore, a direct comparison between our methods and literature that deals with human shape estimation is difficult. In Table 5.4, we compare the runtime of existing methods with our approach. For example, the approach by Wu et al. [165] estimates the illumination map of the environment in order to reconstruct fine details of the human body. Such computations result in a runtime of several minutes per frame. Simpler methods based on silhouette correspondence such as Vlasic et al. [158] adapt a polygonal mesh to image data in a few seconds. Their runtime is mainly limited

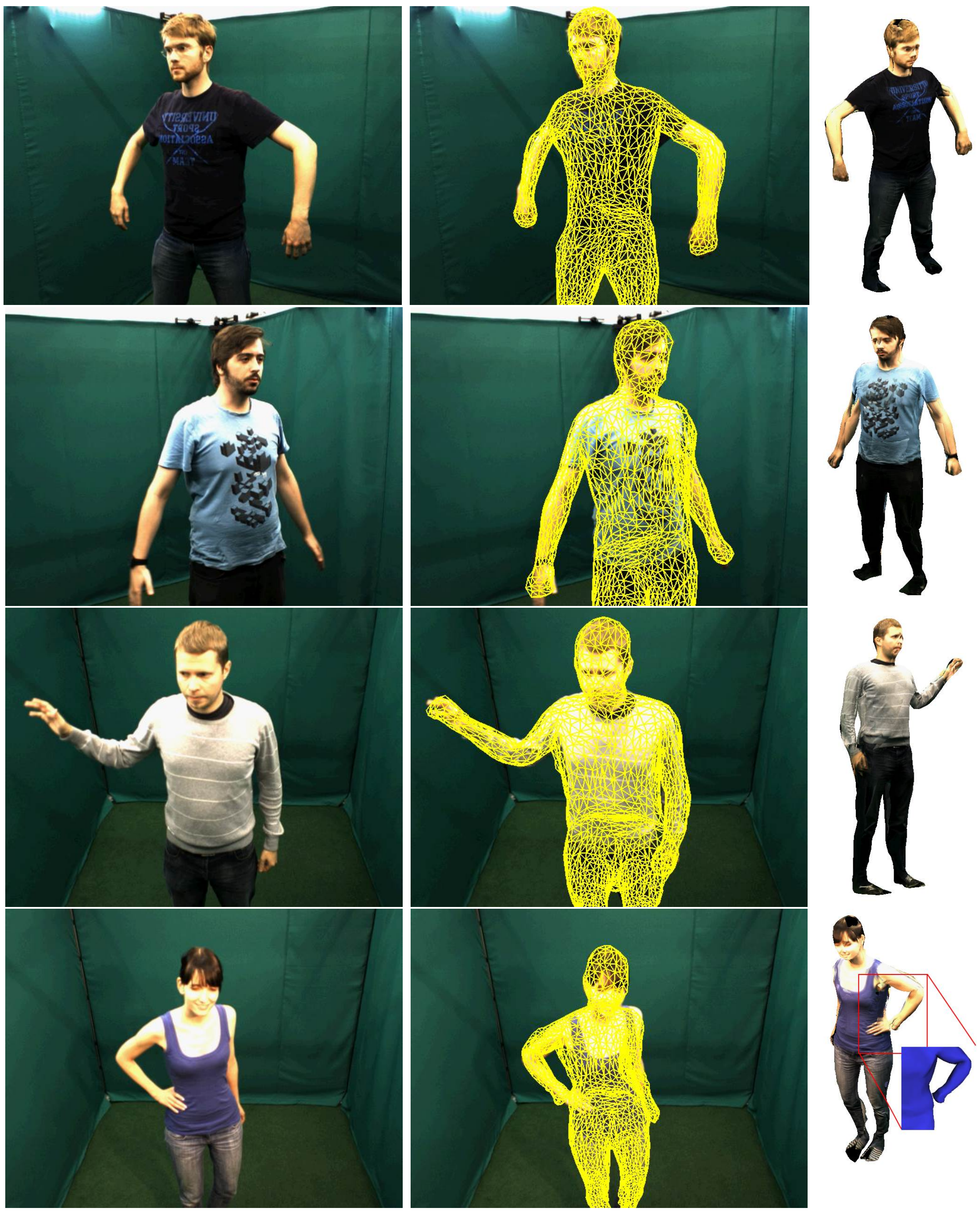


Figure 5.22: One input view (left), the adapted mesh as an image overlay (middle) and the textured 3D model (right). The last row demonstrates a situation where the hand touches the body, which cannot be modeled by our approach.

Method	**Model**	**Corresp.**	**Vertices**	**Time**
Wu et al. [164, 165]	S/M	I	80 000	10 min
Aguiar et al. [4]	V/M	S/D/T	30 000	27 s
Cagniart et al. [34]	M	S	10 000	25 s
Hofmann & Gavrila [79]	P	S/T	-	15 s
Vlasic et al. [158]	M	S	10 000	4.8 s
Huang et al. [81]	M/S	S/P	3 700	3.0 s
Gall et al. [57]	M	S/T	2 000	1.7 s
Our multigrid method (CG solver)	M/S	S	2 500	0.09 s
Our multigrid method (DC solver)			2 500	0.03 s
Our multigrid method (CG solver)			10 000	0.33 s
Our multigrid method (DC solver)			10 000	0.08 s

Model: (M)esh, (P)arametric, (S)keleton, (V)olumetric
Image correspondences: (S)ilhouette, (I)llumination, (T)exture, (D)epth, (P)oint clouds
Number of vertices (for meshes)
Time per frame

Table 5.4: Comparison of the time required to deform a human mesh to multi-camera data in seconds per frame.

by direct minimization of large linear systems of equations. In this thesis, we have shown how to improve on this runtime limitation. The last four rows in Table 5.4 are runtimes measured with our multigrid approach. We report the runtime for two different resolutions of the template mesh (2 500 and 10 000 vertices) and use either a conjugate gradient (CG) optimizer or our decoupled constraint (DC) solver. To the best of our knowledge, only our system is able to perform multi-view shape and pose adaptation of a human body model at interactive frame rates.

There exist other approaches to produce meshes of a scene or a body in real-time. For example, the commercial system built by 4D Views [1] captures a mesh directly from the visual hull. In Alexiadis et al. [6], multiple Kinect cameras are used to capture a human body from multiple viewpoints. Their method records partial surface meshes which are fused into a single mesh of the body. Other Kinect-based approaches such as Kainz et al. [85] enhance the KinectFusion algorithm [110] in order to capture a volumetric representation of a person with more than 10 cameras. However, all these methods produce a new mesh every frame. Thus, these methods cannot capture movement of a person with a consistent model.

An objective comparison of shape adaptation accuracy is difficult. There exists no standard evaluation dataset of moving persons with ground truth shape information because highly accurate body scans are only possible for static poses. Therefore, the most common error measure is the pixel overlap error, which we explained in Section 5.4.1. However, most methods for shape adaptation explicitly aim to optimize the silhouette reprojection error and thus yield a naturally small pixel overlap [81]. As shown in Fig. 5.11, a small overlap error does not imply plausible deformations. Alternatively, some researchers use unpublished synthetic scenes [164] to compute accuracy scores such as the mean vertex position error. Nevertheless, there is a need for a standardized dataset of captured human motion with high accuracy ground truth shape information for comparison.

5.5 A Complete Pose and Shape Estimation System

A complete pose and shape estimation system comprises all hardware and software aspects that have been described in this thesis. This includes components such as image recording, estimation of the initial pose and estimation of the body shape. Most components depend on the result of their predecessors. This leads to a natural sequence of processing steps that cannot be parallelized. We show the main algorithmic blocks of our complete pose and shape estimation system in Fig. 5.23.

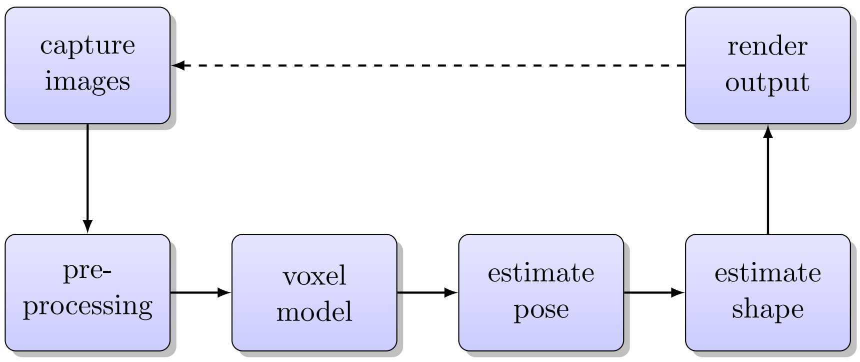


Figure 5.23: Flowchart of our complete pose and shape estimation system.

Our system is built upon the Coin3D [40] framework. In Coin3D, all components of a larger system are expressed in a scene-graph. Each component is a node in the graph which can have several attributes and possibly child nodes. Traditionally, Coin3D is used by the computer graphics community to compose 3D scenes for visualization. We have extended the basic framework by custom nodes that perform tasks such as recording images from a multi-camera setup or nodes that perform shape and pose estimation. All nodes can be configured through a human-readable text-file. Some nodes produce graphical output such as 3D meshes or line drawings which can be visualized. We can configure the nodes of our system in order perform pose and shape estimation from live images or recorded images from a database. At runtime, the complete graph is traversed once per frame and all nodes are executed in their natural sequence.

Similar to traditional visualization systems based on scene-graphs, we can display the graphical output generated by the scene-graph nodes. In Fig. 5.24, we show an example of our graphical user interface. It consists of a large visualization area that displays 3D content and a toolbar to configure parameters at runtime. In the example image, we show an estimated skeleton next to the adapted template mesh.

