Template based shape processing

Carsten Stoll

Max-Planck Institut Informatik

Dissertation
zur Erlangung des Grades des
Doktors der Ingenieurswissenschaften (Dr.-Ing.)
der Naturwissenschaftlich-Technischen Fakultäten
der Universität des Saarlandes

November 2009

Date of Colloquium:

30. September 2009

Dean:

Prof. Dr. Joachim Weickert
Faculty of Mathematics and Computer Science
Saarland University
Saarbrücken, Germany

Examination Board:

Prof. Dr. Philipp Slusallek (Chair)
Saarland University

Prof. Dr. Hans-Peter Seidel
Max-Planck Institut Informatik

Dr. Christian Theobalt
Max-Planck Institut Informatik

Prof Dr. Stefan Gumhold
Technische Universität Dreseden

Dr. Thorsten Thormählen
Max-Planck Institut Informatik

Abstract

As computers can only represent and process discrete data, information gathered from the real world always has to be sampled. While it is nowadays possible to sample many signals accurately and thus generate high-quality reconstructions (for example of images and audio data), accurately and densely sampling 3D geometry is still a challenge. The signal samples may be corrupted by noise and outliers, and contain large holes due to occlusions. These issues become even more pronounced when also considering the temporal domain. Because of this, developing methods for accurate reconstruction of shapes from a sparse set of discrete data is an important aspect of the computer graphics processing pipeline.

In this thesis we propose novel approaches to including semantic knowledge into reconstruction processes using *template based shape processing*. We formulate shape reconstruction as a deformable *template fitting* process, where we try to fit a given template model to the sampled data. This approach allows us to present novel solutions to several fundamental problems in the area of shape reconstruction. We address *static* problems like constrained texture mapping and semantically meaningful hole-filling in surface reconstruction from 3D scans, *temporal* problems such as mesh based performance capture, and finally *dynamic* problems like the estimation of physically based material parameters of animated templates.

Kurzfassung

Analoge Signale müssen digitalisiert werden um sie auf modernen Computern speichern und verarbeiten zu können. Für viele Signale, wie zum Beispiel Bilder oder Tondaten, existieren heutzutage effektive und effiziente Digitalisierungstechniken. Aus den so gewonnenen Daten können die ursprünglichen Signale hinreichend akkurat wiederhergestellt werden. Im Gegensatz dazu stellt das präzise und effiziente Digitalisieren und Rekonstruieren von 3D- oder gar 4D-Geometrie immer noch eine Herausforderung dar. So führen Verdeckungen und Fehler während der Digitalisierung zu Löchern und verrauschten Meßdaten. Die Erforschung von akkuraten Rekonstruktionsmethoden für diese groben digitalen Daten ist daher ein entscheidender Schritt in der Entwicklung moderner Verarbeitungsmethoden in der Computergrafik.

In dieser Dissertation wird veranschaulicht, wie deformierbare geometrische Modelle als Vorlage genutzt werden können, um semantische Informationen in die robuste Rekonstruktion von 3D- und 4D-Geometrie einfließen zu lassen. Dadurch wird es möglich, neue Lösungsansätze für mehrere grundlegenden Probleme der Computergrafik zu entwickeln. So können mit dieser Technik Löcher in digitalisierten 3D Modellen semantisch sinnvoll aufgefüllt, oder detailgetreue virtuelle Kopien von Darstellern und ihrer dynamischen Kleidung zu erzeugt werden.

Acknowledgements

First, I would like to thank my supervisor Prof. Dr. Hans-Peter Seidel who made it possible for me to work and do my research at the inspiring environment of the MPI Informatik, and Prof. Dr. Marc Alexa, who ignited my interest in Computer Graphics in the first place. I would also like to thank all the senior researchers here at the institute who supervised me over the past years: Prof. Dr. Stefan Gumhold, for guiding me through the beginning of my PhD; Dr. Zachi Karni for motivating me to do research on Geometric Modeling; And finally, Dr. Christian Theobalt for sparking my interest in motion capture and all his support in the last few years. I am indebted to all of them for their advice and guidance in my research.

Without the cooperation, advice and discussions of all my former and present colleagues at the MPI many of my research projects would not have been possible. I am especially grateful to all the co-authors of the papers I have worked on during my PhD : Christian Rössl, Hitoshi Yamauchi, Edislon de Aguiar, Naveed Ahmed, Nils Hasler, Martin Sunkel, Bodo Rosenhahn, Thorsten Thormählen and Jürgen Gall. Further, I would like to thank Boris Ajdin, Martin Fuchs and Oliver Schall for the lively discussions, and Conny Liegl and Sabine Budde for their help in managing day to day office life.

I would also like to thank our Maria Jacob, Yvonne Flory, Samir Hammann, Lukas Ahrenberg and Tatjana Feldmann for allowing us to record their performances and use them for research projects, and Derek D. Chan for his help in dubbing our videos.

Finally, I would like to thank my family for their support, and Natascha for just being there for me all the time.

Contents

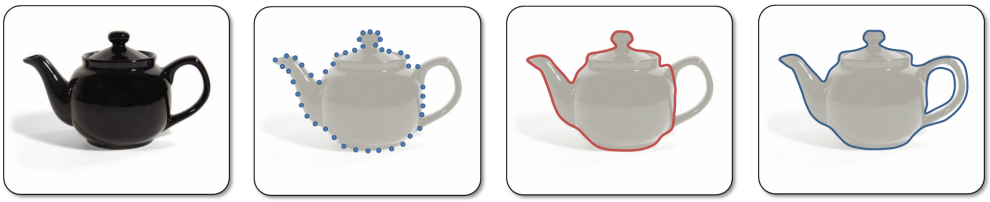
Chapter 1

Introduction

It has been an ongoing effort in the last few decades to enable computers to analyze, understand and recreate our world (and also create and simulate new and imagined worlds). Humans use their senses (for example sight, hearing and touch) to perceive the world around them, while it is necessary to build special devices that emulate these senses by digitizing the physical properties for the computer. The field of *Computer Graphics* is mainly concerned with the sense of sight, i.e. recording, processing and reproducing of what humans can see. Traditionally, *Computer Graphics* was mainly viewed as the process of digitally synthesizing and manipulating visual content. However, nowadays the field is covering the whole pipeline of *input* → *processing* → *output*.

As computers can only represent and process discrete data, *input* information gathered from the real world (for example images or audio signals) always have to be sampled. Whether the sampling is dense enough to reconstruct the original signal depends on the original frequency content and is determined by the Nyquist-Shannon sampling theorem (Shannon (1949)). Due to the improvements in sensor equipment we are nowadays able to sample many signals accurately and are able to generate reasonable reconstructions (especially images and audio data). However, in some areas it is still difficult to achieve a dense and regular sampling of the data, especially concerning 3D data. If our signal is undersampled or even misses data, the only way to recover the original signal is to apply additional knowledge about the original input signal to the reconstruction process.

In this thesis we will present new methods and applications to introducing additional knowledge into reconstruction processes by using a template fitting process.



(a) (b) (c) (d)

Figure 1.1: Illustrating *shape reconstruction* : (a) Real world object. (b) A sparse set of samples derived from the object. Note that the handle of the teapot has not been sampled. (c) If we do not have any further information about the object, the most we can reconstruct from the samples is a handle-less teapot. (d) If we provide additional knowledge (for example in the form of a template), we can reconstruct a teapot with a handle. While the handle does not accurately represent the real handle, the created object is much closer to the real world object than (c).

1.1 Overview

This thesis deals with *template based shape processing*. The definition of *template* according to Wiktionary (2009) is the following :

template (plural templates) :

1. *A physical object whose shape is used as a guide to make other objects*
2. *A generic model or pattern from which other objects are based or derived*

While the first explanation specifically refers to a *physical object*, we reinterpret this in our setting to be a virtual, geometric object represented digitally in a computer. The second definition accurately describes the processing we are going to perform with these virtual objects, namely using them as a basis to derive the shape of other objects from. In real world applications we would use a template to create near identical copies of it. In the context of *template based shape processing*, however, a template will be used to provide the algorithm with additional semantic information about the general object class.

One key example of this is *shape reconstruction*. The goal is to create a virtual shape that best corresponds to a set of sparse input samples taken from a real or virtual object. These sparse samples can be acquired through different methods,

and depending on how they were found they may exhibit different properties. When using a 3D scanner we receive information in the form of the position of a set of unstructured points in 3D space. Analyzing photos yields the position of points in the 2D image plane. There exist methods for extracting time varying information, for example from video or real-time 3D scanners. Finally, it is also possible to incorporate user-defined constraints into this sample set (for example by defining correspondences between the input and template).

These sampling methods often have limitations : the sample set may be noisy, contain outliers and holes, or may be so sparse that it is near impossible to draw conclusions about the original shape. Due to this, it may be difficult to recreate the object accurately using only the given information.

Figure 1.1 (a) shows an image of a real teapot. We use a sampling technique to acquire a sparse set of samples representing the original object (Figure 1.1 (b)). However, due to occlusions during this process we are not able to create any samples on the teapot handle. If we now try to reconstruct the shape of the object using only the sampled data, we will not be able to recreate a full virtual representation of the original object, but at most a teapot missing the handle (Figure 1.1 (c)). This is because our reconstruction method does not have information about the kind of object it is trying to reconstruct, and as such chooses the most conservative strategy. However, if we provide our algorithm with additional knowledge about the general shape (i.e. telling it that the object we want to reconstruct is a teapot, which usually has a handle), we will be able to reconstruct the teapot correctly (Figure 1.1 (d)). As we do not have any information about the missing parts of the original object, the only way we can apply this knowledge is by copying from the template.

As can be seen in this illustration, our use of the template here is different from its purpose in the real world. Usually, a template would be used as basis for generating near exact copies of that object. In our example above however we take the information given by the template and use it to complete a different shape. Using templates for semantically meaningful completion of objects is not a new idea. There exist methods that use a conservative shape reconstruction technique based solely on the input samples wherever they are dense enough. Holes are then filled by copying the information from the template (Sharf *et al.* (2004)). Other methods automatically find the best fitting templates to parts of the object from a large database and stitch them together (Pauly *et al.* (2005)).

1.2 Contributions and structure

In this thesis we propose an approach to template based shape processing that allows us to solve a large number of different tasks using the same underlying idea. We address *static* problems like constrained texture mapping and semantically meaningful hole-filling in surface reconstruction (part I), *temporal* problems such as mesh based performance capture (part I and II), and finally *dynamic* problems like the estimation of physically based material parameters of templates (part III). Instead of using the template to only fill holes, we approach all these problems as a *template fitting* process. Here, we want to change the template in such a way that it fits as accurately as possible to the data samples we have. By approaching the problem in this way, we predefine the topology and connectivity of our object using the template, thereby stabilizing the reconstruction process. Another benefit of this process is that it allows us to guarantee consistency when applying the same template to different reconstructions. The most important factor for the quality of our results is the choice of fitting process. This will be heavily influenced by our input data, the template structure and the desired output.

The rest of this thesis is structured as follows :

Chapter 2 introduces the fundamental data structures and techniques used in this thesis and also gives an overview of related work. In the following chapters we propose three frameworks which combine into one large system for *template based shape processing* and show the uses in different applications. The first two parts will deal with *shape reconstruction* on different scales. The method from part I is particularly suited for modifying the detailed shape of a template (i.e. reconstructing high-frequency information). The framework presented in part II is more suited for low-frequency processing, i.e. changing the fundamental pose of the object while conserving the detail of the template. Combing the frameworks from part I and II allows us to accurately capture detailed shapes undergoing strong deformations. Finally, the framework presented in part III allows for the recovery of physical material properties of the template from animated shape reconstructions.

Part I presents a framework for template processing using triangle meshes based on linear differential coordinates. This framework is particularly suited for *shape reconstruction* given dense samples of the target object. The differential coordinate based deformation allows for a high range of deformations. The

method can be adjusted to allow for very high frequency changes in the structure of the template. One drawback caused by the large freedom of deformation possibilities is that it can be difficult to control the low-frequency deformations of the object if only sparse samples are given. However, if the given set of samples is dense enough we can reconstruct detailed shapes accurately.

Chapter 3 gives a detailed introduction to the linear differential coordinates we use and the shape deformation framework based on them. We also contribute a novel method of dealing with the translational insensitivity this approach is facing (Stoll *et al.* (2006a)).

In Chapter 4 we apply our template deformation framework to introduce a novel approach to constrained texture mapping called *inverse texture mapping* (Stoll *et al.* (2006b)). The traditional approaches to constrained texture mapping unfold a 3D model onto the plane, generating a 2D parameterization of the mesh. The parameterization can then be used to apply the image onto the surface using texture mapping techniques. Instead of this, we propose to regard the texture as a planar template that is wrapped around the target shape under a number of user specified constraints. This approach has several advantages. It is applicable to any type of manifold surface, in particular high-genus surfaces, which needed cutting for traditional texture mapping. It also generates a smoother and more high resolution reproduction of the original shape. This is especially useful for draping apparel onto virtual characters without having to resort to complex cloth simulation algorithms.

Chapter 5 introduces a new approach to surface reconstruction from 3D point clouds based on a template shape (Stoll *et al.* (2006a)). While surface reconstruction methods that are not based on templates will always reconstruct correct surfaces where the sampling density of the input point cloud is high enough, they fail to produce semantically correct reconstructions if parts of the object have not been properly scanned (see Figure 1.1 for an example). Previous work incorporates additional semantic knowledge into surface reconstruction by copying geometry, or by establishing complex cross-parameterizations between a template and scanned geometry (which requires an initial surface reconstruction of the incomplete mesh). Instead, we extend the insights gained in Chapter 4, and formulate surface reconstruction as a template fitting process. This allows us to reconstruct the surface and fill holes in an intuitive and simple manner, resulting in semantically meaningful reconstructions.

Finally, in Chapter 6 we show how template based shape processing can be used for performance capture applications. This extends the system developed

in the previous Chapter to the temporal domain. Traditional marker-less motion capture methods use a kinematic skeleton as underlying shape parameterization, analogous to marker based systems. While this is a very compact representation, it is a very coarse approximation of deformations of the human body, especially if the performer is wearing wide apparel. Instead, we propose replacing the kinematic skeleton by our template fitting framework. This is a more flexible representation that enables us to pose motion capture in terms of a mesh deformed by tracked feature points in 2D images. This allows us to capture the pose of characters in complex apparel (de Aguiar *et al.* (2007a,b,c)), as well as refine surface detail and non-rigid motions of apparel from input videos (de Aguiar *et al.* (2008a); Gall *et al.* (2009)). This level of detail could not be achieved using traditional skeleton-based methods.

Part II presents a novel iterative framework for template based shape processing using volumetric shapes (de Aguiar *et al.* (2008a); Stoll *et al.* (2007)). We extend the linear differential coordinate framework presented in Part I to tetrahedral meshes and introduce an efficient iterative update process. This combinations allows us to efficiently generate stable non-linear deformations that are particularly suited to create low-frequency deformations, i.e. deformations which influence the global pose of the object. This property makes it particularly useful for animation and performance capture purposes, as the approach is very stable and reliable even under a small number of constraints.

Chapter 7 gives a detailed introduction to our non-linear iterative framework. Unlike many other non-linear deformation frameworks, it is conceptually simple and very efficient, allowing for a real-time feedback of the results.

In Chapter 8 we introduce a framework for interactive shape editing based on our deformation method (Stoll *et al.* (2007)). Given a template model and a set of user specified constraints we want to deform our shape to match the constraints as naturally as possible. By running the iterative deformation simultaneously to the user interface and rendering processes the user gains real-time feedback about the deformation. This leads to an intuitive interaction with the model.

Chapter 9 shows how we can apply our non-linear template fitting framework in the context of animation and performance capture. By using a tetrahedral model of the performer as template for the capture process, we can naturally animate virtual characters given a sparse set of constraint points derived from a marker-based motion capture system (Stoll *et al.* (2007)). This allows us to

circumvent embedding a kinematic skeleton into the model, which can be a time-consuming and expensive process. We can apply the motion of the performer to its virtual counterpart directly by using the captured marker positions as deformation constraints and thus produce realistic animations with a less complex system. The stable and locally shape-preserving deformation process also allows us to improve on the mesh-based performance capture methods based on the framework presented in Part I (de Aguiar *et al.* (2008a)). By carefully combining a sophisticated image feature extraction and tracking method with a global optimization scheme, we are able to accurately capture the pose of characters in wide apparel reliably, even under complex and fast motions. We are also able to capture higher geometric detail than with the purely surface-based approaches presented in Part I.

Part III extends the idea of template fitting to *dynamic* data. We presents a method for estimating physical properties using template based shape processing (Stoll *et al.* (2009)). While the previous two parts concentrated on the (temporal) shape of objects, we are now able to estimate the physical material properties of the input object by analyzing temporal data. This enables us to approximate the behavior of the object under previously unobserved external influences. The specific application we are addressing in this part is optical reconstruction of detailed animatable human body models.

In Chapter 10 we use multi-view video of an actor and the performance capture approaches from the previous parts of the thesis to capture reconstruct a detailed animatable human body template (Stoll *et al.* (2009)). We segment a template of the performer into cloth and non-cloth regions, and estimate cloth material parameters. Unlike the performance capture approaches presented in Parts I and II of this thesis, we can now not only play back the recorded performance, but modify it and *simulate* the behavior of the characters apparel under the new impulses accurately. We can create new animations in real-time, with new body motion and cloth deformation that look as realistic as the reference sequence in high quality. This rich performance template is reconstructed from just a 3D scan and a set of multi-view input videos, offering artist and animators a new level of data to work with.

1.3 List of publications

The work presented in this thesis has been published in the following papers:

Geodesics Guided Constrained Texture Deformation

Carsten Stoll, Zachi Karni and Hans-Peter Seidel

Pacific Graphics (2006)

Stoll *et al.* (2006b)

Template Deformation for Point Cloud Fitting

Carsten Stoll, Zachi Karni, Christian Rössl, Hitoshi Yamauchi and Hans-Peter Seidel

IEEE/EG Symposium on Point-Based Graphics (2006)

Stoll *et al.* (2006a)

Rapid Animation of Laser-scanned Humans

Edilson de Aguiar, Christian Theobalt, Carsten Stoll and Hans-Peter Seidel

IEEE Conference on Virtual Reality (2007)

de Aguiar *et al.* (2007c)

Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture

Edilson de Aguiar, Christian Theobalt, Carsten Stoll and Hans-Peter Seidel

IEEE Conference on Computer Vision and Pattern Recognition (2007)

de Aguiar *et al.* (2007b)

Marker-less 3D Feature Tracking for Mesh-based Motion Capture

Edilson de Aguiar, Christian Theobalt, Carsten Stoll and Hans-Peter Seidel

ICCV Workshop on Human Motion - Understanding, Modeling, Capture and Animation (2007)

de Aguiar *et al.* (2007a)

A Volumetric Approach to Interactive Shape Editing

Carsten Stoll, Edilson de Aguiar, Christian Theobalt and Hans-Peter Seidel

MPII Research Report (2007)

Stoll *et al.* (2007)

Performance Capture from Sparse Multi-view Video

Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel and Sebastian Thrun

ACM Transactions on Graphics Special Issue SIGGRAPH (2008)

de Aguiar *et al.* (2008a)

Motion Capture Using Joint Skeleton Tracking and Surface Estimation

Jürgen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn and Hans-Peter Seidel

IEEE Conference on Computer Vision and Pattern Recognition

Gall *et al.* (2009)

Optical Reconstruction of Detailed Animatable Human Body Models

Carsten Stoll, Jürgen Gall, Edilson de Aguiar, Hans-Peter Seidel, Sebastian Thrun and Christian Theobalt

MPII Research Report (2009)

Stoll *et al.* (2009)

Some of these papers have also been presented in the PhD theses of de Aguiar (2008) and Gall (2009). This work focuses mainly on the mathematical frameworks for template processing and deformation, as well as the algorithms to segment the captured animations, and estimate their physical parameters.

1.3 List of publications

Chapter 2

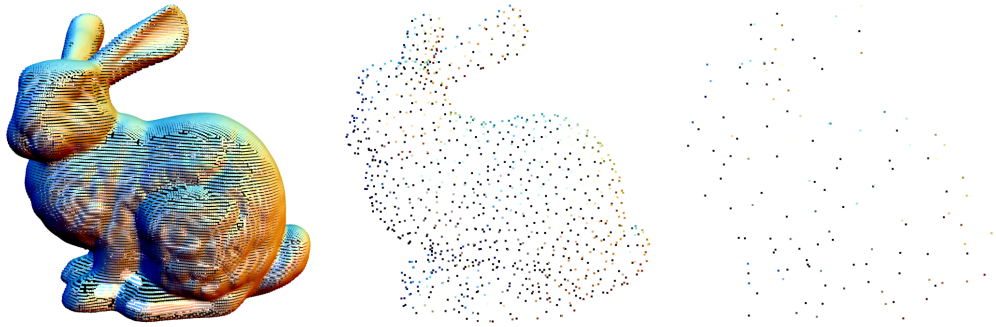
Fundamentals

In this chapter we will introduce some fundamental concepts and notations that the following work is based on as well as related work in the field. We will first introduce the basic 3D and 2D object representations. Here we concentrate on the formats which are relevant in the following chapters. Following this, we will show how to generate various virtual representations by scanning real objects and reconstructing their surfaces. The next section will give an overview over the field of shape editing, followed by a short introduction on physical simulation focused on cloth simulation techniques. Finally we will conclude this chapter with a section on performance capture.

2.1 Basic data structures

2.1.1 3D objects and their representations

To be able to efficiently work with geometric objects it is necessary to have a suitable and efficient data structure. The choice of data structure is not only influenced by the way computers work (being limited in processing power and memory as well as only working on a discrete digital domain), but also by our ability to mathematically describe the geometry correctly. The field of computer graphics is mostly interested in the *appearance* of objects. Because of this representations of objects are often limited to *surfaces*, i.e. disregarding the internal structure of objects and focusing on shape and texture. This is sufficient when we are dealing only with surface interactions, such as traditional rendering. Depending on the application it may become necessary to consider internal volumetric



(a) (b) (c)

Figure 2.1: Stanford Bunny model as a point cloud in different resolutions. (a) 35947 points (b) 1013 points (c) 130 points.

information as well (for example when simulating solid objects, or when visualizing non-homogeneous semi-transparent materials). The representations we will deal with in the following chapters of this thesis are all discretely sampled objects, i.e. they are defined by a finite number of values, and thus belong to the class of piecewise parametric representations. Samples usually consist of a 3D position and may include further information, for example a local normal vector representing the tangent plane of the surface, color or texture coordinates.

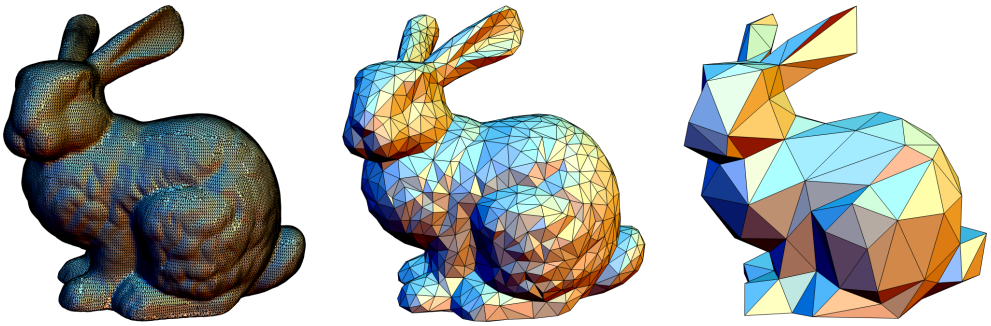
2.1.1.1 Point clouds

The most basic representation we will deal with is the so called *point cloud*. A point cloud is defined as a collection of n sample points :

$$\mathcal{P} = \{\mathbf{p}_1 \dots \mathbf{p}_n\}, \mathbf{p}_i \in \Re^3 \tag{2.1}$$

Point clouds are an *unstructured* representation of an object, meaning that there is no information on how points are related to each other (i.e. if they are in close proximity to each other on the object surface). If there is a sufficient number of sample points n, a point-cloud can be a very accurate representation of the surface. With increasing sparsity of sample points in $\mathcal{P}$ it becomes increasingly difficult to represent an object correctly (see Figure 2.1).

Additional properties enhancing the representation of a point cloud are usually a normal vector $\mathbf{n}_i \in \Re^3$ and/or a color value $\mathbf{c}_i \in \Re^3$ for each sample point $\mathbf{p}_i$.



(a) (b) (c)

Figure 2.2: Stanford Bunny model as a triangle mesh in different resolutions. (a) 35947 vertices / 69451 triangles (b) 1013 vertices / 1999 triangles (c) 130 vertices / 249 triangles.

2.1.1.2 Triangle meshes

Probably the most common 3D object representation in the field of computer graphics is the triangle mesh. A triangle mesh $\mathcal{M}$ is defined as a collection of n sample points (also called vertices) $\mathcal{V}$ and m triangular faces $\mathcal{F}$ connecting them. The vertices of a triangle mesh are defined in the same way as the sample points of a point cloud :

$$\mathcal{V} = \{\mathbf{v}_1 \dots \mathbf{v}_n\}, \mathbf{v}_i \in \Re^3 \tag{2.2}$$

The set of triangular faces is a set of 3Dindex vectors to the set of vertices $\mathcal{V}$:

$$\mathcal{F} = \{\mathbf{f}_1 \dots \mathbf{f}_m\}, \mathbf{f}_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} \tag{2.3}$$

A triangle mesh is a piecewise linear approximation of the object surface. While it is also possible to use more general face sets than triangles (i.e. allowing each face to be a polygon with arbitrary degree), there are several advantages of using triangular meshes over these more general polygonal meshes. Three vertices always from a planar surface, which makes interpolating data and processing of the surface a straightforward task, as the interior of a face $\mathbf{f}_i$ can be interpolated easily. Due to this, the triangle is the preferred rendering primitive of modern graphics hardware.

A triangle mesh is a *structured* representation of a surface, defining neighborhoods on a surface and thus is a richer representation of the surface than a point

cloud. This allows for an efficient calculation of intrinsic properties of the surface such as curvature. Unlike a point cloud, a triangle mesh can represent an object even at very low resolutions in a quality that it is easily recognizable by a human observer (see Figure 2.2).

Similar to a point cloud it is possible to store additional properties for each sample point, including normal vector $\mathbf{n}_i \in \Re^3$ and a color value $\mathbf{c}_i \in \Re^3$. The normal vector can be calculated by taking a weighted average of incident face normal vectors.

2.1.1.3 Tetrahedral meshes

In some cases it is not enough to represent only the surface of an object in the computer. Most real world objects are solid and not just a simple shell, which needs to be taken into account for example when simulating deformation under forces. One of the most simple ways to model solid objects is to use tetrahedral elements to discretize the volume. A tetrahedral mesh $\mathcal{T}$ is defined by a set of n vertices $\mathcal{V}$ and a list of m tetrahedrons $\mathcal{E}$ which connect the sample points. The set of tetrahedral elements is a set of 4D index vectors to the set of vertices $\mathcal{V}$:

$$\mathcal{V} = \{\mathbf{v}_1 \dots \mathbf{v}_n\}, \mathbf{v}_i \in \Re^3 \tag{2.4}$$

$$\mathcal{E} = \{\mathbf{e}_1 \dots \mathbf{e}_m\}, \mathbf{e}_i \in \mathcal{V} \times \mathcal{V} \times \mathcal{V} \times \mathcal{V} \tag{2.5}$$

Like a triangle mesh, a tetrahedral mesh is a *structured* representation of a bounded volume. The surface of the object is defined by the set of triangular faces of the tetrahedra which do not coincide with any other tetrahedra.

2.1.2 Images and videos

Just like 3D objects explained above, it is possible to represent 2D images in digital form in the computer. The two most common approaches are vector-based and pixel-based representations.

Vector images have a strong relation to the representations we have introduced for 3D shapes before. We use a set of geometric shape primitives such as lines, curves and polygons, which are all based on mathematical equations, to represent a scene. This representation is very compact and easy to modify. Vector images can be scaled freely without loss of image quality. However, one of the main drawbacks of vector images is that it is complicated to represent small detail and

texture as vector graphics, as this requires a large amount of primitives. Due to this it is also very difficult to convert real images into vector images.

Pixel images (also called raster image) on the other hand consist of a regular 2D rectangular grid of pixels, where each pixel has a color value. While this means that we cannot zoom into the image arbitrarily (the block structure of the grid will become visible), the representation allows for representation of pixel-sized detail and easy digitization of real images.

In the remainder of this thesis we will only deal with raster images recorded using cameras. We represent an image $\mathcal{I}$ as a 2D array of size $x \times y$. Each array element consists of a 3D color pixel $\mathcal{I}(x, y) \in \Re^3$ representing the intensities in red, green and blue channel respectively. A video $\mathcal{F}$ is a temporal sequence of N images $\mathcal{F}_n$ with constant size $x \times y$ and a constant frame rate f that determines the time that passed between recording each consecutive frame.

2.2 Scanning and surface reconstruction

In recent years we have observed increasing interest in spatial scanning devices and the development of new and improved technologies that enable real-time capturing of real-world objects. This is mostly due to the fact that geometry acquisition and reconstruction are essential to many fields of application: In the life-cycle of industrial product design, prototypes are digitized to serve as feedback to the designer; scanners are used along manufacture lines for quality and process control. In medicine, the shape of internal organs is captured to detect malfunctions and diseases using minimally invasive methods. In security and authentication, spatial scanning introduces an additional dimension upon the traditional image based methods. However, the most evident use of shape digitization is in the entertainment industry, where digital models in games produce realistic scenes and motion, and the movies show realistic special effects.

Independent of scanning technology and application domain, most geometry acquisition results in *unstructured* point cloud data, where each point provides a sample of the acquired object, are typically afflicted with measurement error. Precision of measurement depends on many factors, such as acquisition device and technology, environmental conditions, complexity of the scanned object and many more.

The process of transferring an unstructured point cloud model into a consistent discrete surface model such as a polygonal mesh is commonly referred to as

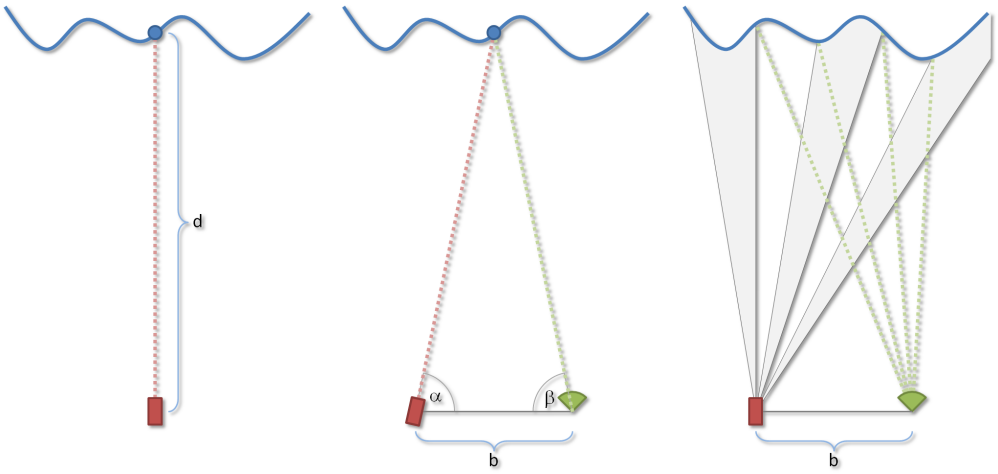
surface reconstruction. Here, the main task consists of the generation of a manifold mesh that approximates the input data, i.e., that captures its global shape and topology together with its fine geometry details (see Kazhdan *et al.* (2006); Kazhdan (2005) for an overview of the field).

We generally have two options to generate 3D templates for use in shape processing. The first is to model the objects by hand using modeling software. This is straightforward for simple shapes like planes or spheres, but quickly becomes more involved for complex objects. Modeling a human body in reasonable accuracy can take a professional artist several days. The process becomes even more complicated when a real object is to be reproduced as a model (for example an existing building). The second possibility is to use 3D scanning to construct a virtual model of a real object.

2.2.1 3D scanning

3D scanning techniques can be roughly separated into two categories. *Active* scanners measure the shape of an object by projecting light into the scene and measuring it, while *passive* scanners estimate 3D information without external interference.

Active techniques include time-of-flight scanners, where a laser pulse is projected into the scene and the time the light needs to return to the sensor is measured, which allows us to calculate the distance of the object the laser hit (Figure 2.3 (a)). Due to the necessary accuracy of time measurement, the first time-of-flight scanners were only suitable for large scale scanning, for example of buildings (Bernardini & Rushmeier (2002)). In the recent years 3D cameras have been developed that are able to record whole depth images of a small region at high frame rates (see Kolb *et al.* (2009) for an overview). However, their measurements are prone to noise and contain systematic errors and outliers. Another possibility for scanning smaller object is the triangulation scanner (Beraldin *et al.* (1995)). Here a laser line is projected on the object and recorded with a camera positioned next to the laser source. The setup has to be calibrated, which allows us to deduce depth information by triangulating the point of intersection on the object (Figure 2.3 (b)). A similar result can be achieved by projecting structured light onto the scene instead of a laser line (König & Gumhold (2008)). Here several images are projected into the scene, whose temporal progression uniquely determines the spatial angle in the projection device. This allows us to identify

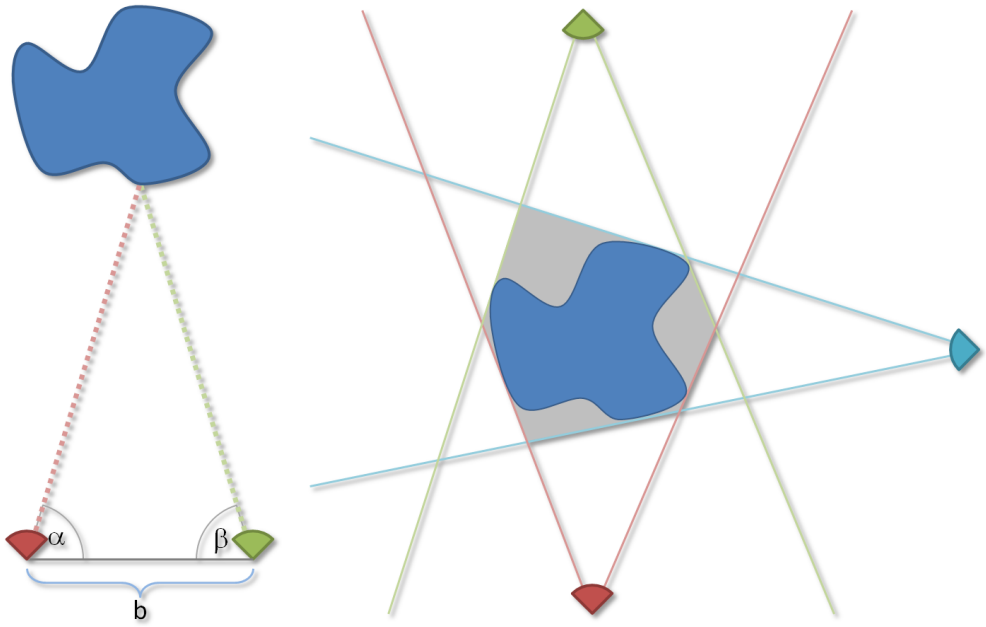


(a) (b) (c)

Figure 2.3: Different 3D scanning devices. Red boxes are light emitters, green devices are cameras. (a) A time-of-flight scanner estimates distance by measuring the time a laser light pulse takes to return to the scanner. (b) Laser triangulation scanners measure depth by triangulating the 3D point position from a known baseline distance **b** between laser and camera and the angles α and β. (c) Structured light scanners work similar to triangulation scanners, however instead of using a laser as light source they use a projector showing several binary patterns which allow for identification of the angles for each point seen by the camera.

the source angle of the projector and thus again to triangulate the intersection point (Figure 2.3 (c)).

Passive scanning techniques do not emit any light into the scene but rather capture the existing light using one or several cameras. For a multi-view stereo reconstruction we take two or more images of the object from different but known positions. By analyzing the image content we now try to identify identical points on the object and triangulate their position (Figure 2.4 (a)). This is also called the *correspondence problem*. There exist a variety of different methods of solving the correspondence problem, ranging from brute force search (Goesele *et al.* (2006)) to sophisticated optimization algorithms based on feature extraction methods (Furukawa & Ponce (2007)). An overview of recent methods can be found in Seitz *et al.* (2006). Another complementary approach is to segment the object in the images from the background and use shape from silhouette to generate a visual hull approximation of the object (Figure 2.4 (b)). These methods can also be combined to achieve better reconstructions (Kutulakos & Seitz (2000)).



(a) (b)

Figure 2.4: Passive 3D scanning techniques. (a) Using two cameras with known baseline width b allows us to estimate the 3D position of a point seen in both views. (b) Visual hull approaches intersect silhouette volumes to approximate the shape of the object (gray volume).

All of the presented 3D scanning techniques only produce semi-structured output. This is usually a 2.5D depth-map which first is converted into a 3D point cloud. All of the outputs contain noise due to the measurement process and possible outliers and holes. As we need to capture several depth-maps to cover the object completely without any major holes, we also need to register the separate scans onto each other (Aiger *et al.* (2008)). After this is done, we can perform surface reconstruction.

In the following we will present an overview of surface reconstruction techniques. Our goal is to create a structured triangle mesh from the input point cloud that represents the scanned object accurately, is smooth and does not contain any significant holes.

2.2.2 Surface reconstruction

Surface reconstruction from unstructured point cloud data has been within interest of computer graphics since its early days. The further development of acquisition techniques with increasing amount of data and the quest for accuracy, correctness and robustness have rendered it a topic of active research. In the following, we cannot go into details of reconstruction methods in general, and for an overview refer to the recent work of Kazhdan (2005) and Kazhdan *et al.* (2006) and the references therein. Following Kazhdan, reconstruction methods can be categorized as: Computational Geometry based methods, implicit function based approaches which fit and extract iso-surfaces, and such methods which fit an explicit surface to the data. Here we will give a short overview of works from the last category.

Early work in this direction are active meshes (Terzopoulos & Vasilescu (1991)), where a grid structure is deformed to fit sampled intensity and range data based on a mass-spring model. Chen & Medioni (1995) apply a dynamic physical model to "inflate" a balloon-like mesh to fit scanned data from the inside; the process is supported by local adaptation. An inverse shrink wrapping method Kobbelt *et al.* (1999) fits meshes to meshes.

Fitting methods are particularly used in reverse engineering together with surface classification, segmentation, and feature detection; often local regions are processed separately as patches (see, e.g., Varady *et al.* (1997)). However, for most methods it is difficult or impossible to generate reasonable results if parts of the input data are missing. On the other hand this situation is typical for most acquisition techniques. Nevertheless, consistency of models and efficiency of the reconstruction process are vital for many applications. Lévy (2003) fills missing parts smoothly by working in the parametric domain. Sharf *et al.* (2004) generate local geometry based on similarity measures to existing parts.

This leads to template-based approaches, where missing or contaminated input data is compensated by additional knowledge in form of a template shape. Template-based methods were frequently used in the previous years especially for reconstructing animated models. Kähler *et al.* (2002) use a detailed model of face anatomy to generate facial animation; this is an example for a highly specialized model. In Allen *et al.* (2002, 2003); Anguelov *et al.* (2005), a combination of templates together with learning techniques is used to reconstruct motion of a human model with a known skeleton. Most recently, Kraevoy & Sheffer (2005) establish

maps between triangulated data and template in order to transfer missing geometry. This work, together with the work by Allen *et al.* (2002, 2003), is built on cross-parameterization between the template and the input model. Pauly *et al.* (2005) apply global deformation, similar to Allen *et al.* (2003), to a collection of templates. They determine the best fitting template parts by segmentation and rebuild the target model from these. For non-rigid registration, warping techniques based on thin-plate splines have been used Brown & Rusinkiewicz (2004); Chui & Rangarajan (2003).

