

Capturing and Modeling of Deformable Objects

by

Tiberiu Popa

B.Sc., The University of Waterloo, 2002

M.Sc., The University of Waterloo, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

November 2009

© Tiberiu Popa 2009

Abstract

Modeling the behavior of deformable virtual objects has important applications in computer graphics. There are two prevalent approaches for modeling deformable objects, an active one by deforming existing virtual models and a passive one by capturing the geometry and motion of real objects. This thesis explores the problem of modeling and acquisition of objects undergoing deformations, and proposes a set of practical deformation and capturing tools.

The first contribution is a new approach to model deformation that incorporates non-uniform materials into the geometric deformation framework. This technique provides a simple and intuitive method to control the deformation using material properties that can be specified by the user with an intuitive interface or can be learned from a sequence of sample deformations facilitating realistic looking results.

Some deformable objects such as garments exhibit a complex behavior under motion and thus are difficult to model or simulate, making them suitable target for capture methods. Methods for capturing garments usually use special markers printed on the fabric to establish temporally coherent correspondences between frames. Unfortunately, this approach is tedious and prevents the capture of interesting, off-the-shelf fabrics. A marker-free approach to capturing garment motion that avoids these problems is presented in chapter three. The method establishes temporally coherent parameterizations between incomplete geometries that are extracted

at each time step using a multiview stereo algorithm, and the missing geometry is filled in using a template.

Garment motion is characterized by dynamic high-frequency folds. However, these folds tend to be shallow, making them difficult to capture. A new method for reintroducing folds into the sequence using data-driven dynamic wrinkling is presented in chapter four. The method first estimates the folds in the video footage and then wrinkle the surface using space-time deformation. The validity of the method is demonstrated on several garments captured using several recent techniques.

While this markerless reconstruction method is tailored specifically for garments, this thesis also proposes a more general method for reconstructing a consistent frame sequence from a sequence of point clouds captured using multiple video streams. The method uses optical flow to guide a local-parameterization based cross-parameterization method. This reconstruction method accumulates geometric information from all the frames using a novel correction and completion mechanism.

Table of Contents

Abstract	ii
Table of Contents	iv
List of Tables	ix
List of Figures	x
Acknowledgements	xiii
Statement of Co-Authorship	xiv
1 Introduction	2
1.1 Modeling and Acquisition of Deformable 3D Digital Models	3
1.1.1 Modeling Static 3D Models	3
1.1.2 Acquisition of Static 3D Models	4
1.1.3 Modeling Deformable Objects	6
1.1.4 Acquisition of Deformable 3D Models	8
1.2 Thesis Contributions	9
Bibliography	15
2 Material-aware Mesh Deformation	20
2.1 Introduction	20

Table of Contents

2.2	Previous Work	23
2.3	Method Overview	26
2.4	Method Details	29
2.4.1	Material Properties	29
2.4.2	Transformation Extrapolation	30
2.4.3	Vertex Repositioning	32
2.5	Positional Constraints	34
2.5.1	Finding Rotations	36
2.5.2	Solving	40
2.6	Anisotropic Materials	41
2.7	Material Learning	44
2.8	Implementation and Results	46
2.9	Summary	50
2.10	Limitations and Future Work	51
	Bibliography	57
3	Markerless Garment Capture	62
3.1	Introduction	62
3.2	Related Work	63
3.2.1	Cloth and Garment Capture	63
3.2.2	Related Geometry Processing Techniques	65
3.3	Overview	66
3.4	Acquisition	68
3.5	Multiview Reconstruction	69
3.6	Consistent Cross-Parameterization	71
3.6.1	Positioning Off-Surface Anchors	74

Table of Contents

3.6.2	Base Mesh Parameterization	77
3.7	Compatible Remeshing and Surface Completion	80
3.8	Results	83
3.9	Conclusion	85
Bibliography		90
4	Wrinkling Captured Garments Using Space-Time Data-Driven Deformation	94
4.1	Introduction	94
4.2	Previous Work	96
4.3	Background and Overview	99
4.3.1	Video Based Fold Estimation	100
4.3.2	Fold Modeling	104
4.4	Video-Based Fold Edge Extraction	105
4.5	Space-Time Deformation	107
4.5.1	Control Mechanism	108
4.5.2	Iterative Space-Time Deformation	109
4.5.3	Space-Time Rotation	109
4.5.4	Vertex Positioning	111
4.5.5	Stretch Reduction	112
4.6	Results	113
4.7	Conclusions	115
Bibliography		121
5	Globally Consistent Space-Time Reconstruction	125
5.1	Introduction	125

Table of Contents

5.2	Related Work	129
5.3	Overview	131
5.4	Cross-Parameterization	136
5.4.1	Optical Flow Tracking	138
5.4.2	Local Patch-Based Parameterization	140
5.4.3	Local Relaxation	142
5.4.4	Anchor Set Refinement	144
5.5	Analysis and Correction	146
5.6	Geometry Completion and Connectivity Combination	147
5.7	Discussion and Results	150
5.7.1	Results	151
5.7.2	Comparisons with Other Methods	152
5.8	Conclusions, Limitations and Future Work	154
	Bibliography	157
6	Conclusions	163
6.1	Contributions	163
6.1.1	Chapter 2 - Material-aware Mesh Deformation	164
6.1.2	Chapter 3 - Markerless Garment Capture	165
6.1.3	Chapter 4 - Wrinkling Captured Garments Using Space-time Data-driven Deformation	166
6.1.4	Chapter 5 - Globally Consistent Space-Time Reconstruction	167
6.2	Future Work	168
6.2.1	Quantitative Analysis and Validation	168
6.2.2	Real-time Dynamic Surface Reconstruction	169

Table of Contents

Bibliography 171

Appendices

A Linear Combinations of Transformations 175

Bibliography 177

B Markerless Garment Capture - Boundary Ellipse Fitting 178

B.1 Automatic Ellipse Fitting 179

B.1.1 Ellipse Candidate List 180

B.1.2 Pruning Criteria 181

B.2 Semi-automatic Off-surface Anchor Placement 183

B.3 More General Boundary Tracking 184

List of Tables

2.1	Model deformation timings	50
4.1	Fold edge parameters	115

List of Figures

1.1	Different types of modeling methods	4
1.2	Static geometry acquisition	5
1.3	Modeling deformable objects	6
1.4	Example of a geometric deformation	7
1.5	Results from this thesis	14
2.1	Material-aware deformation	21
2.2	Material-aware deformation	23
2.3	Algorithm flow	26
2.4	Twisting, bending, and stretching a bar	30
2.5	Deformation of a bar using only positional constraints	34
2.6	Deformation comparison	35
2.7	Comparative results	36
2.8	Error plot	38
2.9	Comparative result	39
2.10	Bending and twisting a bar with anisotropic material	41
2.11	Estimation of material stiffness from sample poses	44
2.12	Refining learned materials	53
2.13	Deformation of articulated models	53
2.14	Scaling and rotating a camel’s head	54

List of Figures

2.15	Deformation comparison for octopus tentacle	55
2.16	Material-aware deformation of cloth	55
2.17	Deformation of a camel using positional constraints	56
3.1	Overview of the technique	66
3.2	Acquisition setup.	70
3.3	Multiview reconstruction	70
3.4	Result of multiview reconstruction	72
3.5	Off-surface anchors	74
3.6	Garment boundary tracking	75
3.7	Automatic off-surface anchor positioning	77
3.8	Smooth hole completion	79
3.9	Two frames parameterized onto the base mesh	80
3.10	Template construction	82
3.11	result of surface completion using template	83
3.12	Capture results of a T-shirt	86
3.13	Capture results of a fleece vest	87
3.14	Capture results of a blue dress	88
3.15	Capture results of a large T-shirt	88
3.16	Capture results of a pink dress	89
3.17	Capture result of a long-sleeve jacket	89
4.1	Reintroducing folds into captured garments	94
4.2	Algorithm overview	96
4.3	Types of garment folds	100
4.4	Fold edge extraction	101
4.5	Deformation setup	103
4.6	Deformation	108

List of Figures

4.7	Wrinkling dresses	117
4.8	Wrinkling dresses	118
4.9	Wrinkling a t-shirt	119
4.10	Wrinkling a jacket	119
4.11	Wrinkling various outfits	120
4.12	Wrinkling various outfits	120
5.1	Topology resolution	127
5.2	Hierarchical assembly of a consistent frame sequence	129
5.3	Combining consecutive frames	133
5.4	Local patch parameterization	137
5.5	Optical Flow Correspondence	137
5.6	Optical Flow Pruning	138
5.7	Local Relaxation	142
5.8	Correction	145
5.9	Completion	148
5.10	Capture results for a hand model	152
5.11	Capture results for a T-Shirt model	153
5.12	Capture results for a hand-puppet model	154
5.13	Capture results for the real two-hands sequence	155
5.14	Comparison to Bradley et al. Due to misalignment between the frames and the template their method exhibits artifacts around the boundaries	156
B.1	Off-surface anchors motion	179
B.2	Fitted ellipse	181
B.3	Ellipse fitting on a TShirt boundary	182
B.4	Degenerate boundary	183

Acknowledgements

First, I wish to express my most sincere thanks to my supervisor Prof. Alla Sheffer for her guidance and support in all aspects of this dissertation.

I wish to thank my supervisory committee members Prof. Wolfgang Heidrich and Prof. Chen Greif for their insightful comments and discussions over the years.

In addition I would like to thank my collaborators Dan Julius, Derek Bradley, Ian South-Dickinson, James Zhou, Vladislav Kraevoy, Hongbo Fu, Tammy Boubekeur as well as all the members of the IMAGER lab that contributed to a wonderful working environment.

I wish to thank the funding organizations: UBC - University Graduate Fellowship (UGF), Natural Sciences and Engineering Research Council of Canada (NSERC) and The Mathematics of Information Technology and Complex Systems (MITACS) research network.

Finally, I wish to thank my wife Paula and my parents Anca and Stelian who were always supportive and encouraged me to enter graduate school and pursue an academic career.

Statement of Co-Authorship

This is a manuscript thesis and Chapters 2 to 5 have been co-written as papers that either have already been published or are nearly ready to be submitted to conferences and journals. I have performed some of the work presented in this thesis in collaboration with other researchers from The Imager Laboratory for Graphics, Visualization and HCI. In addition, my supervisor, Dr. Alla Sheffer, has provided me with invaluable guidance and support over the years.

For Chapter 2, the author of this thesis was the project lead in designing and developing the main deformation method and its refinements. Credit should be given to Dan Julius who contributed to the implementation as well as to the preparation of the manuscript.

For Chapter 3, the author of this thesis worked on the geometric processing algorithms particularly on the Cross Parameterization (Section 3.6) and Compatible Remeshing and Surface Completion (Section 3.7) excluding the template construction. Credit should be given to Derek Bradley whose main contribution was the acquisition setup and the stereographic reconstruction method. Tamy Boubekeur provided the meshing code.

For Chapter 4, the author of this thesis was the project lead and he designed and developed the novel time-space deformation method. Credit should be given to Qingnan Zhou who mostly contributed to the edge detection and the edge pairing (Section 4.4). Derek Bradley and Vladislav Kraevoy pre-processed the input

Statement of Co-Authorship

dataset and prepared some of the figures.

For Chapter 5, the author of this thesis was the project lead in developing the reconstruction pipeline. Credit should be given to Ian South-Dickens who largely focused on the geometry completion module (section 5.6) and partially implemented the simultaneously patch growth (section 5.4.2). Derek Bradley provided some of the input data sets.

Chapter 1

Introduction

The ancient Greeks defined Geometry as the part of mathematics that studies shape. In this digital age, we would like to represent and use shape in a computational framework. To achieve this we need a toolbox of fundamental algorithms and mathematics to process this digital geometric data, a challenge that led to the formation of the research field of digital geometry processing [CGS⁺01]. There are several fundamental problems in digital geometry processing such as shape representation, shape modeling and shape editing. This thesis focuses on the problem of *creating* or modeling digital geometry.

Digital 3D shapes are already ubiquitous in fields such as computer graphics, engineering, architecture and medicine. Nowadays, most engineering and architectural design for such ubiquitous objects as houses, bridges, boats, automobiles or planes is performed almost entirely on computers. In the medical field, diagnostics tools, such as MRI scans that yield 3D reconstruction of human organs, are now routine. Movies are deploying more and more virtual actors, and with the increasing speed of game consoles, games are using more and more complex 3D digital models. With such growing demand, creating 3D digital geometry is becoming an increasingly important and fundamental problem.

1.1 Modeling and Acquisition of Deformable 3D Digital Models

As mentioned earlier, the focus of this thesis is creation of 3D digital models, and specifically deformable models: models that change their shape over time. While the shape of many objects such as buildings, furniture and statues is static, others, for instance people or animals (performing some activity such as walking or running), tend to change their shape - that is, to deform - over time. Modeling such changes is crucial for many applications such as video games or featured films, which use dynamically changing models.

The approaches used to create 3D digital models can be classified into *modeling* and *acquisition*. Modeling approaches assume that the user starts with a "blank" canvas and creates the models from scratch. Acquisition approaches allow the user to create a digital replica of a real object. The following sub-sections present a brief overview of modeling and acquisition methods for static and deformable 3D digital models and place the work presented in this thesis in context.

1.1.1 Modeling Static 3D Models

Despite their increasing popularity, creating detailed 3D digital models remains a complex and time-consuming task. Professional commercial modeling systems such as Maya [Inc09b], Softimage XSI [Inc09c] and 3D Studio Max [Inc09a] often use splines [Gal00, DKT98] or subdivision surfaces [CC78, Kob00]. Although powerful, these sophisticated systems are generally difficult to use and they are often limited to trained professionals.

A more accessible solution is proposed by sketch-based modeling tools. These modeling tools take their cues from intuitive drawing practices and the human abil-

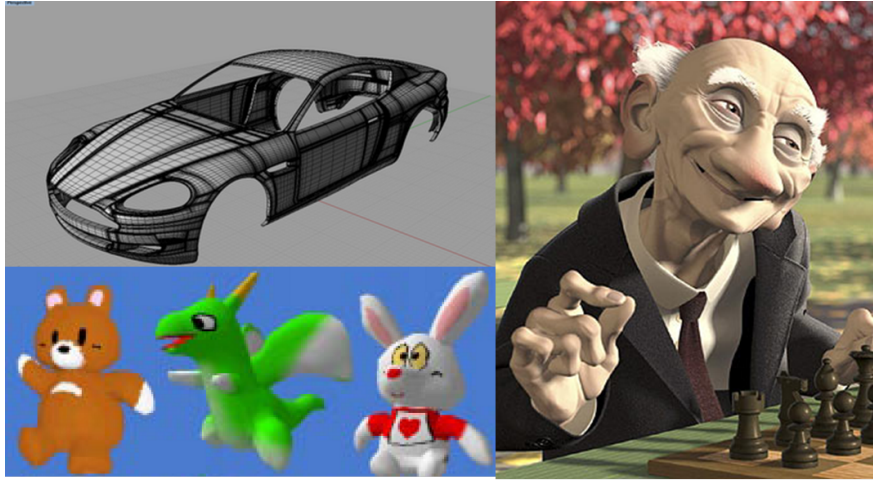


Figure 1.1: 3D geometries generated using spline surfaces(top, left), subdivision surfaces (right), and sketch-based modeling system (bottom, left)

ity to use 2D information to rapidly convey shape [IMT99, ZHH06]. These methods enable even novice users to create 3D models using a simple and intuitive 2D interface. Figure 1.3 shows static geometries created using the two types of methods.

1.1.2 Acquisition of Static 3D Models

In many scenarios the user does not have to model an object from scratch, as it is possible to create a digital replica of the real thing. Most acquisition methods begin by using a scanning hardware device, such as a laser scanner [INC09d], to generate a dense set of points on the surface. This initial point set is generally noisy and has outliers. Therefore, model reconstruction techniques are subsequently used to remove noise and outliers and create the final surface of the model [Cur00, ABK98, DG03].

One popular class of scanning devices are laser scanners [INC09d]. A laser line is projected onto the real object and a camera offset from the laser source

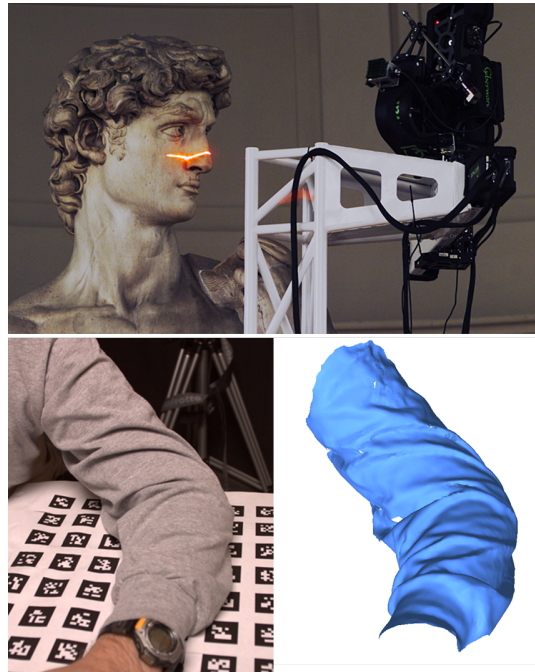


Figure 1.2: Static geometry acquisition. Top: acquisition of a statue using a laser scanner [LPC⁺00]. Bottom: acquisition of the geometry of a sleeve using multi-view stereo reconstruction [BBH08]. One of the 16 images used in the reconstruction (left). The final geometry (right)

views the laser light on the object being scanned (figure 1.2(top)). Points far away from the laser source show up at a different location in the image than points closer to the laser conveying depth information. Another popular technique to acquire 3D geometry is using multi-view stereo algorithms [BBH08] (figure 1.2(bottom)) The basic principle is to recover the depth information by simultaneously matching the same region in multiple pictures of the captured object. The multi-view stereo approach can also be used for dynamic acquisition setup by using synchronized video cameras.

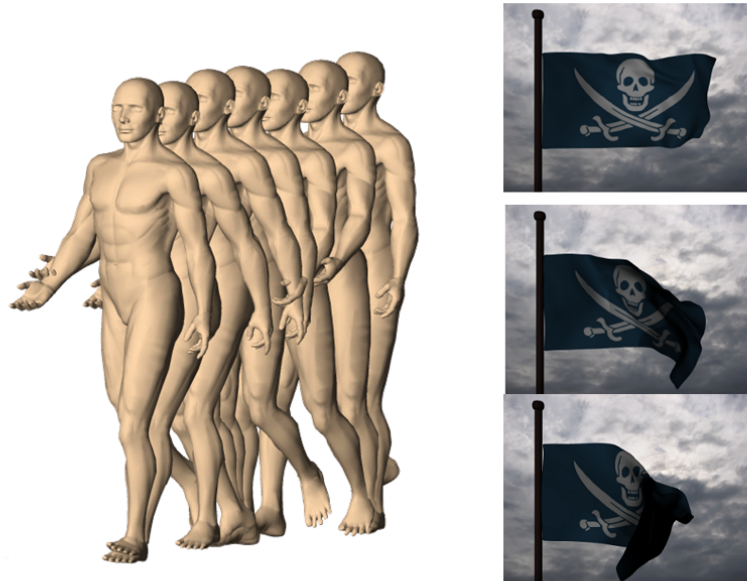


Figure 1.3: Modeling deformable objects. Left: a sequence of virtual human motion [SK04]. Right: a pirate flag blowing in the wind [BWH⁺06]

1.1.3 Modeling Deformable Objects

The modeling techniques mentioned so far are targeted at creating static digital models. But many of the objects that surround us tend to change their shape - that is, to deform. Modeling the dynamic behavior of such deformable object is an important problem with a variety of applications. Some of the popular methods for modeling deformable surfaces include skeleton methods, physics methods and geometric methods and are briefly surveyed below.