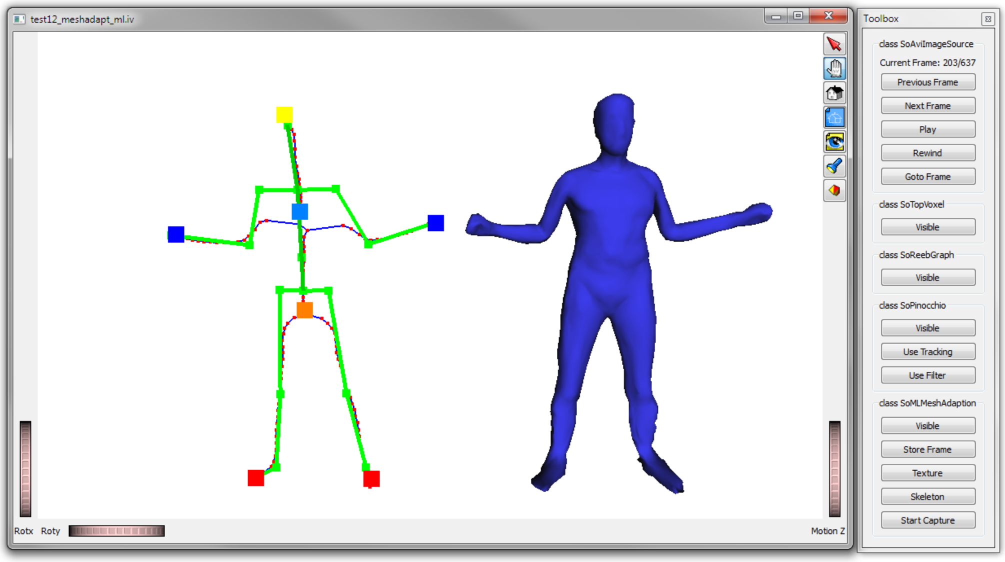


Figure 5.24: Our graphical user interface allows us to set parameters of the components and displays rendered results.

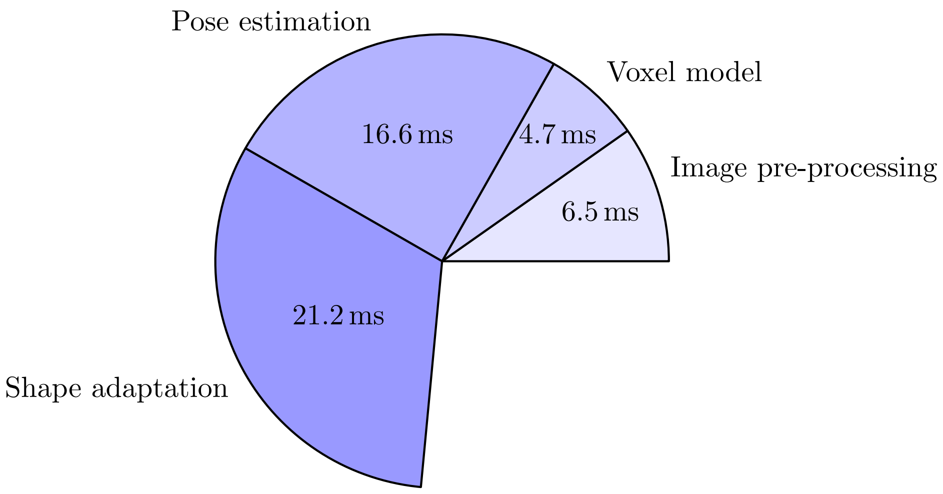


Figure 5.25: Analysis of the runtime of all components required for pose and shape estimation. The total area of the pie chart amounts to 66 ms, which is the maximum processing time possible when cameras are recording at 15 fps. Thus, the empty area is the available time for additional processing in every frame.

In a live system which performs real-time pose and shape estimation, the camera frame rate limits the total processing time allowed for all components. Our camera setup captures images at 15 frames per second. This means that the total processing time for each frame must not exceed 66 ms. In Fig. 5.25, we analyze the combined runtime of all components of our system. The presented time values are average runtimes of a typical interactive application where the user is captured by our camera setup and the adapted polygonal model is shown on a display. The combined runtime of our components is well below the real-time limit. This means that there are more than 15 ms available for additional computations such as rendering, cloth simulation or application specific tasks.

An important aspect for interactive systems is a low delay between the recording of images and a rendered output on the display. A delay that is too long can cause simulator sickness [137]. We have measured a delay of about 250 ms between actions in the real world and their mirror reaction in the artificial rendering. This delay was measured by recording the user and monitor simultaneously with an external high frame-rate camera. This delay is not only caused by processing times but includes the time for transmission of camera images, buffering of rendered images at the output stage and delays introduced by the TV monitor. Considering that the user does not have to perform any critical task in a virtual mirror scenario, this latency is acceptable.

We conclude that with our hardware setup and software runtime environment, it is possible to create interactive applications. Due to the short delay between image recording and graphical output, we prevent that users become annoyed by a visible lag. In the next section, we present typical applications that are feasible with our real-time adapted mesh.

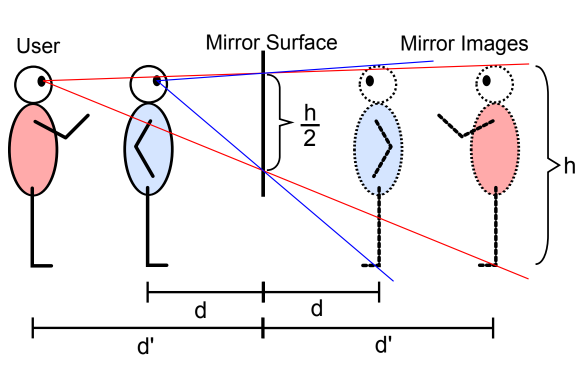
5.6 Applications

This section presents several applications that can be created using our recording hardware and real-time pose and shape estimation algorithms. Many applications are based on recording a 3D model of the user and producing a rendered output which shows the user in a new setting. The focus of this thesis is on the reconstruction of a such 3D model without an emphasis on rendering methods. There exist several methods to create a rendered image of the person without an explicit 3D model. Such methods are described in detail in the PhD thesis of Dr. Stefan Hauswiesner [72], which contains rendering methods tailored to hardware the system described in this thesis.

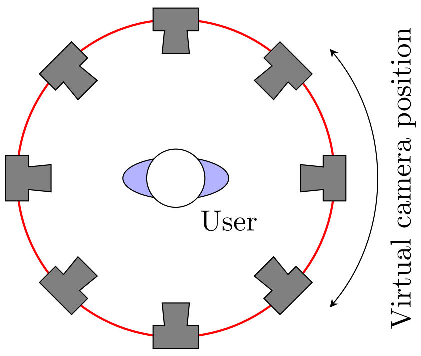
5.6.1 Virtual Mirrors

In Section 1.2, we have described the concept of the virtual mirror. A virtual mirror is a monitor that displays a live image of the user. Similar to a traditional mirror, the image of the user is mirrored. However, there is complete control over the graphical content that can be shown together with the mirror image. Examples are virtual clothing, augmented objects or a complete artificial scene in the background.

A virtual mirror needs to simulate the properties of a real mirror as accurately as possible. When standing in front of a mirror at a distance d, the mirror image of oneself appears exactly at the same distance but behind the mirror. Even though the distance d can change, the size of the user in the mirror image when projected on the mirror surface stays constant at half the height h of the user [21]. We illustrate this observation in



(a) How a mirror works



(b) Virtual camera setup

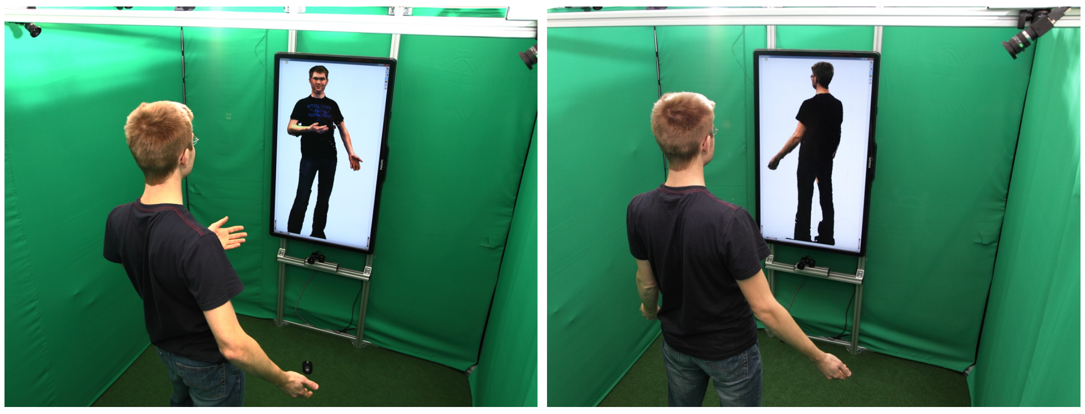
Figure 5.26: (a) Illustration of the mirror effect: The size of the image of the user remains constant at $h/2$, independent of the distance d between the user and the mirror. (b) A virtual camera can be positioned anywhere around the user.

Fig. 5.26(a). The maximum height of a person influences the choice of a suitable monitor for displaying the mirror image (see Section 5.2). Our television monitor measures 104 cm on the longer edge. Therefore, it can display the full-body mirror image of users up to two meters in height, which includes 99 percent of all people [150].

In order to simulate the effect of a virtual mirror, we use a virtual rendering camera to render the captured 3D model of the user. We place the rendering camera at the mirrored position of the eyes of the user and let the camera look in the direction of the mirror's surface normal. Thus, the distance between the user and the virtual camera is twice the distance d. The mirror image of a person will be half the height h of the real person when the focal length f of the virtual camera is twice the distance d between the person and the mirror surface. Details and derivations of the exact camera parameters can be found in Straka et al. [140].

Apart from classic mirror simulations, our system can be used to display a full 360° view of the user by rotating the virtual mirror camera around the body as seen in Fig. 5.26(b). In Fig. 5.27, we show how the user can view his mirror image from the front and even from the back. We propose a simple and intuitive way of controlling the view selection through hand gestures. The positions of the hands can be estimated using our pose estimation algorithm from Chapter 3. The user can trigger a clockwise or counterclockwise rotation of the mirror image by stretching his left or right hand sideways from his torso, respectively. Stretching out both hands resets the rotation and sets the camera back to the normal mirror mode. In order to minimize unintentional user inputs, we require the user to maintain a certain pose for at least one second before an action is triggered.