Recently, technological advances in scanning devices have made it possible to acquire 3D scans at interactive frame rates. These animated point clouds represent a whole new challenge for shape reconstruction algorithms. As all of these devices only capture the object from a single point of view (effectively only providing a single depth map), we never see the whole object at a single point in time. While we can use this input data to efficiently scan static objects in a short time, the more interesting applications lie in scanning and reconstructing deforming geometry. One of the first methods for animation reconstruction was presented by Wand *et al.* (2007) and later refined in Wand *et al.* (2009). Here, the authors try to progressively estimate a template (called ur-shape) from the deforming geometry and use this information to complete the shape in all time steps. Li *et al.* (2000) on the other hand take a given smooth template mesh and fit it to the animated point cloud data.

2.3 Shape editing

One of the most active fields in the area of computer graphics and geometry processing in the recent years has been the area of shape editing. We want to deform a shape (usually in the form of triangle meshes) to a set of (user specified) constraints in an as natural and plausible way as possible. Unlike simulation where we are concerned with the time-varying, physically accurate behavior of objects under external forces, we only want to find a plausible steady state solution given the external constraints. Another important aspect is that the deformation control itself needs to be intuitive and user-friendly. There are many applications for this kind of approaches, reaching from modeling over shape reconstruction to animation.

The earliest approaches to shape modeling are free-form deformation methods as presented by Coquillart (1990); Sederberg & Scott (1986). The shape is

embedded in a lattice grid and the vertex positions are encoded as linear combinations of the surrounding lattice cell. The user then can freely modify the lattice and transfer this deformation back onto the mesh. These approaches enable high-quality shape modeling, but typically fail to reproduce physically plausible transformation results. Additionally, the modeling metaphor of a lattice grid is limiting and lacks intuitiveness.

A more intuitive modeling metaphor is to define *handles* and influence regions directly on the shape (Kobbelt *et al.* (1998)). The user can then freely move and modify the handles, influencing the region of interest while keeping the remaining shape static. Also, physical plausibility is a highly-desirable property, as it leads to a deformation behavior of the edited objects that a user is familiar with from real-world experience. Therefore, it has recently become very popular to model deformation by minimizing physically related energies.

Shape editing methods can be classified in two groups, *linear* methods and *non-linear* methods. While linear methods are usually easy to implement and very fast, they often lack an intuitive user control paradigm or produce unexpected results under larger deformation results. Non-linear methods on the other hand are computationally more involved and have only recently become viable for real-time applications, but usually produce more natural deformation results.

2.3.1 Linear methods

Most linear shape editing methods are linearized versions of originally non-linear energies. One of the most popular choices here in the recent years has been to model deformation using physically based thin membrane and thin shell energies. The linearization of these energies lead to linear Laplacian or Poisson systems and are sometimes also called differential methods (as they are based on differential properties of the mesh).

The main advantage of this methods is that calculating a deformation under a given set of constraints reduces to finding the solution of a simple sparse linear equation system $\mathbf{Lx} = \delta$. Here $\mathbf{L}$ is the Laplacian system matrix, δ are the Laplacian/differential coordinates and $\mathbf{x}$ represent the absolute vertex positions of the mesh. This linear system can be solved easily using iterative or direct solvers (for example the conjugate gradient method or a Cholesky decomposition). We will explain this representation in more detail in Chapter 3. One major drawback of this process is that this leads to artifacts under larger deformations, as linear methods cannot properly couple translational and rotational components

(which is usually called translational insensitivity). While it is possible to specify rotational constraints in linear settings (i.e. by explicitly rotating a handle around a given axis) and also find plausible deformation results for these constraints, it is not possible to induce rotational deformation effects by translation of a handle.

Using linear Laplacian coordinates was first suggested by Alexa (2001) in the context of shape morphing. Lipman *et al.* (2004) suggested to include the representation within an interactive modeling environment. They tried to address the issue of rotational insensitivity by explicitly estimating rotations from an initial deformation. Sorkine *et al.* (2004) and Fu *et al.* (2007) implicitly included a linear rotation optimization derived from a one-ring neighborhood of each vertex into the system. Their implicit solution has the disadvantage of also including isotropic scaling in the solution, which is removed by performing a second deformation step with re-scaled differential coordinates. Igarashi *et al.* (2005) perform 2D shape deformation with a two step approach similar to that of Lipman *et al.* (2004).

Lipman *et al.* (2005) developed a frame-based representation of the differential coordinates. The coordinates are encoded in a locally defined frame at each vertex. When the user specifies a deformation they first solve for new local frame orientations followed by the actual reconstruction of the deformation. This method was developed further in Lipman *et al.* (2007). It allows for handling of much larger rotations than the previous methods while still preserving local detail.

Other methods forgo the traditional handle-based interaction metaphor and use sketch based interfaces in combination with linear differential representations. Nealen *et al.* (2005) first proposed a sketch based editing system where the user can select and modify silhouettes of the shape. Zhou *et al.* (2005) use a similar modeling approach and extend it by integrating a volumetric graph Laplacian into the model representing the interior of the object. This internal graph helps preserving the local volume of the mesh during deformation.

A different approach to mesh editing was introduced by Yu *et al.* (2004) by extending the gradient based methods that have been used before for images processing (for example by Fattal *et al.* (2002) for HDR compression and Pérez *et al.* (2003) for more general image processing) to 3D meshes. Here the gradients of the surface coordinate functions defined over the base shape were used to build a Poisson equation system. By modifying the gradients by applying transformations to them and then solving the Poisson system it is possible to deform the shape. This can be seen as exploding the mesh into its components

(in this case triangles), transforming each independently and finally solving for a new connected configuration where the gradients are as close as possible to the prescribed ones ("gluing" the mesh back together). Botsch *et al.* (2006b) showed that the resulting Poisson equation system is actually just a different way of constructing the Laplacian system **L** used in other methods.

Yu *et al.* (2004) use geodesic propagation for interpolation of transformations between handles. This was extended by Zayer *et al.* (2005) by using harmonic interpolation instead of geodesics. They also showed how this approach can be used to transfer the deformation of one mesh to another where local correspondences have been specified. A related approach was used by Sumner & Popovic (2004) for deformation transfer between meshes. Popa *et al.* (2007) generalized the harmonic propagation to also handle material properties, allowing the user to specify the stiffness of the material locally.

Botsch *et al.* (2006b) implemented a two step gradient based approach similar to Lipman *et al.* (2004) by first performing a linear deformation, estimating local rotations from this and then applying them to a gradient based approach, thereby achieving a certain degree of translational dependence. Yoshizawa *et al.* (2007) addresses the issue by calculating a medial skeleton of the mesh, which is deformed linearly in a first step and then used to extract local rotations.

In part II of this thesis we will explain the linear Laplacian deformation approach in more detail. We will introduce a method to explicitly handle the rotational insensitivity based on a skeleton graph and show several applications which benefit of this representation (de Aguiar *et al.* (2007a,b,c, 2008a); Gall *et al.* (2009); Stoll *et al.* (2006a,b)).

2.3.2 Non-linear methods

In contrast to the linear shape editing methods presented in the previous section, non-linear methods try to minimize deformation energies without simplifying them to a system of linear equations. This means that they require more complex calculations and thus take a longer time to produce a result, often not allowing for real-time interaction. However, the quality of the modified shapes, especially under large deformations, is usually much higher. There exists a wide variety of non-linear methods and we will only briefly mention some of the most recent approaches.

The work in Sheffer & Kraevoy (2004) introduces a non-linear differential coordinate representation based on angles and edge lengths. Huang *et al.* (2006)

embed the shape into a control mesh using mean value coordinates (Ju *et al.* (2005)) as a means of simplifying the dimension of their non-linear energy minimization. Botsch *et al.* (2006a) create a shell of prisms from a mesh and then uses a global/local approach to minimize bending and stretching energies between the prisms. This was extended in Botsch *et al.* (2007) for adaptive rigid grids that also cover the internal volume of a shape. A method which optimizes for an optimal non-linear combination of example shapes was proposed in Sumner *et al.* (2005) and later extended in Der *et al.* (2006). A vector-field-based framework which guarantees volume preservation was proposed by von Funck *et al.* (2006). Although their approach enables the definition of advanced implicit deformation tools, it is not well-suited for handle-based shape modification. Sumner *et al.* (2007) deform shapes by embedding a deformation graph that specifies local transformation matrices. Through optimization of non-linear energies they can generate natural and intuitive deformations of the graph and transfer them back to the shape. Sorkine & Alexa (2007) and Au *et al.* (2006) iterate a linear Laplacian deformation with a differential update step to achieve non-linear deformation behavior.

In part II of this thesis we introduce a non-linear shape editing method that is based on a simple iteration scheme (de Aguiar *et al.* (2008a); Stoll *et al.* (2007)). It was developed concurrently to Sorkine & Alexa (2007) and is closely related to their approach. By iterating a linear Laplacian deformation of a tetrahedral mesh and a differential update step it is possible to efficiently generate natural and plausible non-linear deformations.

2.4 Physical simulation

The field of physical simulation overlaps the field of computer graphics to a certain degree. In physical simulation applications, we are concerned with predicting the behavior of objects and materials under external forces. This is achieved by modeling the processes using the laws of physics in the computer. Physical simulation is used for a wide range of applications, such as crash test simulations, calculating strains and stresses for statics simulations or weather forecasts. The main applications of physical simulation in the area of computer graphics are special effects for movies and games. Some of the active fields here include fluid simulation, rigid and non-rigid body simulations or simulation of muscles and tendons for creation of realistic virtual characters. Many shape editing methods are based on simplifications of elastic body simulations. As we are not concerned

with the dynamic behavior of objects in this field, the methods are often modified to directly find steady state solutions.

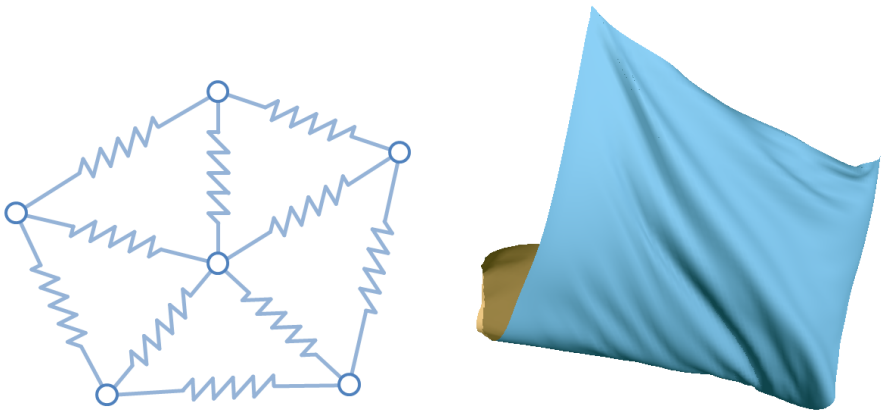
One particular field of physical simulation that will be important for part III of this thesis is the area of cloth simulation. Because of this, we will give a brief overview over the field in the next section. For a more in-depth review of the common techniques we refer the reader to the extensive reports by Volino *et al.* (2005) and Choi & Ko (2005).

2.4.1 Cloth simulation

Realistically simulating garments for virtual characters has been a goal in computer graphics for a long time. Traditionally, cloth has been simulated as a sheet of elastic material. By combining the Lagrange equations of motion and elastic surface energy, one can derive formulas for simulating cloth (Terzopoulos & Fleischer (1988)). The two main ways of integrating these are using finite element methods or particle systems. While finite element methods represent a more accurate discretization of the problem, their complexity has limited their use so far. Most of today's cloth simulation techniques therefore rely on a particle based representation (Breen *et al.* (1994)). Here the cloth is represented by the vertices of a polygonal mesh constituting the cloth surface. The vertices are moved by applying forces that represent the behavior of the cloth. The most commonly used variation of this principle is the mass-spring model (see Figure 2.5). Vertices are modeled as infinitely small points, each having a mass, that are connected to their neighboring vertices by springs. This allows us to compute forces acting on the vertices and moving them through space.

Given position, velocity and forces acting on a point, there exist several methods of integrating this information along time to obtain the respective values at the next time-step. While explicit integration schemes, like the first-order Euler or fourth-order Runge-Kutta method (Eberhardt *et al.* (1996)), are simple to implement and fast, they suffer from instability problems, requiring the time-step of the simulation to be set to a small value. Implicit integration schemes (Baraff & Witkin (1998)) on the other hand circumvent this problem, but require solving large sparse linear equation systems that change at every time-step. Because of this, implicit schemes are often not applicable to real-time systems.

Most real world cloth is nearly inextensible. However, many simulation schemes share the common problem that the system becomes numerically unstable when the material stiffness is increased (i.e. less stretching is allowed). Only in recent



(a) (b)

Figure 2.5: (a) Simple mass-spring representation of a piece of cloth. (b) Simulation of a piece of cloth hanging from two corners.

years the issue of simulating near inextensible cloth has been addressed (Goldenthal *et al.* (2007); Provot (1996); Thomaszewski *et al.* (2009)), allowing for more realistic and very natural simulations. There has also been some developments in simulating cloth not as a solid sheet but rather on the level of individual fibers (Kaldor *et al.* (2008)). While this dramatically increases the computational effort required for simulation, these methods provide a much more accurate representation of the actual cloth behavior.

Another important factor influencing the quality of cloth simulation is the employed collision resolution method. During a typical simulation we have to resolve many cloth-environment and cloth-cloth collisions. Detecting these collisions accurately and resolving them (including friction and restitution) is one of the most time consuming tasks of cloth simulation. As the simulation runs in discrete time-steps, collisions need to be detected not only in the spatial domain (i.e. using bounding volume hierarchies or similar approaches for acceleration), but also in the time-dimension (i.e. resolving when and where exactly the moving triangles/edges/vertices penetrate each other). Speed-ups can be achieved by grouping together particles involved in multiple collisions and handling them as a single object (Provot (1997)), adding repulsive forces and applying geometric treatment of collisions (Baraff *et al.* (2003); Bridson *et al.* (2002)). Recent methods are able to reliable handle collisions occurring in very large meshes and when layering many levels of cloth on top of each other (Selle *et al.* (2009)).

2.5 Performance capture

Performance capture is the process of reconstructing a dynamic scene of one or several performers from input video. The main applications for performance capture lie in the field of entertainment. Using these techniques allows us to capture the motion of an actor and transfer it to a virtual avatar. This is mostly used for special effects purposes in movies and games, where convincing and realistic motion of computer generated characters is important. Recently, these approaches have also been used to capture the detailed facial motion of performers. A second application field for performance capture can be found in 3D television. Here the goal is to allow users to view a recorded scene from arbitrary points in 3D space. First simple 3DTV approaches are already seen for analysis of sports games, but may also have future applications in the movie industry. Performance capture can be viewed as shape reconstruction problem. Given only a very sparse set of samples (images or even just marker trajectories), we want to reconstruct a detailed 3D model of the scene and performance as accurately as possible.

In the following sections we will give a brief overview of the field of performance capture, from traditional motion capture over 3D video techniques to recent mesh-based performance capture techniques.

2.5.1 Motion capture

Following Gleicher & Ferrier (2002) the goal of motion capture is "*to record the movement of a performer [...] in a compact and usable manner*". Traditionally this was achieved by approximating the human body as a small number rigid segments that are connected along joints, called kinematic skeleton. This approach reduces the task of motion capture to finding the correct 3D skeletal configuration given a stream of video observations of a performer (Menache (1999)). However, the reduction of the motion of a person to a set of skeletal joint parameters is inaccurate. Not all anatomic joints in the human body can be accurately represented by rotational joints (for example the shoulders). Additionally, the human body cannot be accurately approximated as a set of piece-wise rigid elements. Even if the performer wears skin-tight clothes we can observe strongly non-rigid deformations during motion. If we allow for arbitrary clothes like for example wide pants or even skirts, this approximation becomes even more inaccurate. However, the skeletal representation makes the problem of capturing the movement tractable, as it reduces the dimensionality of the representation drastically

(typical skeletal representations used for motion capture have somewhere between 30 and 50 degrees of freedom). Additionally, a skeleton representation simplifies processing and editing the recorded motion. When we are using an input model parameterized with a kinematic skeleton, motion capture is also a template-based shape fitting process where we want to reconstruct the global pose and motion of a performer using the template as accurately as possible.

2.5.1.1 Marker-based systems

The industry standard for approaching this problem is by using marker-based motion capture systems. The performer is required to wear a special suit to which a set of markers have been attached. The markers are designed so that they can be easily located in the video streams of the cameras recording the scene. Examples for this are passive markers, which usually consist of retro-reflective tape reflecting under infrared lights (Vicon (2009)), or active markers consisting of infra-red LEDs (Phasespace (2009)). The position of the markers on the body is known and associated with the bones of the kinematic skeleton. Using this setup it is possible triangulate the 3D position of the markers in each frame and estimate the pose of the skeleton. While there are a lot of difficulties to overcome with this type of setup (disambiguation, occlusions and missing markers), and we often need to perform manual post-processing, marker-based systems allow us to record the pose and motion of a performer very accurately. However, these systems are also limited in their application range. The user is required to wear the special marker suit, which is an intrusive process. Additionally, it is not possible to capture dynamic shape, motion and textural appearance of the actor in arbitrary apparel synchronously.

2.5.1.2 Marker-less systems

A first step to address some of the limitations of marker-based systems was the introduction of marker-less motion capture system (see Moeslund *et al.* (2006) and Poppe (2007) for an overview of methods). Instead of using the markers in the images to estimate the skeletal pose these system use *computer vision* techniques to extract features directly from the video without optical scene modification. Marker-less systems are more flexible than marker-based systems. However, it remains difficult for them to achieve the same level of accuracy and application range. Image features may be very difficult to extract from the input videos and contain a high level of noise and inaccuracies, limiting the quality of the resulting

motion unless recorded in a controlled studio environment. As these systems still rely on a kinematic skeleton as body model, it is also difficult to capture motion or detailed shape unless the performer wears skin tight apparel.

2.5.2 3D video

A related goal is pursued in recent years by *3D video* methods. Here the aim is to render a captured real-world scene from new synthetic camera viewpoints that were never seen by a virtual camera. The input here is similar to motion capture approaches. Given a stream of video observations we want to render an image from a novel viewpoint. As most 3D video methods try to be as general as possible, they usually reject a template based approach in favor of more general scene reconstruction techniques. These include intersecting multi-view silhouette cones (Gross *et al.* (2003); Matusik *et al.* (2000)) and multi-view stereo based approaches (Waschbüsch *et al.* (2005); Zitnick *et al.* (2004)). However, as these methods process each frame separately they are not able to generate spatio-temporally coherent geometry (i.e. models with constant connectivity). A coherent geometry simplifies post-processing and storage of the data and is helpful for editing the recorded scenes. Starck & Hilton (2007) and Ahmed *et al.* (2008b) perform a post-processing on their data to establish a common parameterization over time. Carranza *et al.* (2003) use a template model to capture the scene, but due to the kinematic structure used, they cannot capture performers in arbitrary clothes.

2.5.3 Performance capture

Recently, mesh-based performance capture approaches have been introduced which are able to overcome some of the limitations of marker-based and traditional marker-less motion capture systems. Similar to other marker-less motion capture systems they track the motion of a given mesh using computer vision techniques. Instead of only relying on a kinematic skeleton as underlying shape parameterization they also incorporate shape editing methods to adapt a template to the input data.

Carranza *et al.* (2003) use a template model consisting of separate parts for each bone and adjust the sizes to best fit to the input data. Allen *et al.* (2006) learn a deformation model of the naked shoulder and torso from body scans in different postures. Sand *et al.* (2003) use a similar idea and learn pose-dependent

body deformations by using marker-based motion capture and silhouette matching. Similarly, the SCAPE model learns variations in human body shape and surface deformation from laser scans of people (Anguelov *et al.* (2005)) and builds a statistical model. Park & Hodgins (2008) learn a data-driven model of skin and muscle deformation from dense marker-based motion capture data. All these methods are suitable for more or less tightly clothed performers, but none of them can handle people in wide apparel. Balan *et al.* (2007) use a template to capture detailed motion and deformations of a piece of apparel, however are not able to capture a whole performer or fast motions.

To facilitate reconstruction of coherent geometry of people in arbitrary clothing we propose to use deformable meshes created from static full-body laser scans for tracking (de Aguiar *et al.* (2007b, 2008a)). Similar to our work in Gall *et al.* (2009), the method by Vlasic *et al.* (2008) also reconstructs spatio-temporally coherent geometry of people in everyday apparel by fitting a skeleton to visual hulls and non-rigidly deforming a surface mesh to capture time-varying surface geometry. We will explain our methods in more detail in the following parts of this thesis.

Part I

Differential coordinate based shape processing using surfaces

In this part of the thesis we present several novel methods using template based shape processing based on surface meshes. We introduce a method for constrained texture mapping using an image template, Chapter 4 (Stoll *et al.* (2006b)). In contrast to previous work it is applicable to any type of manifold surface, in particular high-genus surfaces, without any additional processing. Following this, we extend the template plane fitting process to arbitrary shapes, and show how to reconstruct surfaces from incomplete 3D scans using a template, Chapter 5 (Stoll *et al.* (2006a)). By formulating surface reconstruction as a template fitting problem we are able to fill holes in a semantically meaningful way and accurately reconstruct the surface using a simple iterative deformation process. Finally, we propose several novel applications in the field of performance capture applications, Chapter 6 (de Aguiar *et al.* (2007a,b,c, 2008a); Gall *et al.* (2009)). Instead of relying on a kinematic skeleton, as traditional methods do, we use a template fitting approach to capture both motion and surface deformation of performers. This allows us to even reconstruct the motion actors wearing wide apparel, which was difficult with traditional methods. All these applications have in common that we need an efficient and practical way of processing meshes. The primary task we need to perform is to deform them under a given set of constraints while preserving the overall quality of the mesh. In Chapter 3 we present an efficient framework based on differential coordinates for mesh deformation that enables us to perform these tasks.

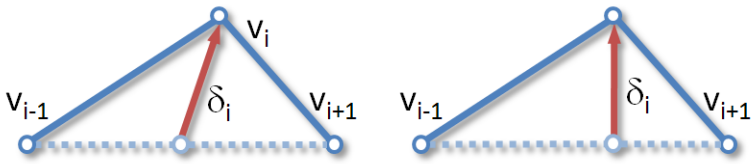
Chapter 3

A deformation framework for triangle mesh based templates

In this chapter we will present a method for deforming a template under a set of user constraints by means of linear differential coordinates. This method will enable us to formulate a variety of different tasks as deformable template fitting process.

We will first discuss the choice of shape representation and explain the deformation framework in more detail. The choice of shape representation greatly influences the range of processing one can perform with the object. In computer graphics it has been common to use triangle meshes because this piecewise linear representation provides a straightforward way to display a shape (by rasterization or ray-tracing), analyze its structure (through the connectivity graph), and extract local geometric properties (such as curvature). While other representations such as splines and subdivision surfaces are more commonly used for modeling, all of these are converted into triangle meshes for display purposes in the end. Most real-time applications (for example games) rely on them as a means of representing their objects. This makes triangle meshes the most natural choice for our template shape representation. Additionally, a vast amount of triangle meshes are available online in shape libraries for use in shape processing, simplifying the search for a suitable template.

Our goal is to be able to deform a triangle mesh $\mathcal{M}$ under a given set of constraints $\mathbf{C}$. While it is possible to define deformation methods that work using just the triangle mesh in its native representation (i.e. absolute vertex positions and connectivity), switching to another form of mesh representation will simplify our task. Differential representations have gained increased popularity in the



(a) (b)

Figure 3.1: 2D example of different weighting schemes for calculating differential coordinates. (a) Uniform weights. (b) Geometry aware weights. Note how the uniform differential coordinate on the left includes normal and tangential components, while the geometry aware coordinate on the right is limited to the normal direction.

fields of computer graphics in the recent years. They are based on the idea that it is possible to describe a triangle mesh by representing it through differential means (i.e. using relative coordinates instead of absolute).

In the following we first introduce the general differential representation we use. Following this, we explain how we can apply it to deform an object using certain constraints. Finally, we will suggest a novel solution to prevent artifacts due to the translational insensitivity caused by the linear formulation. This novel solution is based on a feature point graph.

3.1 Differential representation

Linear differential coordinates for a triangle mesh $\mathcal{M}$ are defined as the difference between the absolute coordinates of its vertices v_i and the center of mass of its one ring neighborhood (Botsch & Sorkine (2008); Sorkine (2005)) :

$$\delta_i = \sum_{j \in N(i)} w_{ij} \left(v_j - v_i \right). \tag{3.1}$$

Here w_{ij} are weights for the edge connecting the vertices v_i and v_j. The differential geometry interpretation of these coordinates is that they are a discretization of the continuous Laplace-Beltrami operator δ considering that our mesh is a piecewise linear approximation of a smooth surface (Carmo (1976)). The Laplace-Beltrami operator in the limit case is the mean curvature normal of the surface (i.e. the normal vector $\mathbf{n}$ scaled by the mean curvature H). This also holds for the limit case of the discretized differential coordinates (when we

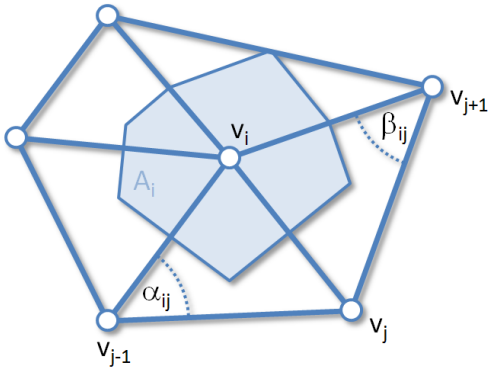


Figure 3.2: Illustration of the angles α_{ij} and β_{ij} as well as the Voronoi area A_i necessary to calculate the discretized Laplace-Beltrami operator using cotangent weights.

refine the mesh infinitely), meaning that then $\delta_i = -H_i \mathbf{n}_i$ (Taubin (1995)) . This means that the differential coordinate approximates the local normal direction and curvature of the surface.

The choice of weighting scheme greatly influences the accuracy of this approximation. While in the limit case (i.e. infinite subdivision of the triangle mesh) all reasonable weighting choices will converge toward the original Laplace-Beltrami integral, we want to choose our weights in such a way that they approximate the operator reasonably well at a coarse level. The most simple choice for weights are uniform weights, i.e. setting $w_{ij} = 1$ for all edges. This however does not take the geometric properties of the mesh into account. If we look at the resulting differential vector in a 2D example (see Figure 3.1 (a)) we can see that the coordinate δ_i contains tangential as well as normal components. The most common choice for geometrically motivated weights are the so called cotangent-weights (Meyer *et al.* (2002); Pinkall & Polthier (1993)). Here the weights w_{ij} are dependent on the Voronoi area A_i of the vertex v_i and the angles α and β opposing the edge (see Figure 3.2) :

$$w_{ij} = \frac{1}{2A_i} \left(cot\alpha_{ij} + cot\beta_{ij} \right) . \tag{3.2}$$

The area A_i is defined as the area of the surface region built by connecting incident edges' midpoints with triangle circumcenters (for acute triangles) or midpoints of opposite edges (for obtuse triangles), as shown in Figure 3.2. If we have a look at the differential vector in 2D again (see Figure 3.1 (b)) we can now

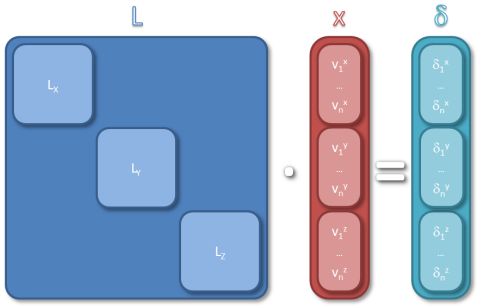


Figure 3.3: The linear Equation system from 3.3. Note that if there are no further Equations that connect the dimensions x, y and z the system can be split into three smaller linear systems and solved separately, increasing processing speed. The sub matrices $\mathbf{L}_x$, $\mathbf{L}_y$ and $\mathbf{L}_z$ are identical.

see that δ_i only includes a normal component while all tangential information is "stored" in the weights w_{ij}.

Using Equation 3.1 we can write the discrete Laplace-Beltrami operator for our triangle mesh $\mathcal{M}$ as a single sparse matrix $\mathbf{L}$. Using this matrix and our vertex positions $\mathcal{V}$ written in a single large vector we can formulate this as a large Equation system

$$\delta = \mathbf{L}\mathcal{V}. \tag{3.3}$$

Calculating the differential coordinates is now a simple matrix-vector multiplication of $\mathbf{L}$ and $\mathcal{V}$. As we are using a local relative representation of our mesh now, the matrix $\mathbf{L}$ will be rank-deficient and cannot be inverted as is. This is caused by the fact that the differential coordinates δ are invariant to translations of $\mathcal{V}$, i.e. $\mathbf{L}\mathcal{V} = \mathbf{L}(\mathcal{V} + \mathbf{o})$, where $\mathbf{o} \in \Re^3$ is an arbitrary displacement vector. The consequence of this is that we can easily calculate δ given $\mathbf{L}$ and $\mathcal{V}$, but cannot uniquely reconstruct $\mathcal{V}$ given $\mathbf{L}$ and δ only.

One drawback of using the linear formulation from Equation 3.3 is that the differential coordinates are not invariant to transformations other than translations (i.e. rotation or scale). This means that if we compare the vector δ_a of a mesh $\mathcal{M}_a$ to the vector δ_b computed from a mesh $\mathcal{M}_b$ which is just $\mathcal{T}_a$ globally rotated or scaled, then $\delta_a \neq \delta_b$ (see Figure 3.4). While it is possible to formulate non-linear differential representations (Huang *et al.* (2006); Sheffer & Kraevoy

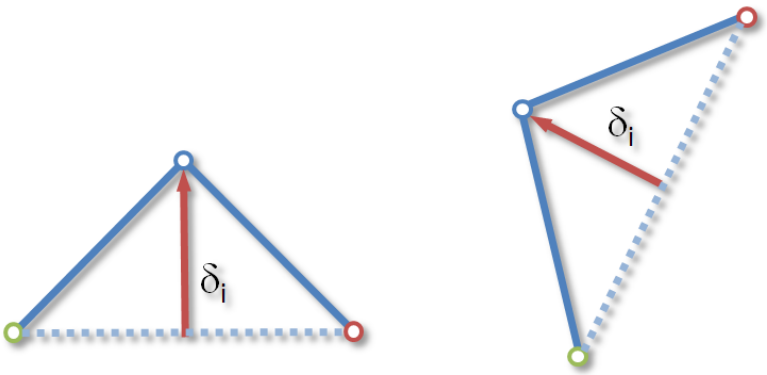


Figure 3.4: Differential coordinates under rigid transformation. Although the geometry for the left and right object are identical up to a rotation, the differential coordinates are not the same, illustrating the sensitivity of linear differential coordinates to transformations.

(2004)) which do not suffer from this limitation, complex optimization problems need to be solved to process them.

For a more detailed overview of differential coordinates and their derivation, please refer to Sorkine (2005) and Botsch & Sorkine (2008).

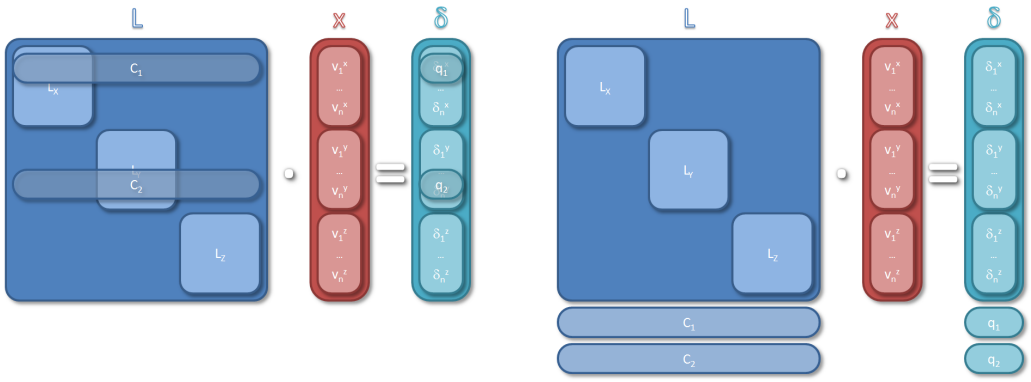
3.2 Reconstruction and deformation

Given the differential coordinates δ and the Laplacian matrix $\mathbf{L}$ of a mesh it is not right away possible to reconstruct the original global coordinates $\mathbf{x} = \mathcal{V}$. As $\mathbf{L}$ is rank deficient the Equation system

$$\mathbf{x} = \mathbf{L}^{-1}\delta \tag{3.4}$$

cannot be solved as $\mathbf{L}$ is singular and cannot be inverted. We need to add at least one positional constraint on a vertex to generate a full rank matrix (thereby resolving the translation invariance of the coordinates), allowing it to reconstruct the absolute mesh coordinates. If we define more constraints than necessary to resolve this invariance, the solution of the system will be a deformed version of the original mesh that adheres to the prescribed constraints.

Given linear constraints of the form $\mathbf{Cx} = \mathbf{q}$ there are two major ways to define constraints in our differential linear system. The first is to specify them as hard constraints, i.e. constraints that *have* to be satisfied by the solution. This



(a) (b)

Figure 3.5: Building the linear Laplacian systems. (a) Linear system with hard constraints replacing the respective rows. (b) Adding soft constraints to the linear system.

can be achieved by replacing rows corresponding to the constrained vertex with the desired constraint (see Figure 3.5 (a)).

The second possibility is to specify soft constraints, i.e. constraints that do not need to be satisfied exactly but only approximated. This can be achieved by adding additional rows to the original linear system from Equation 3.3 (see Figure 3.5 (b)), resulting in an Equation system of the form

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{C} \end{pmatrix} \mathbf{x} = \begin{pmatrix} \delta \\ \mathbf{q} \end{pmatrix}. \tag{3.5}$$

This Equation system is overdetermined, meaning that there usually does not exist an exact solution. However, it is possible to find a solution that minimizes the residual in the least squares sense :

$$\underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \|\mathbf{L}\mathbf{x} - \delta\|^2 + \|\mathbf{C}\mathbf{x} - \mathbf{q}\|^2 \right\}. \tag{3.6}$$

Using soft constraints is better suited for processing data containing uncertainties, such as the 3D scans we will be processing later, as it allows us to specify weights for constraints that regulate their importance or reliability.

We can find the minimum of Equation 3.6 analytically by solving the normal Equations of the linear system 3.5 :

$$(\mathbf{L}^\top \mathbf{L} + \mathbf{C}^\top \mathbf{C})\mathbf{x} \;=\; \mathbf{L}^\top \mathbf{d} + \mathbf{C}^\top \mathbf{q} \;.. \tag{3.7}$$

The normal Equation system is symmetric and positive definite and has a unique solution. There exist a multitude of strategies to find the solution of the Equation system efficiently (Botsch *et al.* (2005)).

When the left hand side of the system (i.e. the matrix) will often stay constant throughout the processing it is of advantage to use a direct solver. Direct solvers pre-factorize the matrix and afterward find the solution of the Equation system by a simple back-substitution. The most expensive part of this process is the factorization which does not involve the right hand side of the Equation system. As such a factorized system can be solved very efficiently several times for different right hand sides. This is of importance when the constrained vertices are the same over several deformations but their target positions change. We employ a Cholesky-decomposition as a direct solver, as it allows for very efficient back-substitutions in this setting (Golub & Loan (1996)).

In the case of constantly changing constraints, it may be more sensible to use iterative solvers to find the solution of the system, such as conjugate gradients. While they are usually slower than direct approaches, they do not require a factorization of the matrix and as such are more efficient in these cases (Golub & Loan (1996)).

The minimum of Equation 3.6 are the vertex positions of our mesh deformed under the given constraints in the global coordinate system. The reconstruction finds a balance between satisfying the constraints and preserving the local detail of the mesh defined by the differential coordinates δ.

3.2.1 Constraint types

To deform the mesh the user needs to be able to specify constraints in an intuitive way. Constraints in general can be any relation that can be described as a linear Equation of the form

$$w_j \mathbf{c}_j = w_j \mathbf{q}_j, \tag{3.8}$$

where w_j is a weight assigned to that specific constraint, specifying its importance for the deformation. While it is possible to prescribe relations between several vertices (for example the differential coordinate is such a constraint), we will concentrate here on constraints which influence the behavior of a single vertex.

3D vertex position The most simple constraint is a 3D positional constraint. It has the form

$$w_j \mathbf{v}_j = w_j \mathbf{p}_j, \tag{3.9}$$

which essentially pulls vertex $\mathbf{v}_j$ toward the absolute 3D position $\mathbf{p}_j$. This kind of constraint is used when it is possible to estimate the full 3D target position of a vertex.

Line features The second type of constraint we are concerned with is the 2D line constraint, which pulls a vertex toward a given parametric line in 3D space. We can describe a line in 3D as the intersection of two planes. We can thus constrain a point by specifying that it should lie on each plane :

$$\begin{aligned} w_j N_1 \mathbf{v}_j &= -w_j d_1 \\ w_j N_2 \mathbf{v}_j &= -w_j d_2 \end{aligned} . \tag{3.10}$$

Here, N_1, d_1 and N_2, d_2 specify the implicit plane Equations. As a line constraint only fixes two positional dimensions and leaves the third open, it is necessary to specify at least two of them to be able to solve the linear system in Equation 3.7 uniquely (as the system will be rank deficient otherwise).

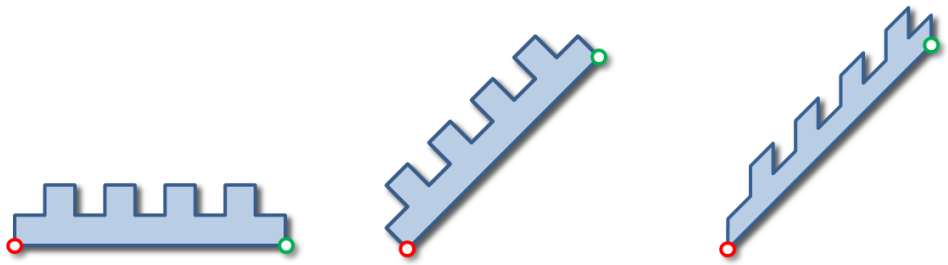
3.2.2 Harmonic interpolation

We can also use the Laplacian formulation introduced above to interpolate sparse data specified on some vertices across the entire surface. Instead of solving the system from equation 3.3 under soft constraints, we can solve the equation

$$\mathbf{L}\S = 0 \tag{3.11}$$

under Dirichlet boundary constraints, i.e. hard constraints as introduced above and shown in Figure 3.5. Not using the differential coordinates δ when solving this guarantees a $\mathcal{C}^1$ smooth (with the exception of the constraint points themselves) interpolation of the constrained values over the mesh. Physically the result of this process is the steady-state solution of the heat-equation : we apply a fixed temperature at the constraint points and wait until the temperature on the remaining surface does not change anymore.

This process can be used to find smooth distribution of sparse values over the mesh. For example if we are given a color value at a few vertices, we can find colors for the remaining vertices using harmonic interpolation.