The behavior of articulated characters has been traditionally modeled using a skeleton and deformed using linear blend skinning [LAN98] or similar methods [Kv05, KCvO07]. These methods, however, cannot be extended to objects that exhibit complex deformation behavior such as cloth, because they do not have

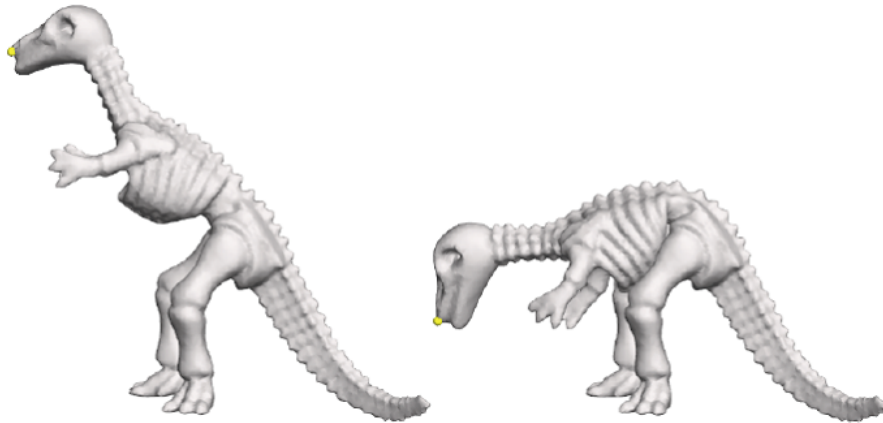


Figure 1.4: Example of a geometric deformation [BPGK06]. The deformation *handle* shown in yellow is dragged to a different position using the mouse and the shape of the model follows

a skeleton .

Physics-based approaches offer a rich and flexible way of defining the behavior of animated objects and characters, by allowing the laws of physics to determine or guide their motion [WB01]. Physically based modeling of deformable objects has a long history in mechanical engineering and materials science. In those disciplines, the main objective is to model real-world objects as accurately as possible, but the priorities are different in computer graphics, where what is often crucial is generating plausible behaviors in a computationally efficient manner while providing a high degree of user control. The methods devised in computational physics are often not fast enough or don't provide the user with enough control to be used in computer graphics.

Geometric deformation techniques [YZX⁺04, SK04, LSLCO05, ZRKS05, BPGK06, NISA07] allow the user to change the overall shape of an existing object while retaining the fine, difficult to model, local details relying only on the geometric in-

formation about the object (i.e. its shape). The user selects a handle on the object, moves it using a simple mouse interface, and the shape of the object follows in an intuitive and realistic way (figure 1.4). Deformations are typically considered intuitive if they imitate the real-world surface behavior in a physically plausible way. Since physically accurate simulations are often not required, physical correctness can be traded for higher performance and ease of use. Geometric deformations are appealing because they allow even an untrained user to produce plausible deformation by directly manipulating only a small set of controls at a fraction of the computational cost of physics-based methods. This thesis propose in chapter two a hybrid method that incorporates elements from physics based methods into a geometric deformation framework.

1.1.4 Acquisition of Deformable 3D Models

An alternative way to create deformable models including both geometry and motion is to capture them. Accurate acquisition of the geometry of such deformable objects has many important applications. In engineering it can be used to observe the behavior of complex surfaces, for example the wings of an airplane under stress, and to validate physical theories and simulations. In the medical field, it can be used to conveniently visualize the mechanics of human organs such as the heart. In game and movie industries, it can be used as a substitute for expensive simulations or time-consuming artist-driven animations. With so many applications, acquisition of deformable objects makes for a very attractive research topic. Currently there are two prevalent technologies for capturing deformable surfaces: one using real-time scanners [WAO⁺09] and the second using multiple video streams [dAST⁺08, VBMP08]. The real-time scanning technology, however, currently outputs only a few frames per second as opposed to the video setting that

can output 60 or more frames per second providing more information. Both these technologies, however, require complex complementary algorithms to convert the raw data provided by the hardware setup into both *geometry* and *motion* of the captured object through time. This thesis makes several contributions in this direction using a video-streaming approach.

1.2 Thesis Contributions

This thesis explores the problem of modeling and acquisition of deformable objects, and proposes a set of methods and tools that have a wide range of applications in computer graphics. More specifically, this thesis presents a novel method for geometric deformation (chapter 2), a system for detailed capture of the geometry of moving garments (chapter 3), a method for augmenting the capture data adding difficult to acquire high-frequency folds (chapter 4), and a system for capturing more general deformable objects (chapter 5). This section presents an overview of these contributions.

As noted in [BS08] an important challenge for geometric deformations is to design tools that provide predictable, physically plausible and esthetically pleasing results efficiently and with minimal effort from the user. The deformation behavior of an object is closely linked to the physical properties of its material. This behavior cannot always be inferred solely from the shape of an object as was largely done by previous methods [BS08]. Therefore, this thesis propose a hybrid technique that incorporates elements from physics-based methods into a geometric deformation framework for more realistic results.

Chapter 2 introduces material-aware model deformation, a novel technique that uses material properties to guide geometric deformations. More specifically, this approach provides a simple and intuitive method to control the local resistance

1.2. Thesis Contributions

to *bending* and *shearing* throughout the model. These material properties can be specified by the user with an intuitive paint-like interface or else learned from a sequence of sample deformations. By combining elements from physics based modeling and geometric deformation methods, this thesis presents an efficient and intuitive deformation method that allows even novice users to generate physically plausible deformations.

While deformation techniques can be used to conveniently edit or modify existing models, they still require a detailed input geometry to start from. A popular alternative to creating such models from scratch is to acquire 3D virtual replicas of real life objects. Until recently, 3D acquisition methods focused on capture of static shapes. Over the past few years, however, scanning methods for dynamic geometries have improved because of the development of both experimental research systems and commercial products.

Capturing dynamic deformable objects has two main interconnected challenges. One is to reconstruct the complete *geometry* in each observed frame and the second is to reconstruct the *motion* of this geometry through time. The geometry reconstructed separately at each individual time frame often will have part of the data missing due to self-occlusions. This missing geometry has to be filled. Reconstructing the motion of the garment implies finding the location of every point on the geometry in each individual frame. Therefore, given the geometry of each individual frame, reconstructing the motion is equivalent to finding a correspondence between the geometries at different time frames. Finding these correspondences is difficult in general and usually requires some knowledge of the temporal behavior of the object to use as a prior. Stronger priors should make the problem more tractable, but may also limit the scope of a method.

One example of deformable surface acquisition, or capture, that has recently seen significant research interest is the acquisition of garments. Moving garments

are ubiquitous in our everyday life and therefore modeling realistic garments is an essential part of modeling a realistic virtual environment. Garments exhibit a very complex behavior that depends not only on the movement of the person wearing them, but also on specific physical properties of the fabric they are made of. For instance the motion of a silk garment can be quite different than the motion of a cotton or a Gore-Tex one. Thus, artist driven modeling of garment motion is very challenging. Physical simulation of garments delivers realistic garment motion, but usually requires fabric specific parameters and the motion is difficult to control. Garment acquisition provides a data-driven alternative to cloth simulation that is independent of any intrinsic parameters of the fabric.

Earlier works on garment acquisition used special patterns printed on the garments to track the motion of the garment through time [WCF07, GKB03]. This approach poses several important limitations on the garment capture process. First, some materials resist the application of patterns, while for others the printing process is too expensive. Second, the resolution of the captured garment motion is usually limited by the resolution of the printing process. Lastly, placing markers on the garment may change the natural behavior of fabric. Therefore markerless methods for garment capture are desirable. Chapter 3 presents such a method. The approach uses, as a prior, the specific geometric properties of cloth to capture both the shape and motion of generic off-the-shelf garments. The captured sequences can be re-rendered with different materials or textures as needed and can be used to augment the scene with virtual elements such as additional synthetic lights.

Garment motion is characterized by a dense network of small dynamic high-frequency folds (Figure 1.5-middle). To appear realistic, a captured garment should exhibit such folds. However, these folds tend to be very shallow, making them difficult to capture and separate from noise. To increase the realism the capture is augmented by introducing folds. Adding believable folds procedurally into the

capture is difficult because the formation of folds is closely linked to the motion speed of the character and the material of the cloth. Physics based simulation could be used to add realistic folds, but requires complete knowledge of the motion and fabric of the garment, something that might not be available.

Instead a hybrid method is chosen which uses a data-driven approach to add believable, dynamic fine-folds into the captured sequence (Chapter 4). The method first estimates the shape and position of the folds from the video sequence and then wrinkles the surface based on those estimates using a novel space-time deformation scheme. It generates realistic looking, temporally consistent networks of folds. The proposed technique is quite generic and can be applied to garments captured using many existing video-based techniques.

The markerless garment reconstruction and cloth wrinkling methods provide powerful tools for modeling detailed garment motion down to the fine high-frequency details. But, they are designed explicitly for garments. Capture of other dynamic shapes would require different approaches. One important, but also very general observation that can be applied for most typically scanned objects is that the changes in the object's shape are typically very gradual over time. This observation is true for any articulated shape, for humans or animals, garments, and many other everyday objects. The *gradual change* assumption effectively implies that no discrete changes which drastically affect the intrinsic shape, such as a change in the object genus, are possible. This observation can be used as a strong prior to capture the motion of the object.

Chapter 5 presents a general surface reconstruction method that relies on this very general prior and thus can be applied to a wide class of objects. The method starts by reconstructing individual frame geometry. Then it proceeds to assemble the individual frames into a consistent space-time frame sequence, simultaneously completing missing information across time and correcting local inconsistencies.

1.2. Thesis Contributions

The method can handle a wider variety of models than previous techniques, while better preserving the input geometric details, robustly handling noise and inconsistencies in the data, and preserving important properties of the object motion across time.



Figure 1.5: Results from this thesis. Top: Deformation of an octopus tentacle using the deformation technique presented in chapter 2. Middle: Detailed garment reconstruction result using a reconstruction method presented in chapters 3 and 4. Video stream (left) and reconstructed geometry (right). Bottom: Reconstruction of geometry and motion of a deformable hand-puppet using an acquisition technique presented in chapter 5. Initial reconstruction (left), final result (right)

Bibliography

- [ABK98] Nina Amenta, Marshall Bern, and Manolis Kamvyselis. A new voronoi-based surface reconstruction algorithm. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 415–421, New York, NY, USA, 1998. ACM.
- [BBH08] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. CVPR*, 2008.
- [BPGK06] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [BS08] Mario Botsch and Olga Sorkine. On linear variational surface deformation methods. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):213–230, 2008.
- [BWH⁺06] Miklós Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. A Quadratic Bending Model for Inextensible Surfaces.

- In *Fourth Eurographics Symposium on Geometry Processing*, pages 227–230, Jun 2006.
- [CC78] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350 – 355, 1978.
- [CGS⁺01] Brian Curless, Igor Guskov, Peter Schröder, Wim Sweldens, and Denis Zorin. Digital geometry processing. In *SIGGRAPH '01: ACM SIGGRAPH 2001 Courses*, 2001.
- [Cur00] Brian Curless. From range scans to 3d models. *SIGGRAPH Comput. Graph.*, 33(4):38–41, 2000.
- [dAST⁺08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98, 2008.
- [DG03] Tamal K. Dey and Samrat Goswami. Tight cocone: a water-tight surface reconstructor. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 127–134, New York, NY, USA, 2003. ACM.
- [DKT98] Tony DeRose, Michael Kass, and Tien Truong. Subdivision surfaces in character animation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 85–94, New York, NY, USA, 1998. ACM.
- [Gal00] Jean Gallier. *Curves and surfaces in geometric modeling: theory and algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.

- [GKB03] I. Guskov, S. Klibanov, and B. Bryant. Trackable surfaces. In *Proc. SCA*, pages 251–257, 2003.
- [IMT99] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. Teddy: A sketching interface for 3d freeform design. pages 409–416, 1999.
- [Inc09a] Autodesk Inc. 3ds max. <http://www.autodesk.com/>, 2009.
- [Inc09b] Autodesk Inc. Maya. <http://www.autodesk.com/>, 2009.
- [Inc09c] Autodesk Inc. Xsi. <http://www.autodesk.com/>, 2009.
- [INC09d] CYBERWARE INC. Cyberware inc. 2009.
- [KCvO07] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O’Sullivan. Skinning with dual quaternions. In *I3D ’07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 39–46, New York, NY, USA, 2007. ACM.
- [Kob00] Leif Kobbelt. \mathbb{S}^3 -subdivision. In *SIGGRAPH ’00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 103–112, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [Kv05] Ladislav Kavan and Jiří Žára. Spherical blend skinning: a real-time deformation of articulated models. In *I3D ’05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, New York, NY, USA, 2005. ACM.
- [LAN98] J. LANDER. Skin them bones: Game programming for the web generation. *Game Developer Magazine*, pages 11–16, 1998.

- [LPC⁺00] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.
- [LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005*, page accepted for publication. ACM Press, 2005.
- [NISA07] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. FiberMesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 26(3):article no. 41, 2007.
- [SK04] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 68–75, Washington, DC, USA, 2004. IEEE Computer Society.
- [VBMP08] Daniel Vlastic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):97, 2008.
- [WAO⁺09] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel,

- and Andreas Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Trans. Graph.*, 28(2):1–15, 2009.
- [WB01] Andrew Witkin and David Baraff. Physically based modeling. In *SIGGRAPH '01: ACM SIGGRAPH 2001 Courses*, 2001.
- [WCF07] R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 34, 2007.
- [YZX⁺04] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.
- [ZHH06] Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: an interface for sketching 3d scenes. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 9, New York, NY, USA, 2006. ACM.
- [ZRKS05] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.

Chapter 2

Material-aware Mesh

Deformation

2.1 Introduction

¹ Mesh deformation is an important task in the modeling and the animation of digital models for computer graphics. Since most real world objects are made up of non-uniform materials, their behavior during deformation varies across the surface depending on the local material properties. Ideally, a mesh deformation tool should satisfy the following key requirements: physical plausibility of the results, ease of control, efficiency, and high degree of automation. Existing deformation approaches typically satisfy only a sub-set of these requirements. For example, physics based methods provide accurate model behavior, but they are often not intuitive to control and are usually relatively slow. Skeleton deformations are simple to control and can be implemented efficiently, but their range is typically limited to only a subset of models. Purely geometric deformation techniques are general, efficient and intuitive to control; however they usually ignore the properties of the underlying materials, and thus make it difficult to generate physically plausible deformations.

This chapter introduces material-aware mesh deformation, a novel technique

¹A version of this chapter has been published. Popa, T. and Julius, D. and Sheffer, A. (2007) An Interactive and Linear Material Aware Deformations, International Journal of Shape Modeling, 2007

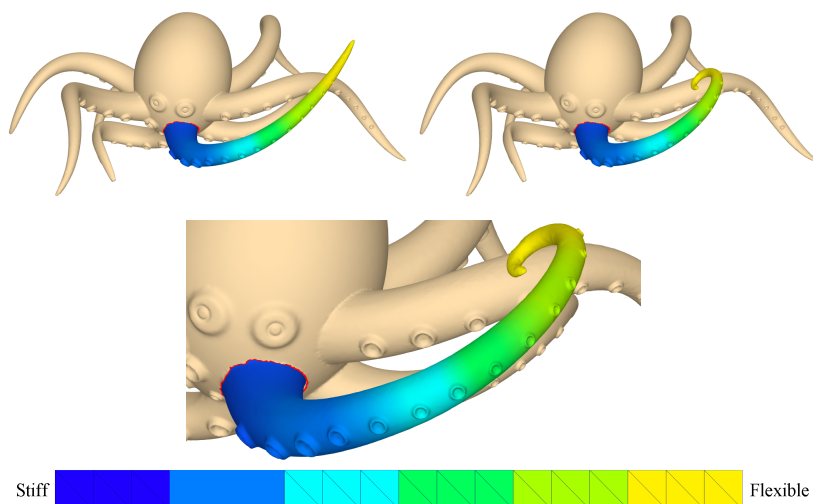


Figure 2.1: Material-aware deformation. The stiffness of the tentacle is set to be proportional to its girth (left) resulting in a spiral like shape (right and bottom)

that uses material properties to guide geometric deformations. Material properties are used to specify the stiffness of the surface, and hence to provide continuous fine control of the surface behavior during deformation, while maintaining the efficiency, simplicity and control specific to geometric methods. The stiffnesses with respect to bending and shearing are represented as scalar fields over the surface. These scalar fields are used within the geometric deformation framework to distribute the deformation according to the local material properties to yield realistic-looking results (Figure 2.1). Often materials may exhibit anisotropic stiffness, for instance articulated models often have joints with only one degree of freedom. The formulation supports such anisotropic behavior by allowing three different scalar fields for the three orthogonal axes of rotation. This is the first method, to my knowledge, to support this feature.

To control the deformation, users can specify the material properties using an intuitive paint-like interface. By simply marking a horse's head as stiff (Fig-

2.1. Introduction

ure 2.2(c)), the deformation is applied mostly to the neck of the horse and achieve more realistic results than in Figure 2.2(b) where the deformation is distributed uniformly. Although some existing geometric methods are capable of achieving similar results, typically they require more user effort to guide the deformation.

In many situations, physically or anatomically correct deformation samples of a given model may be available. In such cases the method can automatically learn the material properties from the sample set, thereby allowing users to create new deformations which are consistent with the sample set. Each of the deformed sample poses contains implicit knowledge of a subset of the material properties. By combining the information from all samples, it is possible to reconstruct the scalar fields across the surface. For additional control, the user can refine the acquired fields in specific areas of interest where the desired behavior differs from that of the sample poses.

A simple and intuitive modeling metaphor is an important feature of any mesh deformation tool. The method provides a flexible and intuitive user interface that allows the user to choose between simplicity and control. The user can choose between a simple drag and drop interface to position vertices, and a more sophisticated user interface where the user has more control over the deformation behavior.

The main contribution is the introduction of a compact representation for the local stiffness of a surface, and the integration of this material stiffness into the geometric deformation framework. The method is linear, it is simple to use and control, and it creates realistic looking deformations as discussed in the results section.

The rest of this chapter is organized as follows: Section 2.2 reviews previous work on deformation techniques. Sections 2.3 and 2.4 describe the deformation algorithm. Section 2.5 extends the formulation to allow positional constraints on

vertices. Section 2.6 explains how the method is extended to support anisotropic behavior. Section 2.7 describes how material properties may be learned by example. Section 2.8 presents some example results. Finally, Section 2.8 summarizes the work.