A virtual dressing room is a typical application for the 360° view mode. The customer tries on clothing and wants to find out how he or she looks from different sides. With a



(a) Mirror image (b) Rotated mirror image

Figure 5.27: A virtual mirror cannot only show a traditional mirror image but also allow a view from the back.

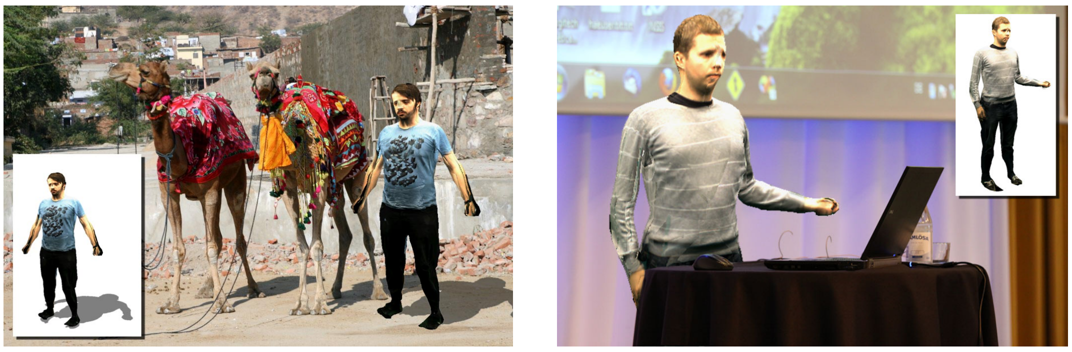
traditional mirror or a single camera virtual mirror, it is fairly easy to look at one's front and side. However, looking at one's own back is challenging as the user would have to rotate his or her head more than $90°$ to the side. This view is very easy to create with our free-viewpoint virtual mirror.

5.6.2 Embedding 3D Models into New Scenes

In the last section, we have shown how to render the user in a virtual mirror setting. We did not include any background in the output images but showed a mirrored image of the user against a white color. A captured 3D model of a human body can easily be integrated into 3D scenes or merged with photographs. There are several advantages to integrating a 3D model into such scenes. When rendering 3D data, it is possible to exploit depth information in order to handle occlusions correctly. Given an artificial light source, a virtual shadow can be generated and merged with the scene to increase realism. Typically, occlusion information and shadows cannot be automatically generated from 2D images. Given a 3D model, this information can be computed with negligible computation time. Moreover, due to the free choice of the viewpoint at render time, it is possible to match the perspective of the person and the new scene.

In Fig. 5.28(a), we show an example of how captured models can be embedded into a new scene. We combine a virtual person and a picture of camels to create the impression that this person was visiting India. For this image, we used the recorded 3D model and adjusted the viewpoint, scale and shadows during compositing.

The second example in Fig. 5.28(b) shows a person giving a talk at a conference while standing behind a desk. In reality, this persons was never at this conference. However, he can give his talk there virtually as he is seamlessly integrated into the scene. This is



(a) Virtual Tourism (b) Telepresence

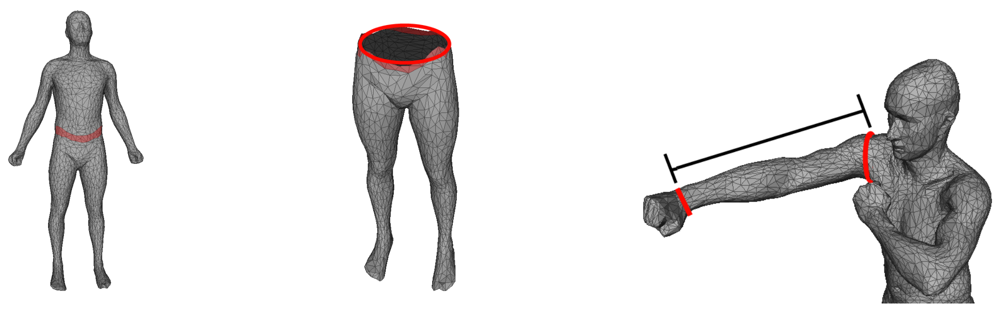
Figure 5.28: Two examples of how 3D models of people can be embedded in a new scene.

a typical application of tele-presence. A 3D model of the speaker and his voice can be recorded in our studio anywhere in the world. The 3D model, texture and audio data is then transferred to the conference via a network. Finally, the model is merged with the existing scene of the speaker's desk at the conference. With only a short delay, the audience of the talk will be able to listen to the speaker and see his image on a screen or projection. Such an image is not significantly different from live video of a physically present speaker.

Another field of application for rendered 3D models is cinematography and performance capture. In recent years, it has become popular to use dynamically captured bodies in movies [1]. Captured people can be used in scenes of crowds or as background actors. While foreground actors require a maximum of visual quality to look realistic, the quality of automatically captured actors is sufficient if they occupy only a small region of the screen. Note that there is no real-time requirement when recording animations of people for movies. Therefore, it may be beneficial to use a longer processing time in order to maximize visual quality [99]. A real-time capable shape estimation system can, however, be used to quickly preview recorded material without hours of processing.

5.6.3 Body Measurements

A 3D model of a body cannot only be used for rendering purposes. It can act as input data to measurement applications. Anthropomorphic measurements can be performed directly on a polygonal mesh. In Fig. 5.29, we show that several measurement points can be defined according to measurement standards such as ISO 8559 [82]. Measurements can be performed by analyzing the cross section of the polygonal mesh or by performing length measurements. When the mesh with defined measurement points is adapted to multi-view images of a person, measurement points are automatically moved to the correct positions. This eliminates the error prone direct detection of measurement points.



(a) Selection of the waistline (b) Measurement of the waistline (c) Measurement of arm lengths

Figure 5.29: Measurements points defined on the SCAPE mesh [10] enable predefined measurements once the mesh is adapted to images.

Worn clothing can change body dimensions significantly. This effect becomes even more challenging with loose clothing such as wide pants or long dresses. An easy solution to avoid this problem is to require that persons must wear tight clothing or be scanned semi-naked. While this may be a suitable requirement in a laboratory setting, it may not be accepted by customers in a store. Xu et al. [166] present a method that can estimate body dimensions from an adapted 3D model even in the presence of loose clothing. Alternatively, there exist methods to fit a parametric model of a person into the captured mesh [70].

While adapting a mesh for anthropomorphic measurements with our approach is possible, it may not be the best choice. Body measurements do not require a real-time capable system that performs body measurements at several frames per second. A slower but more accurate approach to adapt a template mesh to multi-view images should be used. In addition to silhouette correspondences, computing dense depth maps from input cameras should be considered [55]. Nevertheless, our mesh adaptation process is fast enough to perform multiple measurements and use robust statistics to obtain stable measurements over time. An actual implementation of body measurements with our approach remains, however, future work.

5.6.4 Virtual Cloth Simulations

Last but not least, we want to mention that a 3D model of a human body can be used to dress a person in virtual clothing. Virtual clothing is an essential component in a virtual dressing room. In order to wear virtual clothing, the user of the system selects garment items from a catalog. Then, the user can move around in the virtual dressing room and see the clothing augmented on his or her body image.

There exist several methods for cloth simulation in the literature that can fit and animate polygonal cloth models to a 3D body mesh [64, 109]. We show some examples in Fig. 5.30. Typically, the mesh of the body can be used as a collision object that must not be penetrated by clothing. An important requirement for cloth simulations is that the simulation is stable. Thus, rapid movement of body vertices or outliers must not cause the virtual clothing to behave abnormal. Possible solutions use learning algorithms to model possible cloth deformations [43]. A detailed evaluation of cloth simulation using our captured 3D models is out of the scope of this thesis and remains future work.

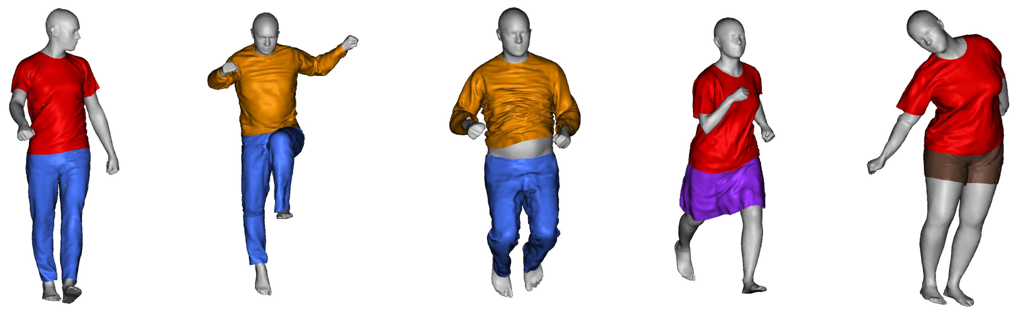


Figure 5.30: Examples of virtual clothing on a polygonal mesh model (images from [64]).

A cloth rendering method based on images is presented in Hauswiesner et al. [74]. It can augment a user's rendered image with virtual clothing that has been recorded with a multi-view setup. This method neither requires a polygonal mesh of the user nor a mesh of the clothing.

CHAPTER 6 CONCLUSION

6.1 Summary

Human pose and shape estimation is an important aspect when creating 3D models of persons. Existing approaches provide solutions for real-time pose estimation while other approaches perform shape estimation in an offline-stage. There is a lack of a combined system which estimates both the human pose and shape in real-time. A real-time-capable system is beneficial for interactive applications such as virtual dressing rooms and telepresence systems. In this thesis, we have addressed the issue of robust and fast pose and shape estimation when using only a single hardware setup. This section gives a summary of our key contributions and findings.