(a) (b) (c)

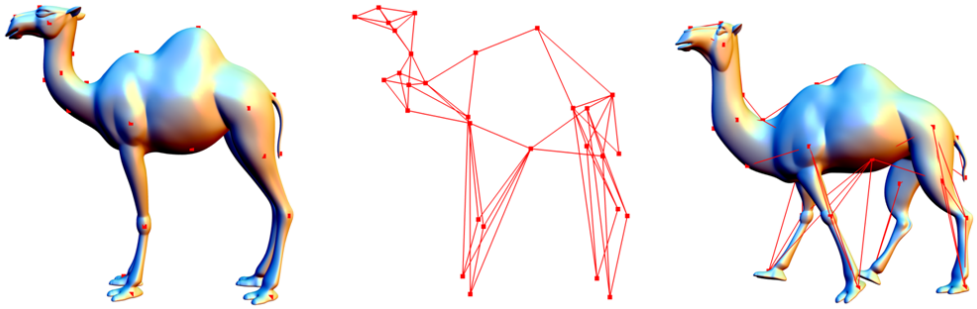
Figure 3.6: Artifacts appearing under larger deformations : The object in (a) is deformed under two point constraints shown as red and green dots. If we move the green constraint up and to the left we would intuitively expect the object to undergo a near rigid rotation as shown in (b). Due to the preservation of the linear differential coordinates however the deformation instead produces the result shown in (c), causing considerable shearing artifacts. This is also called translational insensitivity.

3.2.3 Rotational invariance

The deformation using the linear differential coordinates obtained by finding the minimum of Equation 3.6 generates convincing results for small deformations or objects with low mean curvature (i.e. when all $\|\delta_i\|^2$ are small). When prescribing large deformations or when using complex objects, the system may generate unexpected results (see Figure 3.6). This is due to the fact that the linear differential coordinates are not rotationally invariant. Rotationally invariant representations are by their nature non-linear.

There are several strategies to amend this issue. Sorkine *et al.* (2004) modify the Laplacian matrix to include a linear approximation of similarity transformations. This leads to better behavior for smaller deformations, but introduces scaling effects under larger deformations. Other solutions use a multi-scale approach to solve the problem. Here we first estimate the rotation at each position on the mesh and then incorporate these rotations back into our linear system.

Our solution is related to the work by Zayer *et al.* (2005). They specify sparse rotations at handle points and interpolate them over the mesh using harmonic interpolation (equation 3.11) to generate a dense field of rotations. This field is then incorporated into the linear system to be solved. We will present a very similar approach here, however unlike Zayer *et al.* (2005) we will specify our dense rotation field on the vertices instead of on the triangles.



(a) (b) (c)

Figure 3.7: Skeleton generation for estimation of a dense rotation field. (a) Model with constraint points. (b) Constructed skeleton consisting of the minimal spanning tree and the three closest neighbors of each constraint point based on geodesic distances. (c) Deformed mesh with applied rotation field.

Given a dense rotation field $\mathbf{R}$ consisting of a rotation transformation $\mathbf{r}_i$ for each vertex, we can modify our least squares energy from Equation 3.6 to accommodate these rotations :

$$\underset{\mathbf{x}}{\operatorname{argmin}} \left\{ \|\mathbf{L}\mathbf{x} - \mathbf{R}\delta\|^2 + \|\mathbf{C}\mathbf{x} - \mathbf{q}\|^2 \right\}, \tag{3.12}$$

To find this dense set of rotations $\mathbf{R}$ we first find a sparse set of rotations at a few vertices (which are usually the constrained vertices from $\mathbf{C}$). These sparse rotations may either be defined manually by the user or can be extracted from a "skeleton" of our constraint points.

The skeleton is constructed by first finding the connectivity graph $\mathbf{C}$ whose nodes $\mathbf{n}$ are positioned at the constraint points (Figure 3.7 (a)). Each edge $\mathbf{e}_{ij}$ of the graph is assigned the approximate geodesic distance between the vertices v_i and v_j as weight. We now construct the minimal spanning-tree $\mathbf{G}$ of this graph considering the edge weights. This ensures that there exists a path from each node of the graph to all others. At each node, we further insert an edge to the three closest neighbors (considering the geodesic weights) to the graph (Figure 3.7 (b)), ensuring that each node in the graph has at least three incident edges.

Given the node positions $\mathbf{p}$ before the deformation and the target positions $\mathbf{p}'$ prescribed by the user deformation we can now estimate the rotation at the node by estimating the transformation $\mathbf{T}$ that best matches the incident edges

of $\mathbf{G}$ to the same edges in the target positions of $\mathbf{G}'$. The 3×3 transformation matrix $\mathbf{T}_i$ of a node i can be found by solving the least-squares equation system

$$\begin{pmatrix} \mathbf{p}_i - \mathbf{p}_{N_i(1)} \\ \vdots \\ \mathbf{p}_i - \mathbf{p}_{N_i(j)} \end{pmatrix} \mathbf{T}_i = \begin{pmatrix} \mathbf{p}'_i - \mathbf{p}'_{N_i(1)} \\ \vdots \\ \mathbf{p}'_i - \mathbf{p}'_{N_i(j)} \end{pmatrix} \tag{3.13}$$

where $N_i(1)$ to $N_i(j)$ are the indices of the neighbor nodes of $\mathbf{p}_i$. The matrix $\mathbf{T}_i$ will usually contain anisotropic scaling (stretching) and shearing components and does not represent a rigid transformation as we would desire. As shown by Shoemake & Duff (1992), we can factor $\mathbf{T}_i$ into an orthonormal rotation matrix $\mathbf{R}_i$ and a non-rotational part $\mathbf{S}_i$,

$$\mathbf{T}_i = \mathbf{R}_i \mathbf{S}_i \ . \tag{3.14}$$

Technically, the rotational component can be computed by iteratively averaging $\mathbf{T}_i$ with its inverse transpose as

$$\mathbf{H}_0 = \mathbf{T}_i \ , \ \mathbf{H}_{k+1} = \frac{1}{2}(\mathbf{H}_k + \mathbf{H}_k{}^{-T}) \ . \tag{3.15}$$

The decomposition iterates up to a step k after which $\mathbf{H}_k$ does not change anymore, and then sets $\mathbf{R}_i = \mathbf{H}_k$.

We now have a sparse set of rotations defined at the constraint vertices. To generate the dense set of rotations $\mathbf{R}$ we use harmonic interpolation. For the harmonic interpolation the rotations are represented using 4D unit quaternions. We then interpolate the four dimensions separately using equation 3.11. The resulting interpolated quaternions are normalized and converted to rotation matrices, giving us the dense set of rotations.

Using this dense rotation field we can now solve Equation 3.12 for a deformed mesh that adapts to the constraints in a more natural way (see Figure 3.7 (c)).

Note here that this procedure does not generally work for all kinds of deformation constraints, but rather only for systems where we only specify positional constraints such as introduced in the section above. It is also necessary to make sure that there are constraints distributed regularly over the whole mesh. If this is not the case, the rotation interpolation may produce unnatural results (mainly by extrapolating rotations in an unexpected way).

Chapter 4

Inverse texture mapping

In this chapter we introduce a method for constrained texture mapping based on using an image as a planar 3D template that is deformed onto a mesh (Stoll *et al.* (2006b)).

Texture mapping is a common technique in computer graphics that wraps a two-dimensional image around a polygonal mesh model to add details and enhance the visual appearance of the model. Constrained texture mapping is applied when alignments are required between the model and the image. A known application that uses constraints is facial texture mapping that requires the alignment of prominent features of the 3D model (such as eyes, nose, mouth, etc.) with the corresponding sections of the texture image.

In computer graphics the term mapping or parameterization refers to the process of establishing a bijective (one-to-one and onto) correspondence between the 3D model and a 2D domain. Texture mapping is considered as a parameterization problem in which the 3D position of the models vertices are defined as a bi-variant piecewise-affine function and the two independent variables are used as texture-coordinates. In constrained texture mapping the texture-coordinates for part of the vertices are enforced and the result is a bi-variant function that complies with the constraints and parameterizes the remaining vertices. This makes the parameterization problem much more difficult and might even render the problem unfeasible.

Texture mapping improves the visual appearance of a polygonal model. A highly detailed image gives the illusion of a detailed object, although the underlying geometric structure of polygons and vertices may be coarse. Hoverer, when closely examined, the rough geometric structure becomes noticeable and

together with image distortion artifacts, caused by the parameterization process, the model may look less realistic.

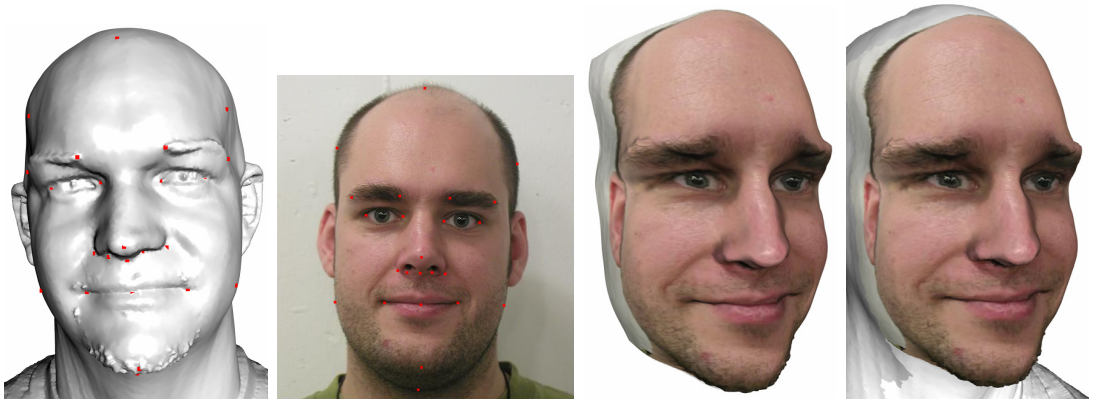
Our method deals with the constrained texture mapping challenge differently. Instead of finding a parameterization of the mesh to the image plane we interpret the 2D image domain as a planar 3D template mesh and fit it to the 3D model. Based on a set of constrained vertices, the image plane is deformed to roughly capture the 3D models pose and shape. Similar to Lévy (2001) the constraints are satisfied in a least-squares manner. A matching based on geodesic distances is then used to couple the unconstrained vertices and the image pixels. Using the matching and the initial constraints, the image plane is deformed again into its final shape. Our technique results in a smooth, visually pleasing and realistic textured model that can be imposed onto or used instead of the original model.

In contrast to parameterization-based approaches, our method has the virtue of being applicable to any type of a manifold surface, in particular high genus models and non-triangulated meshes, as long as it supports the evaluation of geodesic distances. Figure 4.1 shows a texture deformed to meet the shape of a humans head model, which is of genus zero (higher genus models can be used in the same way). Our method is not meant to be a parameterization technique and hence does not guarantee a bijective mapping. However, for many cases it achieves a valid parameterization and traditional texture mapping can be applied as well.

One main application for this process of inverse texture mapping is the fitting of cloth from 2D sewing patterns to 3D human models without having to resort to an expensive physics simulation. For this, the mapped texture image can simply be mapped to an offset surface of the actual mesh.

Mesh parameterization and texture mapping received significant attention from the scientific community for several years. Rather than list them all, we point the interested reader to the excellent survey by Floater & Hormann (2005) and references therein for more details on this topic. We note that common to most methods is their goal of minimizing a distortion measure while guaranteeing a bijective mapping (a mapping that is one-to-one and onto).

Most parameterization methods focus on meshes with a topology homeomorphic to a disc and are limited to triangulated surfaces. To parameterize non-disc-like meshes, it is common to first partition (segment) the mesh into disc-like patches, parameterize each separately and pack them back together in the parameterization space, see Lévy *et al.* (2002) and references therein for more details on atlas-based parameterization methods. A different parameterization approach



(a) (b) (c) (d)

Figure 4.1: Deformed texture and texture mapping on a genus 0 mesh model: (a) The mesh model, (b) Texture image, (c) Deformed texture, (d) Mesh model with texture mapping.

can be applied to genus zero meshes with a topology equivalent to a sphere. Gotsman *et al.* (2003) show how to generate a bijective mapping by generalizing barycentric coordinate methods for planar parameterization. Saba *et al.* (2005) present an optimized numerical scheme that finds a solution solution of Gotsmans formulation efficiently.

While there exist a multitude of parameterization methods, only a few techniques satisfy positional constraints. Lévy (2001) suggests adding the constraints to the linear parameterization system. He satisfies the constraints in a least-squares manner, yielding a "soft" constraints solution. For many applications such a solution is sufficient. However, when adding large number of constraints the method might result in a non bijective parameterization. Eckstein *et al.* (2001) and Kraevoy *et al.* (2003) suggest a method that guarantees the constraints position together with the validity of the embedding. Both methods add new vertices, called Steiner vertices, to the mesh when the constraints cause an over-determined system. Kraevoys method also relies on a valid parameterization of the original mesh. Karni & Gotsman (2000) suggest using a free-boundary linear parameterization method that compels the positional constraints into their place. In case of an invalid solution, they suggest an iterative method that after "several" iterations fixes the invalid areas. However, this method does not guarantee an upper bound on this number of iterations nor that a valid solution will be reached.

To the best of our knowledge, there are no methods for constrained parameterization of high genus models. Alexa (1999) embeds genus zero meshes on a sphere with the presence of constraints. However, in contrast to the approach described in this chapter, their suggestion does not guarantee a solution, even in the event that such a case exists.

4.1 Initial deformation

Input to our method are a $m \times n$ pixel image texture $\mathcal{I}$ with a corresponding image surface mesh $\mathcal{S}$ consisting of $m \times n$ pixels $\mathcal{P}$ embedded in 3D, the target triangle mesh $\mathcal{M}$, and a set of k user-given constraints $\mathbf{K} = (\mathbf{s}_i, \mathbf{v}_i)$ each linking an image surface vertex to a triangle mesh vertex. The image surface $\mathcal{S}$ is in this case the template we want to match to the 3D object and an example of a simple planar template.

Similar to Sorkine & Cohen-Or (2004) we start by deforming our image surface mesh $\mathcal{S}$ to fit the set of 3D constraints. We work on the grid-like connectivity structure of the image and use Equation 3.1 on the pixel neighborhood with a constant weight of $w_{ij} = 1$ and build a linear system of the form

$$\begin{pmatrix} \mathbf{L} \\ \mathbf{C}_k \end{pmatrix} \mathbf{x} = \begin{pmatrix} \delta \\ \mathbf{q}_k \end{pmatrix}, \tag{4.1}$$

where $\mathbf{C}_k$ and $\mathbf{q}_k$ contain the positional constraints of $\mathbf{K}$. Note that due to the planar structure of the image plane it will always have $\delta_i = 0$ for the internal pixels. To deform the image surface $\mathcal{S}$ we now solve for the 3D "pixel" positions $\mathbf{x}$ by minimizing Equation 3.6 (i.e. solving the least-squares linear system).

The deformed surface $\mathcal{S}_d$ now roughly captures the mesh pose as it is imposed by the constraints. Besides mimicking the mesh pose, the initial deformation has an important role in scaling the image surface to approximate the area of the mesh surface. This stage is essential for the matching processes described in the next section. Figure 4.3 shows several examples of deformed image planes based on the constraints points alone. It is easy to notice that the intrinsic parameterization of the image plane is preserved during the deformation.

4.2 Surface Matching

The matching stage couples the non-constrained vertices $\mathcal{V}$ of the mesh with corresponding pixels from the template image $\mathcal{S}$. The matching is one-to-one but

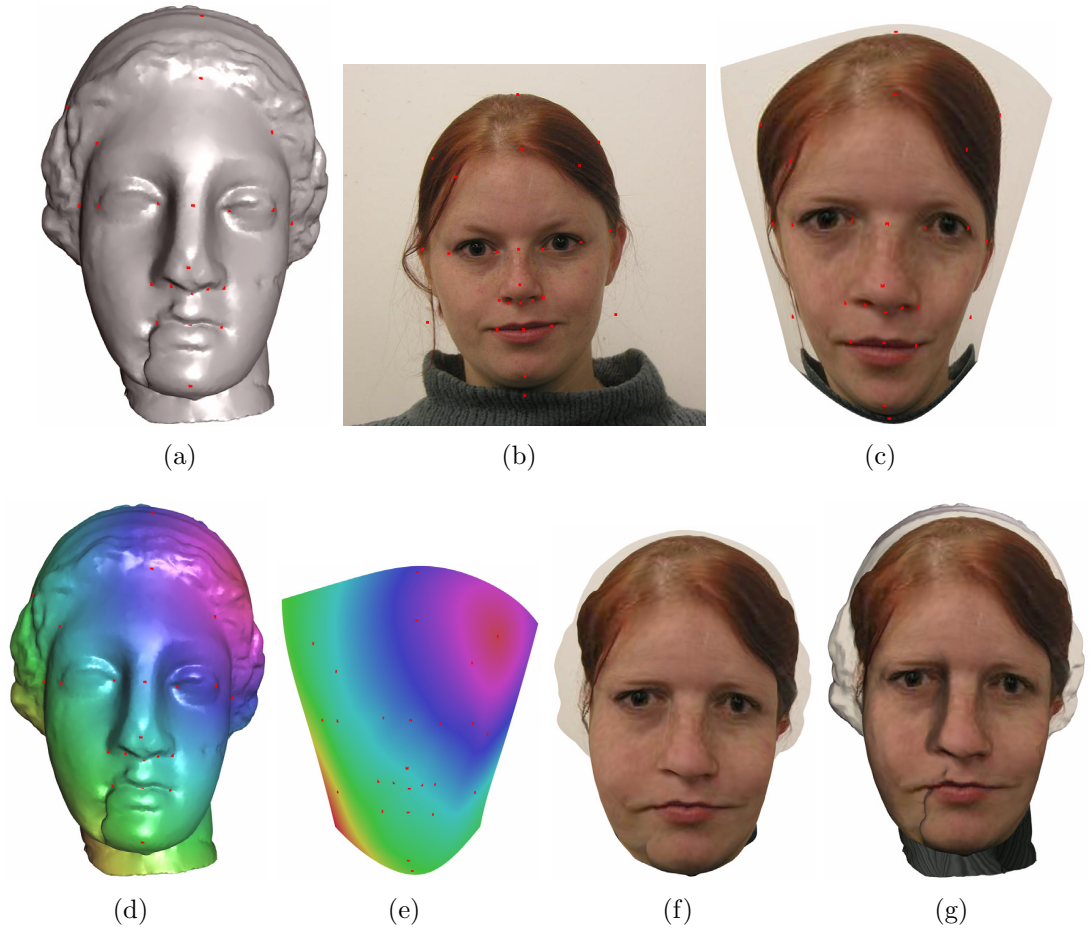


Figure 4.2: Processing steps: (a,b) A genus 0 mesh model and the texture image together with the constrain points, (c) Initially deformed texture, (d,e) Geodesic distances from one constraint point along the mesh and deformed texture surfaces, (f) Final deformed texture, (g) Texture mapping based on the geodesic distances parameterization.

not onto, meaning that several pixels exist without a matching (it is common to assume that the number of pixels in the image is much larger than the number of vertices in the mesh).

The matching should capture the intrinsic properties of the two surfaces. For example: a vertex that lies between two constraint vertices should be matched with a pixel that lies between the corresponding constraint pixels. Zayer *et al.* (2005) used vector fields generated by harmonic maps to match between two meshes for the application of transformation transfer. The principle of the matching is: Let $\mathbf{F}^T$ and $\mathbf{F}^S$ be the vector fields along the triangle mesh and image

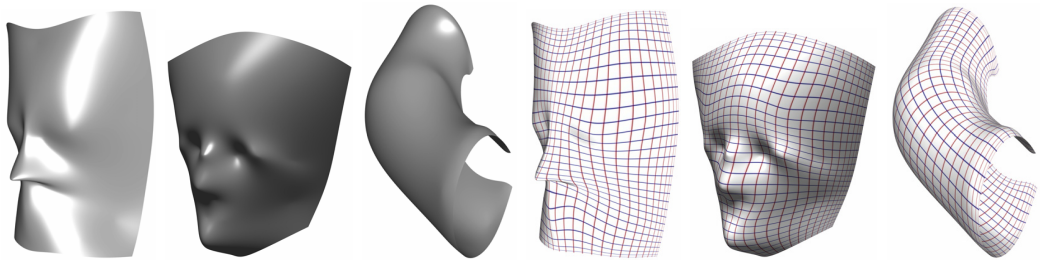


Figure 4.3: Deformed image planes based on the constraint points alone.

surface, respectively. For each vertex $\mathbf{v}_i \in \mathcal{V}$, we match the pixels $\mathbf{p}_j \in \mathcal{P}$ such that

$$j = \underset{j}{\operatorname{argmin}}(\mathbf{F}_i^T - \mathbf{F}_j^S)\ \mathbf{F}_i^T - \mathbf{F}_j^S < \epsilon. \tag{4.2}$$

The second term prevents the matching of points that are too far from any constraints and thus matching of points where the difference between their vector fields is too high. In our implementation ϵ is defined to be one percent of the model's bounding box multiplied by k (the number of constraints).

Harmonic functions have a cyclic nature that can cause the mapping to not be bijective. Therefore, we instead base our matching on two vector fields of geodesic distances. We calculate the distances using the method introduced by Surazhsky *et al.* (2005) , starting from each constrained vertex to the entire mesh vertices of $\mathcal{M}$, and from the constraint pixels to the rest of the image pixels of $\mathcal{S}$ on the deformed image surface. This generates two vector fields of dimension k (each entry is a distance from one constraint vertex or pixel), one along the surfaces of the mesh, the other one on the deformed image template. Figure 4.4 shows a three-dimensional subset of the vector field as [R,G,B] colors on the Igea and face model surfaces together with their deformed image surface.

Deforming an image against a disc-like mesh surface using geodesic distances results in a valid matching. However, when dealing with closed meshes (genus-0) or high-genus models, a trivial geodesic distance matching might lead to a mismatch. Figure 4.5 demonstrates the problem for a simple two-dimensional case. The bar, analog to an image plane, is to deform around the circle, analog to a genus-0 cylinder, using the marked constraints. Observe that the distance along the bar between points 5 and 6 is significantly different than the distance between the corresponding points on the circle. This might lead to matching between points in the surroundings of point 4 on the bar to points in the surroundings of

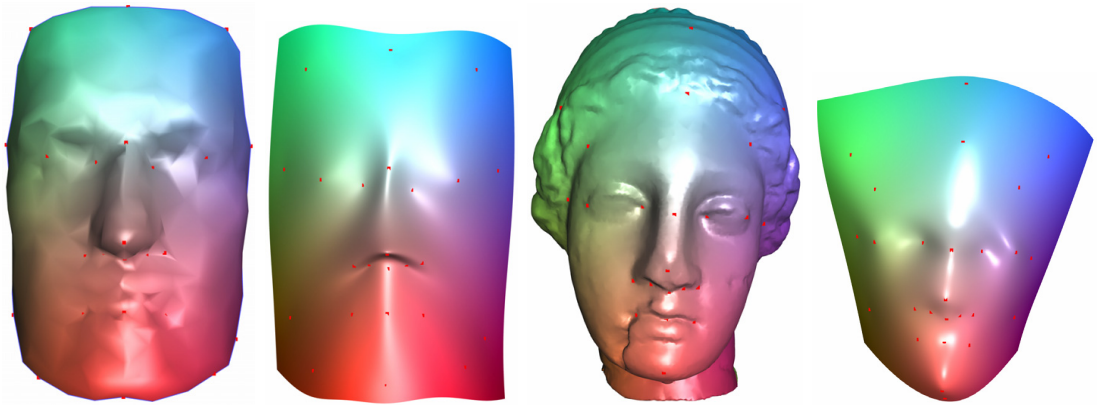


Figure 4.4: Geodesic distances along the surfaces of the Igea and Face models together with their corresponding deformed image surface. Each color represents one entry in the distances vector field.

point 6 on the circle and to a mismatch which will result in a distorted deformation and for sure will lead to an invalid parameterization.

To avoid this problem we recognize a potential mismatch by inspecting the distance vectors at the constraint vertices and pixels for significant variations. A mismatch is recognized when the ratio $\mathbf{F}_i^T/\mathbf{F}_i^S$ at the constraint points $i \in 1, \ldots, k$ is below 0.8 or above 1.25. When such a mismatch is discovered, the suspected entries are eliminated from the distance vectors for pixels and vertices close to the suspected region. This causes the matching to disregard the dimensions of the vector field which have strongly disagreeing entries and focus on the more reliable distance information in the other channels.

Using the dense matching generated with the above outlined procedure we deform the image surface again and bring it into its final pose. Similar to what was described in section 4.1, we use the Laplacian formulation, but instead of considering only the initial constrained points contained in $\mathbf{C}_k$, the entire set of matched points $\mathbf{C}_f$ is added, resulting in a new linear system of the form

$$\begin{pmatrix} \mathbf{L} \\ \omega\mathbf{C}_k \\ \sigma\mathbf{C}_f \end{pmatrix} \mathbf{x} = \begin{pmatrix} 0 \\ \omega\mathbf{q}_k \\ \sigma\mathbf{q}_f \end{pmatrix}, \tag{4.3}$$

By adjusting the value of ω, the user can control the penalty measure for any deviation from the original constraints position. High value of ω can be considered as hard-constraints. However, forcing hard-constraints might result in internal surface intersections and other not visually pleasing results. By adjusting

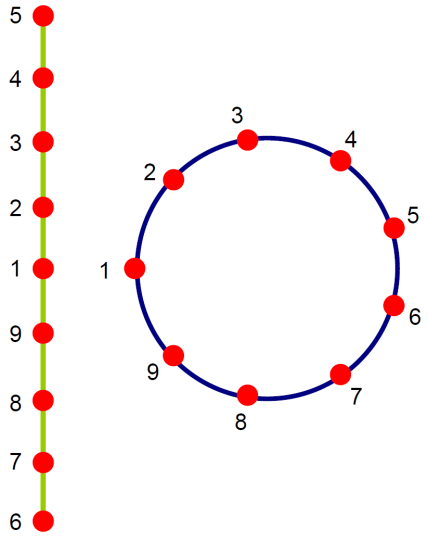


Figure 4.5: A two-dimensional analog to the deformation of a plane (represented as a bar) around a sphere (represented as circle) and the constraint points between them.

the value of σ the user can control the smoothness of the deformed surface. A small value of σ will reduce the influence of the matched points. For example, setting its value to 0 will result in the initial smooth deformed surface.

Matching the vertices of a mesh surface to image pixels establishes texture-coordinates and thus also a planar parameterization of the mesh (see Figure 4.6). Therefore, we can also use the matching results to render the image plane on the mesh surface using traditional texture mapping. Figure 4.6 shows the texture mapping of the tiger onto the face model together with its parameterization. However, the matching stage neither guarantees to cover the entire mesh vertices, nor to establish a valid mapping.

Zigelman *et al.* (2002) used a multi-dimensional-scaling technique on the geodesic distances to embed the mesh vertices on a 2D plane by preserving their original distances along the mesh surface. The embedding on a plane by itself is sufficient to generate texture-coordinates. However, their method that aims on facial texture-mapping also does not guarantee the validity of the mapping, and also does not satisfy positional constraints. An analog to Zigelman's method would be to match the mesh vertices to the image pixels based on the geodesic distances along the mesh surface and the image plane (instead of the deformed image surface). Calculating geodesic distances along a plane sums up to cal-

Figure 4.6: Using the vector fields to calculate a planar parameterization of the mesh. The picture shows the texture mapping of a tiger image onto the face model (left) and the corresponding parameterization domain (right).

culating Euclidean distances. Figure 4.7 shows the differences between the two approaches. It is easy to see that our approach (on the left) visually performs better. The distortions in the Euclidean distances method (right image) and the failure to meet the constraints are due to the differences in the distance measurements, especially around the nose area. As the initial deformation step already brings the image plane close to the shape of the 3D mesh the measured geodesic distance matches the actual mesh distances mush better.

4.3 Results

We implemented the suggested method in an interactive system (Stoll *et al.* (2006b)). The user loads a polygonal model and an image and interactively marks pairs of constraints points between the two. From this point on the system is completely automatic. It generates the initial deformed surface, calculates the geodesic distances and uses them for matching. Finally, it generates the final deformed surface. The user can then interactively add and remove constraints and change the weights of the final least-square system (see Section 4.3) to fine-tune the result and fit it to its needs.

We tested our method with several types of surfaces. Figure 4.9 shows a

Figure 4.7: Deformation based on geodesic distances measured along the deformed image surface (left) compared with deformation based on geodesic distances measured on the image plane (right).

texture-deformation for a disc-like surface. In Figure 4.2 (e) and Figure 4.1 our method is challenged with genus 0 models, and in Figure 4.8, a texture plane is deformed around one handle of the Figure Eight model (a genus-2 model). All the results show that the deformed texture-surface followed the shape of the mesh-surface while keeping the positional constraints in place, as much as they can be held, due to the least-squares nature of the solution.

In Figure 4.10 our method was tested with a noisy mesh. Gaussian noise was added to the Igea model and the same texture plane as in Figure 4.2 was deformed over it. It is evident that in spite of the noise over the mesh surface, the matching based on geodesic distances was successful and the deformed texture-surface inherited the noisy nature of the model surface. By fine-tuning the weights of the second deformation stage, we were able to generate a smooth version of the deformed texture while preserving all the facial details. The smoothed version looks visually better and natural.

In Figures 4.2, 4.8 and 4.10 we provide a texture-mapped model based on

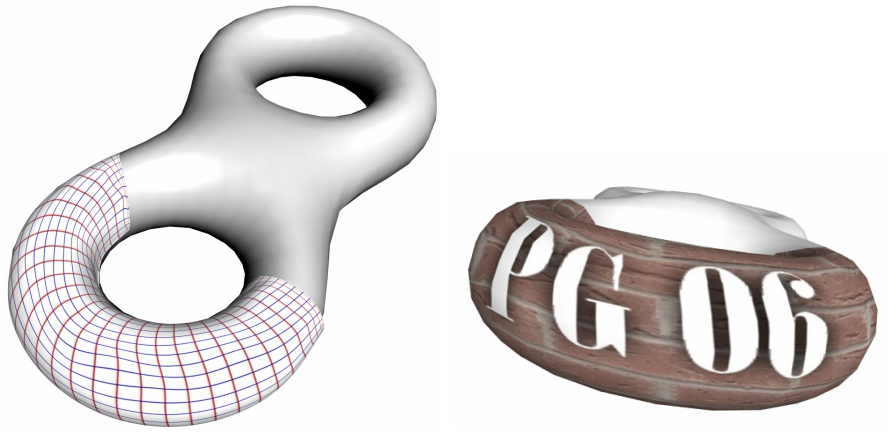


Figure 4.8: Deformed texture along one handle of the Figure Eight, a genus 2 model.

(a) (b) (c) (d) (e)

Figure 4.9: Constrained Texture Deformation: (a) Polygonal surface, (b) Texture image, (c,d,e) Deformed texture surface together with the texture image.

geodesic-distances parameterization. We did not check for the validity of the mapping although visually we could not find any evidence for it being invalid. In all examples the texture is well placed and the model looks visually good. However, the smooth version of the deformed texture-surface seems to us more natural, and in other words, better.

All experiments were preformed using an AMD Opteron-250 system with 2GB of memory. Table 4.1 summarizes the performance of our method on several of the models. We would like to emphasize that the timings listed in Table 4.1 are for the initial constraints points. Adding or removing constraints requires only an update of the already factorized deformation systems and the evaluation of geodesic distances from the new points alone.

It is evident that the major factor in the method's performance is the size of

Model	Face	Igea	Igea	Male	Eight
Vertices	1394	33K	33K	220K	1536
Pixels	40K	65K	262K	65K	65K
Constraints	24	26	26	25	26
Deform 1	2.25	4.45	35.03	4.32	4.42
Mesh Geod	0.12	14.38	14.38	31.2	0.312
Text Geod	15.36	27.56	123.24	25.44	27.56
Matching	0.01	0.49	3.36	1.17	1.17
Deform 2	2.23	4.52	36.51	4.48	4.48
Total	19.97	51.4	212.52	66.61	36.83

Table 4.1: Runtimes for the different stages of our algorithm (in seconds).

the texture image. It influences the runtime of the deformation stages and the geodesic distance calculations. This is not surprising. In most cases the number of pixels in the texture image is significantly larger than the number of vertices in the mesh model. In our experiments we used small and medium sized images. However, we would like our method to handle images with 5 to 10 Megapixels, such as current digital cameras produce.

It is important to note that the texture image surface has the special connectivity structure of a grid. Our solver, which was used in the deformation stages, does not exploit this property. We believe that by using solvers and an algorithm for geodesic distance computation that are dedicated to work on grid structures (e.g., multi-grid methods), we will be able to significantly reduce the computation time of those stages and enable the use of large images.

4.4 Discussion

In this Chapter we proposed a novel approach to constrained texture mapping that forgoes the traditional approach of mesh parameterization. Instead, we view the image as a 3D template surface that is fit to the surface of the target shape. This can be viewed as the inverse process to other texture mapping approaches. If the object to be textured is of particularly high genus or the user wants to replace the original shape geometry with our higher resolution texture object (which can be seen as a resampling of the surface), our method provides a powerful tool to the user. If it is essential to guarantee a valid mapping or preserve the original mesh geometry, it may be preferable to apply a traditional parameterization method that establishes a map from 3D shape to image plane.

(a) (b) (c) (d)

Figure 4.10: Deformed texture and texture mapping on a noisy mesh model: (a) Noisy Igea mesh model, (b) Texture mapping on the noisy Igea model based on the matched geodesic fields, (c) Deformed image plane with smoothing $\sigma = 1.0$, (d) Deformed image plane with smoothing $\sigma = 0.1$.

While it is possible to generate traditional texture mapping coordinates from the results of our fitting procedure, our method does not guarantee to deliver a valid and bijective mapping.

While most images in this Chapter show results created by mapping textures to faces, it is of most interest for mapping sewing patterns to 3D body models. Instead of mapping a whole rectangular grid we can simply omit all image points that do not belong to the pattern and even connect several patterns along the edges easily. Adding a normal offset to the found constraints allows for a straightforward and fast "dressing" of a model with a piece of cloth without having to resort to a more complex simulation.

Our template based approach to constrained texture mapping offers a new view on the problem of texture mapping and parameterization that does not require any handling of special cases like high-genus objects. The method also is a first step in the direction of shape reconstruction as it essentially performs a partial reparameterization of an object. It lays the foundation for more general template based approaches where the template is no longer limited to being a plane. We will introduce a similar approach that incorporates semantic knowledge in the form of a template shape in the area of shape reconstruction in the following chapter.

Chapter 5

Template based shape reconstruction

In this Chapter we show how we can apply template based shape processing to the task of semantic surface reconstruction from 3D scans (Stoll *et al.* (2006a)). We build upon the insights gained in the previous Chapter, and extend the template fitting process from planes to arbitrary template shapes.

Traditional surface reconstruction methods that do not incorporate semantic knowledge about the object will reconstruct correct surfaces only where the sampling density of the input point cloud is high enough. However, they are bound to produce implausible results when reconstructing objects where large parts have not been scanned. There have been previous approaches to incorporate semantic knowledge into surface reconstruction for hole-filling, for example by copying geometry from densely sampled regions (Sharf *et al.* (2004)) or by establishing cross-parameterizations between a template and scanned geometry (Kraevoy & Sheffer (2004)). However, these methods either fail in complex situations, require establishing complex parameterizations, are limited to specific objects like faces or humans (Anguelov *et al.* (2005); Kähler *et al.* (2002)), or require large object databases (Pauly *et al.* (2005)). The method we propose in this Chapter formulates surface reconstruction as a template fitting problem. This allows us to circumvent many of the issues of other methods. Using the deformation framework presented in Chapter 3 we can semantically fill holes and accurately reconstruct the shape surface in an intuitive and simple manner.

Typical data from 3D acquisition devices show missing surface regions even if multiple scans are spatially aligned and combined into a single model. Any reconstruction method is bound to fail for this data in the sense that missing

parts can be filled or extrapolated reasonably only if additional knowledge on the original shape is provided. This applies locally as well as globally: without knowledge, holes are patched smoothly (if at all), and global shape properties such as the genus cannot be detected from incomplete data.

There are various approaches how to apply additional knowledge to surface reconstruction. The method presented in this Chapter uses a template shape for reconstruction of point clouds without any processing such as noise and outlier removal. In particular no high-level tools such as parameterization or maps between surfaces are required. The only restriction is that in order to get reasonable results, the template should share the same global structure or "nature" with the data, e.g., a human model serves as template for reconstructing another human although shapes are in different poses.

Our method uses a small number of correspondence points marked interactively by the user to deform the template shape into the pose of the acquired point cloud model. Successive deformations improve approximation of the data. These deformations are guided by additional local correspondences which are established based on certain geometric criteria, similar to Chapter 4. Throughout the whole process, information is propagated from known to unknown parts of the shape. Our results show that with only minimal user interaction, the approach reconstructs a consistent and smooth surface that approximates the input data. If suitable templates exist, template-based methods like ours carry an advantage compared to those that work only on the plain data. We remark that in practice, templates exist for most common applications, many of them process a certain class of shapes, e.g., humans. This leads to another advantage to our approach: If the same template is used for different data sets we trivially obtain a map of correspondences between the deformed templates. Indeed, such maps are required by many applications, e.g., for texturing, and they are especially valuable for processing time-dependent data.

Our method processes general point data with normals that are either given from data acquisition or can be estimated (see, e.g., Jones *et al.* (2004)). Such data are typically afflicted with noise and outliers, are incomplete (i.e. have missing parts), and hence not directly appropriate for meshing. Our goal is to deform a template mesh such that the input data is approximated in global pose and local surface features, i.e., missing parts are filled from the template.

In order to achieve this, the user specifies a sparse set of corresponding points on point cloud and template (cf. Figure 5.1 (a), (b) and Sec. 5.1). Corresponding

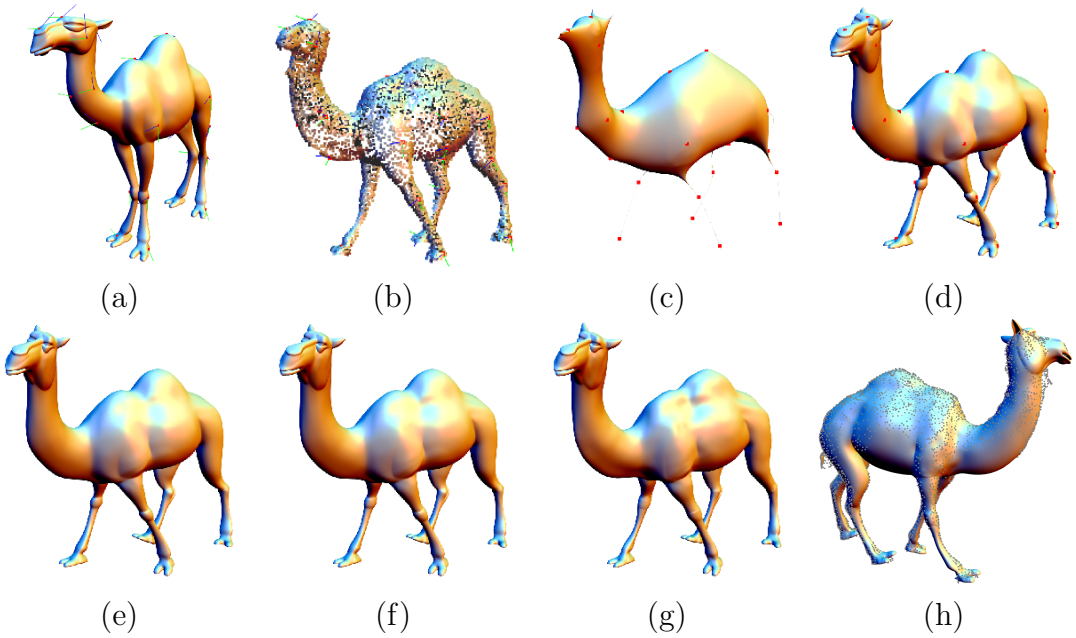


Figure 5.1: Overview of the method for the camel: (a) Template with 30 oriented markers; (b) point cloud with oriented markers; (c) initial deformation (no prior adjustment applied on input data, which are extremely out of scale); (d) after scaling. (e)-(g) show iterations 1,2, and 4, respectively; (h) shows the final result from the backside with points overlaid. Front head, feet, and tail were masked out due to missing data.

pairs are typically placed near shape features. In addition to positions, correspondence in local frames is established for each pair. This is either done manually by the user or can be estimated automatically using the method presented in section 3.2.3.