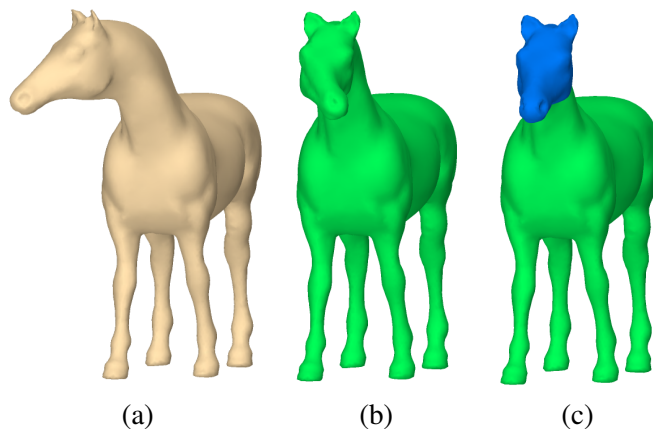


Figure 2.2: Turning a horse’s head: (a) original model; (b) deformation using uniform material; (c) material-aware deformation using two degrees of stiffness

2.2 Previous Work

Researchers have addressed the problems of mesh editing and deformation for over twenty years, creating an impressive body of literature and generating several distinct approaches to the problem. One of the first, yet still actively researched mesh deformation frameworks is that of space warping deformations [Bar84, SP86, ACWK04, BK05]. Since space deformation techniques transform the underlying space, rather than the vertices themselves, it is not possible to incorporate model specific properties such as materials into these techniques.

Another common approach is to use physical simulation methods [TPBF87, TF88, JF03, MHTG05]. These methods naturally incorporate the material prop-

2.2. Previous Work

erties and provide physically accurate behavior; however, they are often computationally intensive, and since their control parameters are typically derived from physical equations, they lack intuitive means of controlling the results.

For articulated models, it is common to use a skeleton in order to simplify the task of defining the deformation [CGC⁺02, ACP02, YBS03]. Most skeleton based deformation methods do not generalize to non-articulated models and often provide only binary gradation of stiffness, thus limiting the type of deformations created. In contrast, this method is not restricted to a particular type of models and supports continuous control over the stiffness of the mesh providing finer control of the deformation.

Geometric deformation techniques [Ale01, SCOL⁺04, LSCO⁺04, SK04, YZX⁺04, IMH05, ZHS⁺05, LSLCO05, ZRKS05, KCVS98, BK04, BPGK06] that operate directly on the meshes have become increasingly popular in recent years. These methods are both efficient and intuitive to control. However, existing geometric techniques do not capture the material properties of the models. This chapter presents a method that uses a geometric deformation approach but introduces material awareness into the framework to provide greater physical plausibility. The geometric deformation methods most related to this work are those of Yu et al. [YZX⁺04], Zayer et al. [ZRKS05] and Igarashi and Moskovitc [IMH05]. Yu et al. [YZX⁺04] perform 3D mesh deformation by means of gradient manipulation. First, the positions of some *anchor* vertices are manually modified by the user. Next, the resulting local triangle transformations are propagated to the rest of the mesh according to geodesic distances. Finally, the new vertex positions are computed using the Poisson equation. Zayer et al. [ZRKS05] show that propagation of the transformations according to geodesic distances is sub-optimal and suggest using harmonic fields as an alternative. Neither method considers material properties in their formulation. Igarashi and Moskovitc [IMH05] deform 2D meshes using

2.2. Previous Work

a formulation based on an earlier morphing technique [ACOL00]. They manipulate the triangles independently and then compute common vertex positions. They show early research results for using material stiffness to control the deformation. A direct extension of their method to 3D would require a volumetric mesh, thus they acknowledge that such an extension may be difficult.

It is often tedious and difficult to define the exact physical properties of an object. One alternative, presented by two recent techniques [JT05, SZGP05], is to create realistic-looking deformations by mimicking existing physically correct example deformations. James and Twigg [JT05] automatically deduce the skeleton of an articulated model from a sample set of deformed models. Using the estimated skeleton and estimated blending weights they are able to create new deformations consistent with the sample set. Sumner et al. [SZGP05] use the set of sample models to create feature vectors that span the space of meaningful deformations. With this method, it is possible to use a set of sample poses as a source for automatically learning the stiffness of the mesh. The learned stiffness is used to create new poses consistent with the samples. In this setting the material properties are derived explicitly, therefore it is very easy for artists to modify and refine those if desired.

Most of the geometry deformation techniques that have linear formulations [SCOL⁺04, LSCO⁺04, LSLCO05, ZRKS05] are limited to a modeling metaphor that requires the user to provide rotational as well as positional constraints to obtain natural looking results. In this method, the user still has the choice of specifying both positional constraints and rotational constraints. But the user also has the choice to only specify positional constraints, in which case the system will compute appropriate rotations for the triangles in the mesh. The user can choose among several criteria to compute these rotations depending on the desired behavior. This is a richer and more flexible front-end to the formulation. Moreover, the method in which the system computes the rotational component is generic and, therefore, can

2.3. Method Overview

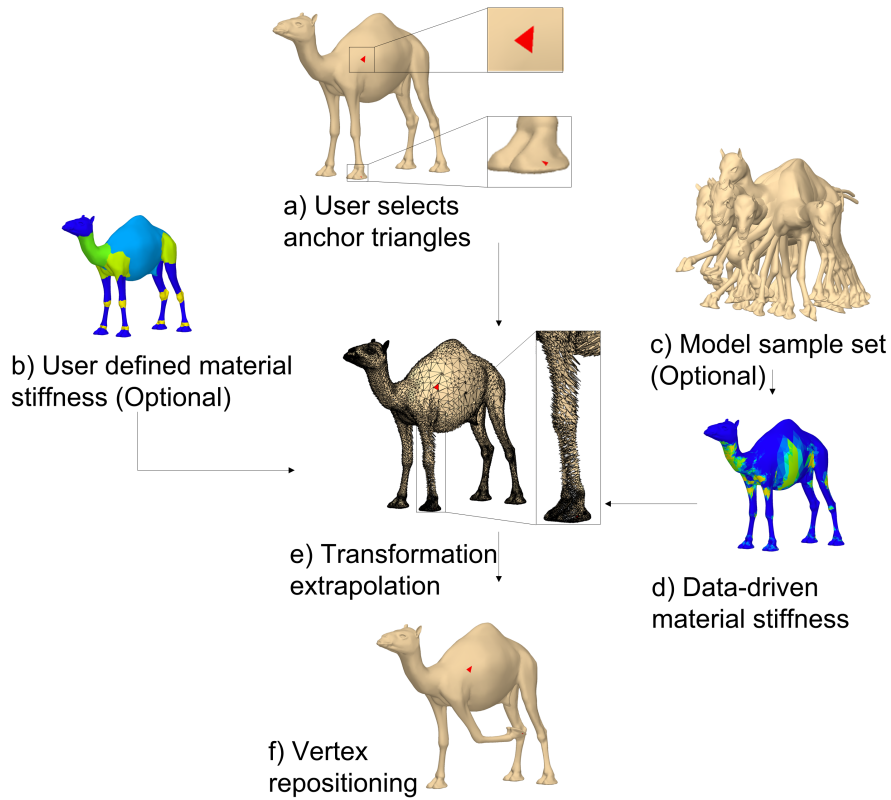


Figure 2.3: Algorithm flow

be applied to other methods that suffer from this limitation.

2.3 Method Overview

This chapter presents a two-step method for 3D mesh deformation that takes into account the intrinsic material properties of the model. To generate the deformation users select a small set of triangles, called *anchor* triangles (Figure 2.3(a)) and apply the desired transformations using a click and drag motion. The system supports anchor transformations that include any combination of rotations and

uniform scales. The transformations for the remaining triangles of the mesh are computed based on the anchor transformations. The calculation takes into account the material properties of the model which are described as follows.

Material properties — To describe the impact of the material on the deformation the stiffness of the material is defined with respect to bending and shearing. These stiffnesses are described by separate scalar fields defined across the mesh. Since materials often bend differently with respect to different directions, the user can define anisotropic bending stiffness fields. The user can define these scalar fields using a paintbrush-like tool (Figure 2.3(b)).

This method introduces a data-driven approach for defining the stiffnesses by automatically learning them from a set of example deformations (Figure 2.3(c)). By examining the shearing of each individual triangle and the difference in the transformations undergone by adjacent triangles within the entire sample set, it is possible to identify degrees of stiffness and flexibility across the mesh, and thus reconstruct the stiffness fields (Figure 2.3(d)). Next, I explain how these stiffness scalar fields, together with the user defined transformations at anchor triangles, are used in the algorithm.

Transformation extrapolation — The first step of the algorithm is to propagate automatically the transformations from the anchors to the remaining triangles in the mesh. Finding optimal transformations for each triangle is non-trivial since these must comply with a number of constraints. First, the transformations must be continuous across the surface to yield a smooth looking deformation. Next, the transformations must also be as-rigid-as-possible in order to maintain the details of the original surface [YZX⁺04]. Finally, the system acquires one new constraint — the transformations must be consistent with the material properties. This final requirement is the one that ensures the deformations behave as desired.

Each triangle transformation is a weighted sum, or blend, of anchor transfor-

mations. Thus, the challenge is to find appropriate weights for blending that comply with the previous requirements. This is formulated as a linear optimization problem where the variables are the blending weights. The stiffness fields are introduced into the formulation to ensure that the deformation is distributed correctly throughout the model. Since the solution depends only on the selection of anchors, the resulting system needs to be solved only once. To perform the actual blending a transformation algebra defined by Alexa [Ale02] is used. For a discussions on the optimality of this method, see Appendix A.

Vertex repositioning — It is easy to see that applying the resulting transformations to each of the triangles in the mesh independently will break the mesh connectivity, since adjacent triangles are not necessarily assigned identical transformations (Figure 2.3(e)). Therefore a second step consists of calculating optimal vertex positions such that each triangle is transformed as closely as possible, in the least squares sense, to the previously calculated transformations (Figure 2.3(f)). The shearing stiffness field is incorporated into the formulation to ensure that most of the resulting shearing is concentrated in the flexible areas of the mesh.

Figure 2.3 summarizes the algorithm, and the following three sections describe it in detail: Section 2.4 explains how transformations are propagated and then optimal vertex positions are found, Section 2.6 explains how the method is extended to support anisotropic stiffnesses, and Section 2.7 explains how the material stiffnesses are estimated from sample deformations.

2.4 Method Details

Subsection 2.4.1 describes how material properties are defined. Subsection 2.4.2 defines the gradient transformations and explain how these are propagated from anchor triangles. Finally, Subsection 2.4.3 explains how optimal vertex positions are found.

2.4.1 Material Properties

Material properties are formulated in terms of material resistance to bending and shearing. This resistance is described by bending and shearing stiffness scalar fields defined across the mesh. This approach allows a high degree of control with smooth variations of stiffness across the mesh.

- The bending stiffness is associated with the mesh edges, reflecting the bending flexibility of each edge. Thus the bending scalar field is defined by a set of values ϕ_{ij} defined on the mesh edges (i, j) . This field is used to propagate the anchor transformations across the mesh (Equation 2.1).
- The shearing stiffness is associated with the mesh faces reflecting the resistance to shearing of each individual face. The shearing stiffnesses ψ_i defined for the mesh faces are used to find optimal vertex positions (Equation 2.3).

The bending stiffness often depends on the rotation axis. For example, articulated models often have joints with a limited number of axes of rotation. Therefore, this approach supports anisotropic stiffness by specifying separate bending stiffness fields for three orthogonal axes of rotation.

This method supports both user-driven and data-driven methods for defining the material properties. In the user-driven setting the user is provided with a simple paintbrush like tool to define the different degrees of stiffness. The user can paint

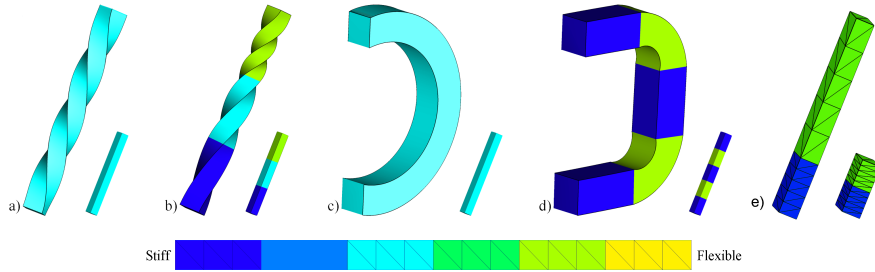


Figure 2.4: Twisting, bending, and stretching a bar using two anchor triangles: (a), (c) uniform material; (b), (d), (e) non-uniform material

two separate fields for bending and shearing. To simplify the interface, since often the two are linked, the user can specify only one field and derive the other one from it. For instance, if the user specifies the shearing stiffness ψ_i , the bending stiffness is obtained by simply setting $\phi_{ij} = (\psi_i + \psi_j)/2$. This simplified interface is used in most of the following examples. Figure 2.4 demonstrates painting areas with different degrees of stiffness on a 3D bar and the resulting deformations.

For the data-driven approach, the material properties are estimated automatically from a sample set of deformations. Additionally, the formulation naturally supports user-intervention after the data-driven material estimation step. This property is important in a production setting where animators require simple user controls to fine-tune automatically generated results.

2.4.2 Transformation Extrapolation

After the material properties and anchor transformations are defined The deformation is created as follows. The entire mesh deformation can be expressed as a set of affine transformations $(Ax + b)$ of local coordinate frames defined per triangle. The deformation gradient of each transformation is the matrix A , which encapsulates the triangle transformation up to the translational component. Since the three

2.4. Method Details

vertices of a triangle do not determine a local frame, the three vertex positions are augmented with a fourth point found by offsetting one of the vertices by the triangle normal [SP04]. Labeling the vertices of a given triangle as v_1, v_2 and v_3 and the additional vertex as v_4 , the three vectors that define the local coordinate frame are $(v_4 - v_1, v_4 - v_2, v_4 - v_3)$.

After the user specifies the transformations for each of the anchor triangles the first step of the algorithm propagates the deformation gradients defined at the anchor triangles to the remaining triangles of the mesh. This is achieved by using a weighted blending of anchor transformations. As previously noted, the challenge is to find appropriate weights for this blending, subject to the material properties. This implies that triangles in areas which are more resistant to bending should be assigned more similar transformations. In the isotropic setting, this is a linear optimization problem:

$$\min_{\omega_i \in R^k} \sum_{(i,j) \in E} \varphi_{ij} \|\omega_i - \omega_j\|_2^2, \quad (2.1)$$

where the unknowns are $\omega_i \in R^k$ the blending weights for face i . Each ω_i is a vector $(\omega_i^1, \omega_i^2, \dots, \omega_i^k)$ where ω_i^a denotes the relative influence of anchor transformation a . k is the number of anchor triangles, E is the set of edges (excluding edges shared by two anchor triangles and boundary edges) and φ_{ij} are the bending stiffness values associated with each edge. The anisotropic setting is slightly different, and is explained in Section 2.6.

When φ_{ij} is large, ω_i and ω_j will have similar values; thus, the resulting transformations of the two adjacent triangles i and j will also be similar, and the mesh may be considered as locally stiff. Similarly, the converse argument can be made for small φ_{ij} .

Note that in this formulation, the weights ω_i depend only on the connectivity

of the mesh and on the selection of anchor triangles. Therefore, the weights need to be computed only once per selection of anchors. Also, note that, since the anchor coefficients have exactly one non-zero entry, the weights are barycentric coordinates with respect to the anchors. As a consequence, the results do not suffer from the propagation problems noted by Zayer [ZRKS05] when using weights based on geodesic distances [YZX⁺04].

In order to perform the actual blending the commutative transformation matrix algebra defined by Alexa [Ale02] is used. By defining two new operations denoted by \oplus and \odot , the blended transformations T_i with weights ω^i are computed as:

$$T_i = \bigoplus_{a=1}^k \omega_i^a \odot T_a. \quad (2.2)$$

Details on these operators are found in Appendix A. Figure 2.3(e) illustrates the propagated transformations between two anchors on the camel’s leg.

2.4.3 Vertex Repositioning

Since the transformations obtained for adjacent triangles are typically not identical, applying each of the transformations as-is would result in ambiguous positions for the two shared vertices. Therefore, a second stage is needed to compute the optimal position for each vertex.

Optimal vertex positions are found such that the gradient transformation of each triangle remains as close as possible, in the least squares sense, to the ideal transformations computed in the extrapolation step [SP04]. Since the process results in triangle shearing, the shearing stiffness is introduced into the formulation to direct the distortion according to the flexibility of the mesh.

As illustrated in [SP04] the transformation gradients can be expressed in terms of the vertex positions before and after the deformation:

2.4. Method Details

$$A = \tilde{V}V^{-1},$$

where

$$\tilde{V} = (\tilde{v}_4 - \tilde{v}_1, \tilde{v}_4 - \tilde{v}_2, \tilde{v}_4 - \tilde{v}_3),$$

$$V = (v_4 - v_1, v_4 - v_2, v_4 - v_3),$$

and \tilde{v}_i is the position of vertex v_i after applying the deformation transformation. Next, the system is reformulated such that the unknowns are the new vertex positions \tilde{v} .

The optimal solution is obtained when the resulting triangle transformations are as close as possible to the previously computed T_i 's. Since the T_i 's are blends of the original anchor transformations, they contain no shearing. Thus, the closer the final and the original transformations are, the smaller the triangle shearing. To account for the shearing stiffness ψ_i is incorporated into the formulation:

$$\min_{\tilde{v}} \sum_i^{n-k} \psi_i \|\tilde{V}_i V_i^{-1} - T_i\|_F^2, \quad (2.3)$$

where V_i and \tilde{V}_i are the local frames before and after applying the deformation and T_i are the previously calculated transformations. This can be reformulated as a linear optimization problem:

$$\min_{\tilde{v}} \|\Psi(A\tilde{v} - t)\|_2^2, \quad (2.4)$$

where A is a sparse matrix constructed using the pre-deformation local frames V , t is a vector composed of all the elements in T_i and Ψ is a diagonal matrix composed of ψ_i .

Large ψ_i will result in transformations which are very close to the originally

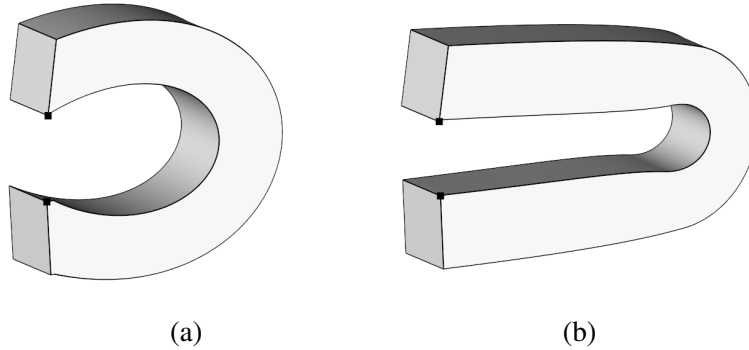


Figure 2.5: Deformation of a bar specifying only positional constraints. The optimal rotations are computed automatically. (a) uniform materials (b) non-uniform materials

computed T_i , and thus exhibit less shearing. Small ψ_i will allow for more shearing to take place. Figure 2.4(e) shows an example of stretching a bar with non-uniform shearing stiffness. As expected, most of the shearing occurs in the flexible region.

The solution of the system in Equation 2.4 provides a new position for each of the vertices up to a global translation of the model. To anchor the model in place, the position of a single vertex is fixed by removing the corresponding variable and pre-multiplying its known position with the appropriate elements of A into the vector T [SZGP05]. In fact, multiple vertices may be set at fixed positions to apply boundary constraints such as regions of influence.