Real-time capturing of people is not possible without a well-designed recording setup. In Section 5.2, we have presented our hardware setup which consists of ten synchronized cameras that are connected to a single computer. From each camera image, we extract the silhouette of the user. By performing all computations on a single computer, we are able to process the captured images with minimal delay. There is no time overhead for network communication or synchronization between multiple devices. This is an important aspect for interactive systems such as virtual mirrors because the user can see himself and expects the mirror to instantly mimic his or her motions.

Human pose estimation is concerned with finding the optimal skeleton configuration for the person in the input images. In Chapter 3, we have presented a novel method to estimate the location of each skeleton joint from a center-line approximation of the body. This so-called skeletal graph is quickly constructed from a volumetric body scan. Our end-node labeling algorithm detects limbs in this graph and fits a skeleton model to the data. Experiments have shown that this method allows automatic initialization and quickly recovers from erroneous poses.

Given a rough pose of a person, we initialize a polygonal surface model and adapt it to the input silhouette images. Our approach in Chapter 4 accurately estimates a person's shape from multi-view images. First, the algorithm searches for correspondences between model vertices and silhouette contours. Then, the vertices are deformed such that the mesh optimally explains the silhouettes in all views. We have introduced a novel linear skeleton term for shape optimization. This term improves the stability of shape optimization and penalizes implausible deformations. Experimental results show that this skeleton term improves robustness and makes it possible to reduce the number of input cameras.

We achieve real-time shape adaptation through the use of a multi-grid structure and a decoupled constraint solver. This combination of multi-resolution processing and iterative optimization reduces the runtime for the most time-consuming tasks in shape adaptation. While related approaches report processing times of several seconds per frame, our method enables shape adaptation at camera frame rate. Many authors who perform runtime evaluations on their systems have to admit that they do not reach the expected performance, yet. A common promise is that faster hardware or GPU implementations can solve their problems in future work. Our algorithms use parallel processors to process raw image data only. We compress image data into a compact representation such that subsequent algorithms can be executed efficiently on a single-threaded CPU. For example, we compact a volumetric body scan into a skeletal graph with only a few nodes and use a coarse-to-fine approach for mesh adaptation.

We have presented a series of experiments in Chapter 5. They prove that our methods for human pose and shape estimation are working in real world settings. In addition, we have shown some applications where a real-time generated model of a human body is useful. In particular, we have pointed out the virtual mirror application which benefits from a minimum delay between image recording and the visualization of a virtual mirror image of the user.

6.2 Conclusion

In this thesis, we have introduced the idea of the virtual dressing room as a new concept to virtually try-on clothes. Our system contains several components that are essential for a virtual dressing room. We are able to capture the non-rigid 3D model of a moving person with a recording studio small enough to be placed in a regular store. We not only capture this model without manual initialization, but also estimate the current shape of the user at every camera frame in real-time. Automatic initialization is an essential requirement because users need not adhere to an initialization protocol or manually enter parameters into the system.

The 3D model of a person can be used for a variety of tasks in a virtual try-on application. For example, it can be rendered from a novel viewpoint to create the impression of a virtual mirror. The captured 3D shape can be used as a collision object for cloth simulations or interaction with virtual objects. A 3D model of a person has additional advantages besides the use in an interactive application. For example, the model of a customer can be recorded in a store, once. At a later time, this model can be used to show the user in new clothing without the need for capturing the customer again. For example, the 3D model of the user can be displayed at his or her home computer. Furthermore, a 3D model allows for body measurements to determine clothing sizes and select fitting garments. These possibilities open up a range of new marketing strategies such as personalized clothing catalogs.

Our contributions to shape and pose estimation are a step in the right direction towards a virtual dressing experience for customers. However, there are some challenges that are not addressed in this thesis. For example, most of our test subjects have been wearing tightly fitting clothing, had short hair and were of a similar height. In the real world and especially in a clothing store environment, there will be people with a wide range of body shape and clothing. With the current system, such variety cannot be handled. A possible solution is to select an appropriate template model for each user. Loose garment such as skirts or ponchos make it impossible for our pose estimation method to report a meaningful pose. Such clothing violates the assumption that the human body has a tubular structure with branches, which is required for voxel scooping. However, it is unclear if other generic pose estimation algorithms can handle such input data, especially when only silhouettes are available.

6.3 Future Work

The methods presented in this thesis yield usable results for estimating the pose and shape of a human body. However, we do not provide a general solution for building a full-featured virtual dressing room application. There are several aspects that we did not address. In this section, we elaborate on possible future work based on our results.

Our recording setup was designed during a time when the Kinect camera was a mere rumor on the internet and depth sensing cameras were extremely expensive. This fact steered our hardware decision towards a multi-view setup. Since the Kinect has been released in late 2010, cheap depth sensing hardware has become widely available. Future work should investigate a combination of depth sensing cameras and classic cameras in a combined installation. Such an installation comes with several challenges such as camera synchronization and interferences between multiple active cameras. However, it may be possible to reduce the total number of required cameras and still capture a full body model. Furthermore, the current research prototype would need a visual redesign and a new graphical user interface that looks more appealing to potential customers.

So far, we did not address self-collision. Self-collision or self-penetration may occur in both shape and pose estimation. For example, it must not be possible that an arm is estimated to be inside the torso. A possible solution to this problem is to include collision constraints to the shape adaptation process [109]. In addition, we cannot handle multiple persons or occluding objects with our framework when input silhouettes are merged. Liu et al. [99] propose to segment multiple persons directly from input silhouettes. Such segmentation will help our algorithms to disambiguate between different persons or even occluding objects.

Cloth simulation is a huge research area with many advances in the recent years. In this thesis, we did not apply cloth simulation on the reconstructed mesh of the person. Our current solution for cloth simulation is image-based [74]. However, it would be interesting to implement a real-time physics simulation for clothing and evaluate how stable and realistic cloth simulations will look. The estimated surface mesh can be used as a collision proxy for virtual clothing such that clothing does not penetrate the body. Alternatively, it may be possible to use the input silhouette images directly for such collision tests.

Our shape adaptation scheme currently adapts a single template mesh to silhouette images in every frame. Even if the adapted shape is systematically different from the template shape, we adapt the template to input data from scratch in every frame. A possible extension to our approach is to modify the template mesh such that systematic adaptations are represented in the template. For example, by iteratively updating the template mesh using existing adaptations, the template mesh will become a personalized mean shape for a specific user.

Finally, to evaluate the acceptance of our system by everyday users, a thorough user study is needed. Even though not evaluated systematically, feedback from users without a technical background have reported that graphical results sometimes look unpleasant. Such reports are most probably related to the *uncanny valley* hypothesis [107]. This hypothesis states that as representations of a human body become more realistic, human observers often respond with revulsion. Unfortunately, this effect is increased when the human model is moving. Therefore, care has to be taken that the model becomes either very realistic or sufficiently unrealistic to increase acceptance from users. In addition, it remains unclear if a company will ever use a camera-based system to promote their clothing. The risk that the presentation of virtual clothing on a personal avatar behaves abnormal can never be fully eliminated. According to companies such as PhiSix Fashion Labs [115], a fashion company desires complete control over the content shown to potential customers. Until now, pre-rendered animations of clothing are preferred over camera-captured user images.

We believe that our work will inspire future work in the area of human pose and shape estimation. For example, Huang et al. [81] have successfully integrated our skeleton constraint into patch-based mesh adaptation. Skeleton constraints could be used by the mesh editing community during design and animation phases. In addition, there is active research in graph-based human pose estimation [15] besides the strong influence of the Kinect camera. Given the results of this thesis and ideas for future directions, we are confident that virtual try-on systems will continue to be developed in the future.

APPENDIX A ACRONYMS AND SYMBOLS

List of Acronyms

AR	augmented reality
CAESAR	Civilian American and European Surface Anthropometry Resource
CD	compact disk
CG	conjugate gradient
CPU	central processing unit
DC	decoupled constraints
DP	dynamic programming
DTW	Dynamic time warping
GICP	Generalized Iterative Closest Point
GPU	graphics processing unit
GS	Gauss-Seidel
GUI	graphical user interface
HD	high definition
ICP	iterative closest point
IR	infra red
LED	light emitting diode
LME	Laplacian Mesh Editing
MAP	maximum a posteriori probability
PCA	principal component analysis
PF	particle filter
RGB	red green blue
RVM	relevance vector machine
SCAPE	Shape Completion and Animation of People
SIFT	scale invariant feature transform

SIMD	single instruction multiple data
SPD	symmetric positive definite
TOF	time-of-flight
VTO	virtual try-on

List of Symbols

Mesh and vertices		
$\mathscr{M}$		A polygonal mesh consisting of vertices $\mathbf{V}$ and faces $\mathbf{F}$
$\mathbf{V}$	$\mathbb{R}^{[3\times n]}$	Collection of n vertices
$\mathbf{v}_i$	$\mathbb{R}^3$	Vertex with index i
$\mathbf{n}_{\mathbf{v}_i}$	$\mathbb{R}^3$	vertex normal
$\mathbf{F}$	$\mathbb{N}^{[3\times m]}$	Collection of m faces
$\mathbf{f}_i$	$\mathbb{N}^3$	triangular face, contains vertex indices
$\mathbf{n}_{\mathbf{f}_i}$	$\mathbb{R}^3$	face normal

Skeleton and Bones		
G		defines a skeleton (collection of joints and bone hierarchy)
b_j		Bone with index j
$\mathbf{g}_j$	$\mathbb{R}^3$	Skeleton joint with index j
$\mathbf{g}_{prev(j)}$	$\mathbb{R}^3$	Parent skeleton joint for joint j
ρ_{ij}	$0 \leq \rho_{ij} \leq 1$	Skinning weight for vertex i that connects to bone j

Camera related symbols		
ℓ		Camera index
$\mathbf{P}_\ell$	$\mathbb{R}^{[3\times 4]}$	Projection matrix of camera ℓ
$\mathbf{K}_\ell$	$\mathbb{R}^{[3\times 3]}$	Intrinsic camera matrix of camera ℓ
$\mathbf{p}_k^\ell$	$\mathbb{R}^2$	Pixel position of point k in the image of camera ℓ

Differential coordinates		
δ_i	$\mathbb{R}^3$	Differential vertex coordinate for vertex i
β_i	$\mathbb{R}^3$	Differential bone coordinate for vertex i
β_{ij}	$\mathbb{R}^3$	Differential bone coordinate for vertex i connected to a single bone b_j

Table A.2: Variable names, their definition domains and a short description.