We first compute Laplacian coordinates of the template and estimate local rotations from the given corresponding local frames (see Chapter 3.2.3). This information is used to obtain an initial deformation using the framework presented in section 3.2. The result mimics the global pose of the data, however, it suffers from improper scaling (cf. Figure 5.1 (c) and Sec. 5.2).

In order to compensate for this, we recover a global scale from averaging ratios of discrete geodesic distances between the user specified points on the original and on the deformed template. The reconstruction from the scaled Laplacian coordinates now captures the global pose of the input data (Figure 5.1 (d)). However, taking only into account the input correspondence so far, the deformed mesh still resembles the template and does not yet provide sufficient local approximation of

the data points.

We address the latter issue with an iterative process of matching and reconstruction. For all vertices of the template, we search for matches on the data. This search is guided by a maximum allowed radius and a maximum allowed angular deviation of normals from template and data. False matches in erroneous or insufficiently sampled data regions are avoided by masking out such regions. For every match a weighted positional constraint is included in the linear system, and the deformed template mesh is reconstructed. This process is iterated based on the new deformation until the deformed template approximates the data sufficiently (cf. Figure 5.1 (e)–(g) and Sec. 5.3).

In a final step, we estimate local displacements for all local matches in the previous set based on the displacements from the initially deformed template and the last deformation. The lengths of displacements are interpolated over the template to generate new positional constraints for the final reconstruction. The rationale of this last step is that we want to propagate the scaling information into regions of the template with no matches or correspondences due to missing data.

A final result is shown in Figure 5.1 (h).

5.1 Experimental setup

Given the point data, the user chooses an appropriate template mesh that will be deformed to approximate the input. Our method is robust, and there are no restrictions on the template in general, however, no meaningful results can be expected from fitting strongly inappropriate templates, for example with different genus. Templates can be obtained in several ways. There exist large free online databases for triangle meshes (for example AIM@SHAPE (2004)) that provide a large data pool for template meshes. It is also possible to have an artist design a template using modeling tools. If it is possible to ensure that one can scan an object without missing data (i.e. by performing a series of range scans and combining them), it is also possible to use traditional surface reconstruction methods to generate the template.

In the next step the user identifies and marks pairs of corresponding points on template and point data. Such correspondences are required on or near shape features, e.g., at feet and knees for human or animal models. Local frames for the rotation are either chosen manually by the user or generated automatically using

the method described in section 3.2.3. Correspondences will be used to globally bring the template into the "pose" of the point data.

In the subsequent step, local correspondences between shapes will be searched based on heuristics. Here, it is important to remove the influence of data regions which are insufficiently sampled and cannot provide reasonable information. For this reason, the user selects a global region of interest either on the point data or on the template (as done in our examples). Points outside this region are ignored for any subsequent matching (Sec. 5.3), i.e., for these masked-out regions the template is deformed rigidly. The choice of this region and hence assessment of the data is straightforward even for a non-trained person.

5.2 Initial deformation and global scaling

From the selected correspondences, we compute the initial deformation of the template. Given the correspondences $\mathbf{C}$ and the rotations of local frames and the respective interpolated rotations $\mathbf{R}$ we can generate an initial deformation by applying the deformation framework we introduced in Chapter 3 and solving Equation 3.12.

The result (see Figure 5.1 (c)) already shows the desired pose, however, ignoring the fact that template and input data may be (and generally are) scaled differently may distort the resulting shape in an unacceptable way. In order to fix this issue, we estimate a global scaling factor λ for the template which is applied to the Laplacian coordinates $\mathbf{d}$. Therefore, we compute discrete geodesic paths (using Dijkstra's algorithm) between marked points on the original template and the initially deformed one. We obtain a global scale as average of the ratios of geodesic distances on both shapes. Our experiments show that a global scale is sufficient, and local diversification does not yield significant improvements except for extreme configurations, which did not occur in any of our examples. On the other hand simple global measures such as ratio of bounding box diagonals are not reliable enough due to different poses. Hence, we now solve the same system (3.12) again with an updated right-hand-side applying λ as scaling factor for the differential coordinates δ. A result is shown in Figure 5.1 (d). In the following we will re-apply the same system with updates in the constraints $\mathbf{C}, \mathbf{q}$.

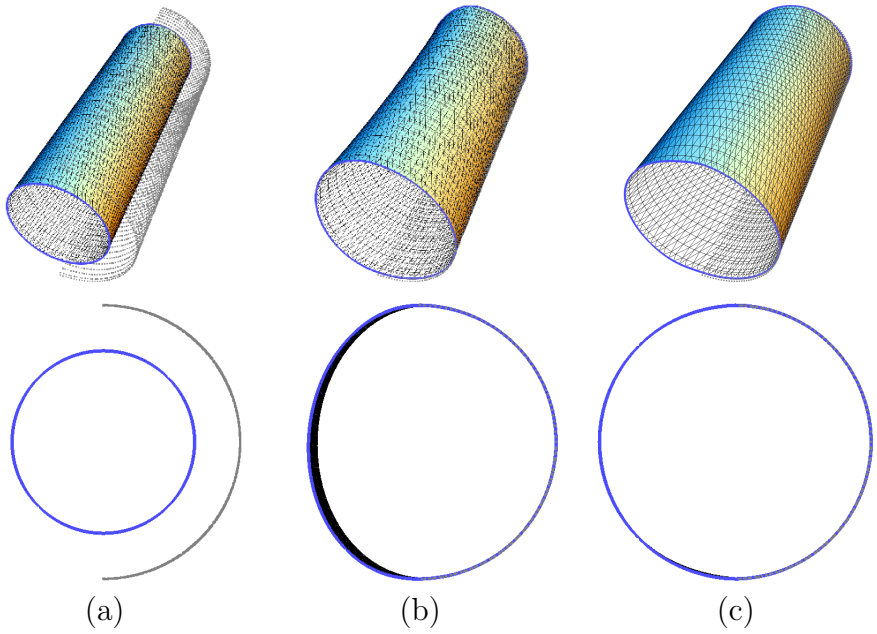


Figure 5.2: The cylinder example illustrates the effect of displacement propagation for the final reconstruction. (a) Template and point cloud, there is no data for the left half; (b) result after iterations using only data points; (c) for the final result, displacements were propagated to the region with missing data.

5.3 Iterative improvement

The initial deformation of the template mesh using proper scaling globally captures the pose of the point cloud. However, the deformed shape does not yet provide sufficient local approximation of the data. The following iterative process moves the template nearer toward the data points guided by local correspondences which are established from simple heuristics. This local matching is motivated by iterative closest point (Besl & McKay (1992)) (ICP) algorithms for finding (rigid) transformations for shape registration.

For every vertex of the template, we search for matches in the nearest data points within a maximum distance $r_{\max}$. Of all points found for a single vertex j those are rejected for which their normal deviates from the vertex normal by more than a maximum angle. From the remaining points we compute a positional constraint $\mathbf{q}_j$ as a weighted average of the point positions. We restrict these displacements to their contribution in direction of template normals to avoid tangential drift. The weighting is based on point-vertex distances mapped by a quadratic B-spline transfer function which maps distances 0 and $r_{\max}$ to 1 and

0, respectively, with C^1 continuity at the interval boundaries. This information is used to update $\mathbf{C}$ and $\mathbf{q}$ in (3.12). The updated system is then solved again, yielding a new deformation of the template. Starting from this new configuration, we search again for local matches and iterate the process. Note that this formulation only involves changing $\mathbf{C}$ and $\mathbf{q}$, where prior constraints are either preserved or overwritten by updates.

The local matching for finding new constraints must fail in regions where the point data is erroneous, e.g., due to measurement error, insufficient sampling, and missing data. This situation cannot be compensated by manually choosing correspondence and because of the lack of data no meaningful deformations can be extracted. Besides the fact that such regions may require manual post-processing, they must be excluded from local matching as they typically lead to false matches. This is done by restricting the search not only by local distance and angular thresholds but also by allowing the user to specify a globally defined region of interest within the point cloud using a simple painting interface.

After a sufficient number of iterations (4-15 for all our examples), the template mesh is deformed in a way that it approximates the point cloud in global pose and local shape. Of course this refers only to shape regions where point data is available (and the region of interest, respectively). In a final step, we improve on the remaining regions of the deformed template for which no counterparts exist in the data. We identify regions with sufficient point data simply as all vertices j for which (local) constraints have been found before. For all such vertices, we measure the displacement in normal direction between the initially deformed template and the result of the last iteration, respectively. These displacements capture shape information of the point cloud, and their lengths are then propagated over the template mesh using harmonic interpolation, similar to the rotations. The interpolated values are then used as normal displacements w.r.t. initial deformation and define additional constraints, which are assigned low weights (we used $w_j = \frac{1}{2}$). We observe that this heuristic provides plausible results, Figure 5.2 illustrates the effect for a simple example.

5.4 Results

We have implemented the method described above in an interactive environment (Stoll *et al.* (2006a)). The user positions constraint points on the template and point data. In rare cases it is necessary to place a few markers freely, away from the data. This situation occurred only for the hand example in Figure 5.4,

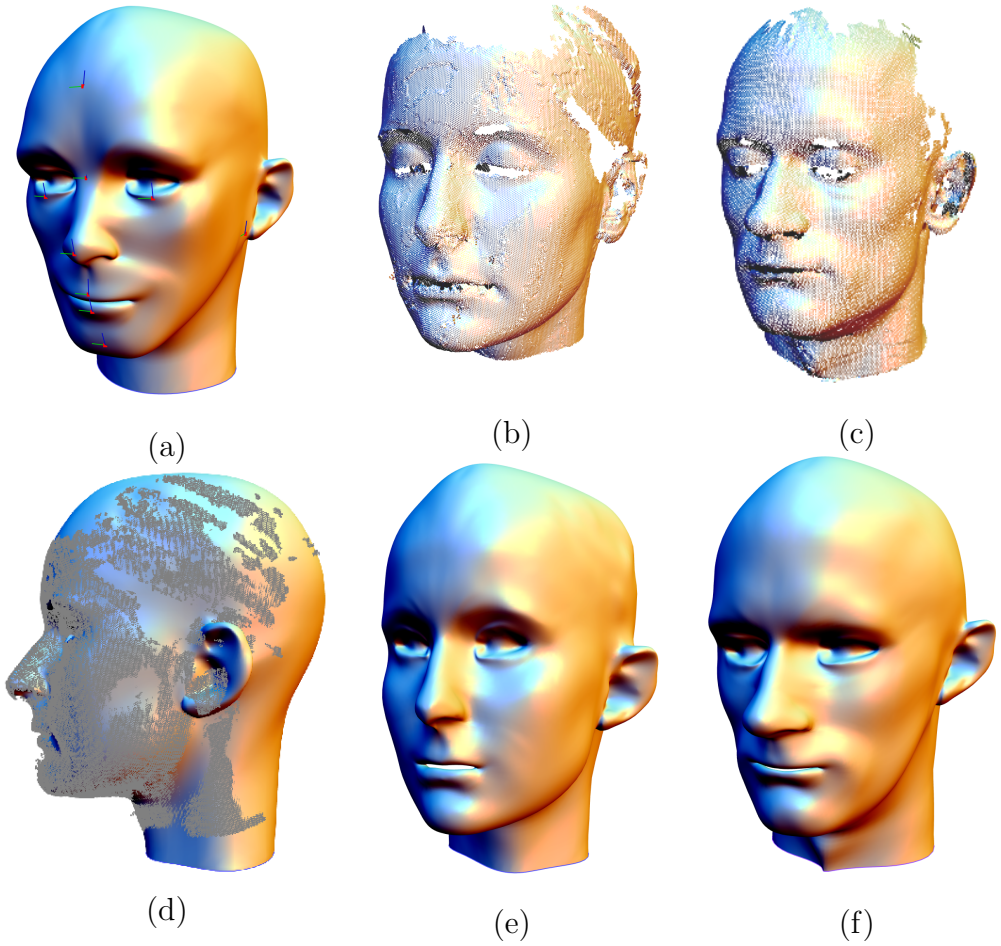
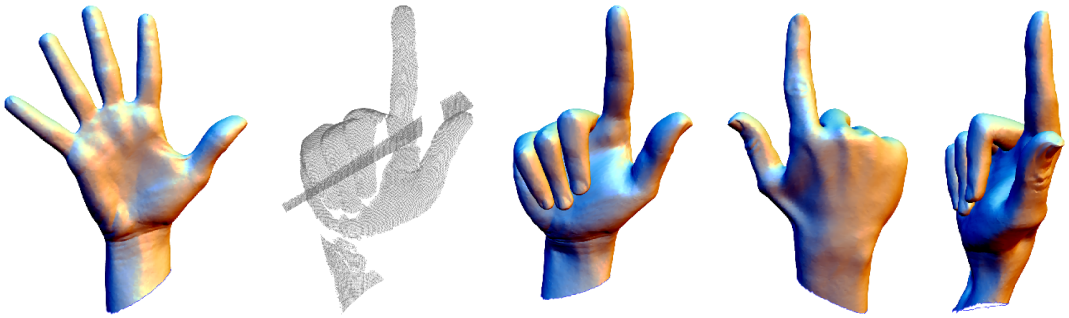


Figure 5.3: Fitting of two heads. (a) Template with 11 markers; (b),(c) data sets; (d), (e) result for (b), the overlay of points illustrates coverage of the data; (f) result for (c).

which consist of a single scan without backside. Here we placed a free marker to constrain the position of the tips of the fingers, which are not visible in the scan. If necessary, the user can the use a brush-interface to mask areas on the model and the point-cloud which should not be processed. The following steps are completely automatic. We generate the initial deformation using our rotation estimation graph (see Chapter 3.2.3) and calculating global scaling. This is followed by the iterative improvement stage and finally interpolating displacements for unmatched vertices, resulting in the final reconstruction. A typical fitting session takes between 5–20 minutes.



(a) (b) (c) (d) (e)

Figure 5.4: A template hand model (a) is fitted to a single range scan (b). The final result is shown in (c)–(e) from different views.

System response is immediate such that the overall timing is mainly influenced by the time the user interaction takes. Computation times such as matrix factorization are negligible; this fact is also known from Laplacian-based interactive shape editing systems (see, e.g., Sorkine (2005)). Computational cost is dominated by matrix operations and thus size of the template (rather than nearest neighbor searches on the point cloud), all automatic stages of iterations were performed in less than 30 seconds for all models.

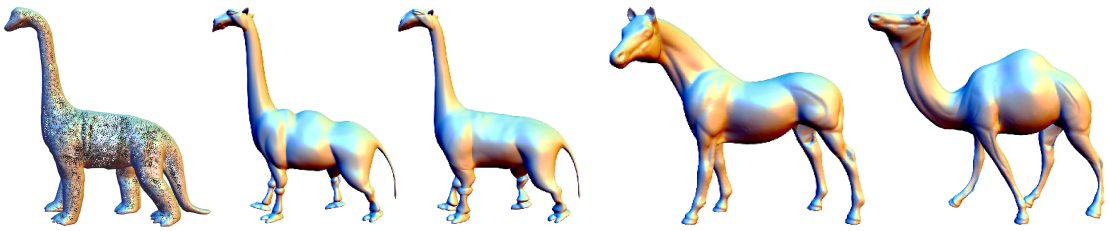
Our method successfully addresses common problems in point cloud reconstruction: noise, outliers, fragmented scans, missing parts. The projection operator and the iterative deformation approach together with smoothness inherited from the Laplacian operator effectively acts as a low-pass filter suppressing artifacts such as high-frequency noise and outliers on the point data. Our method has even proven to be robust toward poorly aligned fragments combined in a point cloud (Figure 5.6).

In all our examples, the scans showed a significant amount of missing data for the scans, e.g., the belly of the camel, the back of the hand/head, and large parts of the human models (see Figure 5.1, 5.3, 5.4, 5.6). These parts are filled with the template shape in a natural way, transitions to such regions are hardly noticeable in the results.

We show that our method is capable of handling extreme deformations (Figure 5.5) to different poses and even shapes. However, depending on the quality of the input and the application it is sometimes not desirable to align each of the small details of the template to the points. Masking out certain parts or focusing on a region of interest enables us to preserve certain details and features of the template model, e.g., the legs in Figure 5.5. For other applications, it is vital to

model	Figures	#marks	#verts	#pts	#err
camel	5.1	30	39024	6195	0.57%
head 1	5.3 (b)	11	16544	52954	0.12%
head 2	5.3 (c)	11	16544	65593	0.16%
hand	5.4	33	25735	114767	0.25%
woman	5.6	26	13784	11798	0.32%
bronto	5.5	41	39024	23982	0.55%
horse	5.5	33	48485	6195	0.33%

Table 5.1: Summary on datasets used. Columns refer number of correspondence markers (# marks), vertices in the template (# verts), points in the point cloud (# pts), and relative approximation error as ratio of one-sided Hausdorff distance from points to results (ROI only) to length of bounding-box diagonal.

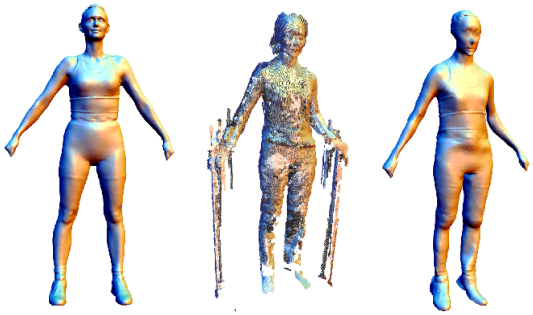


(a) (b) (c) (d) (e)

Figure 5.5: Left: The Brontocamelosaurus is an example of extreme deformation using the camel template (Figure 1(a)). (a) Point cloud; (b) initial deformation; (c) final result. Lower legs, tail and ears were masked out. Right: Use of the horse (d) model as template for the camel point cloud from Figure 1 (b), result shown in (e).

preserve parts of the template that do not match semantically to the point data such as the open and closed eyes in Figure 5.3. Table 5.1 summarizes facts on all of our examples.

One limitation of our method is that we are not able to reconstruct sharp features and small details of the point cloud. This has several reasons. Due to the smoothing required for noise suppression we lose some information. Also, the template may not be of sufficient resolution to capture all details. Finally, since we always regularize our fitting procedure with the initial Laplacian system using Equation 3.12 we try to preserve details of the template at the cost of a more accurate reconstruction. One possible solution to this would consist of extending coating transfer Sorkine *et al.* (2004) to our point cloud settings. Another solution will be presented in the following section.



(a) (b) (c)

Figure 5.6: Reconstruction of a female model from a noisy dataset consisting of two fragments (front and back) which are poorly registered. Note missing parts and artifacts from shadowing in the input data. (a) Template; (b) point cloud, hands and supporting bars were masked out; (c) result.

5.5 Extensions

In this section we want to propose three extensions to the template fitting method proposed before. One of the main limitations of the template fitting method is that it is unable to reconstruct fine details of the point cloud and rather preserves existing detail of the template. Also, if the template shape is too dissimilar to the point cloud to be reconstructed the template might not be able to deform sufficiently to match the object. These problems can be traced back to the way the deformation is handled. The iterative fitting algorithm in section 5.3 regularizes the deformation with the Laplacian matrix $\mathbf{L}$ and the differential coordinates δ of the original template (albeit modified by the rotation-field and the scaling factor). This has the implication that the solution of the linear system will always try to preserve the local detail of the template.

To achieve a greater detail reconstruction of our point cloud we will change the iterative procedure of 5 slightly by adapting the setup of the linear system. We also introduce a method for remeshing, which allows us to modify the mesh connectivity to better accommodate to the point cloud where the resolution is insufficient. Finally we will introduce a method for surface fairing, which can be used to prevent self-intersections when matching the surface. These three techniques allow us to reconstruct point clouds in much higher detail and also allows us to use more general objects for reconstruction (like a low-tessellation sphere for reconstructing the Igea point cloud as in Figure 5.10).

5.5.1 Laplacian updating

The first extension of our method will address the issue of reconstructing finer details of the point cloud. The setup works similarly to the one presented in section 5 up to the iterative refinement step. The user selects correspondence points, rotation and scaling are estimated and an initial deformation of the template mesh is generated, resulting in a triangle-mesh $\mathcal{M}_0$.

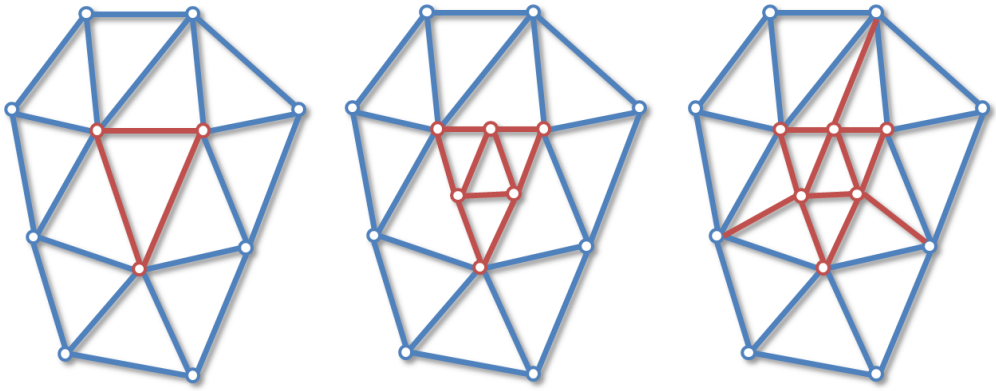
Instead of continuing the fitting procedure with the linear system estimated from the template we now calculate a new Laplacian system at the beginning of each iteration i based on our current deformed template $\mathcal{M}_{i-1}$. We calculate the Laplacian matrix $\mathbf{L}_i$ and the differential coordinates δ_i from $\mathcal{M}_{i-1}$ and build a new linear system as in Equation 3.7. The positional constraints of this system are calculated in the same way as in section 5.3, by finding nearby points that have a similar normal and projecting along the normal. We use a lower weight for those constraints than in the original formulation to compensate for the updating Laplacian system, which allows for a higher range of deformation. Solving the linear system yields a new triangle mesh $\mathcal{M}_{i+1}$.

By iterating this procedure we allow our template to slowly adapt to the fine details of the point cloud. The template mesh acts as a regularization during the Laplacian deformation step, and by slowly changing the template to take into account the deformations we can adapt the mesh much closer to the point cloud geometry.

5.5.2 Remeshing

In this section we will describe how to modify the framework to allow the template to adapt its mesh resolution to the detail of the point cloud by performing an adaptive subdivision.

Following the deformation, we calculate an error term for each triangle of the deformed template by projecting all points of the point cloud onto the mesh and calculating the average distance $\mathbf{d}_i$ of the projected points for each triangle. To ensure that this procedure is not influenced by outliers we disregard all vertices that are further away than a threshold μ. We now select all vertices where $\mathbf{d}_i$ is bigger than a threshold ϵ and subdivide them by simply splitting all edges at the midpoint and replacing the triangle with 4 smaller ones. Following this step we ensure correct triangulation by fixing all resulting T-junctions (see Figure 5.7). This procedure creates finer tessellations for locations where the current triangulation is too coarse to capture the point cloud geometry. Finally we perform



(a) (b) (c)

Figure 5.7: (a) The triangle in red has been marked for subdivision by error thresholding. (b) All marked triangles are split into four smaller triangles by edge splitting. (c) Resulting t-junctions are fixed by inserting edges.

an edge-flipping procedure, inspired by Botsch & Kobbelt (2004). Here we flip edges of the mesh wherever this minimizes the deviation from valence 6 for the surrounding vertices to increase the regularity of the mesh.

While this procedure effectively changes the connectivity of the template mesh, it allows it to adapt the triangle size to the feature size of the point cloud. Regions that can be well approximated by large triangles will keep their resolution, while regions with many details will be represented by smaller triangles.

5.5.3 Surface fairing

Finally, we will explain how to deal with possible self-intersections due to fold-overs that can be produced by the matching procedure.

The new iteration procedure allows for a much wider range of deformations, and as such is prone to self intersections during the projection step (see Figure 5.8). As handling these directly during the projection procedure is difficult we instead use a subsequent fairing step to fix these problems. Similar to Botsch & Kobbelt (2004) we perform an area-based tangential smoothing step. This process moves each vertex towards its area-weighted centroid

$$\mathbf{g}_i = \frac{1}{\sum_{j \in N(i)} A_j} \sum_{j \in N(i)} A_j \mathbf{v}_j, \tag{5.1}$$

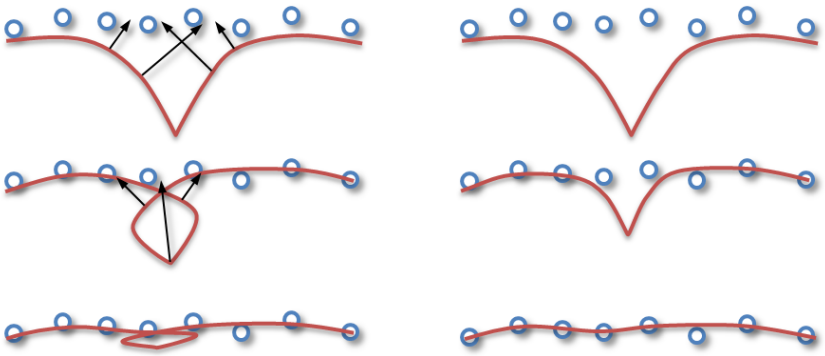


Figure 5.8: The surface mesh (red) is matched to the point cloud (blue). Due to way the matching is performed, under some circumstances self intersections and folding artifacts may be generated (left column). By introducing a surface fairing step which smoothes the mesh along the point cloud surface's tangents these problems can be resolved (right column).

where A_j is the Voronoi area of the vertex as in Equation 3.2. To ensure that we stay on the surface the update vector for each vertex is modified by projecting it onto the tangential plane of the target surface.

$$\mathbf{v}_i = \mathbf{v}_i + \lambda \left(I - \mathbf{n}_i \mathbf{n}_i^\top \right) (\mathbf{g}_i - \mathbf{v}_i) \tag{5.2}$$

Here λ is a dampening factor to avoid oscillations. Instead of using the normal vector of the current deformed template mesh for projection however we us the averaged normal vector $\mathbf{n}_i$ of the closest surrounding point cloud vertices, which ensures that we actually smooth along the tangent of the surface we want to reconstruct and not our current surface (see Figure 5.8).

5.5.4 Results

We have applied our extended processing pipeline to several examples. Figure 5.9 again shows the camel template from Figure 5.1 matched to its respective point cloud using the same constraints, this time though using the Laplacian updating and fairing procedure outlined above. Figure 5.9 shows that the resulting mesh fits much closer to the point cloud and lost all details from the original template in regions where surface matches were found. Also the regions around the head and tail, which were very problematic to handle using the previous method now accurately represent the point cloud's geometry.

(a) (b)

Figure 5.9: Reconstructing the camel scan from Figure 5.1 using a template. (a) Template fitting without Laplacian updating (head and feet were disregarded for matching). (b) Reconstruction with enabled Laplacian updating and surface fairing. Note that the Laplacian updating enables a more detailed reconstruction on the flank and neck of the model.

Figure 5.10 shows the fitting of a low tessellation sphere to the Igea model. Here, the combination of Laplacian updating, remeshing and fairing allows us to create an accurate reconstruction of the head model with sufficient resolution to represent all surface detail present in the mesh.

One drawback of performing the remeshing step is that we lose the original connectivity of our template model, i.e. we no longer have a valid cross-parameterization over all models matched to this template. However, this is a trade-off for achieving a much better surface reconstruction quality in fine details. Depending on the application the user can select which property is more important.

5.6 Discussion

In this chapter we proposed a new approach to surface reconstruction from 3D point clouds that incorporates a template mesh to provide additional semantic information. Approaching shape reconstruction as template fitting process al-

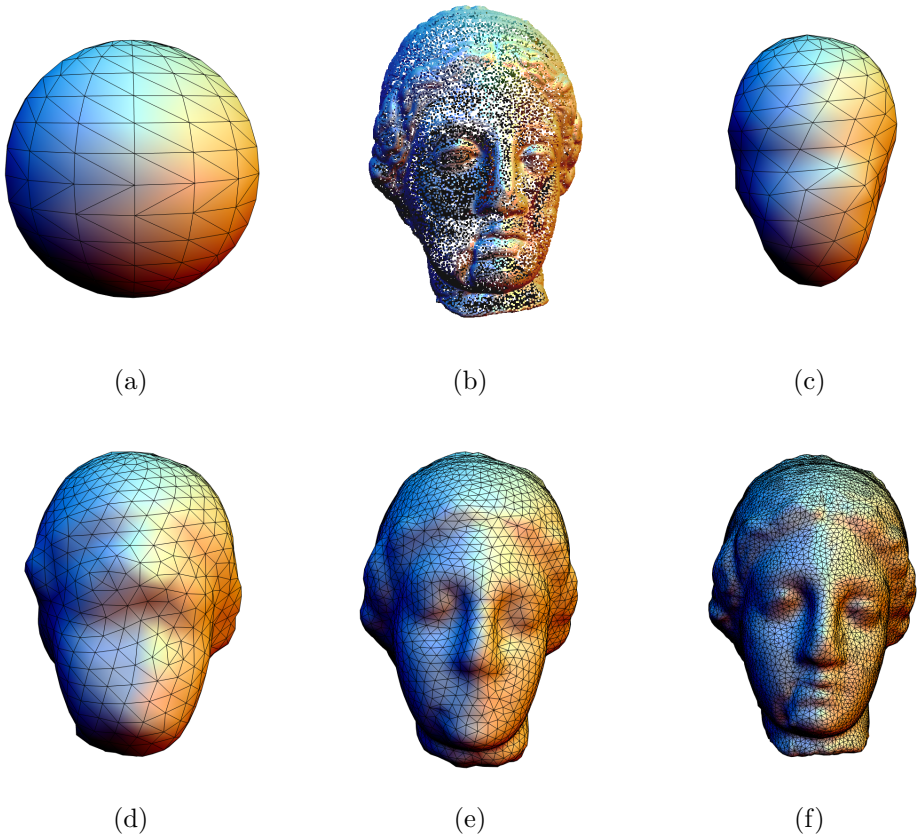


Figure 5.10: Reconstructing the Igea model from a sphere. (a) Low resolution sphere template. (b) Igea model point cloud. (c-f) Several stages of the reconstruction process using Laplacian updating, subdivision and surface fairing to generate an accurate reconstruction.

lows us to reconstruct the surface and semantically fill holes in a single simple framework.

One of the main limitations of the method is that, in contrast to traditional reconstruction approaches, it is not fully automatic. It requires user input in terms of template selection as well as correspondence point selection. The work of Pauly *et al.* (2005) includes an automatic template selection from a database, however they need to manually label correspondence points before. Recent research from Wand *et al.* (2007), Wand *et al.* (2009) and Li *et al.* (2000) use automatic correspondence finding to match partial scans from a sequence of 3D scans. Similar approaches could be added to our method to develop a fully automatic system. Given a templates database we could extract invariant feature

descriptors, for example the Spin-Image descriptor from Johnson (1997), and store them. For the reconstruction, we would extract features from the 3D scan as well and compare them to the entries in the database, finding the most likely shape matches. This would allow us to reconstruct the surface using correspondences that can be established from the features using a robust matching scheme such as RANSAC (Fischler & Bolles (1987)). Finally, we select the reconstruction that provides the minimum reconstruction error or let the user select his preferred result.

We currently are not dealing with sharp features like edges or corners during the reconstruction and will always try to reconstruct a smooth surface. While this can be partially alleviated by applying the template subdivision extension, a more focused approach that detects sharp features in the input data combined with an feature preserving projection operator (like presented in Fleishman *et al.* (2005)) would be the preferable solution.

The linear differential deformation framework presented here is particularly well suited to adapt the mesh to fine details. However, it may produce unintuitive results when the low-frequency deformation (i.e. pose change) becomes too large. While the rotation estimation method introduced in Chapter 3.2.3 can deal with deformations up to a certain degree, the quality of the result is strongly dependent on the distribution of the constraint points. The quaternion interpolation method used there is also not able to deal with rotations larger than π. This limitation can be addressed by replacing the framework with a more powerful, but also more costly, non-linear method, which is able to handle larger deformations more reliably. We will introduce such a framework in Part II of this thesis.

While we concentrated on reconstructions from 3D scans generated with triangulation laser scanners and structured light scanners in this part of the thesis, we want to point out that the method is also applicable to point clouds extracted using image based methods such as multi view stereo or shape from silhouette techniques. We will show how we can incorporate the additional knowledge gained from having image information to shape reconstruction in the following Chapter.

The template based shape reconstruction approach presented in this Chapter is an easy to implement and reliable method to incorporate semantic knowledge into shape reconstruction. Even if large parts of the model are missing, such as in the hand example in Figure 5.4 where the 3D scan consists of a single depth map, we are able to reconstruct a meaningful model. The projection method is robust to noise and outliers and allows reliable reconstruction even of very difficult scans. Our template based approach does not require pre-processing

and high-level tools such as parameterization or mappings between surfaces as previous methods. Additionally, if the same template is used for different data sets we trivially obtain a correspondence map between the deformed templates.

Chapter 6

Surface based animation and performance capture

In this Chapter we will show how we can use template based shape processing for animation and performance capture (de Aguiar *et al.* (2007a,b,c, 2008a); Gall *et al.* (2009)). Traditionally animation and performance capture methods parameterize a model of a subject in terms of a kinematic skeleton. We can then change the joint-angles manually to generate an animation, or estimate them from external input data (marker trajectories or images) for motion capture. Instead of using a kinematic skeleton as underlying scene representation we propose to parameterize a high-quality 3D scan of the performer in terms of constraints of our mesh deformation framework and thus approach animation and motion capture as a deformable template fitting process. Instead of being subject to the limited modeling power of skeleton-based approaches, we can now apply arbitrary surface constraints to our shape. This allows us to not only capture motion, but also surface deformation of our performer, even when wearing wide apparel. The downside of this is that the shape parameterization is much more complex and higher dimensional.

While it is possible to create a shape reconstruction from input images without using any template, this will produce meshes with varying connectivity and topology in every frame. As a constant connectivity is of advantage for processing and storing the mesh, especially for real-time applications, one has to bring the created meshes into correspondence afterwards (Ahmed *et al.* (2008b); Starck & Hilton (2007)). Using a template mesh from the beginning circumvents this step and also can simplify the reconstruction process itself.

On the other hand there are marker-based and marker-less motion capture systems that measure human motion in terms of a kinematic skeleton embedded into a template model (Moeslund *et al.* (2006); Poppe (2007)). While this representation is very compact and has advantages in terms of computational complexity and robustness, it does not allow dealing with situations where a skeleton is not a good approximation of the actual deformation, for example when the performer is wearing a skirt or very wide apparel. In these cases the tracking may even fail completely.

In the following sections we will present methods for template based animation and tracking of the motion and shape of performers in arbitrary apparel (de Aguiar *et al.* (2007a,b,c)). We will also introduce techniques for template refinement that allow us to extract more detailed time-varying detail from input videos (de Aguiar *et al.* (2008a); Gall *et al.* (2009)).

6.1 Data acquisition

The input data for both tracking and refinement were recorded in our multi-view camera studio. The setup consists of $k = 8$ synchronized cameras recording with a resolution of $x \times y = 1004 \times 1004$ pixels. The recorded sequences of a scene contain n frames of video and are denoted as $\mathbf{F}_{k,n}$. We perform a background subtraction for all frames, yielding a set of binary silhouette images $\mathbf{I}_{k,n}$. In the silhouette images 0 denotes the background and 1 the foreground object. For each camera we also reconstruct a 3×4 calibration matrix $\mathbf{C}_k$ which maps 3D world coordinates to 2D images coordinates in the respective camera view. We also acquire a triangle mesh model $\mathcal{M}_{input}$ of the performer using a full body laser scanner.

6.2 Animation and tracking

In this section we will present an overview of methods that allow for animation of template meshes and mesh based marker-less motion capture (de Aguiar *et al.* (2007a,b,c)). Instead of estimating joint angles of a kinematic skeleton, all of these methods rely on tracking 3D positions on the surface of an object and using those to directly deform the mesh. This novel formulation allows us to capture a much wider range of performances than traditional skeleton based techniques.

The vision pipeline used in these methods (i.e. flow calculations, feature extraction and correspondence finding) is explained in detail in the PhD thesis de Aguiar (2008). However, as the template deformation process introduced in this thesis in Chapter 3 is used to parameterize the template models, we will explain the methods only briefly. We refer the reader to de Aguiar (2008) for more details concerning the constraint extraction from the images.

Given the 3D mesh model $\mathcal{M}_{input}$ of the performer and the set of input videos with their segmentations and calibration, we want to generate a mesh animation $\mathcal{M}_{1...n}$ where the mesh is deformed in such a way that the pose and motion is as similar as possible to that of the original performer.

In all three methods we extract 3D deformation constraints for the input mesh from the input videos in different ways and use them to generate a mesh animation $\mathcal{M}_{1...n}$. Given the constraints we use our template processing framework to perform rotation interpolation (see Section 3.2.3) and solve for the deformed state using Equation 3.12 in each frame.

In de Aguiar *et al.* (2007c) we extract deformation constraints for our template mesh from a marker-based system and from the results captured with the marker-less system of Carranza *et al.* (2003). We establish correspondence between the input mesh $\mathcal{M}_{input}$ and the optical marker trajectories/the tracked mesh from the marker-less system. We then apply these points as input to our framework. This allows us to animate 3D laser scans with a simple deformation scheme without having to rely on the complex traditional animation pipeline based on kinematic skeletons.

In both de Aguiar *et al.* (2007a) and de Aguiar *et al.* (2007b) we directly estimate the deformation constraints from the input videos. To achieve this we first have to bring the 3D input model into correspondence with the first frame. This can be achieved by using an input mesh where the pose is already close to the first input video frame and performing an ICP-like alignment, resulting in an initial mesh $\mathcal{M}_1$. Now we need to derive constraints to deform the model from each frame $\mathcal{M}_i$ to the next frame $\mathcal{M}_{i+1}$.

In de Aguiar *et al.* (2007b) we extract constraint points with the help of the optical flow of Brox *et al.* (2004). We first calculate an unconstrained deforming mesh animation by using optical flow to match a projectively textured version of the input mesh to the input images and then simply displacing the vertices according to the estimated 3D scene flow. This process is iterated until the silhouette overlap between mesh and silhouette images is below a threshold ϵ.