2.5 Positional Constraints

The formulation presented so far allows the user to only specify rotational constraints for the anchor triangles. In certain cases the user might want to specify positional constraints as well. Positional constraints on vertices can be added by augmenting the second stage of the system (equation 2.4) by adding more rows to the constraint matrix A to keep the anchor vertices in place. In this setup, the

2.5. Positional Constraints

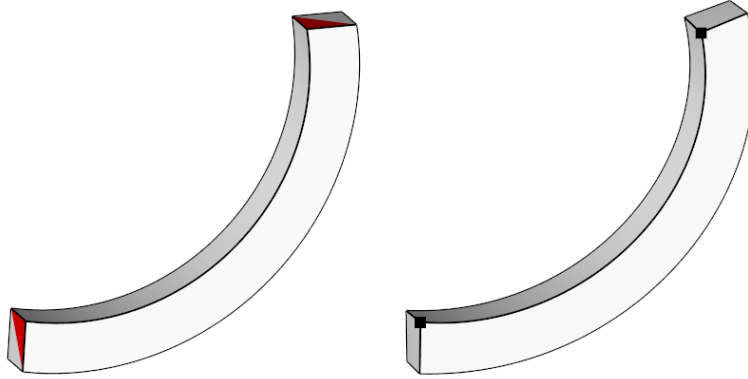


Figure 2.6: Left: rotation of the bar using only rotational constraints. Right: rotation of the bar by dragging one anchor and using a heuristic to determine the rotation

user has to specify positional and rotational constraints simultaneously to achieve the desired results which is less intuitive than a standard drag and drop interface. Most of the deformation methods that use linear formulations [SCOL⁺04, LSCO⁺04, LSLCO05, ZRKS05] also face the same limitation. Non-linear methods [SK04, BPGK06] that employ rotational invariant local coordinates do not have this problem, but non-linear formulations are in general less efficient.

This section describes a method where the user can specify only positional constraints on vertices using a drag and drop interface. The system then computes the optimal rotations automatically. Figure 2.5 shows an example where a straight bar is deformed by specifying a new position for one vertex while keeping the other vertex fixed. Note that the algorithm provided an intuitive rotation for both the uniform and non-uniform case yielding a natural looking result.

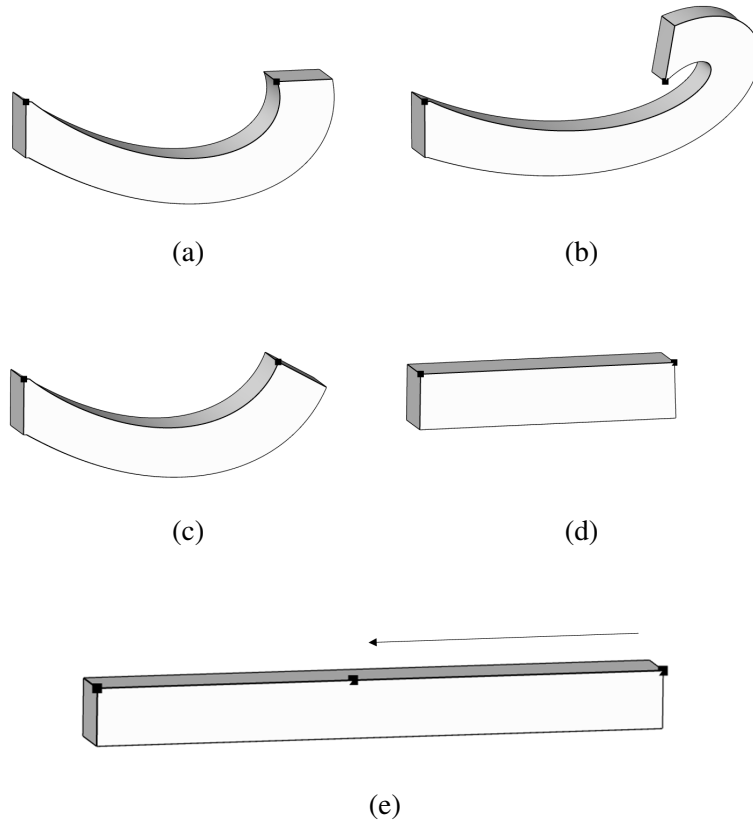


Figure 2.7: Comparative results for various energies used: (a) area, (b) volume, (c) shearing (d) dihedral angle

2.5.1 Finding Rotations

This section describes more formally the method used for computing the rotations. For each user selected anchor vertex v_i the algorithm associates an adjacent anchor triangle t_i . As the user changes the position of v_i , the system computes the optimal rotation R_i for the associated anchor triangle t_i . It then propagates the rotations to all the triangles in the mesh as described in section 2.4.2. After that, it finds vertex positions as described in section 2.4.3 with the amendment that additional constraints are added in the constraint matrix A of Equation 2.4 for all anchor

2.5. Positional Constraints

vertices v_i .

Theoretically, since a 3D rotation has 3 degrees of freedom one would need to solve for 3 new variables. However, in a typical drag and drop interface, an anchor vertex is moved on a plane parallel to the projection plane. This means that the anticipated rotation will typically be around the normal of the projection plane thus reducing the problem to only one degree of freedom. Details on how to solve for this latest degree of freedom are given in section 2.5.2. Figure 2.6 compares the results of a deformation using only rotational constraints and a deformation using only positional constraints. Note that in figure 2.6(b) the system found automatically the appropriate angle.

The optimal rotation angle for the anchor triangle is the angle that yields a result that best preserves certain properties (functionals) between the original object and the deformed object. Several such functionals were considered:

- *Area preserving.* Minimize $\sum_{i \in T} (\psi_i * (A_i - A'_i))^2$ where A_i and A'_i are the areas of the original triangles and the triangles after deformations, respectively. ψ_i is the face stiffness.
- *Volume preserving.* Minimize the variation in total volume. The total volume is estimated using the standard method of summation over the signed tetrahedra volumes obtained by each face with an arbitrary reference point P .
- *Shape preserving.* Minimize $\sum_{i \in T} (\psi_i * s_i)^2$ where s_i is a measures the shearing between the original and the deformed triangles weighted by the stiffness of each triangle. The shearing is computed by first obtaining the 3×3 transformation matrix that transforms the frame of one triangle onto the other. The shearing coefficients are obtained from the polar decomposition.

2.5. Positional Constraints

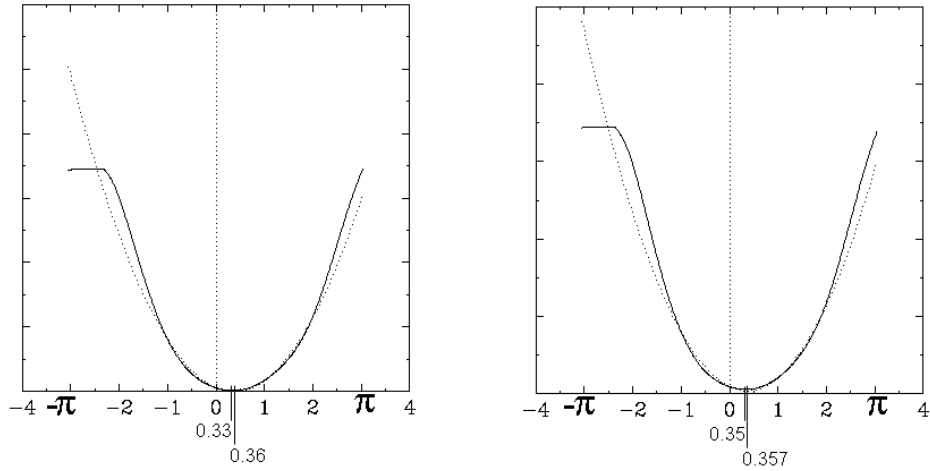


Figure 2.8: The two figures show two plots of the error function against the angle of rotation of the moving anchor. The dotted line represents the parabola that we are trying to fit to the data. The two numbers at the bottom of each figure represent the true solution, and the solution computed using the parabolic strategy respectively.

- *Angle preserving.* Minimize the variation in bending between adjacent triangles: minimize $\sum_{(i,j) \in E} (\varphi_{ij} * (\alpha_{ij} - \alpha'_{ij}))^2$ where α_{ij} and α'_{ij} are the dihedral angle of the faces corresponding to edge (i, j) on the original mesh and deformed mesh respectively, and φ_{ij} is the edge stiffness coefficient of edge (i, j) .

Figure 2.7 shows a comparative result of bending a bar using the different functionals. The deformations in figure 2.7(a-d) are done by keeping an anchor and its associated anchor triangle in place and pushing the other toward it as illustrated in figure 2.7(e).

The *area preserving* and *shape preserving* functionals are in the spirit of the “as rigid as possible” deformations. In both cases we the shapes of the triangles are preserved in the least square sense, giving more priority to stiffer triangles. As

2.5. Positional Constraints

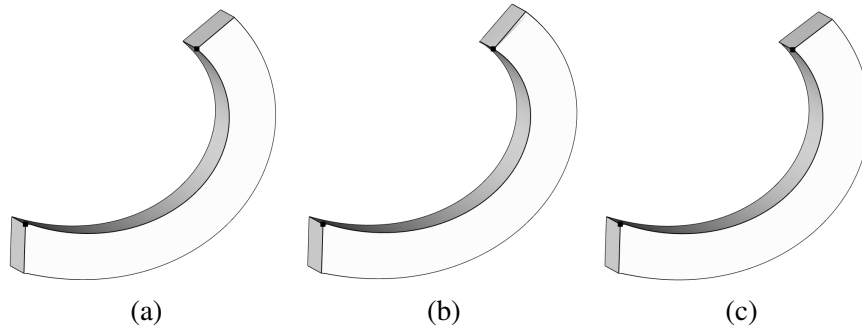


Figure 2.9: Comparative results for the 3 solvers: (a) brute force (b) parabolic solver (c) binary search solver

shown in Figure 2.7, the *area preserving* functional allows for more bending yielding more natural looking results. In addition to that, the *area preserving* is also cheaper to evaluate than the *shape preservation* functional that requires a polar decomposition step for every triangle in the mesh. The *volume preserving* functional finds the rotation in the plane that makes the change in volume minimal. Hence it has to over-rotate in order to add more volume. Note that this functional minimizes the variation in total volume, but it does not guarantee that the volume is preserved exactly. The *angle preserving* functional collapses the bar since a zero rotation of the anchor triangle best preserves the dihedral angles. This functional can be used to simulate plastic deformations like clay. Note that affine combinations of these functionals can be used to get intermediate effects. In practice the *area preserving* functional yields the most natural results and it is also the most numerically stable, so this is the function that is used for all the other examples that use positional constraints.

2.5.2 Solving

Since we are looking for an angle, a brute-force approach to solving this problem is to densely sample the interval $[-\pi, \pi]$ and find which angle gives us optimal results. Moreover, as the user drags the mouse on the screen, the angle change from one evaluation to another is relatively small. This shortens the range of values making the search more efficient. In practice, the $[-1, 1]$ range is usually sufficient. One evaluation of the functional requires solving the two systems described by equations 2.1 and 2.4. Since the two matrices associated with these systems need to be inverted only once, even the brute-force approach to finding the optimal angle yields interactive rates for small models. For larger models two strategies that significantly speed up the process are devised: a binary search strategy and a quadratic fit strategy.

In the binary search case, the derivative of the function is estimated using forward differences and the root of the derivative function is found using a binary search strategy. Note that the search interval is so small that a more sophisticated Newton method would not significantly improve the convergence. Binary search takes usually around 6 iterations to converge, and since it takes two evaluations to compute the derivatives it takes around 12 evaluations.

While the binary search strategy works well in practice, the method can be speeded up even more using the observation that in most cases the function has only one local minimum and the shape of the curve looks very much like a parabola as illustrated in figure 2.8. Therefore, the rotation angle can be found by fitting a parabola to the data and choosing the tip of the parabola as the minimum point. The difference between the estimated solution and the true solution is within a few degrees. This approach always requires a constant number of evaluations, namely 4. Three evaluations are required to find the equation of the parabola and the fourth

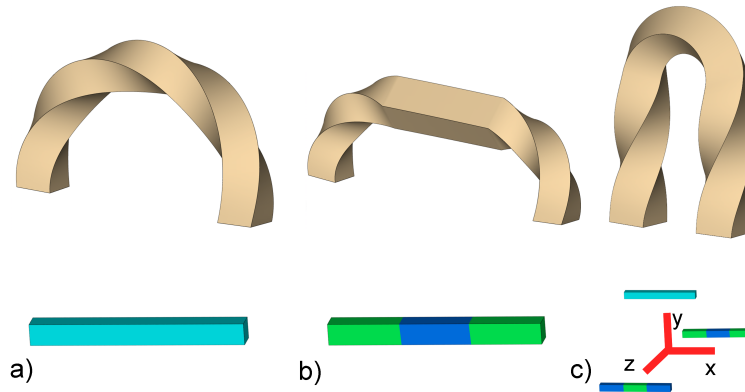


Figure 2.10: Bending and twisting a bar with different material properties and a single transformation applied at the two end faces. (a) Uniform material. (b) Center region is isotropically more stiff. (c) Anisotropic stiffness - allowing the center region to bend around the z axis, and the side regions to twist around the x axis

is the evaluation that gives us the final solution. Using this strategy, the solution is still asymptotically linear with a constant factor of 4. The quadratic fit formulation, while faster, tends to be less stable than the binary search or brute force methods in cases where the true solution is close to $-\pi$ or to π . Figure 2.9 shows a comparison of the solvers.

2.6 Anisotropic Materials

The formulation presented above uses a single scalar bending stiffness field. This formulation assumes that bending flexibility is a non-directional property, which need not be true in practice. For example, a knee joint is only flexible in one direction and rigid in the other two. This section presents an extension of the methods that allows different bending stiffnesses for different axes of rotation.

Instead of using one bending stiffness value per edge, three different values

2.6. Anisotropic Materials

corresponding to rotations around three orthogonal axes x , y and z are defined. To apply those stiffnesses when blending anchor transformations, the algorithm first decomposes the anchor transformations into a scaling transformation and rotation transformations around the three axes x , y and z . Using the polar decomposition $T_a = S_a Q_a$ such that Q_a is a rotation transformation [SD92], the rotational component of each anchor transformation is identified. Using the transformation matrix algebra described by Alexa [Ale02], three matrices R_x^θ , R_y^θ , R_z^θ denoting rotations by an angle $\theta \in [0, \pi]$ around three orthogonal axes x , y , and z form a basis of the sub-space of rotations. Thus, for any given rotation matrix Q one can find a commutative decomposition such that

$$Q = c_x \odot R_x^\theta \oplus c_y \odot R_y^\theta \oplus c_z \odot R_z^\theta.$$

The coefficient c_x is found by simply computing the inner product

$$c_x = \langle \log(Q), \log(R_x^\theta) \rangle,$$

where the inner product of transformation matrices is the sum of products of the corresponding matrix entries

$$\langle A, B \rangle = \sum a_{ij} b_{ij}. \quad c_y \text{ and } c_z \text{ are found in a similar manner.}$$

By defining $X_a = c_x^a \odot R_x^\theta$, $Y_a = c_y^a \odot R_y^\theta$ and $Z_a = c_z^a \odot R_z^\theta$ the method obtains a unique decomposition of each anchor transformation T_a into a scaling matrix S_a and three rotation matrices X_a , Y_a and Z_a around the three orthogonal axes of rotation x , y and z .

To compute the new blending weights and transformations one needs to solve Equation 2.1 separately for each axis of rotation using the appropriate bending stiffness to get three weight vectors ω_i^x , ω_i^y and ω_i^z per triangle. In the isotropic case,

2.7. Material Learning

ω_i is used as weights for blending the anchor transformations. In the anisotropic setting the local triangle transformations T_i is found by combining the per-axis transformations. Equation 2.2 becomes

$$T_i = \left(\bigoplus_{a=1}^k \tilde{\omega}_i^a \odot S_a \right) \left(\bigoplus_{a=1}^k \omega_i^{ax} \odot X_a \oplus \omega_i^{ay} \odot Y_a \oplus \omega_i^{az} \odot Z_a \right)$$

where T_i are the propagated transformations, ω_i^{ax} , ω_i^{ay} and ω_i^{az} are the blending weights, $\tilde{\omega}_i^a$ is their average, and S_a, X_a, Y_a and Z_a are decompositions of the anchor triangle transformations T_a such that $T_a = S_a (X_a \oplus Y_a \oplus Z_a)$.

Since the anisotropic stiffnesses is defined only for bending the rest of the algorithm continues as previously described. Figure 2.10 shows the impact of using anisotropic stiffness fields when bending and twisting a 3D bar. First, a single material with uniform stiffness is used and applied two rotations around the x and z axes at one end of the bar while anchoring the other end. This resulted in a uniform deformation (Figure 2.10(a)). Next, the model's material is changed, specifying the center region as isotropically stiff and the sides as isotropically flexible. The same transformation is then applied. The result is shown in Figure 2.10(b) where the center region remains rigid. Finally, Figure 2.10(c) illustrates an anisotropic deformation. The center region material was defined to be stiff when rotating around the x axis, and flexible when rotating around the z axis. The side region materials were defined to deform in the exact opposite manner. The resulting deformation exhibits twisting (rotation around the x axis) only in the side regions, and bending (rotation around the z axis) only in the center region.

2.7. Material Learning

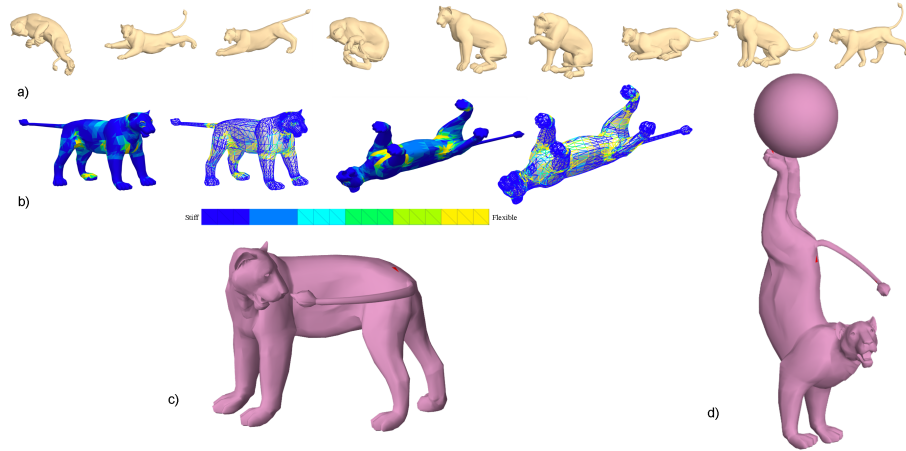


Figure 2.11: Estimation of material stiffness from sample poses. (a) Sample poses. (b) Estimated bending and shearing stiffnesses visualized on the edges and faces respectively. (c), (d) New poses created using the estimated stiffnesses: (c) chasing the tail (4 anchors); (d) handstand (8 anchors)

2.7 Material Learning

In many situations, physically or anatomically correct deformation samples of a given model may already be available. In such cases one would like to derive the material properties from the sample poses to create new deformations which are consistent with the sample set and are therefore also correct. The following section describes a method of automatically acquiring the material properties from sample deformations.

Given a reference mesh P_0 with m triangles and a set of deformed meshes with the same connectivity P_s $s = 1, \dots, l$, the algorithm first computes the deformation gradient for each of the triangles $i = 1, \dots, m$ in P_s as follows:

$$T_{si} = \tilde{V}_{si} V_{0i}^{-1} \quad s = 1, \dots, l, i = 1, \dots, m,$$

2.7. Material Learning

where V and \tilde{V} are the local frames in the reference and deformed meshes.