APPENDIX B LIST OF PUBLICATIONS

My work at the Institute for Computer Graphics and Vision led to the following peer-reviewed publications. They are listed in chronological order along with their abstracts. Primary publications are closely related to this thesis while secondary publications are based on results and developments of this thesis.

B.1 Primary Publications

A Free-Viewpoint Virtual Mirror with Marker-Less User Interaction

Straka Matthias, Hauswiesner Stefan, Rüther Matthias and Bischof Horst
In: *Proceedings of the 17th Scandinavian Conference on Image Analysis (SCIA)*
May 2011, Ystad, Sweden [140]
(Accepted for poster presentation)

Abstract: We present a Virtual Mirror system which is able to simulate a physically correct full-body mirror on a monitor. In addition, users can freely rotate the mirror image which allows them to look at themselves from the side or from the back, for example. This is achieved through a multiple camera system and visual hull based rendering. A real-time 3D reconstruction and rendering pipeline enables us to create a virtual mirror image at 15 frames per second on a single computer. Moreover, it is possible to extract a three dimensional skeleton of the user which is the basis for marker-less interaction with the system.

Skeletal Graph Based Human Pose Estimation in Real-Time

Straka Matthias, Hauswiesner Stefan, Rüther Matthias and Bischof Horst
In: *Proceedings of the British Machine Vision Conference (BMVC)*
August 2011, Dundee, GB [141]
(Accepted for poster presentation)

Abstract: We propose a new method to quickly and robustly estimate the 3D pose of the human skeleton from volumetric body scans without the need for visual markers. The core principle of our algorithm is to apply a fast center-line extraction to 3D voxel data and robustly fit a skeleton model to the resulting graph. Our algorithm allows for automatic, single-frame initialization and tracking of the human pose while being fast enough for real-time applications at up to 30 frames per second. We provide an extensive qualitative and quantitative evaluation of our method on real and synthetic datasets which demonstrates the stability of our algorithm even when applied to long motion sequences.

Simultaneous Shape and Pose Adaption of Articulated Models using Linear Optimization

Straka Matthias, Hauswiesner Stefan, Rüther Matthias and Bischof Horst
In: *Proceedings of the European Conference on Computer Vision (ECCV)*
October 2012, Firenze, Italy [143]
(Accepted for poster presentation)

Abstract: We propose a novel formulation to express the attachment of a polygonal surface to a skeleton using purely linear terms. This enables to simultaneously adapt the pose and shape of an articulated model in an efficient way. Our work is motivated by the difficulty to constrain a mesh when adapting it to multi-view silhouette images. However, such an adaption is essential when capturing the detailed temporal evolution of skin and clothing of a human actor without markers. While related work is only able to ensure surface consistency during mesh adaption, our coupled optimization of the skeleton creates structural stability and minimizes the sensibility to occlusions and outliers in input images. We demonstrate the benefits of our approach in an extensive evaluation. The skeleton attachment considerably reduces implausible deformations, especially when the number of input views is limited.

Rapid Skin: Estimating the 3D Human Pose and Shape in Real-Time

Straka Matthias, Hauswiesner Stefan, Rüther Matthias and Bischof Horst
In: *Proceedings of Conference on 3D Imaging, Processing, Visualization and Transmission (3DimPVT)*
October 2012, Zurich, Switzerland [142]
(Accepted for oral presentation)

Abstract: We present a novel approach to adapt a watertight polygonal model of the human body to multiple synchronized camera views. While previous approaches yield excellent quality for this task, they require processing times of several seconds, especially for high resolution meshes. Our approach delivers high quality results at interactive rates when a roughly initialized pose and a generic articulated body model are available. The key novelty of our approach is to use a Gauss-Seidel type solver to iteratively solve nonlinear constraints that deform the surface of the model according to silhouette images. We evaluate both the visual quality and accuracy of the adapted body shape on multiple test persons. While maintaining a similar reconstruction quality as previous approaches, our algorithm reduces processing times by a factor of 20. Thus it is possible to use a simple human model for representing the body shape of moving people in interactive applications.

B.2 Secondary Publications

Coherent Image-Based Rendering of Real-World Objects

Hauswiesner Stefan, **Straka Matthias** and Reitmayr Gerhard
In: *Proceedings of the 2011 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*
February 2011, San Francisco, CA
(Accepted for oral presentation)

Abstract: Many mixed reality systems require the real-time capture and re-rendering of the real world to integrate real objects more closely with the virtual graphics. This includes novel view-point synthesis for virtual mirror or telepresence applications. For real-time performance, the latency between capturing the real world and producing the virtual output needs to be as little as possible. Image-based visual hull (IBVH) rendering is capable of rendering novel views from segmented images in real time. We improve upon existing IBVH implementations in terms of robustness and performance by reformulating the tasks of major components. Moreover, we enable high resolutions and little latency by exploiting view- and frame coherence. The suggested algorithm includes image warping between successive frames under the constraint of redraw volumes. These volumes form a boundary of the motion and deformation in the scene, and can be constructed efficiently by describing them as the visual hull of a set of bounding rectangles which are cast around

silhouette differences in image-space. As a result, our method can handle arbitrarily moving and deforming foreground objects and free viewpoint motion at the same time, while still being able to reduce workload by reusing previous rendering results.

Image-Based Clothes Transfer

Hauswiesner Stefan, **Straka Matthias** and Reitmayr Gerhard
In: *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*
October 2011, Basel, Switzerland
(Accepted for oral presentation)

Abstract: Virtual dressing rooms for the fashion industry and digital entertainment applications aim at creating an image or a video of a user in which he or she wears different garments than in the real world. Such images can be displayed, for example, in a magic mirror shopping application or in games and movies. Current solutions involve the error-prone task of body pose tracking. We suggest an approach that allows users who are captured by a set of cameras to be virtually dressed with previously recorded garments in 3D. By using image-based algorithms, we can bypass critical components of other systems, especially tracking based on skeleton models. We rather transfer the appearance of a garment from one user to another by image processing and image-based rendering. Using images of real garments allows for photo-realistic rendering quality with high performance.

Free Viewpoint Virtual Try-On With Commodity Depth Cameras

Hauswiesner Stefan, **Straka Matthias** and Reitmayr Gerhard
In: *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI)*
December 2011, Hong Kong, China
(Accepted for oral presentation)

Abstract: We present a system that allows users to interactively control a 3D model of themselves at home using a commodity depth camera. It augments the model with virtual clothes that can be downloaded. As a result, users can enjoy a private, virtual try-on experience in their own homes. As a prerequisite, the user needs to enter or pass through a multi-camera setup that captures him or her in a fraction of a second. From the captured data, a 3D model is created. The model is transmitted to the user's home system to serve as a realistic avatar for the virtual try-on application. The system provides free-viewpoint high quality rendering quality with smooth animation and correct occlusion, and therefore improves the state of the art in terms of quality. It utilizes cheap hardware and therefore is affordable for and accessible to a wide audience.

Multi-GPU Image-based Visual Hull Rendering

Hauswiesner Stefan, Khlebnikov Rostislav, Steinberger Markus, **Straka Matthias** and Reitmayr Gerhard
In: *Proceedings of the Eurographics Symposium on Parallel Graphics and Visualization (EGPGV)*
May 2012, Sardinia, Italy
(Accepted for oral presentation)

Abstract: Many virtual mirror and telepresence applications require novel viewpoint synthesis with little latency to user motion. Image-based visual hull (IBVH) rendering is capable of rendering arbitrary views from segmented images without an explicit intermediate data representation, such as a mesh or a voxel grid. By computing depth images directly from the silhouette images, it usually outperforms indirect methods. GPU-hardware accelerated implementations exist, but due to the lack of an intermediate representation no multi-GPU parallel strategies and implementations are currently available. This paper suggests three ways to parallelize the IBVH-pipeline and maps them to the sorting classification that is often applied to conventional parallel rendering systems. In addition to sort-first parallelization, we suggest a novel sort-last formulation that regards cameras as scene objects. We enhance this method's performance by a block-based encoding of the rendering results. For interactive systems with hard real-time constraints, we combine the algorithm with a multi-frame rate (MFR) system. We suggest a combination of forward and backward image warping to improve the visual quality of the MFR rendering. We observed the runtime behavior of the suggested methods and assessed how their performance scales with respect to input and output resolutions and the number of GPUs. By using additional GPUs, we reduced rendering times by up to 60%. Multi-frame rate viewing can even be ten times faster.

Virtual Try-On Through Image-based Rendering

Hauswiesner Stefan, **Straka Matthias** and Reitmayr Gerhard
In: *IEEE Transactions on Visualization and Computer Graphics*
Volume 19, Issue 9, September 2013
(Accepted for journal publication)

Abstract: Virtual try-on applications have become popular because they allow users to watch themselves wearing different clothes without the effort of changing them physically. This helps users to make quick buying decisions and, thus, improves the sales efficiency of retailers. Previous solutions usually involve motion capture, 3D reconstruction or modeling, which are time consuming and not robust for all body poses. Our method avoids these steps by combining image-based renderings of the user and previously recorded garments.

It transfers the appearance of a garment recorded from one user to another by matching input and recorded frames, image-based visual hull rendering, and online registration methods. Using images of real garments allows for a realistic rendering quality with high performance. It is suitable for a wide range of clothes and complex appearances, allows arbitrary viewing angles, and requires only little manual input. Our system is particularly useful for virtual try-on applications as well as interactive games.