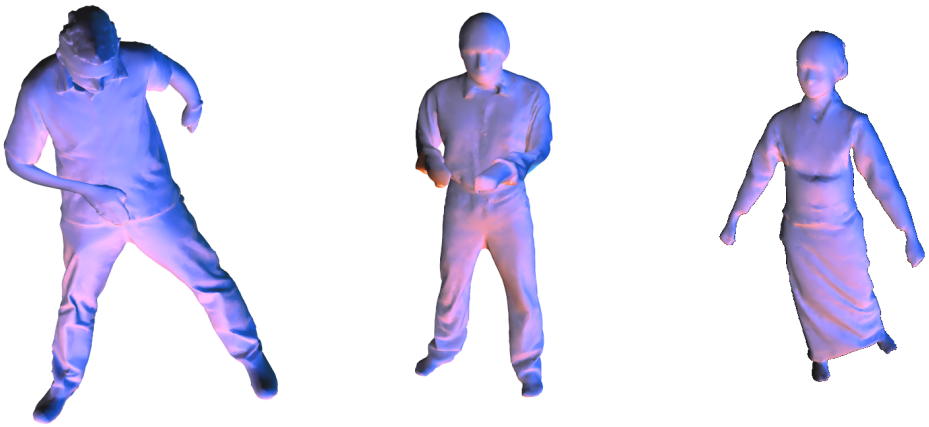
Applying this procedure consecutively to all frames results in a first coarse approximation of the motion. As we do not yet take structural information about the input mesh into account this will often fail to track the object correctly and result in distortions. However, the information can be used to select a subset of mesh vertices that are tracked reliably. We take these feature points and track them again using the optical flow method from before. Instead of simply moving the mesh vertices to the target 3D positions, we use them as constraints for our template deformation framework. This stabilizes the tracking of the mesh, as the differential coordinate solver acts as a regularization on the motion of the mesh, preventing unnatural deformations. In de Aguiar *et al.* (2007a) this work was extended to include SIFT feature extraction (Lowe (1999)) to stabilize the deformation constraint extraction. The tracked local SIFT features are validated with the optical flow. We then apply the confidence-weighted feature trajectories as template deformation constraints to generate the final mesh animation.

6.2.1 Results

We applied the method from de Aguiar *et al.* (2007c) to generate several animations of characters with different input data. Using both marker trajectories extracted from a marker-based and point constraints extracted from the markerless system of Carranza *et al.* (2003) we were able to quickly and intuitively transfer the motion of performers onto models. This allowed us to circumvent the complex process of fitting the skeleton to the marker data and the template model.

We tested the methods presented in de Aguiar *et al.* (2007a) and de Aguiar *et al.* (2007b) on several sequences with different performers wearing a wide range of apparel, from body tight over wide to complex garments like a kimono. Our captured sequences are usually between 100 and 500 frames long and show a wide range of motions from simple walking to yoga exercises.

The results show that our purely passive mesh-based animation and tracking approaches can capture both pose and surface deformation of performers. We are able to track even complex deformations of different materials using the presented frameworks (see Figure 6.1). Most importantly, it is not necessary to embed a kinematic skeleton into the mesh and are able to represent motion only in terms of moving 3D point constraints.



(a) (b) (c)

Figure 6.1: Results from different surface based performance capture methods. (a) Marker based animation transferred to a triangle mesh using de Aguiar *et al.* (2007c). (b) Marker-less motion capture of a performer using optical flow from de Aguiar *et al.* (2007b). (c) Capture of a performer wearing a Kimono from de Aguiar *et al.* (2007a).

6.3 Model refinement

In this section we will show how we can apply our deformation framework to model refinement from images. Unlike the methods presented in the previous section, where we are mainly concerned with finding the low-frequency deformation (i.e. pose) that best matches our input data, we now want to deform the already posed model in each time-step so that it fits the input images as closely as possible also on the finest level of detail (i.e. capture the high-frequency deformations).

In addition to the input videos and segmentations we are also given a mesh animation consisting of n triangle meshes $\mathcal{M}_n$ which have already been tracked (for example using the method presented in the previous section) and roughly capture the pose of the subject in the video.

In the following we will present two different methods for refining our model to match the silhouette in all views, first using projected 3D constraints and then using 2D line constraints. Although the silhouettes only provide reliable information on outer boundaries of the object, the vertices belonging to them are usually evenly distributed on the surface. Hence, in general it is possible to adapt the shape of the whole model also in areas which do not project on image contours realistically. Unless the surface of the object has a complicated shape

with many concavities, the result of silhouette adaptation is already a realistic representation of the correct shape.

Should the target object contain many concavities, it is necessary to employ more sophisticated methods. We will also explain how to use a model guided multi-view stereo approach to match parts of the object which do not lie on silhouette boundaries of the observed image data. This stereo refinement approach can be freely combined with the two silhouette refinement steps if so desired, as all of them rely on our deformation framework and can be combined into a single linear Equation system.

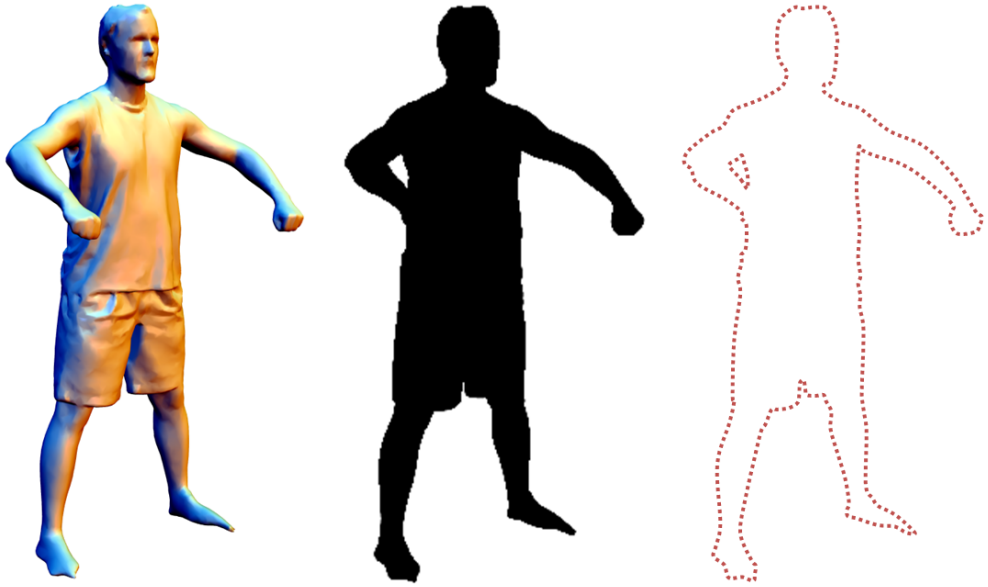
All our refinement calculations will be performed on each frame of the video and animation sequence individually, disregarding temporal coherence. If necessary a temporal smoothing of vertex trajectories can be performed after all frames have been processed, which will remove any noise introduced. While it would be possible to include temporal coherence constraints during processing, this would make the linear Equation systems more complicated and thus increase processing time. In our experiments we could not find any benefit of including implicit temporal coherence in this way over performing a simple explicit smoothing step in the end. Because of this, it is sufficient to approach the problem in terms of processing a single frame only.

6.3.1 Silhouette refinement using positional constraints

The first step for silhouette refinement is to calculate the Laplacian matrix $\mathbf{L}$ and differential coordinates δ of the mesh according to Equation 3.3. We can limit ourselves to the simple Laplacian deformation framework without any rotation estimation as we assume that our model already has the correct low-frequency orientation.

Following this, we identify the rim vertices of our mesh in each camera view. A rim vertex for a camera view is a vertex which lies on the outer silhouette of the 2D projection of the mesh in the respective camera. They can be calculated efficiently by rendering the mesh in a solid color in the camera view and afterwards checking which vertices project exactly onto a border of the rendered image (Figure 6.2).

For each rim vertex the closest 2D point on the silhouette image boundary is found in the camera view that defines its rim status. Now we check if the image gradient at the input silhouette point has a similar orientation to the image gradient in the reprojected model contour image. If this is the case, we back-project the vertex from 2D to 3D world coordinates, keeping the original



(a) (b) (c)

Figure 6.2: Extracting the rim vertices of the mesh : (a) Model in camera view. (b) Solid color rendering. (c) Vertices that lie on the outer silhouette of the mesh.

depth value. This back-projected input contour point defines the target position for the rim vertex. If the distance between back-projection and original position is smaller than threshold ϵ we add it as constraint to our deformation framework in Equation 3.6. Here we use a low weight (between 0.25 and 0.5 depending on the quality of the segmentation) for the rim constraint points. This has a regularizing and dampening effect on the deformation that minimizes implausible shape adaptation in the presence of noise.

After processing all vertices, we solve for the new surface. This rim projection and deformation step is iterated up to 20 times or until silhouette overlap can not be improved further. Note here that we recalculate the Laplacian matrix $\mathbf{L}$ and differential coordinates δ as well as the rim vertices for each iteration. Updating the matrix and coordinates allows us to match the silhouette much closer, similar to section 5.5.1. Updating the rim vertices is necessary as the deformation may cause vertices to change their rim status.

6.3.2 Silhouette refinement using line constraints

In this section we will replace the 3D constrained found by back-projecting the rim vertices with line constraints as introduced in section 3.2.1. Up to the process of back-projecting the vertex the procedure of the algorithm is similar to the one presented in the previous section. To generate the line constraint we first have to split the 3×4 calibration matrix $\mathbf{C}_j$ of a camera j into its translation vector t^j and the remaining 3×3 transformation $\mathbf{N}^j$. Using the 2D target screen space coordinates u_i for a rim vertex i we can now represent the line constraint using the two linear Equations

$$\begin{aligned}(\mathbf{N}_1^j - u_1^i \mathbf{N}_3^j)\mathbf{x}_i &= -t_1^j + u_1^i t_3^j \\ (\mathbf{N}_2^j - u_2^i \mathbf{N}_3^j)\mathbf{x}_i &= -t_2^j + u_2^i t_3^j\end{aligned} \quad . \tag{6.1}$$

Here the subscripts of $\mathbf{N}^j$ and t^j correspond to the respective rows of the matrix/entry of the vector. These Equations force the vertex $\mathbf{x}_i$ to lie somewhere on the ray going through the camera's center of projection and the pixel position u_i. Since the error of this constraint is depth-dependent and thus not linear in the image plane, we weight each constraint such that the error is 1 for a single pixel difference at the original vertex depth.

Similar to refinement with 3D constraints we now solve for the deformed model and iterate the process several times.

6.3.3 Multi-view stereo refinement

In order to recover shape detail of model regions that do not project to silhouette boundaries, such as folds and concavities in a skirt, we resort to photo-consistency information. To serve this purpose, we can derive deformation constraints by applying the multi-view stereo method proposed by Goesele *et al.* (2006). We first estimate a depth map for each image, merge them into a single point cloud and then project nearby vertices of our mesh onto the point cloud, similar to what we proposed in Chapter 5.

To estimate the depth maps for each camera view we first estimate an initial depth for each pixel $\mathbf{d}_i$ by finding the z value of the triangle mesh at the corresponding 2D pixel. We also store the mesh normal $\mathbf{n}_i$. To calculate the photo consistency of a 3D point we first fetch the $l \times l$ pixel back-projection of a small rectangular patch that is tangential to the normal $\mathbf{n}_i$ in all camera views. If the patch is back facing in a camera view it is discarded. For all remaining views, we calculate the normalized cross correlation over all patches. We now find the

optimal depth position by sampling the photo consistency along the viewing ray for each pixel of each depth map in an area ±2 cm away from the initialization. If we can find a distinct maximum in the sampling area, we store the depth, otherwise the pixel is discarded.

As we have far less viewpoints of our subject than Goesele et al. and our actors can wear apparel with little texture, the resulting depth maps are often sparse and noisy. Nonetheless, they provide important additional cues about the object's shape. We merge the depth maps produced by stereo into a single point cloud and then generate deformation constraints for our triangle mesh according to section 5.3. Given the uncertainty in the data, we add the stereo deformation constraints with lower weights.

6.3.4 Results

We have implemented the silhouette and stereo refinement approaches in two research prototypes in the context of performance capture. In de Aguiar *et al.* (2008a) we use a purely mesh-based method for reconstructing spatio-temporally coherent geometry and motion of actors from multi-view video. The method is split into two stages, a coarse pose-capture step and a detail recovery step. The first stage uses a deformable volumetric mesh model to capture the rough pose of the subject without the help of a kinematic skeleton. We will explain this pose capture step in more detail in Part II of this thesis. In the second step, a combination of the presented 3D silhouette refinement and multi-view stereo refinement is used to adapt the models as closely as possible to what can be seen in the input video. This combination allows us to even capture actors performing fast motions and wearing wide apparel or even skirts in high quality. More details about the vision components of this work can also be found in de Aguiar (2008).

Figure 6.3 shows an example of the mesh refinement produced by this method. Subimage (a) shows the initial model and the silhouette constraints, (b) shows the point cloud estimated using the multi-view stereo refinement and (c) the final model after deformation. The mesh now fits much more closely to the original input images than the pure low frequency pose capture result.

In Gall *et al.* (2009) we use a similar coarse-to-fine capture structure as de Aguiar *et al.* (2008a), but use an embedded kinematic skeleton in conjunction with a local/global optimization to estimate the pose of the subject. This is followed by the 2D silhouette refinement as detailed above. The combination of kinematic skeleton and local/global optimization allows for a more reliable tracking of limbs

(a) (b) (c)

Figure 6.3: Model refinement using 3D image constraints and stereo cues : (a) Model with constraints derived from silhouettes overlaid as red lines. (b) Estimated multi-view stereo pointcloud. (c) Model after refinement using the data from (a) and (b).

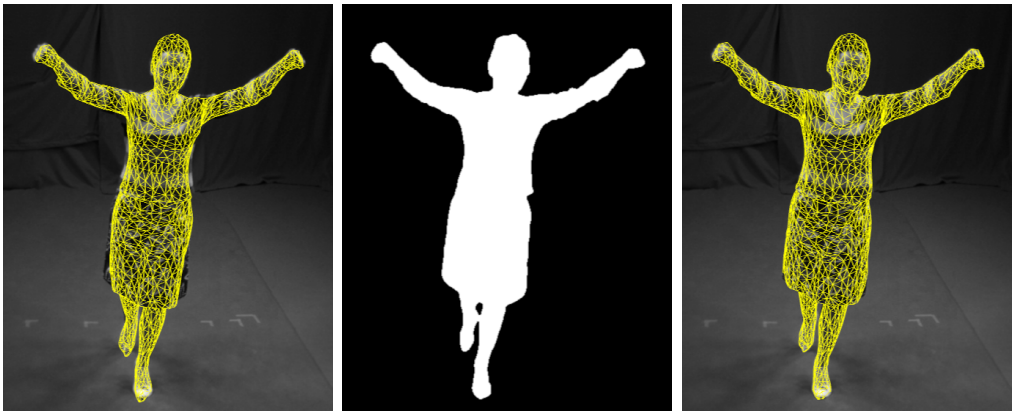
of human actors. On the other hand a kinematic skeleton is not a reliable representation for surface areas whose deformations are only influenced indirectly by the position of limbs, such as a skirt. Because of this the tracking of wide apparel using this method creates slightly less natural results. More details about the kinematic skeleton based marker-less tracking technique can be found in Gall (2009).

Figure 6.4 shows how we adapt the posed model of a performer wearing a skirt using the 2D silhouette constraints to closer match image silhouette, increasing the overlap and accuracy of the reconstruction.

6.4 Discussion

In this Chapter we presented applications of template based shape processing in the context of animation and performance capture. Instead of relying on a kinematic skeleton as template parameterization, we approached both of the applications as deformable template fitting process.

Our work on mesh based performance capture (de Aguiar *et al.* (2007a) and de Aguiar *et al.* (2007b)) illustrates that using a deformable mesh template instead of a traditional kinematic skeleton as underlying scene representation in combination with sophisticated feature point tracking enables us to track complex



(a) (b) (c)

Figure 6.4: Model refinement using 2D image constraints : (a) Input model overlaid on camera image. (b) Segmented silhouette image. (c) Refined model overlaid on camera image. Note the increased accuracy especially on the skirt.

scenes that could not be captured before. We neither require any segmentation of the model into parts, nor explicit specification of material parameters. The methods preserve the mesh's connectivity, which is particularly important when it comes to further processing and storing of the data.

Nonetheless, our approaches are subject to some limitations. Both flow based and feature based tracking are sensitive to the movement speed of the subject (de Aguiar *et al.* (2007a,b)). If the performer moves very quickly, the optical flow that is used in both methods may fail to capture the motion correctly. This can be alleviated by recording with higher frame-rates (i.e. reducing the time-step between frames), however this is not always possible due to hardware restrictions. Similar to the work in the previous Chapter, the deformation framework may produce errors and unnatural deformations if the tracked poses differ too much from the original input model. The rotation interpolation approach can compensate a certain amount of deformation, but will fail if the pose contains rotations larger than π. The surface based approach also does not strive to preserve the local volume of the performer into account, and as such may produce results where the cross-sectional area of a part of the model changes. While this is desirable and even necessary for very non-rigid parts like wide cloth, this effect may also appear in near rigid areas like the head or limbs. Finally, due to the low number of extracted feature points we are not able to capture true shape variation of high-frequency details, such as wrinkles in clothing. While they deform with the

model, they will appear to be *baked* in.

The limitations of sensitivity to large deformations and volume changes can be addressed by using more sophisticated deformation systems, as we will present in Part II of this thesis. Restrictions to movement speed of the actor can be addressed by using more sophisticated feature extraction and tracking algorithms in combination with optimization frameworks. We addressed these issues in de Aguiar *et al.* (2008a) and Gall *et al.* (2009). Both of these methods are two-step approaches and first accurately capture the low-frequency pose of the template, which we then further process using the model refinement approaches presented in Section 6.3. As the template pose is already accurately captured when performing the model refinement, the template deformation method is able to accurately deal with the constraints extracted from silhouettes and stereo cues and produce detailed and natural looking results.

While we use a very simple silhouette matching approach that may produce erroneous deformations in case the topological structure of the input silhouette is too different from the reprojected model silhouette, we never encountered this issue in any of our test scenes. There still is a resolution limit to our deformation capture. Some of the high-frequency detail in our final result, such as fine wrinkles in clothing or details of the face, has been part of the laser scan in the first place. The deformation on this level of detail is not actually captured, but it is baked in to the deforming surface. Consequently, in some isolated frames small local differences in the shape details between ground-truth video footage and our deformed mesh may be observed, in particular if the deformed mesh pose deviates very strongly from the scanned pose. This could be partially alleviated by using image based fold extraction methods like presented in Popa *et al.* (2009) and Ahmed *et al.* (2008a). Another solution to this detail limit will be presented in Part III of this thesis, where we will learn an accurate physical cloth model of the performers apparel to reproduce its behavior.

Both tracking and refinement approaches preserve the topology of the input model over the whole sequence. For this reason, we are not able to track surfaces which arbitrarily change apparent topology over time (e.g. the movement of hair or deep folds with self-collisions). We also do not correct for self-intersections in the output of our capture approach.

Despite the existing limitations, our methods are flexible, easy to implement and reliably capture time-varying shape of a performer in arbitrary cloth with a passive approach. They are the first systems in the literature that can capture arbitrarily deforming meshes in a connectivity preserving way for such complex

scenes. We produce a novel dense and feature-rich output format comprising of spatio-temporally coherent high-quality geometry, and accurate motion data. We thus supplement and exceed the capabilities of marker-based optical capturing systems that are widely used in industry, and can provide animators and CG artists with a new level of flexibility in acquiring and modifying real-world content.

Part II

Differential coordinate based shape processing using volumetric data

In this part of the thesis we will present a method for shape processing using volumetric template meshes. First, we show how to use volumetric template models for shape editing and deformation, Chapter 8 (Stoll *et al.* (2007)). Using our simple iterative deformation scheme we can interactively produce natural deformations of the template under user-defined constraints. Following this, we highlight the advantages of using our non-linear deformable template fitting process over a kinematic skeleton for performance capture, Chapter 9 (de Aguiar *et al.* (2008a); Stoll *et al.* (2007)). It allows us to animate models directly using marker trajectories from motion capture system without having to perform extensive pre-processing to fit a skeleton to the data. We can also use it to reliably track the motion of performers from multi-view video by directly tracking feature points on the surface, allowing us to capture the motion of actors even when wearing wide apparel. To achieve this, we use an efficient iterative method to perform volumetric shape-deformation based on differential coordinates, which we will introduce in Chapter 7. This more sophisticated deformation framework overcomes some of the limitations inherent to the linear method presented in part I of this thesis.

Chapter 7

A deformation framework for tetrahedral meshes

In this Chapter we will introduce a method for non-linear deformation of a volumetric template shape. This method produces more natural and accurate deformations than the linear framework from Part I. This will enable us to perform deformable template fitting for a larger variety of input data, especially when handling large global pose changes.

The biggest drawback of the template shape deformation framework presented in Chapter 3 is its translational insensitivity and the fact that it does not preserve local volume. We either have to limit ourselves to small deformations, as for example during model refinement presented in Chapter 6.3, or use explicit support structures such as the shape skeleton introduced in Chapter 3.2.3. The preferable solution is to have a framework that is able to handle rotations implicitly, allowing for a simpler and more versatile framework.

The goal of implicit rotation handling implies the necessity of performing non-linear operations during processing. Optimizing non-linear energies is computationally expensive and often only reasonable in non-interactive applications. However, using a tetrahedral mesh and linear differential coordinates defined on them we can create a simple non-linear setup that is not very different from the linear case presented in Chapter 3. While finding the energy minimum is expensive, our iterative setup already produces natural and plausible deformations even after a few iterations. In this framework, instead of solving a single linear system to find the deformed mesh, we will perform several iterations of the linear deformation process, each reducing the amount of unwanted distortion introduced and improving the rotational invariance. Additionally, using a volumetric representa-

tion helps preserving the local volume of the shape, especially the cross-sectional areas of parts. In the following, we will first introduce differential coordinates for a tetrahedral mesh $\mathcal{T}$ and then show how we can use an iterative process to find an as-rigid-as-possible deformation given the set of constraints $\mathbf{C}$.

Our work is closely related to the works of Au *et al.* (2006) and Sorkine & Alexa (2007), who also use a linear Laplacian framework in combination with an iterative update procedure. Both methods rely on a triangle mesh as representation and as such do not have the local volume preserving property our formulation provides. The work of Au *et al.* (2006) optimizes the differential coordinates of the dual of the input mesh by performing a linear deformation step and afterwards rescaling the differential coordinates of the resulting mesh to their original length. This process is iterated until convergence, after which the mesh is transfered back to the primal domain. Sorkine & Alexa (2007) iterates a linear differential deformation step with a rotation extraction step based on the one-ring neighborhood of the triangle vertices. This concept is similar to ours and was developed independently and concurrently with our framework.

7.1 Differential representation

Linear differential coordinates for tetrahedral meshes can be described in a similar way to the differential coordinates of triangle meshes. We can construct a linear equation system

$$\mathbf{L}\mathbf{x} = \delta \tag{7.1}$$

where each δ_i can be constructed as a weighted sum of a vertex with its one ring neighborhood

$$\delta_i = \sum_{j \in N(i)} w_{ij} \left(v_j - v_i \right). \tag{7.2}$$

However, there exists an alternative construction method for the Laplacian matrix $\mathbf{L}$ which will allow us to simplify the later iterative steps. The Laplace operator is also defined as the divergence of the gradient of a scalar field. We can construct this operator in the tetrahedral setting by calculating a gradient operator matrix $\mathbf{G}_j$ for each tetrahedron $\mathbf{e}_j$ which contains the gradients of the tetrahedron's shape functions ϕ_i

$$\begin{aligned}\mathbf{G}_j &= (\nabla\phi_1, \ldots, \nabla\phi_4) \\ &= \begin{pmatrix} (\mathbf{v}_1 - \mathbf{v}_4)^T \\ (\mathbf{v}_2 - \mathbf{v}_4)^T \\ (\mathbf{v}_3 - \mathbf{v}_4)^T \end{pmatrix}^{-1} \begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix}\end{aligned}$$

Here $\mathbf{v}_i$ are the four corner vertices of the tetrahedron. The matrices $\mathbf{G}_j$ can be combined into a large $4m \times n$ gradient operator matrix $\mathbf{G}$, which can be used to construct the Laplacian matrix as

$$\mathbf{L} = \mathbf{G}^T\mathbf{D}\mathbf{G} \ , \tag{7.3}$$

where $\mathbf{D}$ is a $4m \times 4m$ diagonal matrix containing the tetrahedra's volumes. In this construction $\mathbf{G}^T\mathbf{D}$ is representing the discrete divergence operator and $\mathbf{G}$ are the gradients of the mesh.

Using this constructions, the differential coordinates δ for the initial mesh can be calculated explicitly using Equation 7.1 or, more importantly, based on the tetrahedra's gradients as in Poisson surface editing Yu *et al.* (2004) as

$$\delta = \mathbf{G}^T\mathbf{D}\mathbf{g} \ . \tag{7.4}$$

Here, $\mathbf{g}$ is the set of tetrahedron gradients, each being calculated as $\mathbf{g}_j = \mathbf{G}_j\mathbf{v}_j$. More details on the construction of the Laplacian from gradients can be found in Botsch *et al.* (2006b).

Similar to the triangle mesh case, the resulting matrix $\mathbf{L}$ is rank deficient, as the differential coordinates for tetrahedral meshes are also translational invariant. This linear construction still suffers from rotational variance as well. However, with this new gradient based reconstruction we can apply a set of tetrahedral transformations $\mathbf{T}$ to the differential coordinate reconstruction : $\delta = \mathbf{G}^T\mathbf{D}\mathbf{T}\mathbf{g}$. The transformation $\mathbf{T}$ can be different for each tetrahedron (in a sense "exploding" the mesh into separate parts). If we now solve the linear equation system and recover the global vertex positions we effectively solve for a connected configuration where each deformed tetrahedrons gradient $\mathbf{g}'$ is as similar as possible to the transformed gradients $\mathbf{T}\mathbf{g}$. Additionally, the construction of the Laplacian matrix $\mathbf{L}$ in Equation 7.3 is invariant under rigid transformations of the tetrahedra, meaning that the matrix will stay constant if we limit ourselves to rotations. This properties allow us to implicitly handle rotational effects in an iterative solver which we will now explain in more detail.

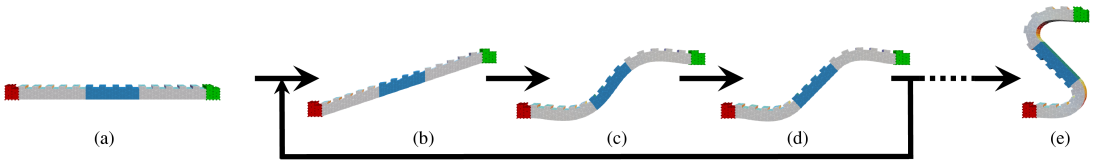


Figure 7.1: Overview of our tetrahedral mesh deformation pipeline - (a) shows the input model with handles in red and green and a rigid section marked in blue. (b),(c) and (d) show the output of the individual iterative processing steps, namely linear Laplacian deformation (b), rotation extraction/interpolation (c), and rigid section handling (d). (e) shows the final deformed model at the end of the user interaction.

7.2 Iterative mesh deformation

We solve the following problem : Given a set of constraints $\mathbf{Cx} = \mathbf{q}$ for a tetrahedral mesh $\mathcal{T}$ we want to find a target configuration of the mesh where the constraints are satisfied and each tetrahedron $\mathbf{e}_j$ deforms as rigidly as possible. We achieve this using an iterative approach comprising of several algorithmic steps that are iterated, Figure 7.1. The individual steps of the method are as follows (processing steps that are optional are printed in italics):

(I) Linear Laplacian deformation

(II) Rotation extraction

(III) *Rotation interpolation*

(IV) *Rigid region handling*

(V) Differential coordinate update

In the following, we explain the the pipeline in more detail.

7.2.1 Iterative processing

The first step in each iteration is a basic linear Laplacian deformation. Since this linear step does not properly handle rotations, we perform an analysis of the linear results to extract for each tetrahedron j an estimate of its rotational transformation $\mathbf{R}_j$. If necessary, user-defined constraint rotations are interpolated across the mesh and added up to the previously extracted per-tetrahedron

rotations $\mathbf{R}_j$. Subsequently, we make sure that all rigid mesh regions rotate in the same way by assigning to each rigid group of tetrahedrons its averaged rotational transformation component. Finally, the resulting $\mathbf{R}_j$ are used to construct an updated set of differential coordinates.

By iterating these steps, we achieve natural and plausible looking deformations of our object. Repeating the iterations minimizes the amount of non-rigid deformation each tetrahedral element undergoes, leading to an as-rigid-as-possible deformation result. The remaining non-rigid energy $E_{nonrigid}$ present in the mesh can be measured from the non-rotational transformation $\mathbf{S}_j$ each tetrahedron undergoes :

$$E_{nonrigid} = \sum_{j=1...n} \|\mathbf{S}_j - I\|_F \tag{7.5}$$

In the following, we detail the individual deformation steps.

(I) Linear Laplacian deformation :

The linear deformation follows the setting outlined in Section 7.1. The matrix $\mathbf{L}$ of Eq. (7.1) can be constructed following Eq. (7.3). Accordingly, the differential coordinates can be computed based on Eq. (7.4).

Constraints are factorized into the matrix $\mathbf{L}$ by eliminating the corresponding row and column in the matrix and incorporating the values into the right hand side δ. The deformation can then be generated by solving the reduced Eq. (7.3). This step is very similar to solving the deformation for the triangle mesh case as detailed in Part I of this thesis.

The resulting deformed mesh $\mathcal{T}'$ only looks natural for very small deformations (see Figure 7.1), as the basic Laplacian setting does not induce any kind of rotation or scaling. The deformed mesh $\mathcal{T}'$ can be computed efficiently with the help of a Cholesky factorization of the left hand side matrix (Golub & Loan (1996)).

(II) Rotation extraction :

We analyze the output of the linear Laplacian deformation in order to extract for each tetrahedron $\mathbf{e}_j$ an estimate of a rotational transformation. These per-tetrahedron rotations are eventually used to update the differential coordinates for the linear deformation.

To put this idea into practice, we compare the deformed mesh $\mathcal{T}'$ to the original mesh $\mathcal{T}$ in its rest pose. By comparing the positions of the vertices $\mathbf{v}_i$ of a single tetrahedron $\mathbf{e}_j$ in $\mathcal{T}$ to their new positions $\mathbf{v}'_i$ in $\mathcal{T}'$, we can calculate a 3×3 transformation matrix $\mathbf{T}_j$ as

$$\mathbf{T}_j = \begin{pmatrix} (\mathbf{v}_1 - \mathbf{v}_4)^T \\ (\mathbf{v}_2 - \mathbf{v}_4)^T \\ (\mathbf{v}_3 - \mathbf{v}_4)^T \end{pmatrix}^{-1} \begin{pmatrix} (\mathbf{v}'_1 - \mathbf{v}'_4)^T \\ (\mathbf{v}'_2 - \mathbf{v}'_4)^T \\ (\mathbf{v}'_3 - \mathbf{v}'_4)^T \end{pmatrix} \tag{7.6}$$

such that $(\mathbf{v}_i - \mathbf{v}_4)\mathbf{T}_j = (\mathbf{v}'_i - \mathbf{v}'_4)$ for $i = 1, \ldots, 3$.

The matrix $\mathbf{T}_j$ usually contains anisotropic scaling (stretching) and shearing components and does not represent a rigid transformation as we would desire. As shown by Shoemake & Duff (1992), $\mathbf{T}_j$ can thus be factored into an orthonormal rotation matrix $\mathbf{R}_j$ and a non-rotational part $\mathbf{S}_j$,

$$\mathbf{T}_j = \mathbf{R}_j \mathbf{S}_j \ . \tag{7.7}$$

Technically, the rotational component can be computed by iteratively averaging $\mathbf{T}_j$ with its inverse transpose as

$$\mathbf{H}_0 = \mathbf{T}_j \ , \ \mathbf{H}_{k+1} = \frac{1}{2}(\mathbf{H}_k + \mathbf{H}_k{}^{-T}) \ . \tag{7.8}$$

The decomposition iterates up to a step k after which $\mathbf{H}_k$ does not change anymore, and then sets $\mathbf{R}_j = \mathbf{H}_k$. In our experiments it was sufficient to perform two computation steps and afterwards normalizing the matrices row vectors to achieve a decent separation. Note that in our three-dimensional setting we also have to make sure that $\mathbf{R}_j$ does not contain a mirroring component to keep tetrahedra from inverting. To serve this purpose, we check the determinant and multiply the matrix by -1 if necessary.

(III) Rotation interpolation :

If a constraint region is rotated and consists of one or more complete tetrahedra in which all vertices were selected we can directly apply these rotations to the mesh to increase convergence speed. While our method adapts very well to translational constraints, rotational constraints (especially twisting) may require a higher number of iterations to propagate through the mesh (see Figure 7.2). We can observe that the limit of the deformation under rotational constraints is usually very similar to the deformation that would have been generated by using

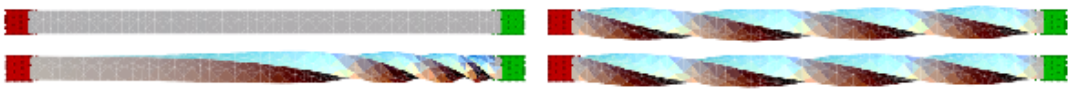


Figure 7.2: Behavior of our deformation technique when a 360° twist is applied to the green handle. Top left: Original model of a bar; Bottom left: result after applying the twist without rotation interpolation; Top right: deformation without rotation interpolation converged after several seconds; Bottom right: instantaneous result *with* rotation interpolation.

a Poisson mesh editing technique where rotations are interpolated between the constraints. Thus, it is only natural to combine the two methods.

We decided to adapt the approach by Zayer *et al.* (2005), which uses a harmonic interpolation of quaternions across the mesh, to our purpose. One major disadvantage of quaternion-based rotation interpolation is its inability to properly reproduce rotations by more than 180 degrees. We solve this problem by interpolating tetrahedron rotations that are relative to the current estimates $\mathbf{R}_j$ instead of absolute rotations with respect to the rest state. These relative rotations are usually very subtle and can thus be easily interpolated with the harmonic interpolation scheme. Technically, the interpolated quaternions are simply added to the current frames rotation matrices $\mathbf{R}_j$.

(IV) Rigid region handling :

Adapting our method to handle rigid parts in the mesh is a straightforward task. For every rigid set of tetrahedra $\mathbf{T}_{rigid} \subseteq \mathbf{T}_{tet}$ we can calculate an average rotation from the $\mathbf{R}_l$ of all the $\mathbf{t}_l \in \mathbf{T}_{rigid}$. Each such $\mathbf{t}_l$ is then assigned this fixed average rotation. By this means, we remove the ability to bend from the set $\mathbf{T}_{rigid}$ while the averaging still allows the set to orient itself in such a way that the deformation's shear is minimized, Figure 7.1. This is similar to replacing the selected region with a single element for deformation, but more efficient as we do not have to change the connectivity of the tetrahedral mesh.

(V) Differential coordinate update :

Now that we have calculated rotations $\mathbf{R}_j$ for each tetrahedron it is straightforward to use them to calculate a new set of differential coordinates. We apply

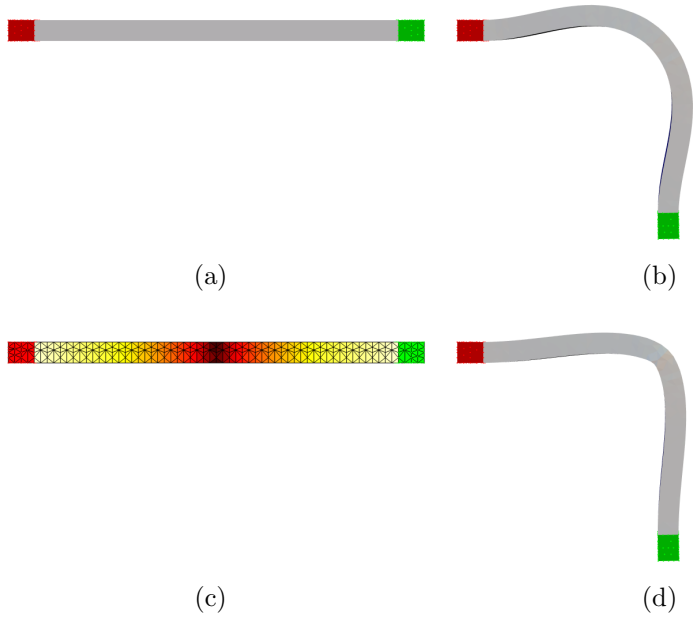


Figure 7.3: Influencing the stiffness of the model. The green handle on the bar in (a) is rotated downwards by 90 degrees. If we specify constant stiffness the deformation will distribute evenly across the whole bar (b). If we specify non-uniform stiffness as in (c), where darker colors represent decreasing stiffness, the bending will be concentrated on the middle of the bar as in (d).

the rotations to their respective tetrahedra $\mathbf{e}_j$ from the original mesh and calculate a new set of transformed tetrahedron gradients $\mathbf{g}_j$, from which the new δ can be calculated using Eq. (7.4). This is similar to the "exploding" of the mesh in Sumner & Popovic (2004).

7.2.2 Controlling deformation behavior

The framework as presented so far does not allow for a closer specification of how the parts of our model are allowed to deform. For some applications it might be necessary to be able to specify stiffness of certain parts. Some parts may have to be completely rigid (if we are deforming a human model we want bending to only occur near skeletal joints) or of varying stiffness (for example when we want to emulate an object made out of different materials). In the following, we will present two simple extensions to the iterative framework which are able to handle both cases in a simple way.

The iterative setup presented also allows us to influence the stiffness of the material in a much more subtle manner than simply deciding between totally rigid or non-rigid. We can introduce a stiffness weight **s** for every tetrahedron, which controls the allowed amount of distortion per element. These weights are used to modify equations 7.3 and 7.4 :

$$\mathbf{L} = \mathbf{G}^T\mathbf{DSG} \ , \tag{7.9}$$

$$\delta = \mathbf{G}^T\mathbf{DSg} \ , \tag{7.10}$$

where **S** is now a diagonal matrix containing the per tetrahedra specified stiffness. This modification downweights the error introduced by non-rigidly deforming low stiffness elements and thus allows us to control the deformation behavior non-uniformly. An example of this is illustrated in Figure 7.3, where we specify a lower stiffness for the elements in the center of the bar. This leads to a concentration of the deformation in the middle of the object. Note that these are no physically based stiffness parameters.

7.2.3 Constraint refinement

For some applications it may be necessary to place constraint points at positions which are not occupied by vertices of the tetrahedral mesh. This can be achieved by simply modifying the structure of the tetrahedral mesh to accommodate for new vertices at the position of the constraints. For internal vertices we simply find the enclosing tetrahedra and subdivide it into four new tetrahedra by adding the constraint point to the mesh. For external points we find the closest surface triangles to the point and create new "virtual" tetrahedra connecting the surface triangles to the constraint point. These virtual tetrahedra are not displayed to the user and used purely for processing needs.

7.3 Processing high resolution meshes

Despite its advantageous properties, our tetrahedral method is not suitable for deformation of very large meshes as processing times may become prohibitively long. Triangle meshes of moderate size can be tetrahedralized such that the surface triangulation is preserved and thus a final pose transfer is not required. To

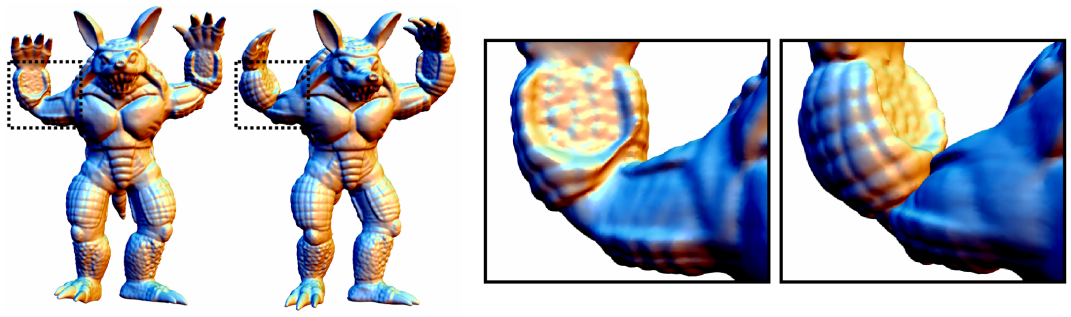


Figure 7.4: Comparison of a real-time deformation result obtained by manipulating a decimated tetrahedral Armadillo and subsequent pose transfer (left image in each pair), and the result of a full-resolution tetrahedral mesh deformation using the same constraints which took several minutes to calculate. The full-resolution deformation leads to an unnatural pose of the arm due to a local minimum.

deform meshes of very high resolution, we first apply a quadric error mesh simplification method (Garland & Heckbert (1997)) to $\mathcal{M}_{input}$ and tetrahedralize the low resolution mesh afterwards using a face-constrained Delaunay tetrahedralization (Si & Gaertner (2005)). We then deform the low resolution tetrahedral mesh and finally transfer the deformation back to the high resolution input mesh.