Next, each deformation gradient matrix is decomposed using the polar decomposition $T_{si} = S_{si}Q_{si}$ such that Q_{si} is a rotation transformation, and S_{si} is a combination of scaling and shearing [SD92, JT05, SZGP05]. This decomposition is used to estimate the bending and shearing stiffness across the mesh.

The amount of bending the model undergoes locally is best reflected by the change in the dihedral angle around a mesh edge, and can be estimated by analyzing the rotations Q_i and Q_j of two triangles sharing the edge. In each example the difference between the two matrices Q_i and Q_j of adjacent triangles is roughly equivalent to the amount of bending around the shared edge. Since each sample only exhibits bending around a subset of the edges, one needs to combine values from multiple samples. The maximum difference observed for each edge is equivalent to the maximum bending around the edge among all samples and therefore is used to define the bending stiffness. The values are then scaled to be in $[\varepsilon, 1]$.

$$\tilde{\varphi}_{ij} = \max_{s=1\dots l} \|Q_{si} - Q_{sj}\|_F^2 \quad (i, j) \in E,$$

$$\varphi_{ij} = 1 - \frac{\tilde{\varphi}_{ij}}{\max_{i,j \in E} \tilde{\varphi}_{ij} + \varepsilon} \quad (i, j) \in E.$$

For each face i , a shearing stiffness ψ_i is estimated based on the maximum distortion of the face found in the set of example deformations. The values are then scaled to be in $[\varepsilon, 1]$.

$$\tilde{\psi}_i = \max_{s=1\dots l} \|S_{si} - I\|_F^2 \quad i = 1 \dots m,$$

$$\psi_i = 1 - \frac{\tilde{\psi}_i}{\max_{i=1\dots m} \tilde{\psi}_i + \varepsilon} \quad i = 1, \dots, m.$$

For both scalar fields clamping the bottom 1% of the values before scaling

greatly improved the results.

Figure 2.11 shows an example of estimated bending and shearing stiffnesses learned from a sample sequence of deformations. As shown in the figure these are used as a basis for creating new poses which were not in the sample set.

The material learning method presented in this thesis is much simpler than the methods of James and Twigg [JT05] and Sumner et al. [SZGP05], who also use polar decomposition as the first step in their methods. However, some problems can occur when stiffness coefficients of different parts of the body are learned independently from different poses. In these cases there is no information on the relative stiffness between these parts and hence the relative scale of the stiffness coefficients may not be correct. Nevertheless, experimental results show that this material learning method appears to correctly capture the model's material properties, and to provide realistic looking deformations. Furthermore, this method exhibits two nice properties. First, the complexity of the learning algorithm is linear in the number of sample poses and second, the learned stiffnesses can be further refined by the user for finer control.

2.8 Implementation and Results

This section describes some implementational details and discusses the results. The deformation method was implemented using Graphite [Gra03]. Material properties are defined using a simple color map with a paintbrush interface. To deform a model, the user selects the anchor triangles using mouse clicks and then transforms them by dragging the mouse onto the screen. Note that in all the examples except Figure 2.4(e) only the rotation and scaling for the anchor triangles are specified without fixing their final positions. The algorithm finds these optimal positions automatically.

2.8. Implementation and Results

The deformation method relies on solving two least squares systems to solve equations 2.2 and 2.3. Both systems can be reduced to solving two linear systems that are very sparse. The system matrix in eq 2.2 has roughly four non-zero entries per row and the system matrix in 2.3 has roughly seven non-zero entries per row, therefore they can be solved very efficiently.

The UMFPACK4.4 [Dav04] package was used to solve both systems. UMFPACK solves the linear system using a LU decomposition. The weights in eq. 2.2 are computed only once and the target transformations for each triangle are computed explicitly using the closed form provided by the transformation algebra [Ale02]. The vertex positions in eq. 2.3 however, are computed every time there is a change in the deformation input. However, the system matrix of eq. 2.3 depends only on the selection of anchor triangles, the mesh connectivity, and the undeformed mesh geometry. Therefore the LU decomposition is precomputed and the position of the vertices is obtained using back-substitution making the system very efficient and allowing it to work at interactive rates.

Since the matrices are symmetric and positive definite, there are more efficient methods to solve the systems such as using Choleski decomposition. However, since the systems are solved essentially once for a given set of anchor triangles, the speed-up would not be critical.

Figures 2.2, 2.3(f), 2.13 and 2.14 demonstrate how a simple coloring scheme for defining the material properties allows us easily to deform articulated models. In the hand example from Figure 2.13, the joints were painted with a flexible material and the user placed three anchors per finger, one at the tip and two at the base, and then deformed each finger independently using a region of influence to speed up the computation. The eagle deformation in Figure 2.13 uses simple coloring of the head, neck and wings to illustrate that the head is the most rigid part while the wings are the most flexible.

2.8. Implementation and Results

Figure 2.14 demonstrates a combination of scaling and rotation applied to the head of the camel. The deformation was created using four anchors on the bottom of the feet, and one anchor on the tip of the nose. The feet anchors were kept in place and the anchor triangle on the nose was rotated and scaled to three times the original size. When using uniform materials (Figure 2.14(top)) the scaling is propagated uniformly through the model. Using different materials it is possible to better control the propagation. In Figure 2.14(bottom), the user defined the neck to be more flexible than the head and the body. With the new materials the head scales to three times the original size, the body remains undeformed, and the distortion is mostly concentrated at the neck. Using standard methods it is not possible to archive such deformation in a single operation. Even defining a region of influence to include only the head and neck would not be sufficient, since the head would not deform uniformly.

For many models the stiffness changes smoothly across the mesh. Tree branches, plant stems, and octopus tentacles are examples of models where the flexibility is proportional to the girth of the model. Figure 2.15 demonstrates how this material-aware deformation framework allows us to define correct bending behavior for an octopus's tentacle. When using a uniform material the tentacle takes the unnatural shape of an arc; however, using non-uniform materials results in the more natural shape of a spiral with the tip bending more than the base. In both deformations the same rotation transformation is applied to an anchor triangle at the tip of the tentacle, while the final position of the tip was computed automatically.

Figure 2.16 demonstrates the application of the algorithm to a non-articulated model. In order to deform the piece of cloth a texture map is used to define the material properties. The bold letters define very stiff areas, while the rest of the model varies in degrees of flexibility.

Figures 2.3(d), 2.11 and 2.12 demonstrate deformation results using material

2.8. Implementation and Results

properties learned from sample poses obtained from [SP04]

For the camel (Figures 2.3(d) and 2.12) the stiffness fields were learned from ten sample poses in 9.5 seconds. For the lion (Figure 2.11) nine sample poses were used and the learning process took 2.1 seconds. As expected, in both cases the estimated stiffness scalar fields are anatomically correct, showing more flexible regions at the joints and stiffer regions along the bones. Figures 2.11(c) and (d) show examples of new anatomically correct poses created using the estimated stiffness fields. Figure 2.12 shows an example of manually modifying the stiffness fields in order to create a desired deformation which does not comply with the sample set. The learned fields allow bending of the leg at the knee and ankle, resulting in an anatomically feasible pose (Figure 2.12 (a)). To handicap the camel, the user manually define these areas as stiff. Applying the same transformation at the tip of the foot causes the leg to raise without bending at the knee or ankle (Figure 2.12 (b)).

Figure 2.17 illustrates deformations using only positional constraints as described in section 2.5 The system automatically computes the appropriate rotation of the head. Since the neck of the camel is more flexible than the rest of the body, most of the rotation will be at the neck, thus preserving the geometric detail of the head. This mesh has 19,536 faces and it was deformed interactively at a frame rate of 3-4 frames per second using the parabolic solver.

Central to this technique is the ability to capture the surface behavior in a pre-processing stage and encode it in the formulation. This deformation method captures the material properties while preserving the simplicity and efficiency common to geometric deformation techniques. This technique can be applied to a wide range of models producing complex results with only few anchors.

Table 2.1 summarizes the deformation statistics of the various models used in the results, measured on a 3GHz Intel Pentium IV with 2Gb of RAM.

2.9. Summary

	Bar	Hand	Lion(c)	Lion(d)	Cloth	Horse	Eagle	Octopus	Camel
#Faces	1596	6274	9996	9996	19602	19996	29232	36542	43775
#Anchors	2	3	4	8	10	2	4	5	5
Preprocessing:									
Weight calculation(s)	0.03	0.23	0.51	1.03	2.91	0.63	1.50	2.76	2.95
Vertex repositioning factorization(s)	0.1	0.48	0.75	0.75	1.58	1.81	2.47	3.16	3.63
Total (s)	0.13	0.71	1.26	1.78	4.49	2.44	3.93	5.92	6.58
Real-time:									
Blending (ms)	10	30	30	30	70	60	100	130	150
Repositioning (ms)	10	60	60	60	90	120	140	210	130
Total (ms)	20	90	90	90	160	180	240	340	280

Table 2.1: Model deformation timings including pre-processing and actual interactive deformation. For the octopus and hand models the number of faces noted is the number of faces in the region of influence.

2.9 Summary

This chapter presented a new mesh deformation technique that incorporates material properties into the geometric deformation framework. Using these properties, the method provides a simple mechanism that allows material-aware behavior of the surface under deformation. This method combines the efficiency, generality and control of geometric methods together with material awareness found in physical and skeleton based methods.

Material properties can be user-driven, where the stiffnesses are specified with a simple brush-like interface, or data-driven where stiffnesses are deduced from a set of given poses. It also can be a combination of the two where the user can override the learned material stiffnesses.

The formulation is simple and efficient. It requires solving only two linear systems, and thus works at interactive rates. The resulting deformations are as-rigid-as-possible, subject to the material stiffness and the user defined transformations,

and therefore maintain the shape details as demonstrated by the results.

2.10 Limitations and Future Work

The decoupling of material properties into bending and shearing coefficients follows the framework of most physically based methods that use a combination of hinge-bending and in-plane stretching energies for the deformation [GHDS03]. However, the current formulation may result in connectivity-dependent deformations. There are several connectivity-independent bending energies available that can be used [GGRZ06], but integrating them into the formulation in equation 2.1 would require dynamic bending coefficients, or in other words φ_{ij} will no longer be a constant. An interesting area of future work is to investigate such a formulation and the possibility of some pre-factorization of the main matrix to enable fast solving when using dynamic material properties.

The quality of a deformation result depends on the material properties that were specified as well as on the selection of anchor vertices and triangles. While in order to achieve a given deformation, the selection of anchors is generally intuitive to the user, the numerical selection of appropriate bending and stiffness coefficients is more obscure. A more intuitive way to qualify the bending and stiffness coefficients might be desirable.

Another limitation of the system is that it does not allow the user to place limits on the bending constraints. For instance, in the bending example in figure 2.3, there is nothing to prevent the user from bending the leg of the camel more than it would be anatomically possible, yielding an unnatural result. Enabling dynamic stiffness constraints as discussed above may solve this problem as well. For instance a very large bending penalty can be added if the leg is already bent a certain amount. This will avoid deforming the camel into an unnatural state.

2.10. Limitations and Future Work

Another direction for future research is to improve the anisotropic model. The current anisotropic model uses a global coordinate frame to decompose local rotations instead of local coordinate frames. Decomposition around local coordinate frames would be preferred because the stiffness maps would be invariant to the position and orientation of an object in space. However, the challenge in doing so is the consistent construction of blending weights across different coordinate frames.

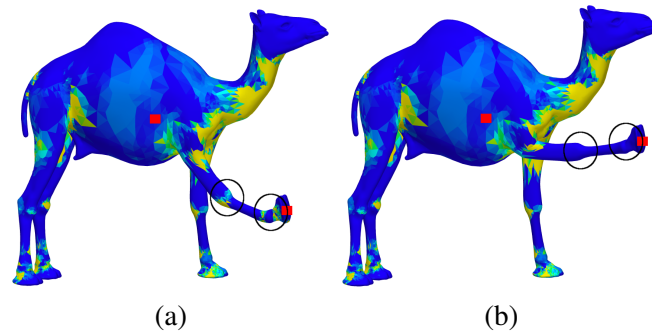


Figure 2.12: Refining learned materials. (a) Bending the front leg of a camel using the estimated stiffness and two anchors. (b) Bending the leg after modifying the stiffness by marking both the knee and ankle joints as stiff

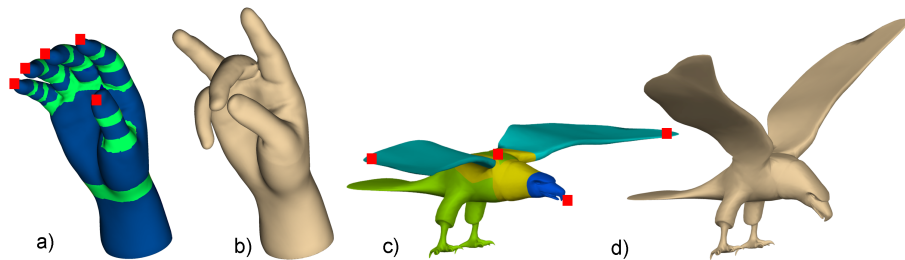


Figure 2.13: Deformation of articulated models using: (a), (c) original models and painted stiffnesses; (b), (d) deformation results

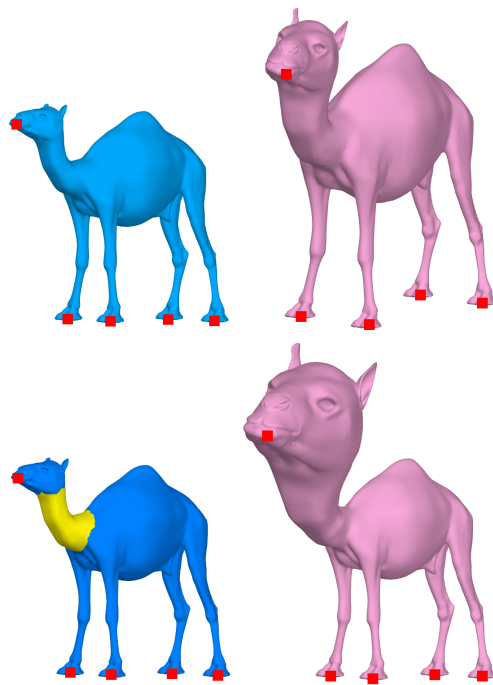


Figure 2.14: Scaling and rotating a camel's head. Top: uniform material causes the entire camel to scale and rotate. Bottom: the stiff head scales uniformly, while the flexible neck absorbs the distortion

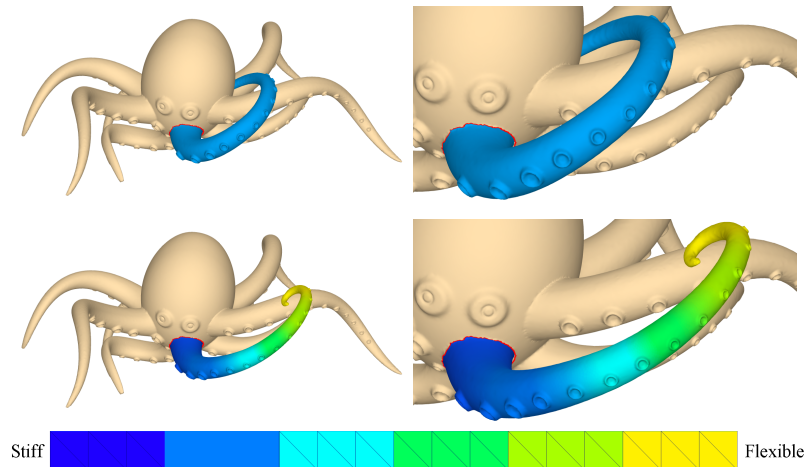


Figure 2.15: Deformation comparison for octopus tentacle in figure 2.1 between uniform stiffness (top) and smooth variation of stiffness (bottom). In both cases the same amount of rotation is applied at the tip of the tentacle while the final position of the tip is calculated automatically

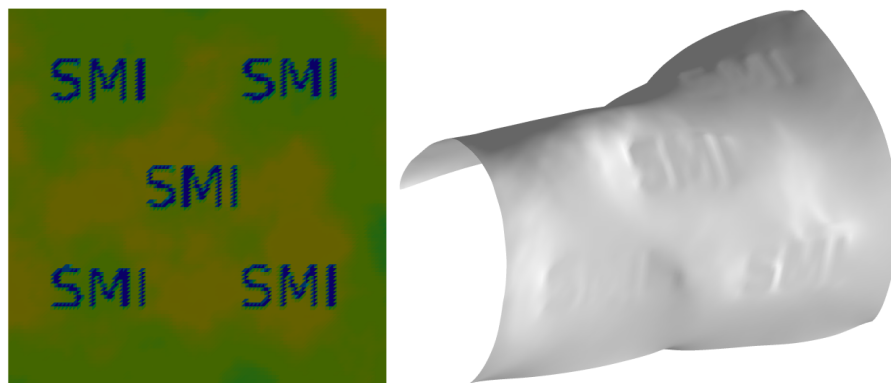


Figure 2.16: Material-aware deformation of cloth. The printed letters are made up of stiffer material; the rest of the model consists of flexible material modulated using Perlin noise to create a natural wrinkled look

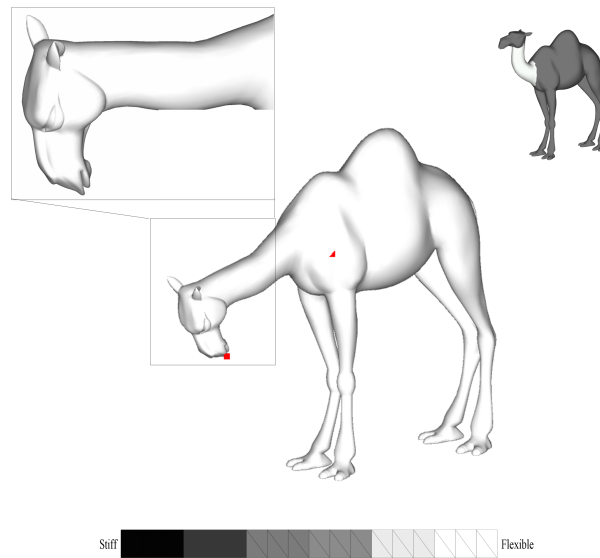


Figure 2.17: Deformation of a camel using positional constraints. The neck of the camel is more flexible than the rest of the body. Three anchors were used: two at the back of the camel on each side of the neck and a third one on the head. The example was generated by simply dragging the vertex on the head to its new position while keeping the other two in place. Note that the geometric details of the head are very well preserved

Bibliography

- [ACOL00] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 157–164. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.
- [ACP02] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. 21(3):612–619, 2002.
- [ACWK04] Alexis Angelidis, Marie-Paule Cani, Geoff Wyvill, and Scott A. King. Swirling-sweepers: Constant-volume modeling. In *Pacific Conference on Computer Graphics and Applications*, pages 10–15, 2004.
- [Ale01] Marc Alexa. Local control for mesh morphing. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 209, Washington, DC, USA, 2001. IEEE Computer Society.
- [Ale02] Marc Alexa. Linear combination of transformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387, New York, NY, USA, 2002. ACM Press.
- [Bar84] Alan H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Com-*

puter graphics and interactive techniques, pages 21–30, New York, NY, USA, 1984. ACM Press.