Temporal Coherence in Image-based Visual Hull Rendering

Hauswiesner Stefan, **Straka Matthias** and Reitmayr Gerhard
In: *IEEE Transactions on Visualization and Computer Graphics*
Volume 19, Issue 10, October 2013
(Accepted for journal publication)

Abstract: Image-based visual hull rendering is a method for generating depth maps of a desired viewpoint from a set of silhouette images captured by calibrated cameras. It does not compute a view-independent data representation, such as a voxel grid or a mesh, which makes it particularly efficient for dynamic scenes. When users are captured, the scene is usually dynamic, but does not change rapidly because people move smoothly within a subsecond time frame. Exploiting this temporal coherence to avoid redundant calculations is challenging because of the lack of an explicit data representation. This paper analyses the image-based visual hull algorithm to find intermediate information that stays valid over time and is therefore worth to make explicit. We then derive methods that exploit this information to improve the rendering performance. Our methods reduce the execution time by up to 25%. When the user's motions are very slow, reductions of up to 50% are achieved.

B.3 Collaborative Statement

Aside from my primary supervisor Prof. Horst Bischof, there has been collaboration with a number of colleagues at the Institute for Computer Vision and Graphics. As the manager of the Robot Vision lab, Matthias Rüther provided help in writing the publications that lead to this thesis and contributed ideas for the evaluation of the presented algorithms. There has been close collaboration with Stefan Hauswiesner during building the experimental hardware setup, in creating graphical output and throughout all publications. In particular, the software framework presented in Section 5.5, which connects all vision and graphics algorithms, is joint work with Stefan.

Bibliography

[1] 4D View Solutions (2013). online. http://www.4dviews.com/.

[2] Agarwal, A. and Triggs, B. (2004). 3D human pose from silhouettes by relevance vector regression. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 882–888.

[3] Agarwal, S., Snavely, N., Simon, I., Seitz, S., and Szeliski, R. (2009). Building rome in a day. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 72–79, Kyoto, JP.

[4] Aguiar, E. d., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S. (2008). Performance capture from sparse multi-view video. *ACM Transactions on Graphics (TOG)*, 27(3):1–10.

[5] Alcoverro, M., Casas, J. R., and Pardàs, M. (2010). Skeleton and shape adjustment and tracking in multicamera environments. In *Proceedings of Articulated Motion and Deformable Objects (AMDO)*, volume 6169 of *Lecture Notes in Computer Science (LNCS)*, pages 88–87. Springer-Verlag.

[6] Alexiadis, D., Zarpalas, D., and Daras, P. (2013). Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339 – 358.

[7] Allen, B., Curless, B., and Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics (TOG)*, 22(3):587–594.

[8] Andriluka, M., Roth, S., and Schiele, B. (2009). Pictorial structures revisited: People detection and articulated pose estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1014–1021.

[9] Andriluka, M., Roth, S., and Schiele, B. (2010). Monocular 3D pose estimation and tracking by detection. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 623–630.

[10] Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., and Davis, J. (2005). SCAPE: shape completion and animation of people. *ACM Transactions on Graphics (TOG)*, 24(3):408–416.

[11] Baak, A., Müller, M., Bharaj, G., Seidel, H.-P., and Theobalt, C. (2011). A data-driven approach for real-time full body pose reconstruction from a depth camera. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1092–1099.

[12] Bai, X. and Latecki, L. J. (2008). Path similarity skeleton graph matching. *Pattern Analysis and Machine Intelligence (PAMI)*, 30(7):1282–1292.

[13] Baillot, Y. and Rolland, J. (1996). Fundamental principles of tracking technology for virtual environments. Technical Report 4, Center for Research and Education in Optics and Lasers (CREOL), Orlando, FL.

[14] Bakken, R. and Eliassen, L. (2012). Real-time 3D skeletonisation in computer vision-based human pose estimation using GPGPU. In *Proceedings of the International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 61–67.

[15] Bakken, R. H. and Hilton, A. (2012a). Real-time pose estimation using constrained dynamics. In *Proceedings of Articulated Motion and Deformable Objects (AMDO)*, volume 7378 of *Lecture Notes in Computer Science (LNCS)*, pages 37–46, Mallorca, SP. Springer-Verlag.

[16] Bakken, R. H. and Hilton, A. (2012b). Real-time pose estimation using tree structures built from skeletonised volume sequences. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP)*, pages 181–190.

[17] Ballan, L. and Cortelazzo, G. M. (2008). Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*.

[18] Baran, I. and Popović, J. (2007). Automatic rigging and animation of 3D characters. *ACM Transactions on Graphics (TOG)*, 26(3):1–8.

[19] Barrett, R., Berry, M., Chan, T. F., Demmel, J., Donato, J., Dongarra, J., Eijkhout, V., Pozo, R., Romine, C., and der Vorst, H. V. (1994). *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. SIAM, 2nd edition edition.

[20] Bennett, K. and Osborne, R. (1986). Interobserver measurement reliability in anthropometry. *Human Biology*, 39:124–130.

[21] Bertaminia, M. and Parks, T. E. (2005). On what people know about images on mirrors. *Cognition*, 98(1):85–104.

[22] Biasotti, S., Falcidieno, B., and Spagnuolo, M. (2000). Extended reeb graphs for surface understanding and description. In Borgefors, G., Nyström, I., and Baja, G. S., editors, *Discrete Geometry for Computer Imagery*, volume 1953 of *Lecture Notes in Computer Science (LNCS)*, pages 185–197. Springer Berlin Heidelberg.

[23] Blum, H. (1967). A transformation for extracting new descriptors of shape. In Wathen-Dunn, W., editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge.

[24] Bodymetrics (2013). Body-mapping technology that makes the perfect fit beautifully simple. online. http://www.bodymetrics.com/.

[25] BodyPal (2013). online. http://www.bodypal.com/.

[26] Bornemann, F. A. and Deuflhard, P. (1996). The cascadic multigrid method for elliptic problems. *Numerische Mathematik*, 75(2):135–152.

[27] Botsch, M., Bommes, D., and Kobbelt, L. (2005). Efficient linear system solvers for mesh processing. In Martin, R., Bez, H., and Sabin, M., editors, *Mathematics of Surfaces XI*, volume 3604 of *Lecture Notes in Computer Science (LNCS)*, pages 62–83. Springer Berlin Heidelberg.

[28] Botsch, M. and Sorkine, O. (2008). On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230.

[29] Bouguet, J.-Y. (2010). Camera calibration toolbox for matlab. online. http://www.vision.caltech.edu/bouguetj/calib_doc/.

[30] Briggs, W. L., Henson, V. E., and McCormick, S. F. (2000). *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics (SIAM), second edition.

[31] Brown, D. (1971). Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866.

[32] Butler, A., Izadi, S., Hilliges, O., Molyneaux, D., Hodges, S., and Kim, D. (2012). Shake'n'sense: Reducing interference for overlapping structured light depth cameras. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1933–1936.

[33] CAESAR (2013). The civilian american and european surface anthropometry resource project. online. http://www.sae.org/standardsdev/tsb/cooperative/caesar.htm.

[34] Cagniart, C., Boyer, E., and Ilic, S. (2010). Probabilistic deformable surface tracking from multiple videos. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume 6314 of *Lecture Notes in Computer Science (LNCS)*, pages 326–339. Springer-Verlag.

[35] Carnegie Mellon Graphics Lab (2013). CMU Graphics Lab Motion Capture Database. online. http://mocap.cs.cmu.edu/.

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

[37] Chen, J. (2009). GPU technology trends and future requirements. In *Proceedings of the IEEE International Electron Devices Meeting (IEDM)*, pages 1–6.

[38] Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. (2008). Algorithm 887: CHOLMOD, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)*, 35(3):22:1–22:14.

[39] Chen, Y., Liu, Z., and Zhang, Z. (2013). Tensor-based human body modeling. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 105–112.

[40] Coin 3D (2011). A high-level, retained-mode toolkit for effective 3D graphics development. online. https://bitbucket.org/Coin3D/coin/wiki/Home.

[41] Cornea, N., Silver, D., and Min, P. (2007). Curve-skeleton properties, applications and algorithms. *IEEE Transactions on Visualization and Computer Graphics*, 13(3):530–548.

[42] Correa, P., Marqués, F., Marichal, X., and Macq, B. (2008). 3D posture estimation using geodesic distance maps. *Multimedia Tools Applications*, 38:365–384.

[43] de Aguiar, E., Sigal, L., Treuille, A., and Hodgins, J. K. (2010). Stable spaces for real-time clothing. *ACM Transactions on Graphics (TOG)*, 29(4):106:1–106:9.

[44] Droeschel, D. and Behnke, S. (2011). 3D body pose estimation using an adaptive person model for articulated ICP. In *Proceedings of the International Conference on Intelligent Robotics and Applications (ICIRA)*, pages 157–167.

[45] Eichner, M., Marin-Jimenez, M., Zisserman, A., and Ferrari, V. (2012). 2D articulated human pose estimation and retrieval in (almost) unconstrained still images. *International Journal of Computer Vision (IJCV)*, 99(2):190–214.

[46] Eigen 3.2 (2013). A C++ template library for linear algebra: matrices, vectors, numerical solvers, and related algorithms. online. http://eigen.tuxfamily.org/.

[47] Fan, S. and Perrie, F. P. (2010). Photo hull regularized stereo. *Image and Vision Computing*, 28(4):724–730.

[48] Fechteler, P., Hilsmann, A., Eisert, P., Broeck, S. V., Stevens, C., Wall, J., Sanna, M., Mauro, D. A., Kuijk, F., Mekuria, R., Cesar, P., Monaghan, D., O'Connor, N. E., Daras, P., Alexiadis, D., and Zahariadis, T. (2013). A framework for realistic 3D tele-immersion. In *Proceedings of the International Conference on Computer Vision/Computer Graphics Collaboration Techniques (MIRAGE)*, pages 12:1–12:8, New York, NY, USA. ACM.

[49] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision (IJCV)*, 61(1):55–79.

[50] Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

[51] Fitting Reality (2013). Online shopping is easier than ever. online. http://fittingreality.com.

[52] Franco, J. and Boyer, E. (2009). Efficient polyhedral modeling from silhouettes. *Pattern Analysis and Machine Intelligence (PAMI)*, 31(3):414–427.