We would like to point out that this is not a principle disadvantage since it is our primary goal to modify the low-frequency components of a model which contain information about its pose. Working on a low resolution mesh is equivalent to performing a low-pass filtering of our geometry and adding the high frequency detail back after the low frequency components have been modified. An added advantage of this strategy is that the energy function we minimize is smoother on the lower resolution mesh and thus it is less likely that we run into a local minimum which is more likely to happen if we work on the high-resolution mesh straight away, Figure 7.4.

A fast and easy-to-implement approach to pose transfer generates for each vertex $\mathbf{v}_i$ of $\mathcal{M}_{input}$ a set of barycentric coordinates with respect to the closest tetrahedron of $\mathcal{T}$ in the rest pose. By applying those coordinates to the deformed tetrahedral mesh, the deformed fine mesh $\mathcal{M}'_{input}$ is reconstructed. Although this approach is fast it has several drawbacks, the main one being that the reconstruction is only piecewise linear and thus artifacts like dents and self intersections may appear in the reconstructed mesh.

Mean value coordinates Ju *et al.* (2005) provide an alternative that circumvents these problems. Here, a set of coefficients $\mathbf{c}_i = \mathbf{c}_i^0 \dots \mathbf{c}_i^m$ is computed for

each vertex $\mathbf{v}_i$ such that ${\mathbf{c}_i}^T\mathbf{V}_{tet} = \mathbf{v}_i$, i.e. each point of the input mesh is represented as a linear combination of all the points of $\mathcal{T}$. Mean value coordinates are $\mathcal{C}^2$-smooth and enable plausible pose transfer to $\mathcal{M}_{input}$ as shown in a similar context in Huang *et al.* (2006). Unfortunately, they are very memory consuming and mesh reconstruction takes significantly longer.

Yoshizawa *et al.* (2003) use displaced subdivision surfaces Lee *et al.* (2000) to transfer their deformation from low resolution mesh to a high resolution input. But this method can only construct a remeshed version of the input mesh with the connectivity of the subdivision surface created from the low resolution version, and therefore it is not suitable for our purpose.

We propose a method that combines the advantages of the first two methods mentioned above. Our approach represents vertices of the input mesh as linear combinations of tetrahedra in the local neighborhood. By selecting the coefficients carefully, we can achieve a smooth deformation transfer with a method that is far less memory consuming and computationally more efficient than mean value coordinates.

To put this into practice, we find for each vertex $\mathbf{v}_i$ in $\mathcal{M}_{input}$ the set $\mathbf{T}_r(\mathbf{v}_i)$ of all tetrahedra that lie within a local spherical neighborhood of radius r (in all our cases r was set to 5% of the mesh's bounding box diagonal). Subsequently, we calculate the barycentric coordinate coefficients $\mathbf{c}_i(j)$ of the vertex with respect to all $\mathbf{t}_j \in \mathbf{T}_r(\mathbf{v}_i)$ and compute the combined coefficient vector $\mathbf{c}_i$ as

$$\mathbf{c}_i = \frac{\sum_{\mathbf{t}_j \in \mathbf{T}_r(\mathbf{v}_i)} \mathbf{c}_i(k)\phi(\mathbf{v}_i, \mathbf{t}_j)}{\sum_{\mathbf{t}_j \in \mathbf{T}_r(\mathbf{v}_i)} \phi(\mathbf{v}_i, \mathbf{t}_j)} \ . \tag{7.11}$$

$\phi(\mathbf{v}_i, \mathbf{t}_j)$ is the Wendland weighting function with respect to the distance of $\mathbf{v}_i$ to the barycenter of tetrahedron $\mathbf{t}_j$

$$\phi(\mathbf{v}_i, \mathbf{t}_j) = \begin{cases} 0 & \text{if } d > r \\ (1 - \frac{d}{r})^4(\frac{4d}{r} + 1) & \text{if } d \leq r \end{cases} \tag{7.12}$$
$$\text{with } d = \|\mathbf{v}_i - center(\mathbf{t}_j)\| \ .$$

The coefficients for all vertices of $\mathcal{M}_{input}$ are combined into a single sparse transfer matrix $\mathbf{B}$. Thanks to the smooth partition of unity definition and the local support of our parametrization, we can quickly compute a smooth and natural looking transformed input $\mathcal{M}'_{input}$ at moderate memory costs by calculating new vertex positions as

$$\mathcal{V}'_{input} = \mathcal{V}_{input}\mathbf{B} \tag{7.13}$$

Chapter 8

Shape editing

In this Chapter we will show that we can view shape editing as a deformable template fitting process based on a volumetric template in combination with the deformation framework described before (Stoll *et al.* (2007)). In recent years, interactive shape deformation and editing has been a very active field of research. The goal is the development of algorithms that enable shape deformation in an intuitive and, more importantly, natural looking way under a given set of constraints. This usually means that the deformation of the given shape must be calculated in a physically plausible manner, i.e. it must satisfy the expectations the user has due to his experience with deforming objects in the real world. Correct physical simulation requires setting up and minimizing complex non-linear energies, which is computationally expensive and thus often only reasonable in non-interactive applications. Since users will usually not be happy with the first deformation they produce and will iteratively modify the deformation constraints until they are satisfied, offline methods are not feasible for shape editing. To enable immediate feedback to the user and reach interactive editing performance it is necessary to use simpler and easy-to-compute deformation techniques that still behave plausibly (see Chapter 2.3 for an overview of linear and non-linear shape editing approaches).

We can view shape editing as a template fitting process where the fitting data is provided by the user in the form of handle transformations. Using the volumetric template deformation framework presented in the previous Chapter we have a powerful tool for interactive shape editing at hand. While it is essentially a non-linear deformation approach, minimizing the remaining non-rigid energy contained in the object (see Equation 7.5), each iteration can be computed very

Figure 8.1: Example of a deformation of a raptor created with our method. From left to right: Reduced tetrahedral model with handles in red, original high resolution input model, deformed input model.

efficiently and produces a valid and plausible deformation result. Directly visualizing the deformation produced after each iteration allows the user to quickly evaluate the deformation and modify the constraints. In combination with the deformation transfer method from Chapter 7.3 this allows us to process high-resolution meshes efficiently.

8.1 Interactive mesh editing

We have implemented the volumetric deformation framework presented in chapter 7 into an interactive shape editing tool. The user can load a triangle mesh, for which we automatically generate a lower resolution tetrahedral mesh using quadric error decimation (Garland & Heckbert (1997)) and then building a face-constrained Delaunay tetrahedralization (Si & Gaertner (2005)). We then calculate the deformation transfer matrix $\mathbf{B}$ detailed in section 7.3.

After all input data is properly set up, the user defines control handles by marking parts of $\mathcal{T}$. On the one hand, handles can be complete regions of the tetrahedral mesh. In this case, rotational constraints are automatically imposed as the orientation for the handle is fixed by the user. On the other hand, handles can be single vertices, which leaves the orientation of the surrounding tetrahedra free to be determined by the deformation method.

Once all handles are set, the user can optionally specify rigid regions (see Chapter 7.2.1(IV)). In this step, sets of tetrahedra that ought to maintain a constant relative orientation can be marked. Conveniently, our system does not impose any constraints on the size, shape or topology of rigid regions. It is even possible to mark disconnected parts as belonging to the same rigid part. This

allows us to quickly specify a kind of *pseudo*-skeleton without having to define joints or a bone hierarchy.

Following this, the user can interactively move/rotate the deformation handles. During this process the user gets immediate feedback of the deformation, as the iterative solver detailed in section 7.2 is running in the background and the current deformation is visualized after each iteration. Once the user is satisfied with the deformation the software transfers the deformation of the tetrahedral mesh back to the high resolution triangle mesh for inspection.

8.2 Results

Our system was used to generate deformations of several high resolution meshes, as shown in Figures 8.1 and 8.2. Tab. 8.1 contains detailed information on the complexity of different models, as well as timing results for the individual processing steps in our approach. We could show that plausible deformations can be quickly achieved using easy-to-specify deformation handles.

In our test cases, both large region handles (prescribing position and orientation) and point handles (prescribing only position) were used to create the various poses. Please note the natural intuitive behavior of the deformation, as well as the authentic look of the deformations also on the high-resolution meshes. Even when using only a few handles it is possible to easily create convincing results that look physically plausible. We would also like to point out that thanks to the tetrahedral setting there is no noticeable loss in volume even after very strong pose changes.

In all our test cases and application scenarios we found that the deformation behaves intuitively and is easy to use even for unexperienced users. The real-time deformation process in conjunction with immediate visual feedback makes it convenient to obtain the envisioned results.

8.3 Discussion

In this Chapter we proposed an interactive approach to shape editing based on our simple iterative volumetric template deformation framework. We view shape editing as template fitting process, where the target shape information is given by the user in the form of handle transformations. Our framework allows the

Figure 8.2: Results of an interactive editing session. Input models and their tetrahedral counterparts with handles on the left, three resulting poses on the right.

user to edit the shape in an intuitive manner, gaining immediate feedback on his handle movements.

One of the main limitations of our method is that it is not suitable for deformation of large models in interactive environments. We circumvent this limitation by working on a decimated version of the original model and transferring the pose back to the high resolution shape after the interaction process. However, we would like to point out that this is not a principle disadvantage since it is our primary goal to modify the low-frequency components of a model which contain information about its pose. Working on the low resolution shape is equivalent to performing a low-pass filtering of our geometry and adding the high frequency detail back after the low frequency components have been modified. We could also leverage the processing power of modern multi-processor architectures like GPUs to further increase processing speed.

As the rotation extraction and update steps (Sections 7.2.1(II) and (V)) are local in their nature, the convergence speed of the optimization is heavily dependent on the number of tetrahedral elements. Depending on the position of the handles, deformations (especially rotations) may only be propagated through

the shape element by element. While this can be partially alleviated using the rotation interpolation step (Sections 7.2.1(III)), in some cases the convergence behavior may still be unexpectedly slow. A solution to this issue would be using a multi-level hierarchical solver. However, it should be noted that this issue only appears under some very special handle placement configurations and will not be a problem in a typical deformation session.

On the other hand, our deformation technique has several distinct advantages over a purely linear deformation method (see Figure 8.3). While the inability of a purely linear approach to handle rotations quickly leads to a deterioration of the deformed geometry, Figure 8.3 (b), our iterative approach handles rotations faithfully lending a plausible appearance of the transformed model, Figure 8.3 (d). Our results are also more pleasing than the ones obtained with single step gradient-based deformation techniques such as Poisson based mesh editing (Yu *et al.* (2004)), Figure 8.3 (c), as we faithfully handle translations. By iteratively computing small pose updates in our framework we can produce similar results as other state-of-the-art non-linear methods, such as Botsch *et al.* (2007) and Sumner *et al.* (2007).

Our volumetric method has more favorable shape preservation properties than approaches working on triangulated manifolds. Due to the tetrahedral construction our approach aims at preserving distances between diametrically opposed vertices on the model's surface. Given an appropriately set up mesh, this can be regarded as a tendency to preserve cross-sectional areas. Although this does not imply global volume preservation, it comes close to a local volume preservation which is very useful when deforming human shapes where one wants, for instance, the thicknesses of limbs to be preserved. This property makes the approach very suitable for animation and performance capture tasks, as we will present in the following Chapter.

Another beneficial property of our deformation is its robustness even under extreme conditions. Even if we completely randomize the differential coordinates of an object constrained by a few handles we can recover the original shape after a number of iterations. This means that it is virtually impossible to break the deformation process by careless constraint placement.

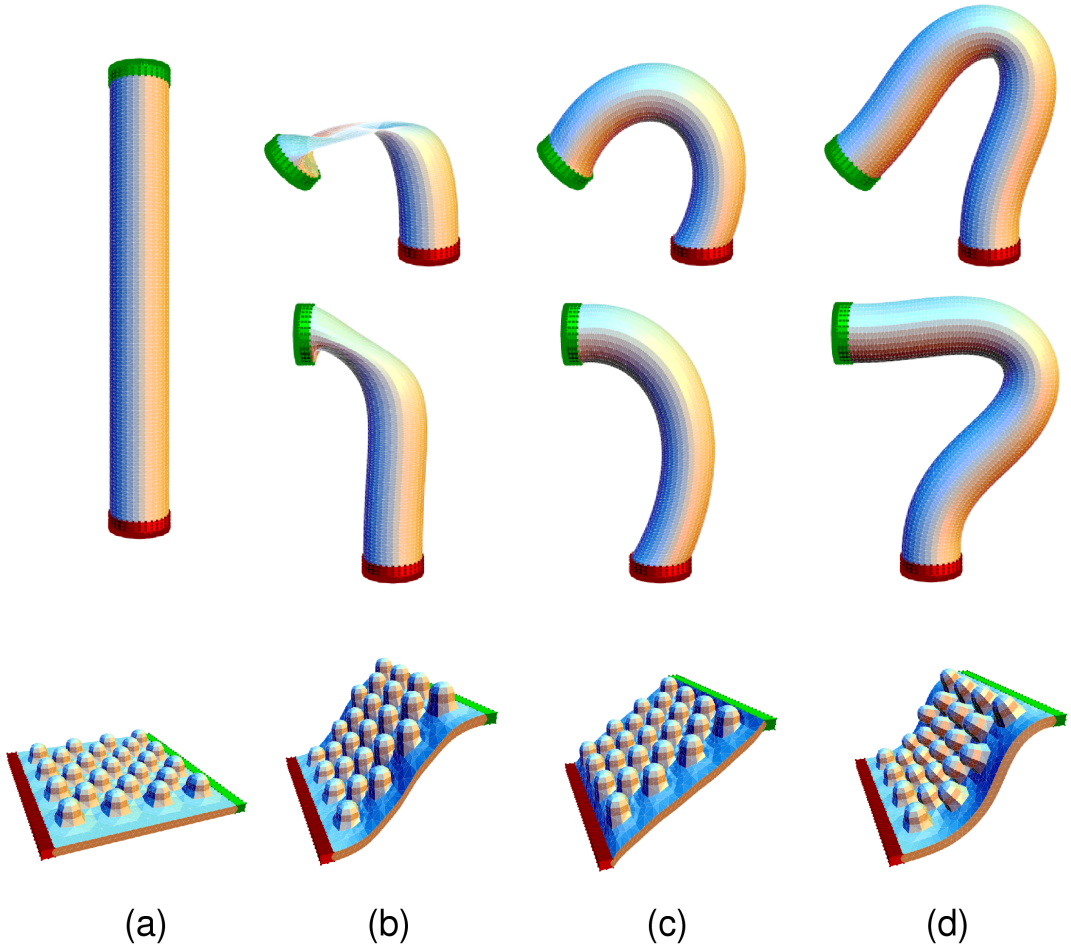


Figure 8.3: Comparison between different deformation methods. Column (a) shows the original objects and the handles in red and green. Different types of constraints are then applied to the green handle: rotation by 135° (top row), rotation by 90° and translation to the right (middle row) and pure translation (bottom row). Column (b) depicts results obtained with a linear least squares Laplacian, column (c) shows results of gradient based deformation, column (d) shows results of our method.

model	fig	input	tet	fact	trans	def	rotext	rotint	rigid	difupd	recons	fps
Bar	7.2	N/A	857 / 2289	0.039	N/A	0.002	0.003	0.004	N/A	0.001	N/A	≥ 120
B. Bar	7.1	N/A	1440 / 4302	0.074	N/A	0.003	0.005	0.006	≤ 0.001	0.003	N/A	$\sim$85
Plane	8.3	N/A	5066 / 16028	0.331	N/A	0.017	0.017	0.026	N/A	0.012	N/A	$\sim$20
Cylinder	8.3	N/A	7235 / 23831	0.565	N/A	0.028	0.025	0.044	N/A	0.015	N/A	$\sim$12
Dragon	8.2	100004	3021 / 9802	0.128	94.5	0.007	0.010	0.011	N/A	0.006	0.442	$\sim$37
Raptor	8.1	175100	2512 / 9372	0.136	167.6	0.006	0.009	0.013	N/A	0.006	0.965	$\sim$37
Armadillo	7.4,8.2	195948	1688 / 5373	0.084	96.0	0.004	0.005	0.007	0.001	0.003	0.484	$\sim$57

Table 8.1: This table shows performance figures for our technique on an AMD X2 5000+. The columns from left to right contain: Model name, figure number in the paper, number of input mesh vertices, number of tet mesh vertices/tetrahedra; thereafter, timings are given for pre-factorization of the left-hand side matrix of Eq. (7.1), pre-calculation of pose transfer coefficients, linear deformation, rotation extraction, rotation interpolation, rigid section handling, differential coordinate update, deformation transfer to input mesh; the rightmost column gives frames per second during interactive editing of the tetrahedral mesh. All timings are given in seconds.

Chapter 9

Animation and performance capture with tetrahedral meshes

In this chapter we will discuss two applications for our non-linear template deformation framework in the context of animation and performance capture (de Aguiar *et al.* (2008a); Stoll *et al.* (2007)). This is an extension to the work presented in Chapter 6 of this thesis. Our goal is to perform deformable template fitting to extracted image and marker constrains. This means that we parameterize the shape of a performer in terms of constraints of our template deformation framework instead of relying on a kinematic skeleton. The two main applications for this are simple direct animation from marker trajectories and pose capture from a set of input videos.

The non-linear framework presented in Chapter 7 is more suited for low-frequency pose deformations than capturing fine shape detail. The main reason for this is that the volumetric approach tries to preserve local volume and cross-sectional areas of the original shape. Additionally, by using lower resolution meshes for processing we are limited to capturing deformations on the detail level of our low resolution mesh. This makes the approach unsuitable for model refinement as in Chapter 6.3, where the flexibility of the linear deformation framework presented in Part I is essential. However, since the non-linear method has better local shape preservation property even under large pose changes and is thus more stable, it is more appropriate for animation and capturing the pose from input videos.

9.1 Animation from marker trajectories

The template deformation framework presented in Chapter 7 allows us to create natural animations from markers extracted from motion capture systems in an intuitive and straight-forward way (Stoll *et al.* (2007)). The flexibility in constraint positioning inside and outside a model, as well as the ability to create plausible deformations from moving point constraints only, enables us to quickly generate realistic animations of models using captured motion data. To demonstrate this, we mapped motion data acquired with a marker-based optical capturing system, as well as data captured with a marker-free motion capture algorithm (Carranza *et al.* (2003)) to high-quality laser scans of humans, Figure 9.1, as well as to the Stanford Armadillo model. We created several animations in which the models perform complex motions, such as a cartwheel. Depending on the type of motion data, different kinds of deformation handles can be specified.

Using our template deformation framework instead of a traditional kinematic skeleton means that we can circumvent the complex process of fitting the skeleton to the marker data and the model and can directly use use the raw 3D marker trajectories as deformation constraints. This simplifies the animation process drastically while still producing high quality results. As a preprocessing step, we align the model to the marker positions in the first frame by hand. After alignment, each marker is added as constraint as explained in section 7.2.3. By this means, we bypass the labor-intensive and often error-prone reconstruction of a kinematic skeleton from the marker data and utilize the raw measurement output straight away. In case we are already given a template skeleton mesh (as for example the default motion template from 3D Studio) we can pre-align our model to it and use vertices of the skeleton joints as deformation handles in a similar manner.

The marker-less motion capture results come already in the form of a coarse moving kinematic body model comprising of individual triangle mesh segments. By specifying corresponding vertices on the template and the mesh $\mathcal{T}$, it becomes straightforward to map the motion onto a scanned model. We used this framework to generate a sequence in which the dancing performance of an actor was captured and applied to a scan of himself.

If only few handles are used, we additionally mark a set of rigid groups on the model to be deformed, Figure 9.1. The whole handle setup-process typically takes only 3-4 minutes for a single animation.

Figure 9.1: Animation of a scanned character using optical MoCap data. From left to right: Input model, reduced tetrahedral model with rigid regions marked roughly, 22 input handles, five frames out of a cartwheel sequence.

The final animation is computed as follows: For each frame we update the position of the handle constraints, run the iterative deformation for a number of cycles (typically 2-3 cycles are sufficient) and afterwards deform the input mesh using the technique from Section 7.3. In practice, we use the tetrahedral mesh model for real-time pre-visualization of the animation, but render the high-quality deformation in a batch process that takes between 0.1 and 0.5 s to write one frame, which is mainly caused by the deformation transfer overhead.

Our results show that our method provides a fast and easy-to-use algorithmic alternative to create high-quality mesh animations from captured motion data. Realistic smooth surface deformations and subtle motion details are faithfully reproduced without having to rely on an intermediate skeleton representation with associated skinning weights.

9.2 Performance capture

In this section we will present how we can use our volumetric template deformation method for tracking a performer in everyday apparel (de Aguiar *et al.* (2008a)). By approaching performance capture as deformable template fitting process and parameterizing the motion of the performer in terms of the template deformation method with arbitrary constraints, we are facing a much harder tracking problem. However, we gain the advantage of now being able to track non-rigidly deforming surfaces (like wide clothing) in the same way as rigidly deforming models and do not require prior assumptions about material distributions or the segmentation of a model.

In combination with the model refinement approach presented in Chapter 6.3 we are able to capture performance data at a high level of detail that is applicable

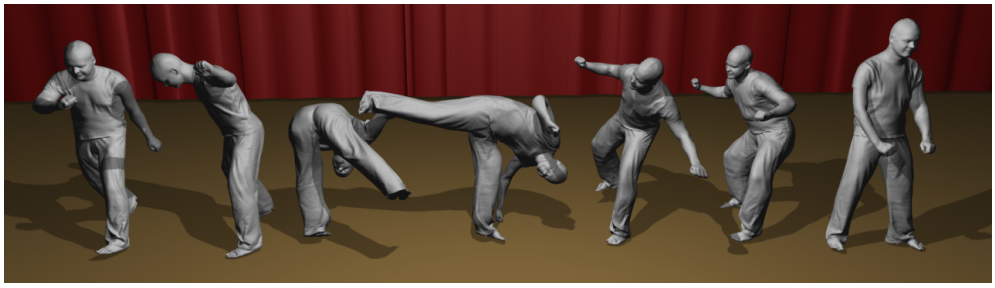


Figure 9.2: Result of the performance capture method from de Aguiar *et al.* (2008a) where the performer does a Capoeira motion.

to many complex scene type that could not be handled by alternative marker-based or marker-less recording techniques. The approach we present here is a continuation of our work in de Aguiar *et al.* (2007a) and de Aguiar *et al.* (2007b), where we directly estimate deformation constraints for a surface-based linear deformation framework from only multi-view video. The volumetric template approach allows us to alleviate many of the drawbacks we faced because of the linear model used in the previous work. We will only outline the method here as it applies the deformation framework developed in Chapter 7 of this thesis. More details about the extraction of the deformation constraints from videos and the global pose optimization can be found in the PhD thesis of de Aguiar (2008).

The main advantage of using our volumetric template over the previously proposed surface based tracking is robustness. The tetrahedral deformation approach is a lot more robust to errors in the constraints and will produce a meaningful low frequency pose even for problematic configurations. Under large deformations the results generated using our iterative method will look a lot more natural, which is mainly due to the implicit formulation of extracting rotations and the preservation of cross-sectional areas. As such it is a lot more suited to capturing the global pose of the performer.

The input data for our performance capture were recorded in our multi-view camera studio. The setup consists of $k = 8$ synchronized cameras recording with a resolution of $x \times y = 1004 \times 1004$ pixels. The recorded sequences of a scene contain n frames of video and are denoted as $\mathbf{F}_{k,n}$. We perform a background subtraction for all frames, yielding a set of binary silhouette images $\mathbf{I}_{k,n}$. In the silhouette images 0 denotes the background and 1 the foreground object. For each camera we also reconstruct a 3×4 calibration matrix $\mathbf{C}_k$ which maps 3D world coordinates to 2D images coordinates in the respective camera view. We

also acquire a triangle mesh model $\mathcal{M}_{input}$ of the performer using a full body laser scanner, convert this to a lower resolution tetrahedral mesh $\mathcal{T}_{input}$ and acquire the deformation transfer matrix $\mathbf{B}$ as described in Chapter 7.

Once all of the data has been capture we automatically register the tetrahedral template model to the first pose of the actor in the input footage using an iterative closest points based method. Since we asked the actor to strike a pose similar to the one she/he was scanned in, this step is greatly simplified. This step gives us the initial mesh $\mathcal{T}_1$. We now use a two-step analysis-through-synthesis approach based on image and silhouette cues to capture the shape and motion of the performer. We first estimate the global pose of the performer from each frame $\mathcal{T}_i$ to $\mathcal{T}_{i+1}$ and then apply the model refinement techniques presented in Chapter 6.3 to capture finer detail.

In a first step, we capture reliable deformation constraints using SIFT features (Lowe (1999)). We extract the features and match them across camera views and time to generate a set of 3D vertex constraints. The template mesh is deformed using those constraints, bringing it into a configuration close to the target pose even if scene motion is rapid. However, as the distribution of 3D features on the model surface is dependent on the scene structure (e.g. texture), it can be non-uniform and sparse. This may lead to the pose not being entirely correct. The feature set may also contain outliers in the correspondences, which makes additional pose update steps unavoidable. In the next steps we thus exploit the silhouette data to fully recover the globally correct pose.

We derive additional deformation constraints from matching the rim vertices of the model to the image silhouettes. Rim vertices are vertices that lie on the outer silhouette of the 2D projection of the mesh in the respective camera k. We pull the rim vertices to the closest match of the image silhouette $\mathbf{I}_{k,n}$ and back project the data into 3D coordinates (this process is similar to the model refinement in Chapter 6.3.1). We now deform the mesh using the extracted constraints, improving the global pose.

In the majority of cases, the pose of the model is now already a good match to the input images. However, in some cases, in particular if the scene motion was fast or the initial pose estimation using SIFT was not entirely correct, residual pose errors remain. We therefore perform an optimization step that corrects such errors by optimizing a subset of deformation handles until the silhouette overlap is maximized. Before processing we handpick somewhere between 15 and 25 key handle vertices and minimize an error term containing silhouette overlap, deformation energy $E_{nonrigid}$ (see equation 7.5) and key handle distance using

(a) (b) (c) (d)

Figure 9.3: Results of the performance capture algorithm from de Aguiar *et al.* (2008a). (a),(c) One input frame. (b),(d) Captured model from a similar camera viewpoint.

a black-box non-linear minimization technique (see de Aguiar *et al.* (2008a) for details on the energy functional).

The output of this is the template model deformed from time step i to $i+1$, i.e. a mesh $\mathcal{T}_{i+1}$. We iterate this process for all frames, resulting in a volumetric template sequence $\mathcal{T}_{1...n}$. We now apply the deformation transfer technique from Chapter 7.3 to generate the high resolution triangle mesh sequence $\mathcal{M}_{1...n}$.

This is a representation of the performer comprising of spatio-temporally coherent geometry that captures the rough global pose and motion of the original input. We can now apply the model refinement method from Chapter 6.3 to further increase the quality of our capture. Results of this combination are shown in Figure 9.3.

9.3 Discussion

In this Chapter we presented applications for template based shape processing with volumetric templates in the context of animation and performance capture. By replacing the traditional kinematic skeleton model with a point constraint based parameterization and approach the tasks as deformable template fitting process allows us to capture a much wider range of deformations. Additionally, we can simplify some of the processes involved in animation and performance capture. Nevertheless, the framework also brings some limitations.

In the context of animation from marker data (Stoll *et al.* (2007)) we are able to directly use the raw 3D marker trajectories as deformation constraints and circumvent the complex process of fitting the skeleton to the marker data and the model. However, as we do not impose any constraints on the topology of

the constraint points (i.e. we have not defined any connectivity), it is possible that constraint points will change the distance to each other even when originally placed on a near rigid body part. This may lead to limbs changing their length. While this is usually not an issue with animation from marker data, it will become apparent if users want to edit the motion, as the system will not impose bone length restrictions on the user. If we do not specify rigid regions on arms or legs we may observe strongly non-rigid behavior in otherwise near rigid areas, i.e. bones may appear to bend. This is a side effect of the as-rigid-as-possible deformation optimization, which essentially mimics elastic objects. Because of this, the method is not immediately suitable for editing of motions.

Despite this, our approach offers a light-weight and easy-to-use algorithmic alternative to create high-quality mesh animations from captured motion data. Realistic smooth surface deformations and subtle motion details are faithfully reproduced without having to rely on an intermediate skeleton representation with associated skinning weights.

Using the volumetric template approach for performance capture (de Aguiar *et al.* (2008a)) allows us to capture the pose of a performer from just a sparse set of calibrated multi-view video streams accurately, even in difficult situations such as when the subject is wearing wide apparel (see Figure 9.3). Even with noisy input data and in scenes containing fast motions and complex self occlusions we are able to track the subject reliably.

In about 13 out of over 3500 captured input frames the tracker failed to correctly align the template to the actors pose. This was mainly caused by motion blur and incorrect silhouette segmentations due to shadows. However, the method recovered automatically and was able to track the motion of the performer correctly until the end.

Analogous to the animation process our template deformation method does not guarantee to preserve the length of limbs and bones. In practice, this was not noticeable in any of our sequences. Rubbery effects due to the elastic deformation process may appear in limbs. For instance, an arm may not only bend at the elbow, but rather bend along its entire length. This artifact can usually be fixed by applying the model refinement step from Chapter 6.3, except for in cases with strong self occlusions.

The biggest limitation in comparison to skeleton based performance capture however is that our final representation is an animated mesh that cannot be edited easily. While there have been recent advances in mesh based animation editing, for example by using mesh deformation approaches on animations Kircher &

Garland (2008); Xu *et al.* (2007), the tools available for skeleton based animation processing are much more sophisticated today. There exist methods that allow us to convert mesh based animations to skeletal animations such as de Aguiar *et al.* (2008b). However, this conversion process may not yield an intuitive skeleton for performers wearing apparel such as a skirt. Additionaly, we will not be able to generate new realistic cloth deformations for wide apparel, as its behaviour will be controlled only by the rigid skeleton. In Part III of this thesis we will address this issue and introduce a method that allows us to capture the physical properties of our template from a mesh animation. Knowing these properties we can use simulation techniques to generate arbitrary new animations that behave similar to the originally captured data.

Despite these limitations the volumetric template processing approach allows us to develop non-intrusive approach to spatio-temporally dense performance capture from video. Abandoning traditional motion skeletons allows us to reconstruct a large range of real-world scenes in a spatio-temporally coherent way. In combination with further model refinement we can exceed the capabilities of marker-based optical capturing systems that are widely used in industry, and capture natural and life-like animation sequences.

Part III

Physically based template shape processing

In this part of the thesis we will present a method for estimating physical material properties using template based shape processing (Stoll *et al.* (2009)). While the previous two parts focused on fitting templates to (temporal) shapes or constraints, we now introduce a method for estimating the material properties of a template fitted shape by analyzing temporal data. This allows us to learn how the template reacts to external impulses over time. We can then not only play back or show reconstructions of data we have recorded, but also generate new sequences by specifying new impulses that were not observed before. This newly generated data will then convincingly exhibit the same properties as the originally reconstructed data. While we will present a specific example for the field of motion capture and animation (estimating dynamic cloth worn by a performer), the methodology presented in this part of the thesis could also be applied to other dynamic models, for example to estimate the dynamic behavior of elastic objects. In Chapter 10 we will explain how we can use performance capture data reconstructed with methods included in Parts I and II of this thesis to segment the template into rigid and non-rigid (i.e. cloth) parts and estimate the apparels physical simulation parameters. This allows us to reconstruct an detailed animatable human body model from just a 3D scan and a set of multi-view videos.

Chapter 10

Optical reconstruction of animatable human body models

In this Chapter we propose a novel method for estimating physical parameters of a template model from a set of multi-view videos with help of the performance capture methods introduced before (Chapters 6 and 9). This allows us to create a virtual animatable human body model (Stoll *et al.* (2009)). Using a template based reconstruction is essential here, as the constant connectivity and spatio-temporal coherence is what allows us to gather information about the physical properties. Our template now not only consists of the shape of the performer, but rather also includes a kinematic skeleton, is segmented into cloth parts, and features physical material parameters for each cloth part. Our goal is to fit this complex template representation to our input data as accurately as possible.

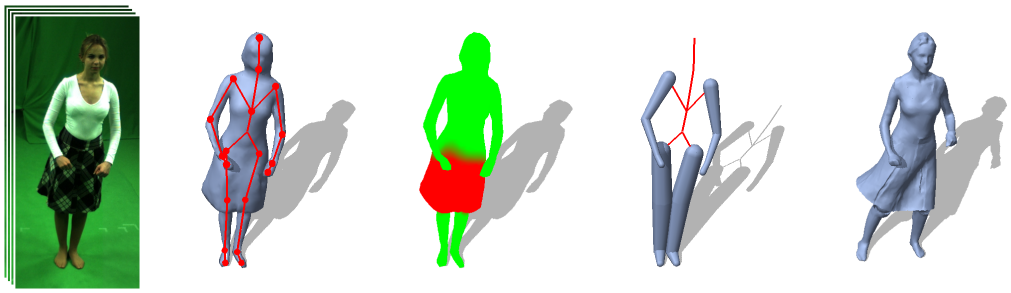
The main motivation for this research stems from increasingly powerful computing hardware, which will soon make it feasible to render characters of ever higher visual complexity and realism even in real-time applications like games or networked virtual worlds. Certain visual effects that were previously only seen in feature films, such as detailed cloth animation or body models of high geometry and texture detail, now also slowly find their way into consumer market applications. The recent advent of hardware accelerated physics simulation has expedited this trend, and more advanced simulated characters will soon become part of many computer games.

The ability to render more complex characters also makes the animation design process a more challenging one. Each design element of a character requires greater attention to detail, resulting in an overall more labor-intensive and costly process. More detailed surface geometry will have to be designed or scanned,

more detailed textures will have to be painted or photographed, more detailed motion will have to be keyframed or captured, and cloth simulation models will have to be properly tuned to enable real-time rendering. Various acquisition technologies exist to support the animator in parts of the design process, such as motion capture systems to get skeleton motion, 3D scanners to measure static geometry, or specialized measurement devices to analyze material parameters. Unfortunately, using such tools can be very cumbersome and time-consuming, and therefore often complicates animation design rather than to accelerate it. Capture technology would be much more practical if all aspects of an animation that were mentioned above could be obtained from real subjects in a simple, fast, and cost-effective unified process. Being able to start off such rich new captured performance models, animation professionals would not only save valuable time during initial scene design. They would also be given the possibility to easily create complete virtual doubles of real actors and their apparel even under tight budget and time constraints. Additionally, the creative freedom of animators is greatly enhanced since they can either directly work with unmodified captured scenes or freely modify any aspect of a captured performance in their favorite animation tool.

The research that has come closest to achieving this goal are template based performance capture approaches which enable the reconstruction of detailed motion and time-varying geometry of entire human actors or pieces of apparel from multi-view video recordings, such as de Aguiar *et al.* (2008a) and Gall *et al.* (2009) presented partially in Part I and II of this thesis, or the work of Bradley *et al.* (2008) and Vlasic *et al.* (2008). Unfortunately, performances captured with these approaches can merely be played back and it is impossible to create entirely new motion sequences with similar lifelike surface deformation. We are only able to capture static template animation.

Contrary to those approaches, our goal is to reconstruct a *fully-animatable* dynamic human character in general clothing from unmodified multi-view video streams and an input template scan, without requiring much manual intervention by the user. We first record a so-called *reference sequence* of the person whose model is to be reconstructed (Figure 10.1(a)). In this sequence, the person walks around for a few seconds in the same attire that should be part of the *animatable model*. From this reference sequence, we reconstruct a *reference performance* which comprises of a deforming mesh sequence that represents the detailed dynamic surface geometry of the moving actor over time, as well as a sequence of joint angle parameters of the underlying skeleton, Figure 10.1(b). The reference



(a) (b) (c) (d) (e)

Figure 10.1: Overview of our processing pipeline: (a) multi-view video sequence of a reference performance; (b) performance capture result: skeleton motion + deforming surface; (c) cloth segmentation (red=regions of loose apparel); (d) estimated collision proxies; (e) After optimal cloth simulation parameters are found, arbitrary new animations can be created.

performance is captured by a marker-less performance capture technique we presented in Gall *et al.* (2009) and includes the model refinement step presented in Part II of this thesis. It requires no manual intervention, even if the input shows difficult motion of a person in loose attire.

We then analyze the dynamic template surface geometry of the reference performance and automatically decompose it into approximately rigid body parts and non-rigidly deforming pieces of attire, Figure 10.1(c). In the same stage of the algorithm, we reconstruct approximate collision proxy geometry for both occluded and non-occluded parts of the body, Figure 10.1(d). These proxies are needed for accurate and fast cloth simulation, Section 10.3. After segmentation, we fit to each region of the deforming fabric a physics-based cloth simulation model. The parameters of this model are numerically optimized to best reproduce the cloth motion in the reference sequence.

The end result of this process is an enriched template model, containing a skeleton with skinning weights for non-cloth regions, collision data, the cloth segmentation and a physics based simulation model with optimized parameters. Given this, arbitrary new animations of the character with realistic dynamic surface appearance can be created by simply changing skeletal motion parameters, Figure 10.1(e).

Many previous approaches for cloth estimation from real world samples reconstruct deforming cloth geometry only, but not a physically-based forward simulation model. Some methods use multi-view feature and stereo for vision-based de-

forming geometry capture of square cloth samples (Pritchard & Heidrich (2003)). Other algorithms rely on special marker patterns printed on the fabric, as well as on a priori geometry model of a piece of apparel, to measure time-varying cloth geometry from multi-view video (Scholz *et al.* (2005); White *et al.* (2007)). The latter approach also learns a simple data-driven deformation model which can approximate the wrinkling of fabric in new poses. Recently, Bradley *et al.* (2008) proposed a new approach to capture deforming cloth geometry from multi-view video without explicit markers in the scene. So far only few algorithms estimated parameters of a physics-based cloth model. Bhat *et al.* (2003) learn such parameters from waving square fabric samples that were recorded with a real-time structured light system. Hasler *et al.* (2006) use a similar approach but rely solely on texture information from videos. Conceptually related is the approach by Shi *et al.* (2008) who estimate parameters for simulating secondary skin deformation. In contrast, our method does not require an a priori shape model for individual pieces of apparel, does not require any form of visual pattern in the scene, and does not require off-line physical material measurements from fabric samples. Our method fully-automatically identifies all cloth regions on the entire moving human and recovers plausible forward simulation parameters.