- [BK04] Mario Bostch and Leif Kobbelt. An intuitive framework for real-time freeform modeling. In *ACM Transactions on Graphics (ACM SIGGRAPH 2004)*, volume 23, pages 628–632, 2004.
- [BK05] Mario Botsch and Leif Kobbelt. Real-time shape editing using radial basis functions. In *Computer Graphics Forum, (Eurographics 2005 proceedings)*, volume 24, pages 611–621, 2005.
- [CGC⁺02] Steve Capell, Seth Green, Brian Curless, Tom Duchamp, and Zoran Popovic;. Interactive skeleton-driven dynamic deformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 586–593, New York, NY, USA, 2002. ACM Press.
- [Dav04] Timothy A. Davis. Algorithm 832: Umfpack — an unsymmetric-pattern multifrontal method. *ACM Transactions on Mathematical Software*, 30(2):196–199, June 2004.
- [GGRZ06] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Zorin d.: Computing discrete shape operators on general meshes. In *Eurographics (Computer Graphics Forum) (2006, 25:2006, 2006*.
- [GHDS03] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

- [Gra03] Graphite, 2003. <http://www.loria.fr/~levy/Graphite/index.html>.
- [IMH05] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. Graph.*, 24(3):1134–1141, 2005.
- [JF03] Doug L. James and Kayvon Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph.*, 22(3):879–887, 2003.
- [JT05] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, 2005.
- [KCVS98] Leif Kobbelt, Swen Campagna, Jens Vorsatz, and Hans-Peter Seidel. Interactive multi-resolution modeling on arbitrary meshes. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 105–114, New York, NY, USA, 1998. ACM Press.
- [LSCO⁺04] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, pages 181–190. IEEE Computer Society Press, 2004.
- [LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005*, page accepted for publication. ACM Press, 2005.
- [MHTG05] Matthias Muller, Bruno Heidelberger, Matthias Teschner, and Markus

- Gross. Meshless deformations based on shape matching. *ACM Trans. Graph.*, 24(3):471–478, 2005.
- [SCOL⁺04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *SGP '04: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 175–184, New York, NY, USA, 2004. ACM Press.
- [SD92] Ken Shoemake and Tom Duff. Matrix animation and polar decomposition. In *Proceedings of the conference on Graphics interface '92*, pages 258–264, San Francisco, CA, USA, 1992. Morgan Kaufmann Publishers Inc.
- [SK04] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT*, pages 68–75, 2004.
- [SP86] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. In *SIGGRAPH '86: Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 151–160, New York, NY, USA, 1986. ACM Press.
- [SP04] Robert W. Sumner and Jovan Popovic. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.
- [SZGP05] Robert W. Sumner, Matthias Zwicker, Craig Gotsman, and Jovan Popovic;. Mesh-based inverse kinematics. *ACM Trans. Graph.*, 24(3):488–495, 2005.
- [TF88] Demetri Terzopoulos and Kurt Fleischer. Modeling inelastic deformation: viscoelasticity, plasticity, fracture. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and*

interactive techniques, pages 269–278, New York, NY, USA, 1988. ACM Press.

- [TPBF87] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM Press.
- [YBS03] Shin Yoshizawa, Alexander G. Belyaev, and Hans-Peter Seidel. Free-form skeleton-driven mesh deformations. In *SM '03: Proceedings of the eighth ACM symposium on Solid modeling and applications*, pages 247–253, New York, NY, USA, 2003. ACM Press.
- [YZX⁺04] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.
- [ZHS⁺05] Kun Zhou, Jin Huang, John Snyder, Xinguo Liu, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Large mesh deformation using the volumetric graph laplacian. volume 24, pages 496–503, New York, NY, USA, 2005. ACM Press.
- [ZRKS05] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.

Chapter 3

Markerless Garment Capture

3.1 Introduction

² Capturing the geometry of moving garments and other cloth is a problem that has recently seen a significant amount of research interest. One goal of such research is to provide a data-driven alternative to cloth simulation in much the same way motion capture provides an alternative to character animation. Perhaps more importantly, cloth capture promises to become an indispensable tool for postprocessing and augmenting live action sequences with computer rendered content.

Consider, for example, a live action sequence in which the texture or material properties of a garment are to be changed in post production (see Figure 4.1). In order for synthetic surface texture to move correctly with the real cloth geometry, it is not only necessary to capture the 3D shape of the garment in each time step, but also to establish a temporally consistent parameterization of the surface that prevents the texture from floating on it.

For this reason, the state of the art in cloth and garment capture is to use markers printed on the garment to simultaneously track geometry and parameterization. The need for such markers unfortunately makes existing cloth capture approaches unattractive for several reasons. First, the effort to create garments with unique marker patterns can be prohibitive for some applications. Second, the types of

²A version of this chapter has been published. Bradley, D. and Popa, T. and Sheffer, A. and Heidrich, W. and Boubekeur, T. (2008) Markerless Garment Capture, ACM Trans. Graphics (Proc. SIGGRAPH)

fabrics on which markers can be printed are typically limited, making it difficult to capture fabric-specific differences in garment motion. However, the characteristic differences between, say, silk, cotton, and fleece, are precisely the reason why one would want to capture cloth as opposed to simulating it. A key goal of this work is thus to capture the motion of specific existing garments, with a specific design, and made from a specific fabric.

This chapter presents a marker-free approach to garment capture that realizes this goal of capturing off-the-shelf garments. First, a multiview stereo approach is used to obtain initial garment geometry for each time step. Although this initial geometry can have a significant number of holes, by leveraging low fabric stretch and utilizing the topology of the scanned garments, the algorithm produces a parameterization of the geometry that is consistent across time, without explicitly tracking motion. This temporally consistent parameterization helps us fill the holes in the geometry, producing a high quality final result. A detailed overview of the method is provided in Section 5.3.

3.2 Related Work

This section describes related methods for cloth capture and provides a short overview of geometry processing techniques related to this work.

3.2.1 Cloth and Garment Capture

Over the last several years, techniques for capturing the motion of cloth have evolved from methods for tracking a single cloth sheet to systems for reconstructing full garments under complex motion.

Sheets of cloth and partial garments: Early work on this topic focused on single sheets of cloth with existing textures [PH03, SM04], or small subsets of a full

3.2. Related Work

garment with patterns that were custom-printed onto the fabric [GKB03]. Hasler et al. [HAR⁺06] use an analysis-by-synthesis approach that sets their method apart from the other work. They too, however, rely on markers in the form of patterned cloth.

This early work identifies the basic task of combining the geometry of the cloth with a temporally coherent parameterization and introduces the use of markers as a solution to this problem. Capturing full garments with large areas of occlusion is, however, significantly more challenging.

Full garments: A first solution to this problem was proposed by Scholz et al. [SSK⁺05], who suggested a special marker pattern composed of a matrix of colored dots, in which each 3×3 neighborhood of dots is uniquely identifiable. A simple thin-plate approach was used to fill any holes in the geometry. White et al. [WCF07] improved on these results by thoroughly analyzing the design of suitable marker patterns. They also improve on the hole filling, using a new data-driven technique. This work is currently the state of the art in garment capture and produces excellent results. However, like the work by Scholz et al., White et al.’s method requires custom-tailored garments made from fabric on which the marker patterns have been printed. It thus cannot deal with arbitrary off-the-shelf clothing.

Markerless approaches: Recently, some markerless capture techniques were proposed, albeit with a somewhat different focus. Bhat et al. [BTH⁺03] capture the motion of sheet samples made from different fabric to estimate parameters for a cloth simulation. Rosenhahn et al. [RKP⁺07] focus on estimating the pose of a human wearing a garment using an analysis-by-synthesis approach similar to Hasler et al. [HAR⁺06]. While they do show some approximate skirt geometry, they do not discuss how to capture the exact shape of complicated garments in the presence of occlusions. In a recent paper, Hernandez et al. [HVB⁺07] propose a

photometric stereo approach for capturing deformable geometry using garments as an example. While their approach is technically marker-free, they require patterned cloth mesostructure such as knitting patterns to compute optical flow. They do not discuss how to handle occlusion, and due to the interaction of differently colored light sources it is not clear how to extend their method to 360° surround capture.

Face capture: The overall design of the garment capture system presented in this thesis has similarities to facial capture systems such as [ZSCS04] or [BBA⁺07]. However, garment capture is fundamentally a different problem due to inherent occlusions and chaotic folding and rippling nature of fabrics.

Unlike previous methods, this technique is able to reconstruct high resolution models of full garments under normal human motion without the need for printed on-surface markers or high-frequency surface texture.

3.2.2 Related Geometry Processing Techniques

From a geometry perspective, garment motion capture can be posed as a problem of tracking a deformable surface over time. Recently, two solutions have been presented for this problem, starting from an initial set of point clouds [WJH⁺07, MFO⁺07]. While Wand et al. reconstruct both the geometry and temporal correspondences, and Mitra et al. register the point clouds without computing explicit correspondences, both methods assume that the motion is locally rigid. Therefore, these methods would not perform well for cloth capture.

Establishing temporal correspondences between acquired frames of a moving surface requires a bijective mapping, commonly referred to as *cross-parameterization* [KS04] or *inter-surface mapping* [SAPH04], between the frames. In this chapter, the term *consistent cross-parameterization* of multiple frames is defined as a cross-parameterization from-frame-to-frame that maps a point in one frame to the same point, subject to motion, in all other frames. These meshes are *compatible* if they

3.3. Overview

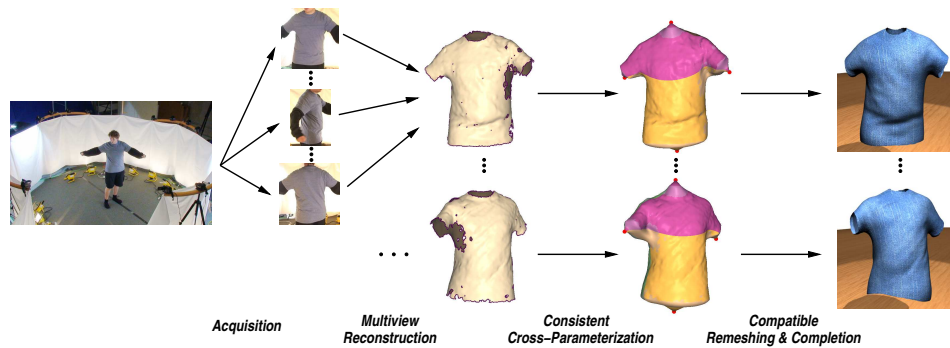


Figure 3.1: Overview of the technique

have the same connectivity and an explicit vertex correspondence. Most methods for cross-parameterization and compatible remeshing of surfaces rely on accurate user-provided correspondences for a set of markers on the surfaces [ACP03, SAPH04, KS04, ASK⁺05, KS05]. This manual positioning of markers is impractical for hundreds or thousands of frames produced by a capture process, which is the main reason why existing cloth capture approaches rely on printed markers.

In contrast, a consistent cross-parameterization is established between acquired frames of a moving surface through the use of a small number of *off-surface anchors* computed on-the-fly, using user-assisted keyframing when necessary. Thus markers printed on the fabric are no longer needed.

3.3 Overview

The method for markerless garment capture presented in this thesis consists of four key components, illustrated in Figure 3.1:

Acquisition: A unique 360° high-resolution acquisition setup using sixteen inexpensive high-definition consumer video cameras is deployed. This lighting setup avoids strong shadows by using indirect illumination. The cameras are set up in

a ring configuration in order to capture the full garment undergoing a range of motions.

Multiview Reconstruction: The input images from the sixteen viewpoints are fed into a custom-designed multiview stereo reconstruction algorithm [BBH08] to obtain an initial 3D mesh for each frame. The resulting meshes contain holes in regions occluded from the cameras, and each mesh has a different connectivity.

Consistent Cross-Parameterization: A consistent cross-parameterization among all input meshes is computed. To this end, I used strategically positioned *off-surface anchors* that correspond to the natural boundaries of the garment. Depending on the quality of boundaries extracted in the multiview stereo step, the anchors are placed either fully automatically, or with a small amount of user intervention.

Compatible Remeshing and Surface Completion: An effective mechanism for compatible remeshing and hole completion using a template mesh is introduced. The template is constructed from a photo of the garment, laid out on a flat surface. The template is cross-parameterized with the input meshes, and then deformed into the pose of each frame mesh. This deformed template is used as the final per-frame surface mesh. As a result, all the reconstructed per-frame meshes have the same compatible connectivity, and the holes are completed using the appropriate garment geometry.

Sections 3.4 to 3.7 elaborate the garment capture method, section 3.8 shows results using this method and a discussion of the method and conclusions are presented in section 3.9.

3.4 Acquisition

³This section describes the acquisition setup, including the hardware, calibration, and synchronization.

Cameras. The acquisition setup consists of sixteen high-definition Sony HDR-SR7 video cameras, mounted in a ring around the actor wearing the garment to be captured. External and internal camera parameters are calibrated using ARTag markers and the automatic calibration technique by Fiala and Shu [FS05].

The cameras record 29.97 frames per second, stored as approximately sixty fields per second on camera-internal hard drives. Each field is converted to a full resolution image using a simple spatio-temporal de-interlacing technique applied to the missing scanlines. The method weights spatial interpolation higher in regions with fast motion, and uses simple temporal interpolation where there is no motion. A pixel $I(x, y, t)$ on a missing scanline is reconstructed as

$$I(x, y, t) = w \cdot \frac{I(x, y, t - 1) + I(x, y, t + 1)}{2} + (1 - w) \cdot \frac{I(x, y - 1, t) + I(x, y + 1, t)}{2},$$

where w is a Gaussian function of the difference $I(x, y, t - 1) - I(x, y, t + 1)$, a choice that is inspired by the Bilateral filter [TM98]. The final data stream is in the form of sixteen high-resolution (1920×1080) images, captured at sixty images per second.

Synchronization. Since all cameras operate independently, and cannot be synchronized using any external trigger, software synchronization needs to be applied in a post-process. A *moving-object* synchronization method is used. At the beginning of each capture sequence the ARTag calibration grid is rotated approximately 360° on a manually operated turn table over a period of ten seconds. Each camera

³The work described in this section is not part of this thesis, it has been added for completeness only.

auto-detects the grid in each frame and the amount of rotation is used to determine a temporal offset. Although this method yields offsets with sub-frame precision, we choose to round the offsets to the nearest integer field. At sixty fields per second, no artifacts due to sub-field synchronization differences are noticed during moderately fast motion. For high speed motion, optical flow could be computed from frame to frame for each camera and the sub-frame offset could be used to interpolate between frames. We leave such interpolation for future work.

Lighting. Some care has to be taken in the lighting setup as well. As in all movie and video shots, bright illumination is necessary to avoid noise and motion blur by reducing both the camera gain and the exposure time. Moreover, direct illumination by a few bright spot-lights causes hard shadows that exceed the dynamic range of the cameras, thus preventing geometry reconstruction in the shadowed regions. The ideal illumination is therefore bright, diffuse lighting, not unlike that used on movie sets.

To achieve this goal, an indirect lighting setup was devised as depicted in Figure 3.2. The outer ring of the setup is lined with white cloth, which was used as a diffusing reflector. Sixteen inexpensive 500W halogen lights are placed inside the ring, pointing outwards at the cloth. This setup results in a bright, uniform illumination of the ring interior.

3.5 Multiview Reconstruction

⁴The captured and processed images are used as input for a multiview stereo reconstruction algorithm that was developed for this purpose. The method, which is described in full detail elsewhere [BBH08], was specifically designed to work

⁴The work described in this section is not part of this thesis, it has been added for completeness only.

3.5. Multiview Reconstruction



Figure 3.2: Acquisition setup.

well with the relatively small number of views available in the setup. According to the Middlebury multiview stereo evaluation [mvi, SCD⁺06], this method is currently the best multiview algorithm for such datasets in terms of both precision and performance.

For the sake of completeness, a brief outline of the multi-view stereo algorithm follows. The algorithm has two stages, binocular stereo on image pairs, followed by surface reconstruction. Figure 3.3 shows a diagram of the individual stages.

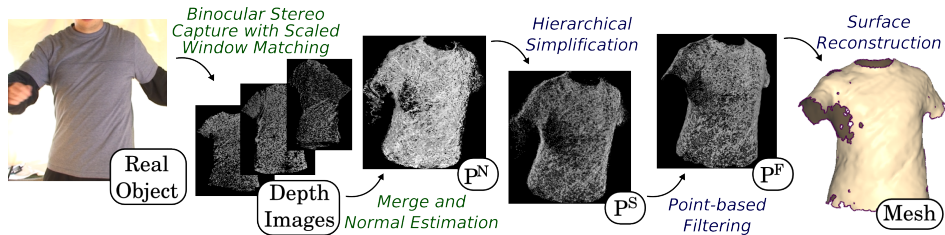


Figure 3.3: Multiview reconstruction: starting from a set of depth images, a 3D point cloud is generated (P^N), simplified (P^S), filtered (P^F) and meshed.

The binocular stereo part of the algorithm creates depth maps from pairs of adjacent viewpoints. After rectifying the image pairs, the relationship between the views causes distortions of the comparison windows. A *scaled-window matching* technique is used to compensate for these distortions, especially in high curvature

regions and for sparse viewpoints (i.e., large baselines). The depth images from the binocular stereo pairs are converted to 3D points and merged into a single dense point cloud.

The second part of the algorithm aims at reconstructing a triangle mesh from the initial point cloud. Instead of implicit reconstructions, which smoothly interpolate over holes but introduce stretch in doing so, a reconstruction method that keeps the holes so that they can be filled based on parameterization information generated further down the pipeline is preferred. First, a hierarchical vertex clustering approach is used to eliminate excessive sampling density. After downsampling, the simplified point cloud still contains some noise. Unlike other methods, which integrate noise removal into the meshing algorithm, the noise removal is explicitly handled in a separate phase. The processed samples are finally triangulated using an improved version of the method by Boubekeur et al. [BRS05].

All stages of the algorithm are designed to scale well. The output of the reconstruction is a triangle mesh for each video frame. Even though the multiview stereo algorithm is the state of the art for sparse views according to the Middlebury benchmark, significant holes remain in occluded and saturated image regions. The subsequent stages of the cloth capture system therefore have to robustly deal with missing data in the cross-parameterization between the frames, as well as fill in the missing regions from a template. Figure 3.4 shows the multiview reconstructed frames from three different garments in motion, along with the final reconstruction result after all stages of the algorithm.

3.6 Consistent Cross-Parameterization

The next step of the method constructs a consistent cross-parameterization between the incomplete per-frame meshes. In order for cloth capture to be useful,

3.6. Consistent Cross-Parameterization



Figure 3.4: Result of multiview reconstruction for different garments (middle row). One of the sixteen input images is shown above each result, and the final reconstruction after parameterization and completion is shown below.

the parameterization needs to capture the motion of the garment, mapping a point on the garment at one frame to the position of the same point on the subsequent frame. Inconsistencies in the mapping are likely to lead to visible artifacts such as “floating” texture on the garment surface. There is no explicit information on the motion of the garment, and estimating such motion is quite challenging. Instead this method relies on geometric properties of the garments to compute a consistent parameterization. Many real fabrics exhibit very low stretch in regular use (see e.g. [GHF⁺07]), and thus one can assume that a consistent parameterization of garments is (nearly) isometric. Moreover, since most garments exhibit no ro-

3.6. Consistent Cross-Parameterization

tational symmetries, such an isometric parameterization is unique. A key insight of this work is therefore that a consistent parameterization between frames can be obtained simply by finding an as-isometric-as-possible, i.e. stretch-minimizing, parameterization between the per-frame meshes, instead of employing some form of tracking.