[53] Franco, J.-S. and Boyer, E. (2003). Exact polyhedral visual hulls. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 329–338.

[54] Furukawa, Y. and Ponce, J. (2008). Dense 3D motion capture from synchronized video streams. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1–8.

[55] Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multi-view stereopsis. *Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376.

[56] Gall, J., Rosenhahn, B., Brox, T., and Seidel, H.-P. (2010). Optimization and filtering for human motion capture. *International Journal of Computer Vision (IJCV)*, 87(1–2):75–92.

[57] Gall, J., Stoll, C., de Aguiar, E., Theobalt, C., Rosenhahn, B., and Seidel, H.-P. (2009). Motion capture using joint skeleton tracking and surface estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1746–1753.

[58] Ganapathi, V., Plagemann, C., Koller, D., and Thrun, S. (2010). Real time motion capture using a single time-of-flight camera. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 755–762.

[59] Garland, M. and Heckbert, P. S. (1997). Surface simplification using quadric error metrics. In *Proceedings of the ACM SIGGRAPH*, pages 209–216, New York, NY, USA.

[60] Giovanni, S., Choi, Y., Huang, J., Khoo, E., and Yin, K. (2012). Virtual try-on using kinect and HD camera. In Kallmann, M. and Bekris, K., editors, *Proceedings of Motion in Games (MIG)*, volume 7660 of *Lecture Notes in Computer Science (LNCS)*, pages 55–65.

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

[62] Gond, L., Sayd, P., Chateau, T., and Dhome, M. (2008). A 3D shape descriptor for human pose recovery. In *Proceedings of Articulated Motion and Deformable Objects (AMDO)*, volume 5098 of *Lecture Notes in Computer Science (LNCS)*, pages 370–379. Springer-Verlag.

[63] Graber, G., Pock, T., and Bischof, H. (2011). Online 3D reconstruction using convex optimization. In *Proceedings of the Workshop on Live Dense Reconstruction From Moving Cameras (ICCVW)*, pages 708–711.

[64] Guan, P., Reiss, L., Hirshberg, D., Weiss, A., and Black, M. (2012). Drape: Dressing any person. *ACM Transactions on Graphics (TOG)*, 31(4):35:1–35:10.

[65] Guan, P., Weiss, A., Bălan, A. O., and Black, M. J. (2009). Estimating human shape and pose from a single image. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1381–1388.

[66] Gyles, D. B. (1986). *Your Vital Statistics: The Ultimate Book About the Average Human Being*. Lyle Stuart.

[67] Hansard, M., Lee, S., Choi, O., and Horaud, R. (2013). *Time-of-Flight cameras: Principles, Methods and Applications*. Springer Briefs in Computer Science. Springer-Verlag.

[68] Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition.

[69] Hasler, N., Ackermann, H., Rosenhahn, B., Thormählen, T., and Seidel, H.-P. (2010). Multilinear pose and body shape estimation of dressed subjects from image sets. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1823–1830, San Francisco, USA.

[70] Hasler, N., Stoll, C., Rosenhahn, B., Thormählen, T., and Seidel, H.-P. (2009a). Estimating body shape of dressed humans. *Computer Graphics*, 33(3):211–216.

[71] Hasler, N., Stoll, C., Sunkel, M., Rosenhahn, B., and Seidel, H.-P. (2009b). A statistical model of human pose and body shape. *Computer Graphics Forum*, 28(2):337–346.

[72] Hauswiesner, S. (2013). *Efficient Image-based Augmentations*. PhD thesis, Graz University of Technology.

[73] Hauswiesner, S., Straka, M., and Reitmayr, G. (2011). Coherent image-based rendering of real-world objects. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D)*, pages 183–190, San Francisco, CA, USA. ACM.

[74] Hauswiesner, S., Straka, M., and Reitmayr, G. (2013). Virtual try-on through image-based rendering. *IEEE Transactions on Visualization and Computer Graphics*, 19(9):1552–1565.

[75] Hen, Y. W. and Paramesran, R. (2009). Single camera 3D human pose estimation: A review of current techniques. In *Proceedings of the International Conference for Technical Postgraduates*, pages 1–8.

[76] Hilsmann, A. and Eisert, P. (2009). Tracking and retexturing cloth for real-time virtual clothing applications. In *Proceedings of the International Conference on Computer Vision/Computer Graphics Collaboration Techniques (MIRAGE)*, pages 94–105.

[77] Hilsmann, A., Schneider, D., and Eisert, P. (2011). Template-free shape from texture with perspective cameras. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 26.1–26.10, Dundee, Scotland.

[78] Hirai, M., Ukita, N., and Kidode, M. (2010). Real-time pose regression with fast volume descriptor computation. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 1852–1855.

[79] Hofmann, M. and Gavrila, D. M. (2011). 3D human model adaptation by frame selection and shape-texture optimization. *Computer Vision and Image Understanding (CVIU)*, 115(11):1559–1570.

[80] Hoppe, H. (1996). Progressive meshes. In *Proceedings of the ACM SIGGRAPH*, pages 99–108, New York, NY, USA. ACM.

[81] Huang, C., Boyer, E., and Ilic, S. (2013). Robust human body shape and pose tracking. In *Proceedings of the International Conference on 3D Vision (3DV)*.

[82] ISO 8559:1989 (E) (1989). *Garment construction and anthropometric surveys – Body dimensions*. International Organization for Standardization, Geneva, CH.

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

[84] Jones, P. R., West, G. M., Harris, D. H., and Read, J. B. (1989). The loughborough anthropometric shadow scanner (LASS). *Endeavour*, 13(4):162–168.

[85] Kainz, B., Hauswiesner, S., Reitmayr, G., Steinberger, M., Grasset, R., Gruber, L., Veas, E., Kalkofen, D., Seichter, H., and Schmalstieg, D. (2012). OmniKinect: Real-time dense volumetric data acquisition and applications. In *Proceedings of the Symposium on Virtual Reality Software and Technology (VRST)*, pages 25–32.

[86] Kalman, R. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45.

[87] Kanaujia, A., Haering, N., Taylor, G., and Bregler, C. (2011). 3D human pose and shape estimation from multi-view imagery. In *Proceedings of Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 49–56, Colorado Springs, CO, USA.

[88] Kavan, L., Collins, S., Žára, J., and O'Sullivan, C. (2007). Skinning with dual quaternions. In *Proceedings of the Symposium on Interactive 3D Graphics and Games (I3D)*, pages 39–46. ACM.

[89] Kazhdan, M., Bolitho, M., and Hoppe, H. (2006). Poisson surface reconstruction. In *Symposium on Geometry Processing*, pages 61–70.

[90] Kuhn, H. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

[91] Kutulakos, K. N. and Seitz, S. M. (2000). A theory of shape by space carving. *International Journal of Computer Vision (IJCV)*, 38(3):199–218.

[92] Ladikos, A., Benhimane, S., and Navab, N. (2008). Efficient visual hull computation for real-time 3D reconstruction using CUDA. In *Proceedings of Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, Anchorage, AK, USA.

[93] Laurentini, A. (1994). The visual hull concept for silhouette-based image understanding. *Pattern Analysis and Machine Intelligence (PAMI)*, 16(2):150–162.

[94] Lee, V. W., Kim, C., Chhugani, J., Deisher, M., Kim, D., Nguyen, A. D., Satish, N., Smelyanskiy, M., Chennupaty, S., Hammarlund, P., et al. (2010). Debunking the 100x GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU. *ACM SIGARCH Computer Architecture News*, 38(3):451–460.

[95] Li, J. and Lu, G. (2011). Skeleton driven animation based on implicit skinning. *Computers & Graphics*, 35(5):945–954.

[96] Li, K., Dai, Q., and Xu, W. (2011). Markerless shape and motion capture from multi-view video sequences. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(3):320–334.

[97] Lindeberg, T. (1994). *Scale-Space Theory in Computer Vision*. Springer-Verlag.

[98] Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rössl, C., and Seidel, H.-P. (2004). Differential coordinates for interactive mesh editing. In *Proceedings of the International Conference on Shape Modeling and Applications*, pages 181–190.

[99] Liu, Y., Gall, J., Stoll, C., Dai, Q., Seidel, H.-P., and Theobalt, C. (2013). Markerless motion capture of multiple characters using multi-view image segmentation. *Pattern Analysis and Machine Intelligence (PAMI)*, 35(11):2720–2735.

[100] Lorensen, W. E. and Cline, H. E. (1987). Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, 21(4):163–169.

[101] Lucas, B. D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679.

[102] Luo, X., Berendsen, B., Tan, R. T., and Veltkamp, R. C. (2010). Human pose estimation for multiple persons based on volume reconstruction. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 3591–3594.

[103] Maimone, A. and Fuchs, H. (2011). Encumbrance-free telepresence system with real-time 3D capture and display using commodity depth cameras. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 137–146.

[104] Menier, C., Boyer, E., and Raffin, B. (2006). 3D skeleton-based body pose recovery. In *Proceedings of the International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT)*, pages 389–396.

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

[106] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding (CVIU)*, 104(2–3):90–126.

[107] Mori, M. (1970). The uncanny valley. *Energy*, 7(4):33–35.

[108] Müller, M. (2008). Hierachical position based dynamics. In *Proceedings of the Workshop on Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, pages 1–10.

[109] Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. (2007). Position based dynamics. *Journal of Visual Communication Image Representation*, 18(2):109–118.

[110] Newcombe, R. A., Davison, A. J., Izadi, S., Kohli, P., Hilliges, O., Shotton, J., Molyneaux, D., Hodges, S., Kim, D., and Fitzgibbon, A. (2011). Kinectfusion: Real-time dense surface mapping and tracking. In *Proceedings of the International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136.

[111] Nickolls, J. and Dally, W. (2010). The GPU computing era. *IEEE Micro*, 30(2):56–69.

[112] Niu, F., Recht, B., Ré, C., and Wright, S. J. (2011). Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. Technical report, Computer Sciences Department, University of Wisconsin-Madison.

[113] Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer-Verlag, second edition.