10.1 Experimental setup

The performance of an actor is captured by $k = 8$ synchronized and calibrated cameras running at 40 fps and providing $x \times y = 1004 \times 1004$ pixel resolution. This yields multi-view video frames $\mathbf{F}_{k,n}$ for each camera and each frame $n = 1, \ldots, N$. The image silhouettes are extracted by chroma-keying, yielding N silhouette images $\mathbf{I}_{k,n}$. Prior to recording, we acquire a model of the subject that comprises of two components, a surface mesh $\mathcal{M}$ and an underlying bone skeleton $\mathcal{S}$. To obtain the initial geometry, we take a static full-body laser scan $\mathcal{M}_{high}$ of the actor wearing the attire for the reference sequence, which we then decimate to roughly 5000 triangles to obtain our mesh template $\mathcal{M}$. The decimation is necessary to keep both performance capture and, later, performance simulation times in reasonable bounds. The skeleton with 36 degrees-of-freedom is inserted into the surface mesh by manually marking the joint positions. Thereafter, the method of Baran & Popović (2007) is employed to assign each vertex a weight that is used to deform the surface according to the skeleton pose. The latter paper also proposes a fully-automatic skeleton insertion approach which could be used instead of our manual procedure. For the deformation, we rely on quaternion

blend skinning (Kavan *et al.* (2007)) that approximates the non-linear surface deformation better than linear blend skinning methods.

Please note that the original laser scan may contain holes, some of them due to bad measurements and some of them due to occlusion from the scanner, such as at the underneath of the skirt. We first automatically fill in all holes by means of Poisson surface reconstruction to create a closed surface. Some originally occluded regions may have been filled in a "semantically" wrong way, such as the sheet of triangles connecting the lower rim of skirt and the legs. We have developed a semi-automatic tool that proposes regions of "semantically" incorrect triangles. The user can optionally refine this proposal manually. Implausible triangles are thereby marked as invalid. They are weighted down in the reference performance capture and excluded from the animatable model estimation.

10.2 Performance capture

The first step after pre-processing is capturing the reference performance using a template based approach (Gall *et al.* (2009)). Unlike the method we presented in de Aguiar *et al.* (2008a), the method we apply here relies on a kinematic skeleton as template parameterization for coarse pose capture and combines it with the model refinement step explained in Chapter 6.3.2.

There are several reasons why we prefer a skeleton-based coarse template parameterization for our task at hand. Skeleton-based animation is the predominant character animation paradigm in real-time applications. It is easier to edit animations of performers using a skeletal representation, as it intrinsically preserves bone lengths. Since we will represent the non-rigid motion of our template using an additional physical simulation, we will only use the skeleton to represent near rigid parts which can be approximated well using skinning techniques. Finally, estimating the hidden collision geometry of the performer under the potentially wide apparel is simplified by the skeletal structure. It should be noted that it would also be possible to use our results from de Aguiar *et al.* (2008a) in combination with the skeleton extraction presented in de Aguiar *et al.* (2008b) to generate the reference performance for our method. However, the tracking of the method presented in Gall *et al.* (2009) is more reliable in sequences where people wear wide apparel.

In the following we will only briefly outline the main aspects of our approach from Gall *et al.* (2009). For each frame n of the reference sequence, the surface geometry in terms of the mesh $\mathcal{M}(n)$ and the joint motion parameters $\mathcal{S}(t)$ are

estimated. To find the body pose in the current frame, the skeletal pose $\mathcal{S}(n)$ is optimized and quaternion blend skinning deforms the detailed surface $\mathcal{M}(n-1)$ of the previous time step into the current time step. Once converged, we apply the model refinement step presented in Chapter 6.3.2 to estimate the fine surface deformation of $\mathcal{M}(n)$ without limiting the deformation to comply with the skeleton.

We estimate the globally optimal skeleton pose in two phases: The first phase searches for the nearest local minimum of an energy functional that assesses the model-to-image alignment based on silhouettes and texture features. To this end, the whole articulated skeleton is optimized locally. Subsequently, misaligned bones are detected by evaluating the energy of each body part and the skeleton is traversed to label all affected bones. The labeled bones are then re-estimated by a particle-based global optimization similar to Gall *et al.* (2008) that is also used to initialize the tracker.

The output reference performance captures the geometry of the actor in each pose, and also captures the true look of both coarse and medium scale folds in clothing. At the same time, the bone skeleton configurations are reliably captured, even under occlusion by loose apparel, Figure 10.1(b).

10.3 Cloth segmentation

Our method of identifying which parts of the moving template geometry are likely to be part of the actual body and which are likely to be cloth is inspired by motion segmentation methods that identify approximately rigidly moving sections in mesh animations (de Aguiar *et al.* (2008b); James & Twigg (2005)).

Although human motion is mainly controlled by a kinematic skeleton, even the surface of a naked person never deforms entirely like a collection of rigid bodies. In practice, the non-rigid surface deformation of naked skin or areas with rather tight clothing, which is often due to muscle bulging, can be plausibly approximated in real-time by surface skinning methods. This does, however, not hold true for surface regions which represent loose pieces of apparel, since fabric exhibits starkly different non-rigid deformation behavior.

We therefore analyze the deforming surface geometry of the reference performance and automatically segment it into cloth and non-cloth regions, Figure 10.2. To identify cloth regions, we search for surface areas that deform mainly non-rigidly. If a body part deforms approximately rigidly, mutual distances of vertices

on that part hardly change over time. We take this insight and derive the following simplified but efficient rule to decide if a vertex $\mathbf{v_i}$ lies on a cloth segment: In the first mesh pose $\mathcal{M}(1)$ of the reference sequence, a ray is shot from vertex $\mathbf{v_i}$ in the negative local normal direction and intersected with the mesh surface on the opposite side. Vertex $\mathbf{v_i}$ is paired with the intersection point $\mathbf{p_i}$, which is described on the intersected face at sub-triangulation accuracy using barycentric coordinates. We now compute the standard deviation $\sigma(\mathbf{v_i}) = \sigma(d(\mathbf{v_i}, \mathbf{p_i}))$ of the Euclidean distance $d(\mathbf{v_i}, \mathbf{p_i})$ over the entire reference sequence. If this standard deviation exceeds a threshold t_{rigid} we conclude that $\mathbf{v_i}$ lies on a non-rigidly deforming surface area likely to be cloth. In other words, we find non-rigidly deforming surface regions by searching for sections with non-persistent cross-sectional areas. This is a weaker criterion than testing preservation of mutual distances between all points on a segment, but it can be computed more efficiently and performs well in practice.

Please note that we could also search for non-rigid segments by finding regions with starkly varying vertex to bone distances over time. However, in practice we found that this leads to frequent misclassification of actual body parts, in particular the arms, as cloth. The reason for this is that the skeleton of the actor does not provide all anatomical degrees of freedom (e.g. all shoulder and spine joints) which makes the least squares poses of the bones deviate from the true physiological positions. This deviation can lead to unintended bone distance variations which would result in wrong classification, in particular in the arms. Our purely surface-based decision criterion does not suffer from this problem and it also works for performances captured with skeleton-free measurement approaches such as presented in previous parts.

Since our final animatable performance model combines physically-based and skeleton-based animation, we need to make sure that the transition between the two modalities is seamless. We therefore transform the $\sigma(\mathbf{v_i})$ values into "*cloth-ness weights*" $\delta(\mathbf{v_i}) \in [0, 1]$:

$$\delta(\mathbf{v_i}) = cl\left(\frac{\sigma(\mathbf{v_i}) - t_{rigid}}{t_{cloth} - t_{rigid}}\right) . \tag{10.1}$$

Here, the function cl clamps the weights to $[0, 1]$. t_{cloth} represents a threshold on $\sigma(\mathbf{v_i})$ above which vertex motion is fully determined by the physics simulation. The motion of a vertex with $\delta(\mathbf{v_i}) = 0$ is fully-controlled by skeleton motion + skinning, a vertex with $\delta(\mathbf{v_i}) = 1$ is fully cloth model controlled. Vertices with $\delta(\mathbf{v_i}) \in]0, 1[$ lie in the transition zone between cloth and skeleton simulation

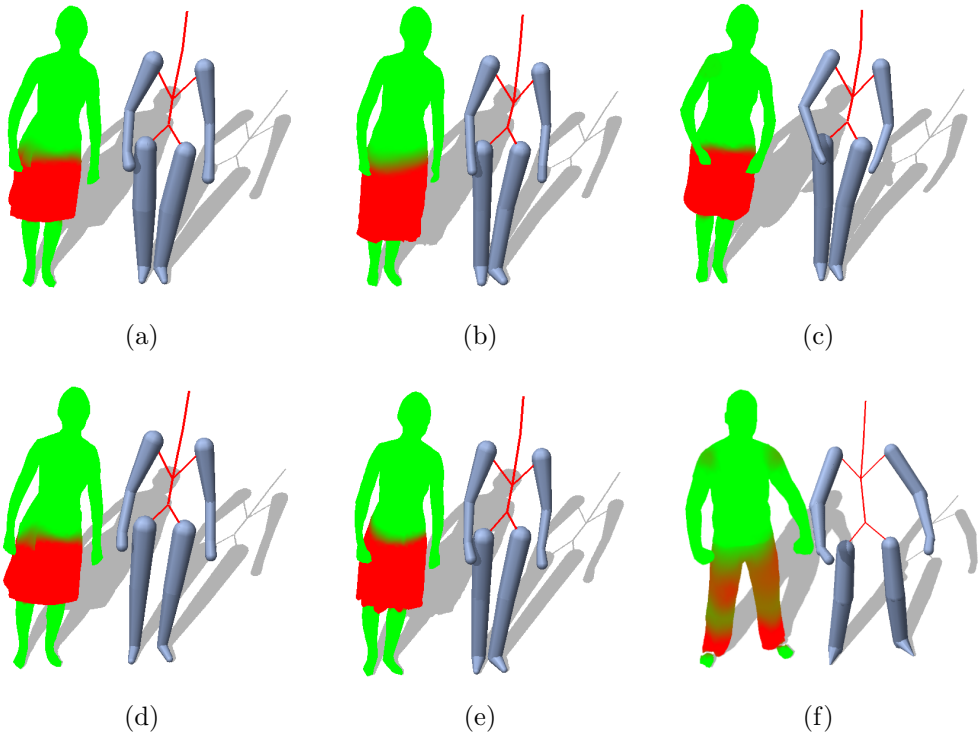
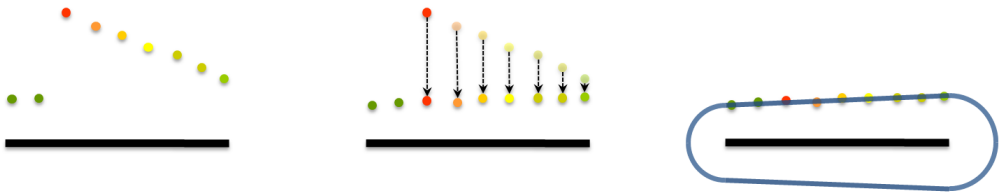


Figure 10.2: Cloth segmentation results and proxies found for several reference sequences showing different clothing styles (red=cloth, green=approximately rigid): (a) skirt 1, dancing sequence; (b) skirt 2, walking sequence; (c) skirt 3, dancing sequence; (d) skirt 1, walking sequence; (e) dress, dancing sequence; (f) capoeira sequence - cloth regions and transition areas at boundaries are reliably identified. Note that in the skirt 1 results the dent in the skirt region boundary is a woolen cord that was automatically excluded from the cloth simulation region since it stays mostly rigid.

(e.g. the rim of the skirt and parts of the pants in Figure 10.2). In Section 10.7 we explain how to blend the two animation types in those areas. To create a smooth clothness distribution and fill in potential holes in the estimation, we also perform a diffusion of the δ's on the surface.

10.4 Estimating hidden geometry

For both the estimation of cloth simulation parameters, as well as for creating new animations it is essential that collisions of the fabric with the body are plausibly simulated. Unfortunately, the true shape of body geometry under wide attire is



(a) (b) (c)

Figure 10.3: 1-D proxy fitting to segmented data: (a) vertices around a bone with weights (green = high, red = low); (b) moving vertices closer according to their distance standard deviation σ; (c) fitting the conical section and end-caps.

not directly visible and is thus neither part of the captured mesh sequence nor the skeleton model itself. However, by analyzing the reference performance and by capitalizing on general assumptions about human anatomy it is feasible to approximate even occluded body geometry.

We approximate collision geometry by fitting a capped conic shape template to the bones in the body. Performing collision tests for these shapes is computationally very efficient. One conic collision proxy can be fitted to each bone in the body. Depending on the sequence captured, however, it is often sufficient to only estimate proxies for arms and legs, such as in our test data with different skirts. The dimensions of each collision proxy are estimated from the segmented reference performance. We base our estimation procedure on the assumption that in areas with strong cloth motion collision proxies are most likely further away from the outer surface, whereas in mostly rigid areas the collision geometry should coincide with the surface of the captured mesh. We therefore use the following efficient 1-D fitting procedure to fit rotationally symmetric conic proxies to the first captured pose of the surface mesh, Figure 10.3: Given a bone, all mesh vertices closest to that bone are transformed into local cylindrical coordinates and the rotational dimension around the bone is neglected. The parameters of the conic in 1-D are fitted to these vertices by means of a weighted linear regression. Since non-cloth vertices are closer to the true collision surface than cloth vertices, each vertex is weighted by the reciprocal of its deviation. Also, the cylindrical distance of each vertex from the bone is reduced by its standard deviation $\sigma(\mathbf{v_i})$ that was computed to find its clothness value. Finally, we post-process all the fitted proxies, and average the conic shapes of the corresponding bones on joints and on the left and right body halves to enforce symmetry.

10.5 Cloth simulation

Having segmented the mesh $\mathcal{M}$ into approximately rigid and non-rigid moving parts, we can now replace the mesh animation of the non-rigid parts using a dynamic cloth simulation. As our segmentation is not binary but continuous, we need to be able to incorporate the animation as soft constraints into our simulation system. In this section we will give a brief overview of the method we chose for our implementation.

Our cloth simulation is based on the *position-based dynamics* system proposed by Müller *et al.* (2007). Their method differs from many other approaches in that it uses vertex positions explicitly as part of the simulation state, instead of only computing them by integrating velocities. In position-based dynamics vertex positions are explicit members of the simulation state which allows for direct position manipulation during simulation. Springs connecting vertices do not apply a force anymore that is used to update velocities, but are instead used in a projection operation that directly moves the vertex positions to a desired target location. The position-based dynamics framework bears a couple of advantages: it allows for fast real-time simulation, it is very stable, and it allows us to formulate a large variety of simulation constraints using the same mathematical framework. Here, we only briefly review the main concepts of our approach and refer the interested reader to the original paper for more details on the physics-based simulation.

A piece of cloth is geometrically represented by a triangular surface mesh. The simulated geometry comprises of vertices with positions $\mathbf{v_i}$, and velocities $\mathbf{u_i}$. Masses m_i for each vertex are approximated by the respective areas on the surface (i.e. $\frac{1}{3}$ of the area of the adjacent triangles). In addition, the solver expects a set of n_c constraints C_j acting on the positions of the mesh vertices, as well as a set of external forces $\mathbf{f}$ acting on the fabric. Each constraint C_j is represented by means of a function defined over a subset of vertex position. The strength of each constraint is further controlled by a stiffness parameter $k_j \in [0, 1]$.

The simulation loop comprises of two steps. First, an explicit Euler integration computes new vertex positions based on the current velocities and external forces. In a second step, a Gauss-Seidel like solver modifies the vertex position estimates such that the constraints are satisfied. This constraint projection step works locally, i.e. it handles all constraints separately from each other. This allows for simple handling of even non-linear constraints. Within the overall simulation loop, the projection of all constraints is iterated several times.

We support four types of simulation constraints in our implementation: collision constraints, stretching and bending constraints, as well as positional constraints. Stretching constraints take the form

$$C_{stretch}(\mathbf{v_a}, \mathbf{v_b}) = \|\mathbf{v_a} - \mathbf{v_b}\| - l_{a,b}\,, \tag{10.2}$$

where $l_{a,b}$ is the rest length of the edge between $\mathbf{v_a}$ and $\mathbf{v_b}$, which is the length of the respective edge in the original surface mesh. The bending constraint evaluates to

$$C_{bend}(\mathbf{v_a}, \mathbf{v_b}, \mathbf{v_c}, \mathbf{v_d}) = acos(\mathbf{n_l} \cdot \mathbf{n_r}) - \phi\,, \tag{10.3}$$

where $\mathbf{v_a}$, $\mathbf{v_b}$, $\mathbf{v_c}$, and $\mathbf{v_d}$ are the positions of vertices of two adjacent triangles that share a common edge, and $\mathbf{n_l}$ and $\mathbf{n_r}$ are the respective triangle normals. ϕ is the rest-angle between the two triangle normals, which we assume to be 0 everywhere. For general pieces of apparel, the rest angle may be different from 0 and should actually be estimated from the reference sequence as well. However, in order to keep the estimation problem tractable, we exclude the rest angles from the optimization and set them to a constant conservative value.

A spatial hashing approach is used to quickly find collisions and self-intersections, yielding additional collision constraints with fixed stiffness $k_{coll} = 1.0$. Finally, we include positional constraints which take the form

$$C_{blend}(\mathbf{v_a}) = \mathbf{p_a}\,, \tag{10.4}$$

where $\mathbf{p_a}$ is the position the vertex should reach. The stiffness of this constraint, $k_{blend}(\mathbf{v_a})$, is vertex-specific. These position constraints can be interpreted as soft attachment and are explained in more detail in Section 10.7.

External forces comprise of gravity $\mathbf{f_g}$ and drag $\mathbf{f_d}$. Unlike the original paper by Müller *et al.* (2007), we use a more sophisticated model for air-resistance similar to Bhat *et al.* (2003). The air drag force depends quadratically on the velocity in direction of the surface normal, and linearly in tangential direction:

$$\mathbf{f_d}(\mathbf{v_i}) = -0.5A(\mathbf{v_i})h(d_n\|\mathbf{u_{i,n}}\|^2\mathbf{n} + d_t\mathbf{u_{i,t}})\,. \tag{10.5}$$

Here, $A(\mathbf{v_i})$ is the area of the surface patch around the vertex $\mathbf{v_i}$ (in our case equal to m_i), and $h = 0.005$ the drag constant. $\mathbf{u_{i,n}}$ and $\mathbf{u_{i,t}}$ are the components of $\mathbf{u_i}$ in direction of normal $\mathbf{n}$ and tangential to it, respectively. The constants d_n and d_t are the drag coefficients.

Friction and restitution are handled by directly manipulating velocities of colliding vertices. To this end, vertex velocities are damped in the direction perpendicular to the collision normal, and reflected in the direction of the collision normal.

If the step size of our simulation is set to coincide with the capture rate of our system it allows easy synchronization between the data. However, it would be a straightforward process to decouple simulation and animation frame-rates by expressing fractional steps using linear blending between two updates, as is usually done in real-time simulations.

For each cloth region on the model, we therefore have to determine the three parameters $\rho = (d_n, d_t, k_{bend})$, i.e. the two drag coefficients, as well as the bending stiffness. Since our simulation is aimed at real-time performance the cloth exhibits a certain amount of stretching, even if the stretching stiffness is set to its maximum. This inherent flexibility is higher than that of all types of cloth we recorded in our experiments. Therefore, we use a fixed stretching stiffness of $k_{stretch} = 1$. In theory, the number of simulation steps n_{iter} per time frame of video could be considered a free parameter of our optimization problem as well. However, we aim for a character that can be simulated in real-time, and in real-time applications the number of iterations is usually fixed in order to balance resources. We therefore employ the same number of iterations ($n_{iter} = 16$) during both parameter optimization and creation of novel animations.

We implemented the method in a prototypical simulation system that runs with interactive frame-rates for meshes of moderate size. Due to the local handling of constraints the method is very suitable for implementation on parallel processors. NVidia provides a version of the simulation as part of its PhysX library (NVidia (2009)), which is accelerated by graphics hardware and provides a significant speed boost over a single threaded CPU implementation.

10.6 Combining simulation and reference performance

Before explaining how to estimate cloth parameters, we describe how new poses of our final fully-animatable performance model are created. Positions of vertices with $\delta(\mathbf{v_i}) = 0$ are solely determined by the current joint parameters of the skeleton and dual quaternion skinning. Similarly, the positions of all pure cloth vertices with $\delta(\mathbf{v_i}) = 1$ are determined by the physical simulation described

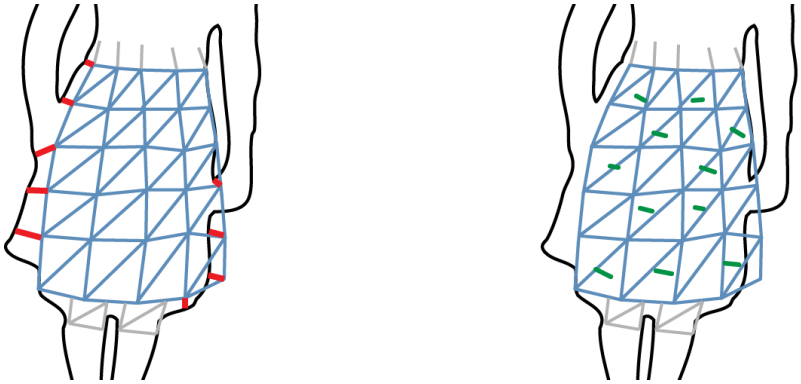


Figure 10.4: Two components of the error function on the cloth section (light blue) shown in 2D. (left) silhouette distance error: red lines between reprojected and measured silhouette points; (right) SIFT distance: green lines between predicted SIFT feature locations and measured SIFT feature locations.

in Section 10.5. For blend vertices, $\delta(\mathbf{v_i}) \in]0,1[$, the new pose is jointly determined by cloth and skeleton simulation. To this end, the position of $\mathbf{v_i}$ according to skeleton motion, $\mathbf{p_i}$, is included as a blend constraint into the physics simulation (Section 10.5) with a stiffness of $k_{blend}(\mathbf{v_i}) = (1 - \delta(\mathbf{v_i}))$. At each time step, collisions of the cloth with collision proxies are tested and included into the simulation.

10.7 Parameter optimization

In order to create new animations, we need to determine the set of three simulation parameters $\rho = (d_n, d_t, k_{bend}, k_{stretch}))$ for each cloth region in our template. It is our goal to find for each coherent region of cloth on the body a set of parameters such that the forward simulation of the cloth exhibits the same behavior as the cloth in the training sequence. The parameters are found by running a numerical optimization that strives to minimize the difference of the simulation to the input videos. Our energy function measures two properties of the simulated animation: **1)** the alignment of the cloth region's silhouette edges with silhouette boundaries in all input images over time, $E_{sil}(\rho, n)$, and **2)** the alignment of the reprojected cloth with robust features in the interior of the fabric in all camera views, $E_{sift}(\rho, n)$. Both error terms are evaluated over the entire reference

sequence and their contributions are combined, yielding the overall fitting error

$$E_{fit}(\rho) = \frac{1}{N}\sum_{n=1}^{N}(\alpha E_{sil}(\rho, n) + \beta E_{sift}(\rho, n))\,. \tag{10.6}$$

Here, the values $\alpha = 1$ and $\beta = 10$ are empirically determined weights that are kept constant for all our estimations.

For each time step of video n and each input camera view k, a certain set of vertices on the cloth segment under consideration should project onto the silhouette boundary of the respective input frame. This set can be easily identified and we call it the set of silhouette rim vertices $\mathbf{V}_{k,n}$, Figure 10.4(left). The silhouette error then evaluates to

$$E_{sil}(\rho, t) = \frac{1}{n_{sil}}\sum_{c}\sum_{\mathbf{v}\in\mathbf{V_{k,n}}} d_{im}^2(\mathbf{q}_{I_{k,n}}(\mathbf{v}), \mathbf{q}'_{I_{k,n}}(\mathbf{v}))\,, \tag{10.7}$$

where $d_{im}(\mathbf{q}_{\mathbf{I}_{k,n}}(\mathbf{v}), \mathbf{q}'_{\mathbf{I}_{k,n}}(\mathbf{v}))$ is the image space distance between the reprojection $\mathbf{q}'(\mathbf{v})$ of silhouette rim vertex $\mathbf{v}$ into silhouette image $\mathbf{I}_{k,n}$ and the closest boundary point of the measured silhouette in the same image, $\mathbf{q}_{\mathbf{I}_{k,n}}(\mathbf{v})$. n_{sil} is the number of all rim vertices from all $\mathbf{V}_{k,n}$ in this frame. For each time step of video and each camera view, we also compute a set of SIFT features (Lowe (1999)), $\mathbf{E}_{k,n}$. In addition, we establish correspondences between features in two subsequent time steps for each camera view, Figure 10.4(right). The feature-based energy term then reads

$$E_{sift}(\rho, t) = \frac{1}{n_{sift}}\sum_{c}\sum_{\mathbf{e}\in\mathbf{E}_{k,n}} d_{im}^2(\mathbf{o}_{\mathbf{F}_{k,n}}(\mathbf{e}), \mathbf{o}'_{\mathbf{F}_{k,n}}(\mathbf{e}))\,, \tag{10.8}$$

where $\mathbf{o}_{\mathbf{F}_{k,n}}(\mathbf{e})$ is the 2D image position of feature $\mathbf{e}$ in image $\mathbf{F}_{k,n}$. $\mathbf{o}'_{\mathbf{F}_{k,n}}(\mathbf{e})$ is the predicted image position of feature $\mathbf{e}$ at time t and n_{sift} is the number of all features from all $\mathcal{F}_{k,n}$ in this frame. For each camera c, these predicted image positions are obtained from feature positions at the previous time step as follows: Each feature $\mathbf{e}$ at time step $t-1$ is projected back onto the final cloth surface at $t-1$ using the camera matrix of camera c. For each feature $\mathbf{e}$, this yields a 3D position on the surface of the mesh $\mathbf{p_{k,n-1}}(\mathbf{e})$, expressed in barycentric coordinates relative to the enclosing mesh triangle. The position at the current time step t, $\mathbf{p'_{k,n}}(\mathbf{e})$, is predicted from $\mathbf{p_{k,n-1}}(\mathbf{e})$ by the cloth simulation. The predicted image positions $\mathbf{o}'_{I_{k,n}}(\mathbf{e})$ are then obtained by reprojecting $\mathbf{p'_{k,n}}(\mathbf{e})$ back into each respective camera view.

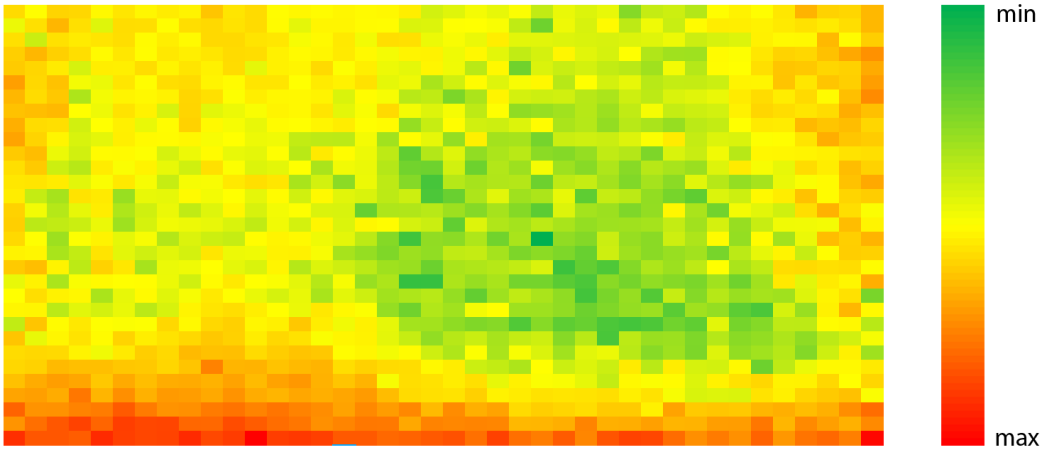


Figure 10.5: Plot of the energy function over k_{bend} on the horizontal and d_n on the vertical axis. Red color represents high energy, green low. A clear minimal energy area can be spotted near the lower right.

When evaluating E_{fit}, both E_{sil} and E_{sift} are evaluated for each frame t after the new model pose is determined according to the method in Section 10.6, using $n_{iter} = 16$ cloth simulation iterations. Please note that we intentionally formulate our energy function in terms of image features of the original reference sequence, and not in terms of a 3D comparison to the tracked template from the reference performance. In this way we stay as close as possible to the measured data and prevent unintended fitting to potential inaccuracies in the tracked 3D reference performance. In addition, our formulations of E_{sil} and E_{sift} are memory efficient as they only require storage of 2D silhouette rim points and 2D feature locations for each input frame. Holding all image data of the reference sequence simultaneously in memory would be impossible, and we therefore refrain from more complex distance functions such as distance-field based evaluations.

The combination of silhouette and SIFT features is essential for our goodness-of-fit measure. Silhouette data alone would not suffice since a lot of information on cloth behavior can be extracted from the inner regions of the cloth in all images. Silhouette information is important to asses the overall appearance of the cloth boundaries. We would also like to note that our feature handling with correspondence re-computation for each frame pair usually manages to make the majority of feature points found in all images accessible to the optimization. Multi-frame correspondence estimation more frequently leads to lost feature tracks, in particular on motions with frequent turns and occlusions.

Unfortunately, the error function E_{fit} is non-convex and exhibits many local minima, as shown in Figure 10.5 for two of the three simulation parameters (k_{bend} and d_n). This multi-modalness comes as no surprise since the cloth simulation behavior is highly non-linear causing potentially large changes in geometric cloth appearance for only small changes in ρ. Globally, there is a distinct minimal parameter region which we have to find. Any form of gradient-based optimization strategy would fail to achieve this goal. We therefore resort to a sampling-based minimization strategy which is suitable for such multi-modal error landscapes. In particular, we employ a greedy stochastic search similar to simulated annealing. The minimization is initialized by placing K^3 (typically $K = 4$) samples on regularly spaced grid positions of the space of simulation parameters, with $d_n \in [0, 10]$, $d_t \in [0, 50]$, and $k_{bend} \in [0, 1]$. The sample with the lowest error, $\tilde{\rho}$, is used as starting point for the greedy optimization $\rho_1 = \tilde{\rho}$. To this end, we first estimate a particle distribution from which to generate new samples. In our case, this distribution is a three-dimensional Gaussian $\mathcal{N}_n(\mu(n), \Gamma(n))$, whose mean $\mu(n)$ and diagonal covariance matrix $\Gamma(n)$ are dependent on the iteration n of the optimizer. Each diagonal entry of $\Gamma(1)$, namely $\sigma_{d_n}(1)$, $\sigma_{d_t}(1)$, and $\sigma_{k_{bend}}(1)$ is set to be the average distance between the grid samples used for initialization along the respective dimension. The covariance entries are reduced over time by multiplying with a linear scaling function $\Gamma(n) = g(n)\Gamma(1)$. Here $g(1) = 1$, and it linearly decreases to $g(n) = 0$.

Starting from $\tilde{\rho}$, the greedy optimizer generates one new sample per iteration by drawing from $\mathcal{N}_n$. In each iteration, $\mu(n)$ is set to the optimal particle from the previous iteration, ρ_{n-1}. A new particle is accepted if its error is lower than the error of ρ_{n-1}, otherwise $\rho_n = \rho_{n-1}$ and a new sample is drawn. After $n_{sam} = 192$ iterations, the optimizer returns the average of the ten last particles as optimum ρ_{max}. More advanced sampling strategies exist, but many of them, such as particle filter variants, are ruled out due to the much larger number of required particle evaluations that would lead to unreasonable running times.

10.8 Results and validation

The output of our reconstruction process is a fully-rigged character template comprising of an animation skeleton, surface skinning weights, collision proxies, and a physically-based cloth model for each region with wavy apparel. In total, we captured 14 performances with a female test subject using the method described in Section 10.2, each comprising of between 580 and 1300 frames. In

those sequences, two actors wear six different types of apparel: a skirt made of a fabric with medium thickness (s1), a long skirt with a rather light material (s2), a medium length skirt with comparably stiff material (s3), a dress (s4), loose track pants (s5), and skin-tight pants (s6). For each attire, several different sequences were captured with motion styles ranging from walking over dancing to kicking and turning in the capoeira scene. Only for the first five clothing styles, we ran our full pipeline to estimate an animatable performance model. Clothing style s6 was only used to acquire motion data for new animations of the other performances. In the following we discuss several aspects of our approach. All final renderings in the paper were done by real-time transfer of the pose of the decimated mesh $\mathcal{M}$ used for simulation to the high resolution body scan $\mathcal{M}_{high}$ using the deformation transfer method proposed in Chapter 7.3. The mesh $\mathcal{M}$ is a surface mesh consisting of triangles. However, we can simply convert the triangles to tetrahedra by adding a point in normal direction. This allows us to apply the method from Chapter 7.3 directly.

10.8.1 Segmentation and Cloth Parameter Estimation

For each style of apparel, we chose one reference sequence to reconstruct an animatable performance from, Table 10.1. For skirt s1, we also reconstructed animatable models from two different reference motions, walking and dancing, to verify stability. Our segmentation approach was able to reliably identify the loose cloth areas in all reference scenes using a fixed set of segmentation thresholds. The segmentation of s1 in two different reference sequences is almost identical demonstrating the stability of our method. The segmentation of the pants in the capoeira sequence shows that our algorithm also faithfully handles tighter clothing which is less wavy than the skirts. The lower end of the pants (which is very wide in comparison to the leg) is marked as loose cloth, while segments further up on the leg receive an intermediate clothness weight since they are closer to the skin and thus less wavy. Here, our algorithm also correctly identifies the sleeves of the t-shirt as slightly wider apparel, Figure 10.2(f).

Table 10.1 lists the set of cloth simulation parameters which we estimated for the different clothing styles. According to our expectations, the stiffest skirt s3 (Figure 10.6(middle)) was found to have the highest bending resistance, the dress s4 (Figure 10.6(bottom)) made of fabric with medium density has a medium bending resistance, and the light long skirt s2 (Figure 10.6(top)) a very low one.

sequence	set name	d_n	d_t	k_{bend}	$frames$
s1a: dancing	ρ_{1a}	0.5	6.8	0.36	1300
s1b: walking	ρ_{1b}	1.3	5.8	0.26	800
s2: walking	ρ_2	1.7	44.1	0.003	1100
s3: dancing	ρ_3	0.15	12.0	0.98	1200
s4: dancing	ρ_4	0.5	0.08	0.43	800
s5: capoeira	ρ_5	2.3	2.8	0.03	400

Table 10.1: Estimated cloth simulation parameters for the skirts and the pants made of different fabrics. ρ_{1a} and ρ_{1b} were estimated for the same skirt from different reference sequences.

The visual draping characteristics of all simulated skirts closely match the draping behaviors of their real world counterparts.

When estimating parameters for the same skirt s1 from two different reference sequences (a simple walking and a dancing sequence), very similar simulation parameters were found (see parameter sets ρ_{1a} and ρ_{1b} in Table 10.1). The resulting animations with both parameter sets are, in turn, visually very similar. Although the reliability of the parameter estimation is certainly influenced by the contents of the reference sequence, this test shows that for reasonably long scenes with sufficient cloth motion, parameters can be faithfully found.

Our simulation model is physically plausible and allows realistic simulation, but it does not allow us to determine physical material parameters accurately. Comparison to measured material properties using material science test methods is therefore not straightforward. To nonetheless test the feasibility of estimated parameters, we performed a cross-validation of the computed optimal parameters for all clothing styles in terms of the cloth estimation error E_{fit}, Table 10.2. As one can see, the parameters found for each attire are the true optimal ones in comparison to all other reconstructed clothing styles.

10.8.2 User Study

Since our goal is realistic simulation of new performances, and realism can ultimately only be judged by a human observer, we performed a survey with 49 test persons. Each of the participants was shown a web page with one input camera view of the reference sequence of skirt s3, as well as three simulated performances from the same camera view. The simulated performances used the reference skeleton motion but different simulation parameters for the skirt: a medium light

material (*video A*: $d_n = 2.0$, $d_t = 4.0$, $k_{bend} = 0.3$), a material exhibiting very high drag (*video B*: $d_n = 8.0$, $d_t = 4.0$, $k_{bend} = 0.8$), and the true estimated material parameters (*video C*: ρ_3). Please note that in terms of fitting error *video A* is closer to the truth than *video B*. The users were asked to rank the simulations according to the closeness of the simulated cloth behavior to the reference input (1=most similar, 3=most dissimilar). 39 (79.6%) participants ranked video C as 1. Out of these participants 32 (82.0%) also ranked video A in second place. From those users which did not rank video C correctly, 7 still ranked video B as the highest and video C as second highest. We therefore conclude that we can capture and simulate cloth behavior realistically, and that users are even able to rank simulations according to closeness of material parameters to the optimum.

10.8.3 Creating New Animations

Given a reconstructed animation model for a person, arbitrary new motions can be simulated by simply modifying the joint motion parameters of the skeleton. All motion sequences used for new animations were either captured by our performance capture algorithm from people wearing s6, or stem from a database of motion capture files. All new skeleton motions we employ are parts of the performances that were captured with the method from Section 10.2, but not used for reconstructing an animatable model. Note that data from any type of motion capture system or key-frame animations created by an animator would be equally feasible. In all new animations, both the overall appearance of the body and the deformation behavior of the cloth are very lifelike since even subtle deformation details are realistically reproduced, Figure 10.6.

Please note that the mesh pre-processing step described in Section 10.2 may lead to holes in the geometry, that may become visible during new animations. For instance, it may become visible that there is no real upper leg geometry under the skirt. For visualization, we therefore render the collision proxies as geometry approximation of the invisible parts of the upper leg to reduce this effect. For the long skirt s2, however, the proxy visualization does not look pleasing and we therefore refrain from it, exposing sometimes the missing leg geometry. In a practical animation setting, any standard 3D package could be used to create more sophisticated geometry in those regions.

	ρ_{1a}	ρ_2	ρ_3	ρ_4
s1	132.97	137.50	137.47	134.70
s2	218.90	214.37	224.89	226.37
s3	168.97	173.54	158.26	169.10
s4	142.39	145.18	145.79	137.62

Table 10.2: Cross-validation matrix: rows = clothing style, columns = parameter set - Entries are simulation parameter fitting errors: for one particular attire, the optimized parameters are also optimal in comparison to all other estimated parameter sets.

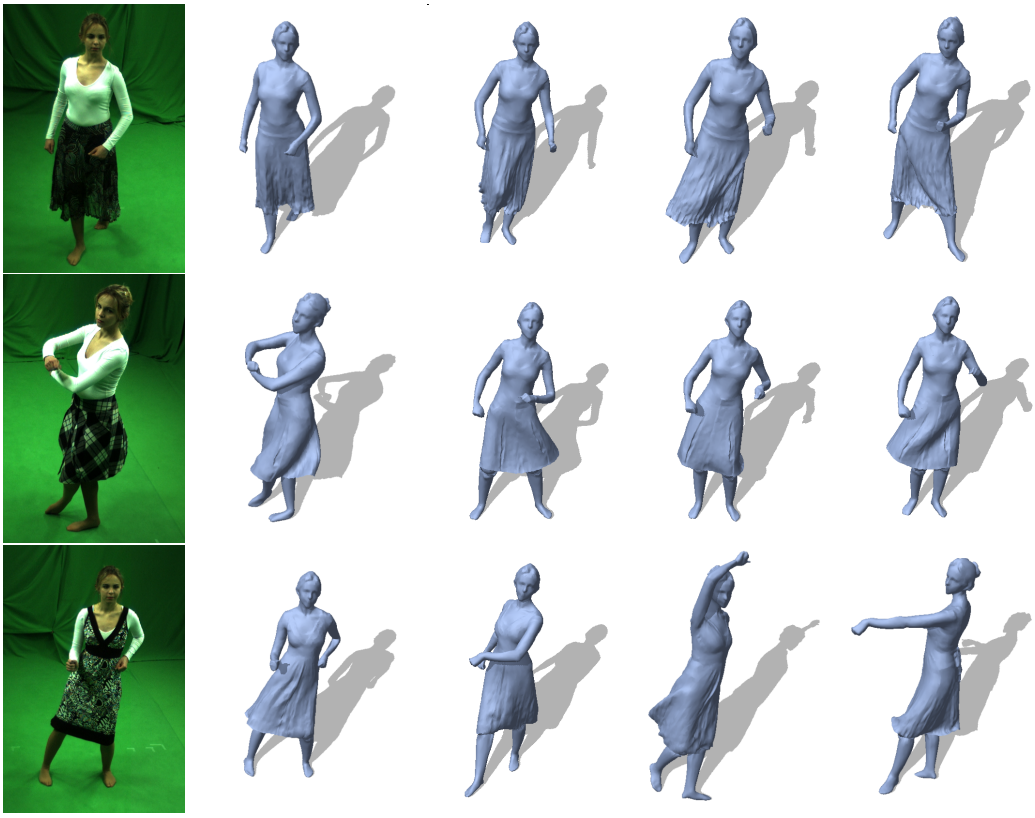
10.8.4 Performance

It takes around 2.5 hours to capture a reference performance of 1000 frames on an Intel Core 2 Duo with 3.0 GHz. All sequences we captured, both reference sequences and skeleton motions only used for new animations, were tracked fully-automatically. Segmentation of cloth takes around 5 seconds. Estimation of cloth simulation parameters takes around 3-4 hours for a reference sequence consisting of roughly 1000 frames.