Several of the cross-parameterization methods reviewed in Section 3.2 can process incomplete meshes. However, in this setup there are additional challenges. Due to the large number of frames to be processed, it is impractical to manually add markers in every frame to assist the parameterization or to use computation-intensive approaches such as the work by Anguelov et al. [ASK⁺05] or Schreiner et al. [SAPH04].

In order to parameterize the meshes in a reasonable time frame, a *base mesh* approach is used [PSS01, SAPH04, KS04, KS05]. A base mesh (Figure 3.5, left) is a low-resolution mesh whose vertices correspond to a consistent set of marker vertices on the input meshes. In base mesh parameterization, each mesh is mapped to the base, such that the marker vertices map to the corresponding base vertices. This provides a cross parameterization between the inputs, where a map from one mesh to another is given by combining the map from the first mesh to the base with the inverse map from the base to the second mesh. In all previous methods, the on-surface marker vertices are manually specified by the user for each input mesh. As mentioned before, it would be infeasible in this setting to provide such markers manually for all frames. However, we observe that garments, in contrast to generic geometric models, have distinct, identifiable, topology with several boundary loops. For instance a T-shirt has four boundaries: a neckline, two sleeves and a waistline. The algorithm identifies the corresponding boundaries across all frames, as discussed in Section 3.6.1, and associates a floating, off-surface anchor vertex with each boundary (see Figure 3.5). The anchors replace the on-surface markers

3.6. Consistent Cross-Parameterization

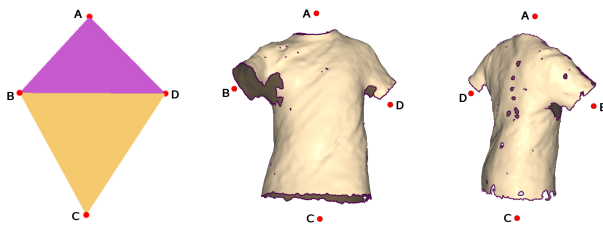


Figure 3.5: Off-surface anchor vertices are used to establish boundary correspondences between the tetrahedral base mesh (left) and the different frames of the same garment (right)

in the base mesh construction and subsequent parameterization (Section 3.6.2).

In general, even when using a base mesh for stretch-minimizing cross-parameterization, one needs to measure and optimize the stretch directly between the input meshes, a fairly time-consuming process. However, given two meshes that are isometric, mapping both of them using stretch-minimization to the same base mesh, the maps should be nearly identical, resulting in a nearly isometric map between the input meshes. Even in the presence of holes, careful design of a low-stretch base mesh parameterization algorithm, discussed in Section 3.6.2, allows us to compute cross-parameterizations that are consistent across the entire sequence by parameterizing the meshes independently to the base. This makes the method fast enough to be used in practice and allows for parallel processing of individual frames.

3.6.1 Positioning Off-Surface Anchors

The off-surface anchors in this setup correspond to the natural boundaries of the processed garment and are mapped to the vertices of the base mesh during parameterization (Figure 3.5). The anchors are placed manually for the first frame of each video sequence and are updated over time to follow the motion. Each anchor is updated automatically, except where the multiview algorithm has failed to reconstruct a sufficient amount of geometry on the corresponding mesh boundary

(see Figure 3.6, right). As mentioned in Section 3.5, missing geometry can be a problem due to occlusion and the relatively small number of cameras.

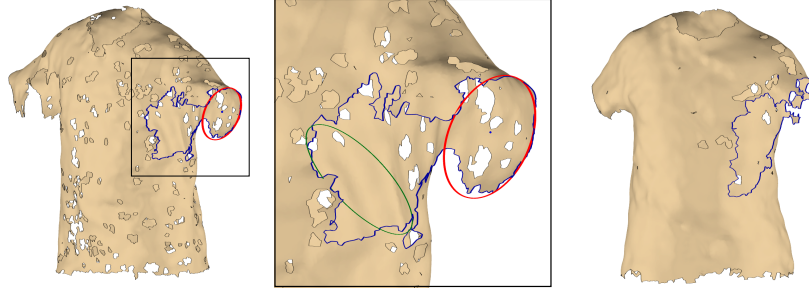


Figure 3.6: Garment boundary tracking. Left: the automatic method correctly chooses the red ellipse based on confidence, size and temporal similarity. Center: using confidence alone, the green ellipse would be incorrectly chosen. Right: the boundary loop contains very little of the actual garment boundary and user-assisted tracking is required

Automatic boundary tracking. Boundary loops of typical garments have an approximately elliptical shape. Given an elliptical approximation of each boundary, the corresponding anchor is placed a fixed distance away from the ellipse center with an associated normal vector pointing away from it. The normals are used in the hole filling process, as discussed in Section 3.6.2. To find the best fitting ellipse, a variant of the RANSAC algorithm [FB81] is used: for each boundary loop of the mesh that corresponds to a garment boundary, six boundary vertices are iteratively chosen at random and then fit an ellipse to these vertices. The *confidence* for any given ellipse is calculated using the number of boundary vertices that agree with that ellipse. The selection of a best fitting ellipse is further guided by size and temporal similarity to the position of the corresponding ellipse in the previous frame. The number of RANSAC iterations required is on the order of $\binom{n}{6}$, where n is the length of a typical boundary. Missing portions of the geometry such as sections under the arm, which are often not captured by the multiview stereo algorithm, can

cause significantly prolonged boundary loops. As a result, 50K to 100K iterations may be required per boundary for such frames, taking fifteen to thirty seconds per frame.

Anchor positions and normals are smoothed in time using Gaussian smoothing. Figure 3.7 shows the resulting anchors for a short sequence of a garment undergoing motion. The anchors with normals are shown for each frame, and the garment geometry is shown for one frame of the sequence.

The assumption of elliptic boundaries can be violated for some garments such as dresses where some boundaries, especially the neck one can have a more arbitrary shape. Since this method can deal with significant holes along the boundary, as shown in Figure 3.6 (left), a non-elliptical boundary can be seen as an elliptical boundary with some geometry missing. Therefore, just like in the case of missing geometry it is sufficient to consistently fit an ellipse only to parts of the boundary. In the dress example in Figure 3.16 the algorithm automatically placed the off-surface anchors for a non-elliptical boundary. A more in-depth discussion of this ellipse-fitting method is presented in Appendix B.

However, for some degenerate boundaries such as the one shown in Figure 3.6 (right), the automatic positioning fails. A short consecutive sequence of poor frames can still be handled gracefully, as the method will auto-detect and skip over the problematic boundaries when no suitable ellipse is found. However, this technique cannot recover from sequences with a large number of consecutive unrecognizable boundaries.

User-assisted boundary tracking. For sequences where the automatic anchor placement fails for a larger number of consecutive frames, a semi-automatic alternative was developed. For a number of keyframes, the user chooses an ellipse for each boundary from a selection of high-confidence ellipses determined by the

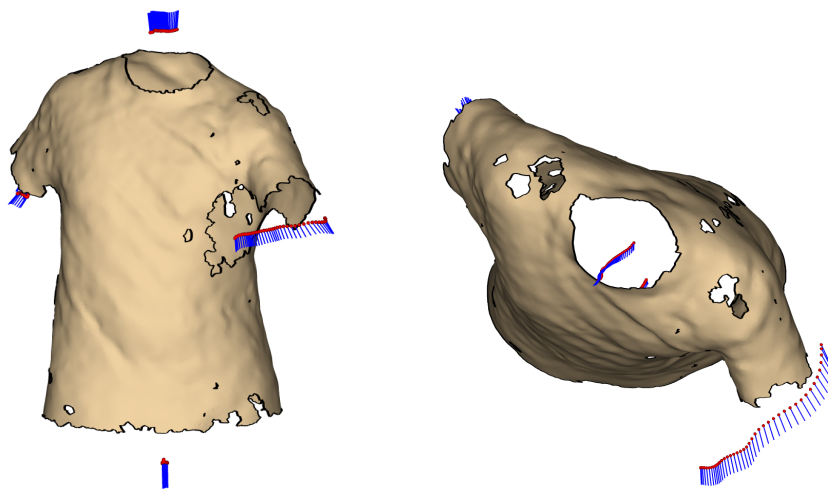


Figure 3.7: Automatic off-surface anchor positioning across a sequence of frames. Anchor vertices are shown in red, while normals are shown in blue. Front view (left) and top view (right)

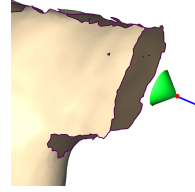
RANSAC algorithm. Choosing an ellipse from a precomputed set typically takes about 10 seconds per boundary. In between keyframes, the anchor positions are interpolated using splines. The required density of keyframes depends on the speed of motion; for fast motions, a keyframe approximately every 30 frames (1/2 second) yields good results.

3.6.2 Base Mesh Parameterization

For each input garment sequence we first construct a base mesh and then map the per-frame meshes onto the base. The base is constructed by positioning the vertices at the locations of the corresponding anchors in a canonical pose (Figure 3.5, left). Given the common base, a stretch-minimizing parameterization of each mesh onto the base is computed using the computed anchors.

3.6. Consistent Cross-Parameterization

As an initialization step, a small cone is created around each anchor, with the apex pointing in the direction of the anchor normal (right). These cones are used later-on to facilitate smooth hole completion. Since the parameterization method used mostly follows the work by Kraevoy and Sheffer [KS05], I focus on the differences with their method and refer the interested reader to their paper for more details.



The algorithm first segments the mesh into charts corresponding to the base mesh facets and constructs an initial parameterization by mapping each chart to the corresponding facet. Since the anchor cones form separate connected components, the gaps between them are triangulated and the corresponding boundaries before computing the segmentation. Given the initial parameterization, the method iterates between re-triangulating the holes in the mesh, including the gaps between the anchor cones and the actual surface, and reparameterizing the mesh onto the base. The main goal of the process is to parameterize the mesh onto the base with minimal stretch.

During re-triangulation, Kraevoy and Sheffer [KS05] first triangulate holes on the base, and then project the new vertices back to 3D using a Laplacian type formulation that minimizes

$$\sum_i (P_i - \sum_{j \in N(P_i)} \omega_{ij} P_j)^2, \quad (3.1)$$

where P_i are the projected vertices with neighborhood $N(P_i)$, and ω_{ij} are the normalized mean value weights [Flo03] from the base embedding. This formulation creates a membrane type surface, filling the holes in 3D with C^0 continuity along hole boundaries (Figure 3.8, center). The discontinuities along the boundaries lead to a sub-optimal estimation of the actual hole shape and area and increase the pa-

parameterization stretch. This problem is rectified by introducing additional terms $(P'_i - \sum_{j \in N(P'_i)} \omega_{ij} P_j)^2$ into the minimized functional in Equation 3.1 for the vertices P'_i on the hole boundary. While the 3D positions of these boundary vertices remain fixed, the new terms influence the positions of the hole vertices adjacent to them. The additional terms effectively eliminate the discontinuities across hole boundaries, creating a C^1 effect (Figure 3.8, right).

Note that the membrane retraction is suitable for establishing the base mesh parameterization, but is not always sufficient for final garment hole completion. Filling holes in complex geometric regions, such as armpits, using the generic membrane generates oversmoothed geometry (Figure 3.11, left). This problem is addressed in Section 3.7, with a template-based remeshing and surface completion technique.

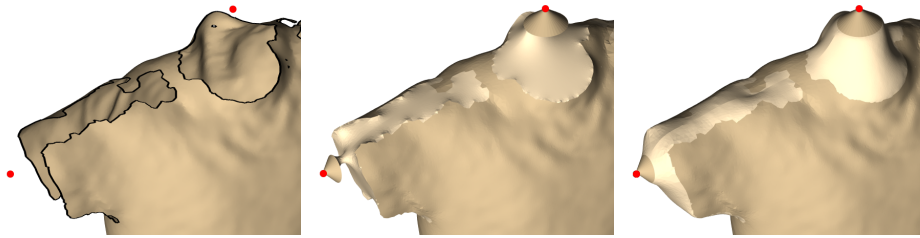


Figure 3.8: Smooth hole completion during parameterization. Left: an incomplete T-shirt. Center: C^0 membrane. Right: C^1 membrane. The membranes are rendered in lighter tones for visualization

Since the geometry connecting the anchors to the mesh is iteratively updated as the parameterization improves, any inaccuracy in the anchor position is automatically corrected, thus the anchors need not be placed as accurately as on-surface markers. Therefore, the anchor-based parameterization is fairly robust to noise and artifacts in the input meshes.

The result of the base mesh parameterization method for two input frames is

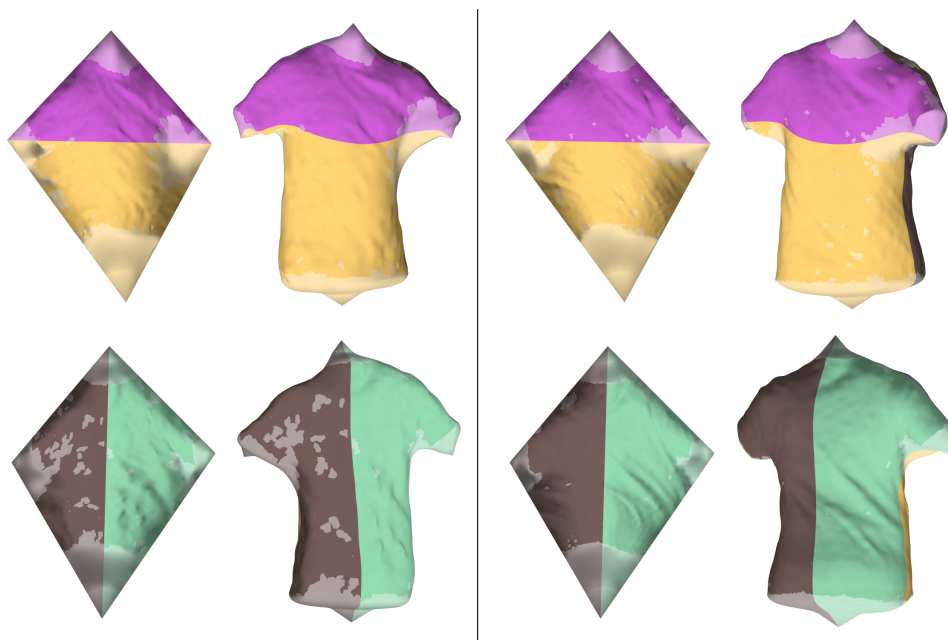


Figure 3.9: Two frames parameterized onto the base mesh, illustrated as normal maps (top row: front view, bottom row: back view). The charts corresponding to the base mesh faces are shown on the garments. Again, hole regions are rendered in lighter tones

illustrated in Figure 3.9. Thanks to the near-isometry between the frames, the natural boundaries on the normal maps are well aligned. Once each mesh has been parameterized onto the common base, a trivial cross-parameterization is established between all frames.

3.7 Compatible Remeshing and Surface Completion

The acquired garment frames typically have numerous holes, which are filled with a C^1 continuous membrane during the parameterization. For final reconstruction, these membranes work well for flat or convex regions, but generate an over-smoothed surface in saddle-like regions (see Figure 3.11). One option for replacing

the membrane with more realistic geometry is data-driven hole filling [WCF07]. White et al. complete hole geometry using the mapping from other input frames that do not contain the hole. However, some regions of the garment may be occluded in all input frames. For this reason, a template mesh is used, which is generated by “inflating” a photo of the garment to provide a more faithful geometry completion. Additionally, a common compatible connectivity is established by using the template triangulation for all frames.

Template construction. The template mesh is constructed from a single photo of the garment, laid out on a flat surface (Figure 3.10, left). The silhouette of the garment is extracted from the image and the interior is triangulated [She96] creating a 2D mesh. The mesh is duplicated for the back surface, and garment boundaries are indicated by the user with brush strokes.

The 2D mesh is then inflated into 3D, using a variant of the iterative physical simulation approach of Mori et al. [MI07]. Each iteration consists of two steps. First, every face is moved slightly in its normal direction, to mimic the effect of internal pressure. Then the length of each edge is adjusted to preserve the stretch of the material. In the work of Mori et al., material stretching is prevented while compression is tolerated, a desired effect for their application. For the garment template, it is necessary to prevent both stretch and compression. Thus, the second step of the algorithm is modified to include a compression prevention condition. In this step, the displacement d_{v_i} of a vertex v_i is computed as a weighted sum of the forces (t_{ij}) from the neighboring edges (E_i) :

$$d_{v_i} = \frac{\sum_{e_{ij} \in E_i} \{A(e.left\ face) + A(e.right\ face)\} t_{ij}}{\sum_{e_{ij} \in E_i} \{A(e.left\ face) + A(e.right\ face)\}} \quad (3.2)$$

3.7. Compatible Remeshing and Surface Completion

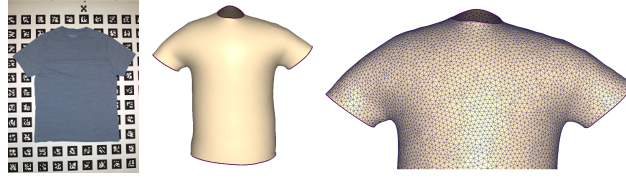


Figure 3.10: Template construction. A photo of the garment (left) is inflated to 3D

$$t_{ij} = \begin{cases} 0.5 \cdot (v_j - v_i) \cdot \frac{|v_i - v_j| - l_{ij}}{|v_i - v_j|} & \text{if } |v_i - v_j| \geq l_{ij} \\ 0.5 \cdot (v_i - v_j) \cdot \frac{l_{ij} - |v_i - v_j|}{l_{ij}} & \text{if } |v_i - v_j| < l_{ij} \end{cases}, \quad (3.3)$$

where $A(f)$ is the area of a face f and l_{ij} is the length of an edge e_{ij} in the initial 2D mesh. This formulation differs from Mori et al. in Equation 3.3, where the first condition is for preventing stretch and the second condition is for preventing compression.

A template is created only once for each garment. The constructed template of a T-shirt is shown in Figure 3.10.

Remeshing and completion The cross-parameterization of the input meshes will be used for consistent texture mapping and temporal smoothing of the geometry. Thus, the meshes for all frames have to be compatible, i.e., have the same connectivity and explicit vertex correspondence. This is achieved by projecting the uniform triangulation of the template mesh onto each frame.