[114] OpenNI (2013). The standard framework for 3D sensing. online. http://www.openni.org/.

[115] PhiSix fashion labs (2013). Virtual.fitting.room. online. http://www.phisix.co/.

[116] Poppe, R. (2007). Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding (CVIU)*, 108(1–2):4–18.

[117] Ratanamahatana, C. A. and Keogh, E. (2004). Everything you know about dynamic time warping is wrong. In *Workshop on Mining Temporal and Sequential Data*, pages 1–11.

[118] Richter, M., Varanasi, K., Hasler, N., and Theobalt, C. (2012). Real-time reshaping of humans. In *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DimPVT)*, pages 340–347, Zürich, CH.

[119] Rodriguez, A., Ehlenberger, D., Hof, P., and Wearne, S. L. (2009). Three-dimensional neuron tracing by voxel scooping. *Journal of Neuroscience Methods*, 184(1):169–175.

[120] Sagawa, Y., Shimosaka, M., Mori, T., and Sato, T. (2007). Fast online human pose estimation via 3D voxel data. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 1034–1040.

[121] Salzmann, M. and Urtasun, R. (2010). Combining discriminative and generative methods for 3D deformable surface and articulated pose reconstruction. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 647–654.

[122] Scharstein, D. and Szeliski, R. (2002). A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision (IJCV)*, 47(1):7–42.

[123] Segal, A. V., Hähnel, D., and Thrun, S. (2009). Generalized ICP. In *Proceedings of Robotics: Science and Systems*, pages 1–8.

[124] Seitz, S. M., Curless, B., Diebel, J., Scharstein, D., and Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 519–528.

[125] Seitz, S. M. and Dyer, C. R. (1999). Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision (IJCV)*, 35(2):151–173.

[126] Shi, L., Yu, Y., Bell, N., and wen Feng, W. (2006). A fast multigrid algorithm for mesh deformation. *ACM Transactions on Graphics (TOG)*, 25(3):1108–1117.

[127] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1297–1304.

[128] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

[129] Sidenbladh, H., Black, M. J., and Fleet, D. J. (2000). Stochastic tracking of 3D human figures using 2D image motion. In Vernon, D., editor, *Proceedings of European Conference on Computer Vision (ECCV)*, volume 1843 of *Lecture Notes in Computer Science (LNCS)*, pages 702–718. Springer-Verlag.

[130] Sigal, L., Balan, A., and Black, M. J. (2010). Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision (IJCV)*, 87(1–2):4–27.

[131] Sigal, L., Isard, M., Haussecker, H., and Black, M. J. (2012). Loose-limbed people: Estimating 3D human pose and motion using non-parametric belief propagation. *International Journal of Computer Vision (IJCV)*, 98(1):15–48.

[132] Simmons, K. P. and Istook, C. L. (2003). Body measurement techniques: Comparing 3D body-scanning and anthropometric methods for apparel applications. *Journal of Fashion Marketing and Management*, 7(3):306–332.

[133] Sonka, M., Hlavac, V., and Boyle, R. (2008). *Image Processing, Analysis, and Machine Vision*. Thomson, 3rd edition.

[134] Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., and Seidel, H.-P. (2004). Laplacian surface editing. In *Proceedings of Eurographics*, pages 175–184, New York, USA. ACM.

[135] Starck, J. and Hilton, A. (2003). Model-based multiple view reconstruction of people. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 915–922.

[136] Starck, J. and Hilton, A. (2007). Surface capture for performance-based animation. *IEEE Computer Graphics and Applications (CG&A)*, 27(3):21–31.

[137] Steed, A. (2008). A simple method for estimating the latency of interactive, real-time graphics simulations. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 123–129. ACM.

[138] Stoll, C., Gall, J., Aguiar, E. d., Thrun, S., and Theobalt, C. (2010). Video-based reconstruction of animatable human characters. *ACM Transactions on Graphics (TOG)*, 29(6):139.

[139] Stoll, C., Hasler, N., Gall, J., Seidel, H.-P., and Theobalt, C. (2011). Fast articulated motion tracking using a sums of gaussians body model. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 951–958.

[140] Straka, M., Hauswiesner, S., Rüther, M., and Bischof, H. (2011a). A free-viewpoint virtual mirror with marker-less user interaction. In *Proceedings of Scandinavian Conference on Image Analysis (SCIA)*, volume 6688 of *Lecture Notes in Computer Science (LNCS)*, pages 635–645, Ystad, SE. Springer-Verlag.

[141] Straka, M., Hauswiesner, S., Rüther, M., and Bischof, H. (2011b). Skeletal graph based human pose estimation in real-time. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 69.1–69.12, Dundee, GB.

[142] Straka, M., Hauswiesner, S., Rüther, M., and Bischof, H. (2012a). Rapid skin: Estimating the 3D human pose and shape in real-time. In *Proceedings of the International Conference on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DimPVT)*, pages 41–48, Zürich, CH.

[143] Straka, M., Hauswiesner, S., Rüther, M., and Bischof, H. (2012b). Simultaneous shape and pose adaption of articulated models using linear optimization. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume 7572 of *Lecture Notes in Computer Science (LNCS)*, pages 724–737, Florence, IT. Springer-Verlag.

[144] Sun, M., Kohli, P., and Shotton, J. (2012). Conditional regression forests for human pose estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 3394–3401.

[145] Sundaresan, A. and Chellappa, R. (2008). Model driven segmentation of articulating humans in laplacian eigenspace. *Pattern Analysis and Machine Intelligence (PAMI)*, 30(10):1771–1785.

[146] Sutter, H. (2005). The free lunch is over: A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3):202–210.

[147] Taylor, J., Shotton, J., Sharp, T., and Fitzgibbon, A. (2012). The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 103–110.

[148] $(TC)^2$ (2013). online. http://www.tc2.com/.

[149] Thome, N., Merad, D., and Miguet, S. (2006). Human body part labeling and tracking using graph matching theory. In *Proceedings of the IEEE Conference on Video and Signal Based Surveillance*, page 38.

[150] Tilley, A. R. and Dreyfuss, H. (2002). *The Measure of Man & Woman*. John Wiley & Sons, revised edition.

[151] Tippetts, B., Lee, D. J., Lillywhite, K., and Archibald, J. (2013). Review of stereo vision algorithms and their suitability for resource-limited systems. *Journal of Real-Time Image Processing*, pages 1–21.

[152] Tognetti, A., Lorussi, F., Bartalesi, R., Tesconi, M., Zupone, G., and De-Rossi, D. (2005). Analysis and synthesis of human movements: wearable kinesthetic interfaces. In *Proceedings of the 9th International Conference on Rehabilitation Robotics (ICORR)*, pages 488–491.

[153] TOMS Shoes (2013). Virtual try-on. online. http://www.toms.com/tryon/l.

[154] Tran, C. and Trivedi, M. M. (2008). Human body modeling and tracking using volumetric representation: Selected recent studies and possibilities for extensions. In *Proceedings of the International Conference on Distributed Smart Cameras*, pages 1–9.

[155] UPcload (2013). Always get the right size. online. http://www.upcload.com/.

[156] Van den Bergh, M., Koller-Meier, E., and Van Gool, L. (2009). Real-time body pose recognition using 2D or 3D haarlets. *International Journal of Computer Vision (IJCV)*, 83:72–84.

[157] Vicon (2013). Motion capture systems. online. http://www.vicon.com/.

[158] Vlasic, D., Baran, I., Matusik, W., and Popović, J. (2008). Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics (TOG)*, 27(3):1–9.

[159] Vlasic, D., Peers, P., Baran, I., Debevec, P., Popović, J., Rusinkiewicz, S., and Matusik, W. (2009). Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics (TOG)*, 28(5):174:1–174:11.

[160] Wagner, D. and Schmalstieg, D. (2007). ARToolKitPlus for pose tracking on mobile devices. In Grabner, M. and Grabner, H., editors, *Proceedings of the Computer Vision Winter Workshop (CVWW)*, pages 139–146.

[161] Wang, Y.-S. and Lee, T.-Y. (2008). Curve-skeleton extraction using iterative least squares optimization. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):926–936.

[162] Weingartner, M. (2011). Better online shopping: Virtually walk the catwalk dressed in the latest fashion. online. http://newsroom.intel.com/docs/DOC-2154.

[163] Weiss, A., Hirshberg, D., and Black, M. J. (2011). Home 3D body scans from noisy image and range data. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1951–1958, Barcelona, SP.

[164] Wu, C., Varanasi, K., Liu, Y., Seidel, H.-P., and Theobalt, C. (2011). Shading-based dynamic shape refinement from multi-view video under general illumination. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1108–1115. IEEE.

[165] Wu, C., Varanasi, K., and Theobalt, C. (2012). Full body performance capture under uncontrolled and varying illumination: A shading-based approach. In *Proceedings of European Conference on Computer Vision (ECCV)*, volume 7575 of *Lecture Notes in Computer Science (LNCS)*, pages 757–770, Florence, IT. Springer-Verlag.

[166] Xu, H., Yu, Y., Zhou, Y., Li, Y., and Du, S. (2013). Measuring accurate body parameters of dressed humans with large-scale motion using a kinect sensor. *Sensors*, 13:11362–11384.

[167] Yuan, M., Farbiz, F., Khan, I., Yao, S., Niswar, A., and Foo, M. (2013). A mixed reality virtual clothes try-on system. *IEEE Transactions on Multimedia*, PP(99):1–24.

[168] Zach, C., Pock, T., and Bischof, H. (2007). A duality based approach for realtime TV-L^1 optical flow. In *Proceedings of the DAGM Symposium on Pattern Recognition*, pages 214–223. Springer-Verlag.

[169] Zhang, S., Huang, J., and Metaxas, D. N. (2011). Robust mesh editing using laplacian coordinates. *Graphical Models*, 73(1):10–19.

[170] Zhou, S., Fu, H., Liu, L., Cohen-Or, D., and Han, X. (2010). Parametric reshaping of human bodies in images. *ACM Transactions on Graphics (TOG)*, 29(4):126:1–126:10.