The actual computation of new poses and the transfer to the higher resolution mesh of about $25k$ triangles run in real-time with ~ 60 fps on the same machine. Please note that our simulation code runs single-threaded and does not use any GPU acceleration for skinning, simulation, or detail transfer. The cloth simulation currently runs with the same step size as the animation input (i.e. 40 fps), but could easily be decoupled in both rendering of new animations as well as in the optimization procedure.

10.9 Discussion

In this Chapter we presented a method for using template based shape processing to estimate physical properties of a shape (Stoll *et al.* (2009)). We capture the performance of an actor wearing wide apparel and build an animatable virtual template model. The output of our method is a fully-rigged character template comprising of an animation skeleton, surface skinning weights, collision proxies, and a physically-based cloth model for each region with wavy apparel. We faithfully capture the behavior of the characters apparel and are able to create new animations by just applying new skeletal joint parameters that can stem from



(a) (b) (c) (d) (e)

Figure 10.6: Results for skirts 2, 3, and the dress model (from top to bottom). Column (a) input frame from reference sequence; (b) the reconstructed animatable model simulating the same pose; (c)-(e) subsequent frames of a newly created animation for which only motion parameters of the underlying skeleton were given.

any source in the standard repertoire of an animator, such as motion-capture or key-frame animations.

Our approach is subject to a few limitations. First, since we start off with a closed full-body laser scan of a person, we are limited in what type of clothing we can handle. Without manual post-processing it will not be possible to handle apparel that shows difficult topology changes while moving, such as the opening up of a coat. However, we believe that additional user input would render our method capable of handling these situations, too. Further on, the quality of cloth reconstruction is dependent on the type of reference sequence captured. If the cloth does not move very much, we will not be able to identify pieces of apparel

automatically. However, this is not a principal limitation of our method since in a general recording scenario it is fair to instruct the person to move in such a way that all pieces of apparel are exposed.

The results of the cloth parameter optimization are dependent on the input sequence, too. If the recorded sequence does not demonstrate the behavior of the material in a reasonable way (for example when recording only a walking sequence without any turns), the optimization will most likely produce a different parameter set than for a highly dynamic scene of the same skirt. Again, properly instructing the actor remedies this problem.

We would also like to note that in a certain range different cloth simulation parameters may lead to very similar cloth behavior. To a certain extent one can see this in the error plot Figure 10.5. For us, however, it is sufficient to find any such optimum within the globally low error region of the error landscape, since either of them will lead to plausible renderings.

The chosen cloth simulation method also imposes a few limitations. The maximal material stiffness we are able to simulate with the step size and number of iterations per frame we selected is limited. With our settings the cloth will always exhibit a certain amount of stretchiness that may not be there in the real fabric, even with maximal stiffness. Sometimes this is visible in our result animations. We could simulate higher stretching stiffnesses by employing hierarchical methods (as presented in Müller (2008)) or by using more sophisticated simulation methods. However, this would usually prevent us from rendering the cloth in real-time. Another consequence of the choice of simulation is that cloth simulation parameters are not independent of step size and number of iterations. This means that we always have to estimate parameters for exactly the same simulation settings that we also use for creating new animations and have to rerun the optimization if any of these settings change.

One might argue that instead of using our motion-driven cloth segmentation, one could identify loose apparel via distance thresholding between corresponding surface points of two laser scans of a subject, one with full attire and one with underwear only. However, this strategy would have several drawbacks. First, the two scans would need to be geometrically aligned. Scan alignment is a very difficult and potentially error prone process, in particular if the two scans exhibit pose differences, or if the apparel is very loose or made of very thick fabric yielding stark geometric differences between the scans. However, even if perfect alignment was achieved, one cannot easily differentiate regions of tight and wavy cloth with simple distance thresholding, let alone create a non-binary mask of

clothness weights. Such a simple geometric strategy would fail to identify regions of loose apparel that are accidentally close to the skin in a single pose, but may move around significantly in other postures. Only with a motion-driven segmentation it is possible to identify those regions, for instance the upper seam of the skirt and the areas of different intermediate clothness weights in the track pants, Figure 10.2. Admittedly, one could use a scan in underwear or fit a parametric body model like to SCAPE (Anguelov *et al.* (2005)) to represent occluded geometry at more detail. In practice we found that any such additional step is unnecessary. Our proxies are very easy-to-fit, allow for faster collision handling than more detailed geometry, and fully suffice to faithfully simulate realistic cloth collisions with the body.

Nevertheless, the level of detail that we can achieve in any of the newly created animations is high, and, unlike the performance capture approaches presented previously, it is trivial to create new sequences using the standard toolbox of the animator. As it was originally our intention, we anticipate that the reconstructed animatable actors will be used to create more realistic real-time characters for games and networked virtual environments. Admittedly, even though the models are fairly detailed, they may still be too coarse for application in feature films, where real-time rendering ability is not the primary concern. However, we are confident that a similar methodology with higher resolution sensors and more advanced manual post-processing will make a similar reconstruction approach also feasible for applications with highest production standards. To the best of our knowledge, this is the first method in the literature to capture such rich performance models.

Chapter 11

Conclusions and future work

In this thesis we proposed novel approaches to template based shape processing for solving problems in the context of shape reconstruction, performance capture and physical material estimation. We showed how to use templates for static problems like texture mapping and semantically meaningful hole-filling for surface reconstruction in Part I. We showed applications for temporal problems in the field of mesh based performance capture in Part I and II. Finally, we explained how we can apply it to dynamic problems by estimating physical parameters of a cloth simulation for optical reconstruction of detailed animatable human body models in Part III.

All of these applications had in common that we approached them as a *template fitting* process. We were given information about our target shape in the form of constraints that were either user defined or extracted from 3D scans, images, and videos. We then fit our template to the data as accurately as possible. In Part I this was achieved with a simple linear deformation scheme based on differential coordinates that performed extremely well for small scale and high-frequency detail fitting. In Part II we used a non-linear but efficient iterative volumetric template deformation process that better preserved the original shape of the template and thus was well suited for low-frequency fitting. Finally, in Part III we combined several of the approaches outlined in previous chapters and found a set of simulation parameters by minimizing a highly non-linear energy function using a stochastic sampling approach.

In Part I of this thesis, we introduced a framework for template deformation based on linear differential coordinates, and showed several applications of it. In Chapter 3 we explained how to calculate the coordinates and use linear constraints

to generate deformations. We also introduced a novel explicit formulation for addressing the translational insensitivity issue inherent to the linear formulation. This offered us a flexible and efficient framework for shape processing, especially when dealing with detailed and high-frequency deformations.

In Chapter 4 (Stoll *et al.* (2006b)), we introduced a novel method for constrained texture mapping. Instead of viewing texture mapping as parameterization problem, we formulated it as template fitting process where the image formed a planar template that was matched to the mesh geometry. This allowed us to directly work with any type of manifold surface, in particular high-genus surfaces, without any additional processing.

In Chapter 5 (Stoll *et al.* (2006a)), we showed how shape reconstruction from 3D scans could be approached as template fitting process. It is a generalization of the image plane fitting process to arbitrary 3D models. This view enabled us to incorporate semantic knowledge into the process of shape reconstruction. Even if large parts of the model were missing we were able to reconstruct a meaningful model, making it also robust to noise and outliers.

In Chapter 6 (de Aguiar *et al.* (2007a,b,c, 2008a); Gall *et al.* (2009)), we used the knowledge gained for shape reconstruction and applied it to the fields of animation and performance capture. Approaching these problems as shape fitting process based on a deformable mesh model allowed us to abandon the traditional parameterization based on a kinematic skeleton. A deformable mesh model represented a much more versatile representation and, in combination with the right feature extraction and constraint generation techniques, allowed us to reconstruct and track performers wearing wide apparel and even a kimono.

In Part II of this thesis, we introduced a novel non-linear framework for template deformation based on a volumetric mesh and an iterative differential coordinates based solver. In Chapter 7 we explained how linear differential coordinates could be extended using a simple iterative process to efficiently produce natural non-linear deformations. The volumetric formulation and implicit rotation handling allowed us to handle large global pose changes during deformation intuitively since local shape preservation was enforced.

In Chapter 8 (Stoll *et al.* (2007)), we showed how to use our non-linear framework to create an interactive shape editing application. It provided an intuitive interface and modeling metaphor to edit shapes and was robust even under extremely large deformation constraints.

In Chapter 9 (de Aguiar *et al.* (2008a); Stoll *et al.* (2007)), we presented how to use our non-linear template fitting framework for performance capture. We used the insights gained in Part I to build a more robust fitting process for global pose capture from multi-view video input. Abandoning traditional motion skeletons allowed us to capture performers in a spatio-temporally coherent way, even when wearing wide apparel like a skirt. In combination with the model refinement techniques from in Chapter 6, we could exceed the capabilities of marker-based optical capturing systems that are widely used in industry, and capture natural and life-like animation sequences. Using the non-linear deformation approach allowed us to alleviate many of the drawbacks that resulted from using the linear framework in Chapter 6. Handling rotations implicitly and preserving local volume enabled us to generate deformations which were more natural and plausible.

In Part III of this thesis, we built upon the results from the previous parts and introduced a method for estimating physical material parameters using template based shape processing. The template based shape processing methods presented in previous parts allowed us to reconstruct high quality spatio-temporally coherent shape animations. The quality of these reconstructions us enabled us to learn how our template shape reacts to external impulses and reproduce the behavior under new conditions.

In Chapter 10 (Stoll *et al.* (2009)), we presented a novel method that allowed us to optically reconstruct detailed animatable human body models. Previous performance capture method were limited to playing back the original reconstructed animation and did not allow for direct editing. Capitalizing upon the techniques presented in previous parts of this thesis allowed us to fit a fully-rigged character template comprising of an animation skeleton, surface skinning weights, collision proxies, and a physically-based cloth model for each region with wavy apparel, to a set of multi-view videos accurately. This rich performance model allowed us to faithfully capture the behavior of the characters apparel and create new animations by simply applying new skeletal joint parameters. This was the first approach in the literature to solve this complex problem.

The template frameworks and applications we introduced in this thesis show that template based shape processing is an important tool for computer graphics and computer vision applications. Whenever it is necessary to incorporate semantic knowledge about a shape into the processing pipeline because the input information is otherwise too sparse for a reconstruction, a template can simplify

processing dramatically. We have shown that template based fitting processes enables us to perform shape reconstruction and performance capture at much higher detail than was previously possible. We are convinced that future research in the area of template based shape processing will enable us to create reconstructions of scenes and actors in a quality that cannot be distinguished from the real input data anymore.

Currently, the process of template selection still requires user input (i.e. selecting a template from a database or providing a 3D scan of the performer). An interesting field of future research is how to automate this process or even estimate the template from the sparse input data themselves. This is especially interesting for temporal data such as animated point clouds extracted from depth cameras and stereo and visual hull reconstructions from multi-view video. While there has been some research on extracting a base template model from these kind of data (Wand *et al.* (2007, 2009)), we are still not able to reconstruct templates that match the quality of a 3D scan or a modeled shape. Being able to reconstruct the template from time varying data would allow handling of topology changes in the data, which is problematic with the current formulation, as the topology is fixed to that of the input template. One of the final goals would be to develop an automatic pipeline that reconstructs a high-quality animatable template of a performer, including accurately simulated layered apparel, muscle deformation during motion, detailed geometry as well as texture and reflectance properties from just a set of input videos.

Another area of future work is the development of more sophisticated fitting processes. All three frameworks introduced in this thesis are based on optimization strategies. Recent work such as Wang & Popović (2009) has shown that it is also possible to approach this issue as an information retrieval problem. Given a large enough database containing the input information connected with the respective desired output shape it is possible to vastly simplify the fitting process by initializing it with the best match from the database before continuing with a simple fitting process. We are convinced that it is possible to develop a system that can estimate the pose of a performer simply based on efficient database lookups in real-time.

References

Ahmed, N., Theobalt, C., Dobrev, P., Seidel, H.P. & Thrun, S. (2008a). Robust fusion of dynamic shape and normal capture for high-quality reconstruction of time-varying geometry. In *IEEE Conference on Computer Vision and Pattern Recognition*. 90

Ahmed, N., Theobalt, C., Rössl, C., Thrun, S. & Seidel, H.P. (2008b). Dense correspondence finding for parametrization-free animation reconstruction from video. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. 29, 79

Aiger, D., Mitra, N.J. & Cohen-Or, D. (2008). 4-points congruent sets for robust pairwise surface registration. In *ACM Transactions on Graphics*, 1–10. 18

AIM@SHAPE (2004). Shape repository. 64

Alexa, M. (1999). Merging polyhedral shapes with scattered features. In *International Conference on Shape Modeling and Applications*, 202. 50

Alexa, M. (2001). Local control for mesh morphing. In *Proceedings of the International Conference on Shape Modeling & Applications*, 209. 22

Allen, B., Curless, B. & Popović, Z. (2002). Articulated body deformation from range scans. *ACM Transactions on Graphics*, **21**, 612–619. 19, 20

Allen, B., Curless, B. & Popović, Z. (2003). The space of human body shapes: reconstruction and parameterization from range scans. *ACM Transactions on Graphics*, **22**, 587–594. 19, 20

Allen, B., Curless, B., Popović, Z. & Hertzmann, A. (2006). Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis. In *Proc. of SCA*, 147–156. 29

REFERENCES

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J. & Davis, J. (2005). SCAPE: Shape completion and animation of people. In *ACM TOG (Proc. SIGGRAPH '05)*. 19, 30, 61, 151

Au, O.K.C., Tai, C.L., Liu, L. & Fu, H. (2006). Dual Laplacian editing for meshes. *IEEE Transaction on Visualization and Computer Graphics*, **12**, 386–395. 24, 98

Balan, A.O., Sigal, L., Black, M.J., Davis, J.E. & Haussecker, H.W. (2007). Detailed human shape and pose from images. In *Proc. CVPR*. 30

Baraff, D. & Witkin, A. (1998). Large steps in cloth simulation. In *Proceedings ACM SIGGRAPH*, 43–54. 25

Baraff, D., Witkin, A. & Kass, M. (2003). Untangling cloth. *ACM Transactions on Graphics*, **22**, 862–870. 26

Baran, I. & Popović, J. (2007). Automatic rigging and animation of 3d characters. *ACM TOG (Proc. SIGGRAPH '07)*. 132

Beraldin, J.A., El-Hakim, S.F. & Blais, F. (1995). Performance evaluation of three active vision systems built at the national research council of canada. 16

Bernardini, F. & Rushmeier, H.E. (2002). The 3d model acquisition pipeline. *Computer Graphics Forum*, **21**, 149–172. 16

Besl, P.J. & McKay, N.D. (1992). A method for registration of 3-d shapes. *IEEE Trans. Pat. Anal. and Mach. Intel.*, **14**, 239–256. 66

Bhat, K.S., Twigg, C.D., Hodgins, J.K., Khosla, P.K., Popović, Z.Z. & Seitz, S.M. (2003). Estimating cloth simulation parameters from video. In *Proc. of SCA*. 132, 139

Botsch, M. & Kobbelt, L. (2004). A remeshing approach to multiresolution modeling. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, 185–192. 73

Botsch, M. & Sorkine, O. (2008). On linear variational surface deformation methods. *IEEE TVCG*, **14**, 213–230. 36, 39

Botsch, M., Bommes, D. & Kobbelt, L. (2005). Efficient linear system solvers for geometry processing. In *11th IMA conference on the Mathematics of Surfaces*. 41

Botsch, M., Pauly, M., Gross, M. & Kobbelt, L. (2006a). Primo: Coupled prisms for intuitive surface modeling. In *Proc. Symposium on Geometry Processing*, 11–20. 24

Botsch, M., Sumner, R., Pauly, M. & Gross, M. (2006b). Deformation transfer for detail-preserving surface editing. In *Proc. Vision, Modeling and Visualization*, 357–364. 23, 99

Botsch, M., Pauly, M., Wicke, M. & Gross, M. (2007). Adaptive space deformations based on rigid cells. *Computer Graphics Forum*, **26**, 339–347. 24, 113

Bradley, D., Popa, T., Sheffer, A., Heidrich, W. & Boubekeur, T. (2008). Markerless garment capture. *ACM TOG (Proc. SIGGRAPH '08)*. 130, 132

Breen, D.E., House, D.H. & Wozny, M.J. (1994). Predicting the drape of woven cloth using interacting particles. In *Proceedings ACM SIGGRAPH*, 365–372. 25

Bridson, R., Fedkiw, R. & Anderson, J. (2002). Robust treatment of collisions, contact and friction for cloth animation. In *Proceedings ACM SIGGRAPH*. 26

Brown, B. & Rusinkiewicz, S. (2004). Non-rigid range-scan alignment using thin-plate splines. In *Symposium on 3D Data Processing, Visualization, and Transmission*. 20

Brox, T., Bruhn, A., Papenberg, N. & Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *European Conference on Computer Vision*, 25–36. 81

Carmo, M.D. (1976). *Differential Geometry of Curves and Surfaces*. Prentice-Hall. 36

Carranza, J., Theobalt, C., Magnor, M. & Seidel, H.P. (2003). Free-viewpoint video of human actors. *Proc. ACM SIGGRAPH*, 569–577. 29, 81, 82, 118

Chen, Y. & Medioni, G. (1995). Description of complex objects from multiple range images using an inflating balloon model. *Computer Vision and Image Understanding*, **61**, 325–334. 19

Choi, K.J. & Ko, H.S. (2005). Research problems in clothing simulation. *Computer-Aided Design*, **37**, 585–592. 25

Chui, H. & Rangarajan, A. (2003). A new point matching algorithm for non-rigid registration. *Comput. Vis. Image Underst.*, **89**, 114–141. 20

Coquillart, S. (1990). Extended free-form deformation : a sculpturing tool for 3D geometric modeling. Research Report RR-1250, INRIA. 20

de Aguiar, E. (2008). Animation and performance capture using digitized models. In *PhD Thesis*. 9, 81, 87, 120

de Aguiar, E., Theobalt, C., Stoll, C. & Seidel, H.P. (2007a). Marker-less 3d feature tracking for mesh-based motion capture. In *Human Motion - Understanding, Modeling, Capture and Animation*, vol. 4814, 1–15. 6, 8, 23, 33, 79, 80, 81, 82, 83, 88, 89, 120, 154

de Aguiar, E., Theobalt, C., Stoll, C. & Seidel, H.P. (2007b). Marker-less deformable mesh tracking for human shape and motion capture. In *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 6, 2502–2509. 6, 8, 23, 30, 33, 79, 80, 81, 82, 83, 88, 89, 120, 154

de Aguiar, E., Theobalt, C., Stoll, C. & Seidel, H.P. (2007c). Rapid animation of laser-scanned humans. In *IEEE Virtual Reality 2007*, 223–226. 6, 8, 23, 33, 79, 80, 81, 82, 83, 154

de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.P. & Thrun, S. (2008a). Performance capture from sparse multi-view video. In *ACM TOG (Proc. SIGGRAPH'08)*. 6, 7, 9, 23, 24, 30, 33, 79, 80, 87, 90, 95, 117, 119, 120, 122, 123, 130, 133, 154, 155

de Aguiar, E., Theobalt, C., Thrun, S. & Seidel, H.P. (2008b). Automatic conversion of mesh animations into skeleton-based animations. *Proc. Eurographics EG'08*. 124, 133, 134

Der, K.G., Sumner, R.W. & Popovic, J. (2006). Inverse kinematics for reduced deformable models. In *Proc. ACM SIGGRAPH*, 1174–1179. 24

Eberhardt, B., Weber, A. & Strasser, W. (1996). A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, **16**, 52–59. 25

Eckstein, I., Surazhsky, V. & Gotsman, C. (2001). Texture mapping with hard constraints. *Computer Graphics Forum*, **20**. 49

Fattal, R., Lischinski, D. & Werman, M. (2002). Gradient domain high dynamic range compression. In *Proceedings SIGGRAPH*, 249–256. 22

Fischler, M.A. & Bolles, R.C. (1987). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Readings in computer vision: issues, problems, principles, and paradigms*, 726–740. 77

Fleishman, S., Cohen-Or, D. & Silva, C.T. (2005). Robust moving least-squares fitting with sharp features. *ACM Transactions on Graphics*, **24**, 544–552. 77

Floater, M.S. & Hormann, K. (2005). Surface parameterization: a tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*, 157–186. 48

Fu, H., Au, O.K.C. & Tai, C.L. (2007). Effective derivation of similarity transformations for implicit Laplacian mesh editing. *Computer Graphics Forum*, **26**, 34–45. 22

Furukawa, Y. & Ponce, J. (2007). Accurate, dense, and robust multi-view stereopsis. In *CVPR*. 17

Gall, J. (2009). Filtering and optimization strategies for markerless human motion capture with skeleton-based shape models. In *PhD Thesis*. 9, 88

Gall, J., Rosenhahn, B., Brox, T. & Seidel, H.P. (2008). Optimization and filtering for human motion capture – a multi-layer framework. *International Journal of Computer Vision*. 134

Gall, J., Stoll, C., de Aguiar, E., Theobalt, C., Rosenhahn, B. & Seidel, H.P. (2009). Motion capture using joint skeleton tracking and surface estimation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8. 6, 9, 23, 30, 33, 79, 80, 87, 90, 130, 131, 133, 154

Garland, M. & Heckbert, P.S. (1997). Surface simplification using quadric error metrics. In *Proc. ACM SIGGRAPH*, 209–216. 106, 110

Gleicher, M. & Ferrier, N. (2002). Evaluating video-based motion capture. *Computer Animation*, **0**, 75. 27

Goesele, M., Curless, B. & Seitz, S.M. (2006). Multi-view stereo revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2402–2409. 17, 86

Goldenthal, R., Harmon, D., Fattal, R., Bercovier, M. & Grinspun, E. (2007). Efficient simulation of inextensible cloth. *ACM Transactions on Graphics*, **26**, 49. 26

Golub, G.H. & Loan, C.F.V. (1996). *Matrix Computations*. The Johns Hopkins University Press, 3rd edn. 41, 101

Gotsman, C., Gu, X. & Sheffer, A. (2003). Fundamentals of spherical parameterization for 3d meshes. *ACM Transactions on Graphics*, **22**, 358–363. 49

Gross, M., Würmlin, S., Näf, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Gool, L.V., Lang, S., Strehlke, K., Moere, A.V. & Staadt, O. (2003). blue-c: a spatially immersive display and 3d video portal for telepresence. *ACM TOG*, **22**, 819–827. 29

Hasler, N., Asbach, M., Rosenhahn, B., Ohm, J.R. & Seidel, H.P. (2006). Physically based tracking of cloth. In *In Proc. VMV 2006*, 49–56. 132

Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B. & Shum, H.Y. (2006). Subspace gradient domain mesh deformation. In *Proc. ACM SIGGRAPH*, 1126–1134. 23, 38, 107

Igarashi, T., Moscovich, T. & Hughes, J.F. (2005). As-rigid-as-possible shape manipulation. In *Proc. ACM SIGGRAPH*, 1134–1141. 22

James, D.L. & Twigg, C.D. (2005). Skinning mesh animations. *ACM TOG (Proc. SIGGRAPH'05)*. 134

Johnson, A. (1997). *Spin-Images: A Representation for 3-D Surface Matching*. Ph.D. thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. 77

Jones, T.R., Durand, F. & Zwicker, M. (2004). Normal improvement for point rendering. *IEEE Computer Graphics and Applications*, **24**, 53–56. 62

Ju, T., Schaefer, S. & Warren, J. (2005). Mean value coordinates for closed triangular meshes. In *Proc. ACM SIGGRAPH*, 561–566. 24, 106

Kähler, K., Haber, J., Yamauchi, H. & Seidel, H.P. (2002). Head shop: Generating animated head models with anatomical structure. In *ACM SIGGRAPH Symposium on Computer Animation*, 55–64. 19, 61

Kaldor, J.M., James, D.L. & Marschner, S. (2008). Simulating knitted cloth at the yarn level. In *ACM Transactions on Graphics*, 1–9. 26

Karni, Z. & Gotsman, C. (2000). Spectral compression of mesh geometry. In *SIGGRAPH*, 279–286. 49

Kavan, L., Collins, S., Žára, J. & O'Sullivan, C. (2007). Skinning with dual quaternions. In *Symposium on Interactive 3D graphics and games*, 39–46. 133

Kazhdan, M., Bolitho, M. & Hoppe, H. (2006). Poisson surface reconstruction. In *Proceedings Symposium on Geometry Processing*, 61–70. 16, 19

Kazhdan, M.M. (2005). Reconstruction of solid models from oriented point sets. In *Symposium on Geometry Processing*, 73–82. 16, 19

Kircher, S. & Garland, M. (2008). Free-form motion processing. *ACM Transactions on Graphics*, **27**, 1–13. 123

Kobbelt, L., Campagna, S., Vorsatz, J. & Seidel, H.P. (1998). Interactive multi-resolution modeling on arbitrary meshes. In *Proceedings SIGGRAPH*, 105–114. 21

REFERENCES

Kobbelt, L., Vorsatz, J., Labsik, U. & Seidel, H.P. (1999). A shrink wrapping approach to remeshing polygonal surfaces. *Compututer Graphics Forum*, **18**, 119–130. 19

Kolb, A., Barth, E., Koch, R. & Larsen, R. (2009). Time-of-flight sensors in computer graphics. In *Proceedings Eurographics (State-of-the-Art Report)*. 16

König, S. & Gumhold, S. (2008). Image-based motion compensation for structured light scanning of dynamic surfaces. *International Journal of Intelligent Systems Technologies and Applications*, **5**, 434–441. 16

Kraevoy, V. & Sheffer, A. (2004). Cross-parameterization and compatible remeshing of 3D models. *ACM Transactions on Graphics*, **23**, 861–869. 61

Kraevoy, V. & Sheffer, A. (2005). Template-based mesh completion. In *Symposium on Geometry Processing*, 13–22. 19

Kraevoy, V., Sheffer, A. & Gotsman, C. (2003). Matchmaker: constructing constrained texture maps. *ACM Trans. Graph.*, **22**, 326–333. 49

Kutulakos, K.N. & Seitz, S.M. (2000). A theory of shape by space carving. *Int. J. Comput. Vision*, **38**, 199–218. 17

Lee, A., Moreton, H. & Hoppe, H. (2000). Displaced subdivision surfaces. In *Proc. ACM SIGGRAPH*, 85–94. 107

Lévy, B. (2001). Constrained texture mapping for polygonal meshes. In *SIGGRAPH*, 417–424. 48, 49

Lévy, B. (2003). Dual domain extrapolation. *ACM Transactions on Graphics*, **22**, 364–369. 19

Lévy, B., Petitjean, S., Ray, N. & Maillot, J. (2002). Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics*, **21**, 362–371. 48

Li, H., Adams, B., Guibas, L.J. & Pauly, M. (2000). Robust single view geometry and motion reconstruction. In *ACM Transactions on Graphics (SIGGRAPH ASIA)*. 20, 76

Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rössl, C. & Seidel, H.P. (2004). Differential coordinates for interactive mesh editing. In *Proc. of Shape Modeling International*, 181–190. 22, 23

Lipman, Y., Sorkine, O., Levin, D. & Cohen-Or, D. (2005). Linear rotation-invariant coordinates for meshes. *ACM Transactions on Graphics*, **24**, 479–487. 22

Lipman, Y., Cohen-Or, D., Ran, G. & Levin, D. (2007). Volume and shape preservation via moving frame manipulation. *ACM Transactions on Graphics*, **26**. 22

Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proc. ICCV*, 1150–1157. 82, 121, 142

Matusik, W., Buehler, C., Raskar, R., Gortler, S. & McMillan, L. (2000). Image-based visual hulls. In *ACM TOG (Proc. SIGGRAPH'00)*. 29

Menache, A. (1999). *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann. 27

Meyer, M., Desbrun, M., Schröder, P. & Barr, A. (2002). Discrete differential-geometry operators for triangulated 2-manifolds. In *Proc. VisMath*, 35–57. 37

Moeslund, T.B., Hilton, A. & Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Comput. Vis. Image Underst.*, **104**, 90–126. 28, 80

Müller, M. (2008). Hierarchical position based dynamics. In *Proc. VRIPhys*. 150

Müller, M., Heidelberger, B., Hennix, M. & Ratcliff, J. (2007). Position based dynamics. *J. Vis. Comun. Image Represent.*, **18**, 109–118. 138, 139

Nealen, A., Sorkine, O., Alexa, M. & Cohen-Or, D. (2005). A sketch-based interface for detail-preserving mesh editing. In *Proc. ACM SIGGRAPH*, 1142–1147. 22

NVidia (2009). Physx simulation framework. http://www.nvidia.com. 140

Park, S.I. & Hodgins, J.K. (2008). Data-driven modeling of skin and muscle deformation. *ACM TOG (Proc. SIGGRAPH'08)*. 30

Pauly, M., Mitra, N.J., Giesen, J., Gross, M. & Guibas, L. (2005). Example-based 3d scan completion. In *Symposium on Geometry Processing*, 23–32. 3, 20, 61, 76

Pérez, P., Gangnet, M. & Blake, A. (2003). Poisson image editing. *ACM Transactions on Graphics*, **22**, 313–318. 22

Phasespace (2009). Motion capture systems. http://www.phasespace.com. 28

Pinkall, U. & Polthier, K. (1993). Computing discrete minimal surfaces and their conjugates. *Experiment. Math.*, **2**, 15–36. 37

Popa, T., Julius, D. & Sheffer, A. (2007). Interactive and linear material aware deformations. *International Journal of Shape modeling*. 23

Popa, T., Zhou, Q., Bradley, D., Kraevoy, V., Fu, H., Sheffer, A. & Heidrich, W. (2009). Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics Forum (Proc. Eurographics)*, **28**. 90

Poppe, R. (2007). Vision-based human motion analysis: An overview. *CVIU*, **108**, 4–18. 28, 80

Pritchard, D. & Heidrich, W. (2003). Cloth motion capture. In *Proc. Eurographics EG'03*. 132

Provot, X. (1996). Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface*, 147–154. 26

Provot, X. (1997). Collision and self-collision handling in cloth model dedicated to design garments. In *Graphics interface*, 177–189. 26

Saba, S., Yavneh, I., Gotsman, C. & Sheffer, A. (2005). Practical spherical embedding of manifold triangle meshes. In *International Conference on Shape Modeling and Applications*, 258–267. 49

Sand, P., McMillan, L. & Popović, J. (2003). Continuous capture of skin deformation. *ACM TOG (Proc. SIGGRAPH'03)*. 29

Scholz, V., Stich, T., Keckeisen, M., Wacker, M. & Magnor, M. (2005). Garment motion capture using color-coded patterns. In *Proc. Eurographics EG'05*. 132

Sederberg, T. & Scott, R. (1986). Free-form deformation of solid geometric models. In *Proc. ACM SIGGRAPH*, 151–160. 20

Seitz, S.M., Curless, B., Diebel, J., Scharstein, D. & Szeliski, R. (2006). A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition*, 519–528. 17

Selle, A., Su, J., Irving, G. & Fedkiw, R. (2009). Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction. *IEEE Transactions on Visualization and Computer Graphics*, **15**, 339–350. 26

Shannon, C.E. (1949). Communication in the Presence of Noise. *Proceedings of the IRE*, **37**, 10–21. 1

Sharf, A., Alexa, M. & Cohen-Or, D. (2004). Context-based surface completion. *ACM Transactions on Graphics*, **23**, 878–887. 3, 19, 61

Sheffer, A. & Kraevoy, V. (2004). Pyramid coordinates for morphing and deformation. In *Proc. 3D Data Processing, Visualization, and Transmission*, 68–75. 23, 38

Shi, X., Zhou, K., Tong, Y., Desbrun, M., Bao, H. & Guo, B. (2008). Example-based dynamic skinning in real time. *ACM TOG (Proc. SIGGRAPH '08)*. 132

Shoemake, K. & Duff, T. (1992). Matrix animation and polar decomposition. In *Proc. of Graphics Interface*, 258–264. 45, 102

Si, H. & Gaertner, K. (2005). Meshing piecewise linear complexes by constrained delaunay tetrahedralizations. In *Proc. International Meshing Roundtable*, 147–163. 106, 110

Sorkine, O. (2005). Laplacian mesh processing. In *Eurographics STAR*, 53–70. 36, 39, 69

Sorkine, O. & Alexa, M. (2007). As-rigid-as-possible surface modeling. In *Proceedings of the fifth Eurographics symposium on Geometry processing*, 109–116. 24, 98

Sorkine, O. & Cohen-Or, D. (2004). Least-squares meshes. *International Conference on Shape Modeling and Applications*, 191–199. 50

Sorkine, O., Lipman, Y., Cohen-Or, D., Alexa, M., Rössl, C. & Seidel, H.P. (2004). Laplacian surface editing. In *Proc. Symposium on Geometry Processing*, 179–188. 22, 43, 70

Starck, J. & Hilton, A. (2007). Surface capture for performance based animation. *IEEE CGAA*, **27(3)**, 21–31. 29, 79

Stoll, C., Karni, Z., Rössl, C., Yamauchi, H. & Seidel, H.P. (2006a). Template deformation for point cloud fitting. In *Symposium on Point-Based Graphics*, 27–35. 5, 8, 23, 33, 61, 67, 154

Stoll, C., Karni, Z. & Seidel, H.P. (2006b). Geodesics guided constrained texture deformation. In *Pacific Graphics*, vol. 14, 144–152. 5, 8, 23, 33, 47, 55, 154

Stoll, C., de Aguiar, E., Theobalt, C. & Seidel, H.P. (2007). A volumetric approach to interactive shape editing. Research Report MPI-I-2007-4-004, Max-Planck-Institut für Informatik. 6, 9, 24, 95, 109, 117, 118, 122, 154, 155

Stoll, C., Gall, J., de Aguiar, E., Seidel, H.P., Thrun, S. & Theobalt, C. (2009). Optical reconstruction of detailed animatable human body models. Research Report MPI-I-2009-4-006, Max-Planck-Institut für Informatik. 7, 9, 127, 129, 148, 155

Sumner, R.W. & Popovic, J. (2004). Deformation transfer for triangle meshes. In *Proc. ACM SIGGRAPH*, 399–405. 23, 104

Sumner, R.W., Zwicker, M., Gotsman, C. & Popovic, J. (2005). Mesh-based inverse kinematics. In *Proc. ACM SIGGRAPH*, 488–495. 24

Sumner, R.W., Schmid, J. & Pauly, M. (2007). Embedded deformation for shape manipulation. In *ACM Transactions on Graphics*, 80. 24, 113

Surazhsky, V., Surazhsky, T., Kirsanov, D., Gortler, S.J. & Hoppe, H. (2005). Fast exact and approximate geodesics on meshes. *ACM Transactions on Graphics*, **24**, 553–560. 52

Taubin, G. (1995). A signal processing approach to fair surface design. In *SIGGRAPH*, 351–358. 37

Terzopoulos, D. & Fleischer, K. (1988). Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proceedings ACM SIGGRAPH*, 269–278. 25

Terzopoulos, D. & Vasilescu, M. (1991). Sampling and reconstruction with adaptive meshes. In *IEEE Computer Vision and Pattern Recognition*, 70–75. 19

Thomaszewski, B., Pabst, S. & Straer, W. (2009). Continuum-based strain limiting. *Computer Graphics Forum*, **28**, 569–576. 26

Varady, T., Martin, R.R. & Cox, J. (1997). Reverse engineering of geometric-models: An introduction. *Computer-Aided Design*, **29**, 255–268. 19

Vicon (2009). Motion capture systems. http://www.vicon.com. 28

Vlasic, D., Baran, I., Matusik, W. & Popović, J. (2008). Articulated mesh animation from multi-view silhouettes. *ACM TOG (Proc. SIGGRAPH '08)*. 30, 130

Volino, P., Cordier, F. & Magnenat-thalmann, N. (2005). From early virtual garment simulation to interactive fashion design. *Computer-Aided Design*, **37**, 593–608. 25

von Funck, W., Theisel, H. & Seidel, H.P. (2006). Vector field based shape deformations. In *Proc. ACM SIGGRAPH*, 1118–1125. 24

Wand, M., Jenke, P., Huang, Q., Bokeloh, M., Guibas, L. & Schilling, A. (2007). Reconstruction of deforming geometry from time-varying point clouds. In *Proc. SGP*, 49–58. 20, 76, 156

Wand, M., Adams, B., Ovsjanikov, M., Berner, A., Bokeloh, M., Jenke, P., Guibas, L., Seidel, H.P. & Schilling, A. (2009). Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Transactions on Graphics*, **28**, 1–15. 20, 76, 156

Wang, R.Y. & Popović, J. (2009). Real-time hand-tracking with a color glove. *ACM Trans. Graph.*, **28**, 1–8. 156

Waschbüsch, M., Würmlin, S., Cotting, D., Sadlo, F. & Gross, M. (2005). Scalable 3D video of dynamic scenes. In *Proc. Pacific Graphics*, 629–638. 29

White, R., Crane, K. & Forsyth, D. (2007). Capturing and animating occluded cloth. In *ACM TOG (Proc. SIGGRAPH'07)*. 132

Wiktionary (2009). Template. http://en.wiktionary.org/wiki/template. 2

Xu, W., Zhou, K., Yu, Y., Tan, Q., Peng, Q. & Guo, B. (2007). Gradient domain editing of deforming mesh sequences. In *ACM TOG (Proc. SIGGRAPH '07)*. 124

Yoshizawa, S., Belyaev, A.G. & Seidel, H.P. (2003). Free-form skeleton-driven mesh deformations. In *Proc. Symposium on Solid modeling and applications*, 247–253. 107

Yoshizawa, S., Belyaev, A. & Seidel, H.P. (2007). Skeleton-based variational mesh deformations. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, **26**, 255–264. 23

Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B. & Shum, H.Y. (2004). Mesh editing with poisson-based gradient field manipulation. In *Proceedings ACM SIGGRAPH*, 644–651. 22, 23, 99, 113

Zayer, R., Rössl, C., Karni, Z. & Seidel, H.P. (2005). Harmonic guidance for surface deformation. In *Proceedings Eurographics*, 601–609. 23, 43, 51, 103

Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B. & Shum, H.Y. (2005). Large mesh deformation using the volumetric graph laplacian. In *Proc. ACM SIGGRAPH*, 496–503. 22

Zigelman, G., Kimmel, R. & Kiryati, N. (2002). Texture mapping using surface flattening via multidimensional scaling. *IEEE Transactions on Visualization and Computer Graphics*, **8**, 198–207. 54

Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S. & Szeliski, R. (2004). High-quality video view interpolation using a layered representation. *ACM TOG (Proc. SIGGRAPH '04)*. 29