The template is parameterized onto the base mesh in the same way as the input frames, thus establishing a mapping between the template and each individual frame. This mapping is sufficient for remeshing each frame with the template connectivity by simply mapping the vertices of the template to each frame. However, for the hole regions in each frame the geometry of the template is used instead of the membrane. Therefore only those vertices of the template that map to the original frame geometry are kept. To obtain the position of the remaining vertices, the

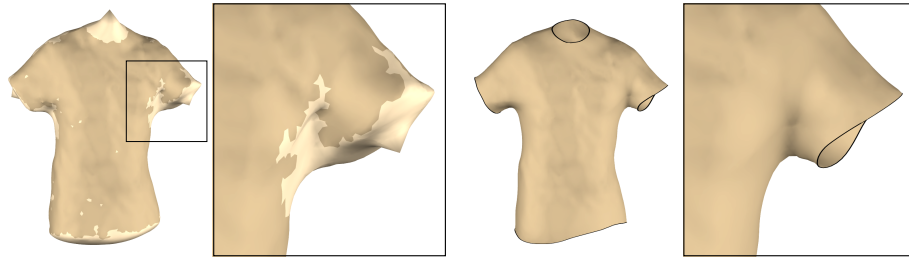


Figure 3.11: Initial C^1 smooth membrane (left) and result of surface completion using the template from Figure 3.10 (right)

template is deformed into the pose of the frame, using the mapped vertex positions as constraints. The deformation method of Sheffer and Kraevoy [SK04] was used, although other algorithms could be used instead. Figure 3.11 shows the result of the completion for one frame. This technique produces a compatible, uniform triangulation of the whole frame sequence, while completing the holes with surface patches that match the captured garment.

Finally, the frame sequence is smoothed in the temporal domain using Gaussian smoothing to remove temporal noise.

3.8 Results

The results of different garment captures are shown in Figures 3.12- 3.17. It is possible to render the captured geometry separately, as a replacement of the original garment, and as an augmentation to the original video frame. This technique is, to my knowledge, the first method that is able to produce all three of these results from a single capture. In particular, augmentation would not be possible without markers remaining visible in most of the previous work. The reader is encouraged to view the full sequences in the video here:

<http://www.cs.ubc.ca/labs/imager/tr/2008/MarkerlessGarmentCapture/>

Figure 3.12 shows different frames from a T-shirt sequence. This is the same T-

3.8. Results

shirt that was used in illustrations throughout the chapter. Due to boundary issues discussed in Section 5.4, the user assisted anchor placement had to be employed for parts of this sequence. Note that this garment is rather tight, and the actor's stomach motion is correctly reconstructed.

For a fleece vest, shown in Figure 3.13, off-surface anchors were placed entirely using the automatic boundary tracking technique. Another important observation is that the multiview algorithm produced only small holes for the vest, unlike the large holes caused by occlusion on the T-shirt. As a consequence, it was possible to reconstruct the vest without creating a template mesh. Instead, a compatible remeshing was achieved by performing a uniform triangulation on the base mesh and then projecting the geometry to each frame. The fleece vest also illustrates that the system can indeed capture garments made from different fabrics. While it would be relatively straightforward to print marker patterns onto the cotton fabric of the T-shirt, producing patterned fleece is not as easy, due to the fuzzy nature of this fabric. Note that this fuzziness does not prevent reconstruction of features such as folds with this method.

A different T-shirt is shown in Figure 3.15. The user assisted anchor placement was also used for this sequence. This garment is larger and more loosely fitting than the others, resulting in more ripples, which this capture method successfully reconstructs.

Two full-body garments were also reconstructed, demonstrating the versatility of this approach. A pink dress is shown in Figure 3.16, and a blue dress in Figure 3.14. For the pink dress, off-surface anchors were placed with the automatic boundary tracking method, while the user-assisted method was used for the blue dress.

Figure 3.17 shows another result of garments made from different fabric, in this case a long-sleeve nylon-shell down jacket. Anchor placement in this case was

again partially manual.

Processing a sequence from captured video to the final result requires approximately one hour per frame, which can be easily parallelized. The majority of the time is spent in the multiview reconstruction step, since the input data is in the form of high-definition video and the multiview stereo algorithm establishes dense correspondences between images. A typical frame for the T-shirt data sets reconstructs over a million surface points, which are used to construct an initial triangulation with over 100K vertices. The very large resolution is required at this point in order to preserve high-frequency detail in the garments, particularly since the mesh reconstruction is non-uniform. The re-meshed sequences typically have around 20K vertices.

3.9 Conclusion

In this chapter, I have presented the first method for markerless capture of full garments and demonstrated its viability on a number of examples. The acquisition and reconstruction methods generate per-frame meshes that capture the complex, changing structure of garments during human motion. The parameterization and completion algorithms leverage fabric incompressibility and garment topology to consistently parameterize the frame sequences and generate realistic garment geometry and motion faithful to the input.

The garment capture system presented in this chapter has two main limitations. First, while the technique can handle significant occlusions, it does not handle situations where the garment surface comes in contact with itself, e.g. in case a sleeve touches the torso. The initial multiview reconstructions in these situations can contain incorrect surface topology, which is not handled in the subsequent process. For this reason, the actors were asked to keep their arms away from their torso. Second,

3.9. Conclusion

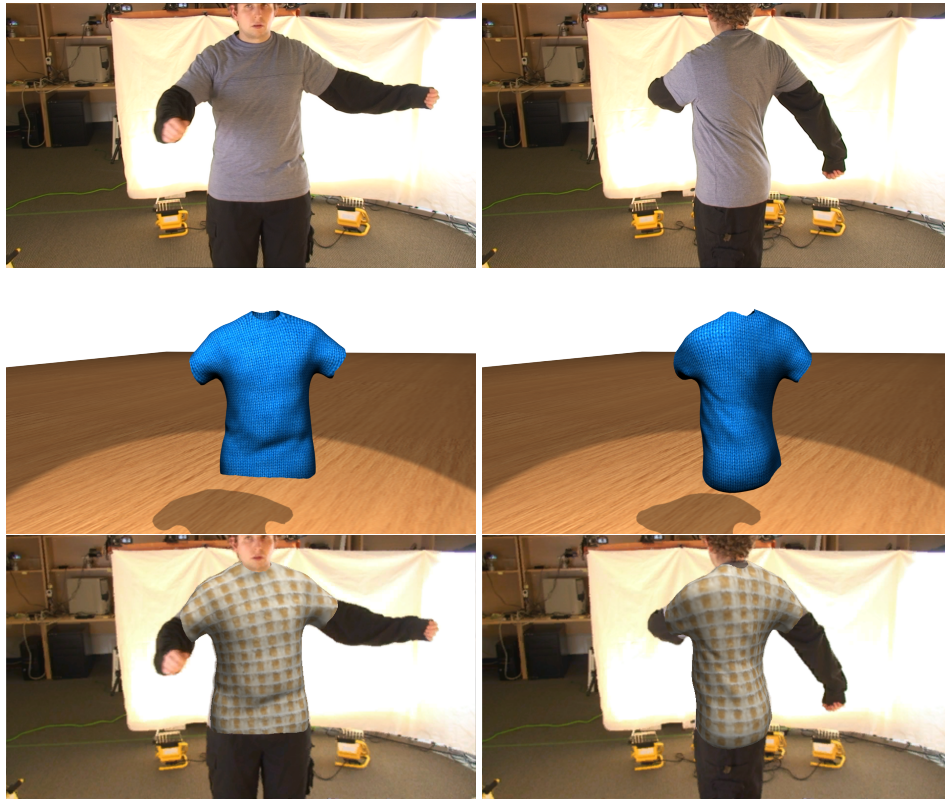


Figure 3.12: Capture results for five frames of a T-shirt. From top to bottom: input images, captured geometry, T-shirt is replaced in the original images

the necessary spatial and temporal smoothing steps in order to produce noise-free results tend to remove the fine details in the geometry. As a result, the reconstructed garments appear to have fewer wrinkles than the original input images. Despite these limitations, this method greatly advances the current state-of-the-art in garment capture.

This method was used to reconstruct several different garments, with and without sleeves. In the future I would like to test it on more types of garments. For example, pants have only three boundary loops rather than the four boundaries

3.9. Conclusion



Figure 3.13: Capture results for a fleece vest. From top-left to bottom-right: one input frame, captured geometry, vest is replaced in the original image, the scene is augmented by adding a light source and making the vest more specular

of the examples in this chapter. Such different types of garments will require a different structure for the base mesh. In the case of pants, the base mesh would be composed of two co-planar triangles, one for the front, and one for the back. While in this thesis such garments have not been yet investigated, no fundamental difficulties in processing them can be foreseen.

Another area of future work is the replacement of the boundary tracking method by a fully automatic method. It may be possible to use 2D image segmentation results to improve the reliability of the 3D anchor positions even in cases where the multiview stereo algorithm has not produced clean boundaries.

3.9. Conclusion

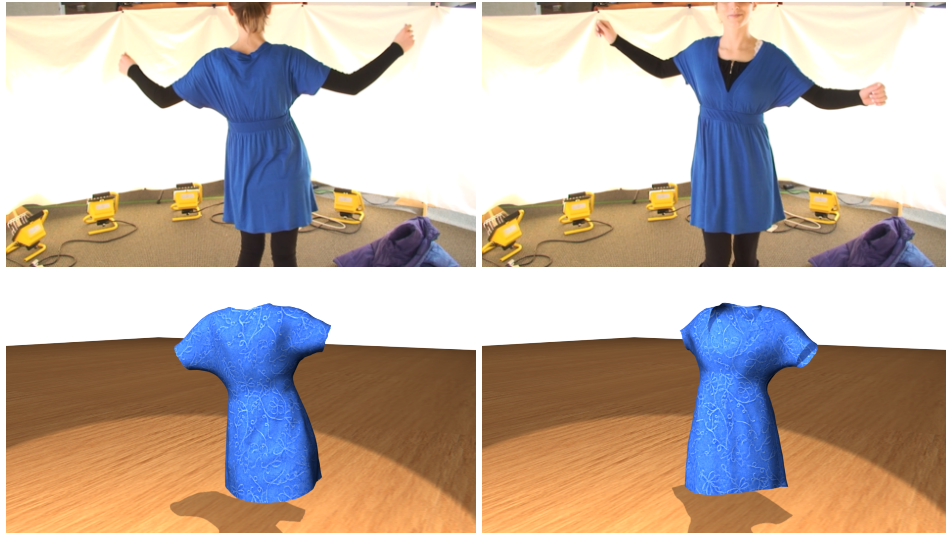


Figure 3.14: Capture results for five frames of a blue dress. Input images (top) and reconstructed geometry (bottom)

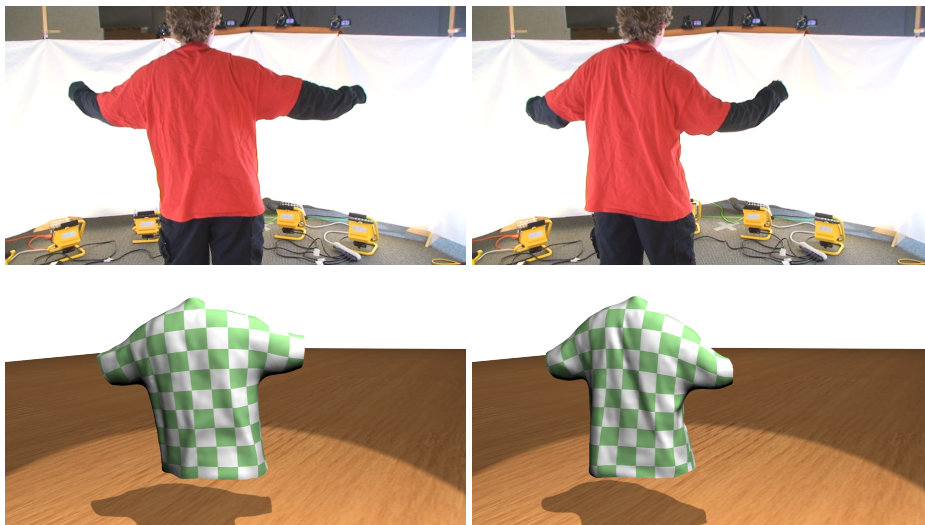


Figure 3.15: Capture results for two frames of a large T-shirt. Input images (top) and reconstructed geometry (bottom)

3.9. Conclusion

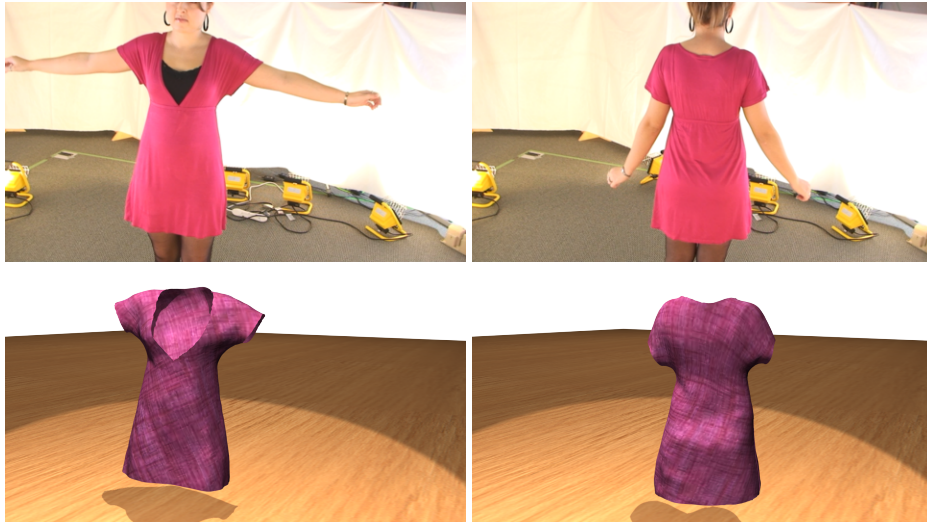


Figure 3.16: Capture results for two frames of a pink dress. Input images (top) and reconstructed geometry (bottom)



Figure 3.17: Capture result for a long-sleeve nylon-shell down jacket

Bibliography

- [ACP03] B. Allen, B. Curless, and Z. Popović. The space of human body shapes: reconstruction and parameterization from range scans. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 587–594, 2003.
- [ASK⁺05] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. SCAPE: shape completion and animation of people. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 408–416, 2005.
- [BBA⁺07] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale capture of facial geometry and motion. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 33, 2007.
- [BBH08] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. CVPR*, 2008.
- [BRS05] Tamy Boubekeur, Patrick Reuter, and Christophe Schlick. Visualization of point-based surfaces with locally reconstructed subdivision surfaces. *Shape Modeling and Applications, International Conference on*, 0:23–32, 2005.
- [BTH⁺03] K. Bhat, C. Twigg, J. Hodgins, P. Khosla, Z. Popovic, and S. Seitz.

- Estimating cloth simulation parameters from video. In *Proc. SCA*, pages 37–51, 2003.
- [FB81] M. Fischler and R. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.
- [Flo03] M. Floater. Mean value coordinates. *Comput. Aided Geom. Des.*, 20(1):19–27, 2003.
- [FS05] M. Fiala and C. Shu. Fully automatic camera calibration using self-identifying calibration targets. Technical Report NRC-48306 ERB-1130, National Research Council of Canada, 2005.
- [GHF⁺07] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient simulation of inextensible cloth. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 49, 2007.
- [GKB03] I. Guskov, S. Klibanov, and B. Bryant. Trackable surfaces. In *Proc. SCA*, pages 251–257, 2003.
- [HAR⁺06] N. Hasler, M. Asbach, B. Rosenhahn, J-R Ohm, and H-P Seidel. Physically based tracking of cloth. In *Proc. Workshop on Vision, Modeling, and Visualization*, pages 49–56, 2006.
- [HVB⁺07] C. Hernandez, G. Vogiatzis, G. J. Brostow, B. Stenger, and R. Cipolla. Non-rigid photometric stereo with colored lights. In *Proc. ICCV*, 2007.
- [KS04] V. Kraevoy and A. Sheffer. Cross-parameterization and compatible remeshing of 3d models. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 861–869, 2004.

- [KS05] V. Kraevoy and A. Sheffer. Template-based mesh completion. In *Proc. SGP*, page 13, 2005.
- [MFO⁺07] N. Mitra, S. Flöry, M. Ovsjanikov, N. Gelfand, L. Guibas, and H. Pottmann. Dynamic geometry registration. In *Proc. SGP*, pages 173–182, 2007.
- [MI07] Y. Mori and T. Igarashi. Plushie: an interactive design system for plush toys. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 45, 2007.
- [mvi] mview. <http://vision.middlebury.edu/mview/>.
- [PH03] D. Pritchard and W. Heidrich. Cloth motion capture. In *Proc. Eurographics*, pages 263–271, September 2003.
- [PSS01] E. Praun, W. Sweldens, and P. Schröder. Consistent mesh parameterizations. In *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 179–184, 2001.
- [RKP⁺07] B. Rosenhahn, U. Kersting, K. Powell, R. Klette, G. Klette, and H-P Seidel. A system for articulated tracking incorporating a clothing model. *Machine Vision and Applications*, 18(1):25–40, 2007.
- [SAPH04] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 870–877, 2004.
- [SCD⁺06] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR*, pages 519–528, 2006.

- [She96] J. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, 1996.
- [SK04] A. Sheffer and V. Kraevoy. Pyramid coordinates for morphing and deformation. In *Proc. 3DPVT*, pages 68–75, 2004.
- [SM04] Volker Scholz and Marcus A. Magnor. Cloth motion from optical flow. In *Proc. Vision, Modeling and Visualization 2004*, pages 117–124, November 2004.
- [SSK⁺05] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor. Garment motion capture using color-coded patterns. In *Proc. Eurographics*, pages 439–448, 2005.
- [TM98] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *Proc. ICCV*, page 839, 1998.
- [WCF07] R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 34, 2007.
- [WJH⁺07] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. SGP*, pages 49–58, 2007.
- [ZSCS04] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Space-time faces: high resolution capture for modeling and animation. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 548–558, 2004.

Chapter 4

Wrinkling Captured Garments Using Space-Time Data-Driven Deformation

4.1 Introduction



Figure 4.1: Reintroducing folds into captured garments: (left) input video-frames, (center) typical capture result [BPS⁺08], and (right) t-shirt wrinkled by this method.

⁵ Capturing the geometry of moving garments provides a data-driven alter-

⁵A version of this chapter has been published. Popa, T. and Zhou, Q. and Bradley, D. and Kraevoy, V. and Fu, H. and Sheffer, A. and Heidrich, W. (2009) Computer Graphics Forum (Proc. Eurographics)

native to cloth simulation in much the same way as motion capture provides an alternative to character animation. Recent garment capture techniques [BPS⁺08, VBMP08, dAST⁺08] are based on multi-view video capture and are quite successful at capturing the low-frequency garment shape and motion over time, and at establishing consistent correspondences across frames. However, as demonstrated in Figure 4.1 (center), they often fail to capture the numerous high-frequency folds characteristic of garments, reducing the realism of the reconstructed dynamic models. Folds tend to be very shallow, making them hard to capture and separate from noise. However, due to large normal variation they are highly noticeable. Thus, garments that lack folds look unnatural.

Marker-based capture methods such as [WCF07] capture folds more accurately, but are restricted to custom made garments. Active lighting approaches, e.g. [HVB⁺07], do reasonably well in capturing fine details in single camera setups. However, in practice, due to interference of the light sources, such methods do not generalize well to 360° acquisition of moving targets. The use of active lighting also adds considerably to the complexity, and in some cases, cost, of the setup.

In this work, a simple, yet effective, method is proposed for reintroducing folds into models captured by any multi-view video technique using data-driven, dynamic wrinkling (Figure 4.1 (right)). To generate believable folds the algorithm leverages unique geometric properties of garments, most notably the fact that most garments exhibit very low stretch under normal wear. This property imposes very strong constraints on the shape of garment folds, leading to the characteristic fold shapes we are all familiar with. To generate folds consistent with the captured garment motion this method takes advantage of the available video footage and use the video to guide fold modeling. This method first analyzes the video capture to estimate the position and shape of folds, using their distinguishing shape charac-

teristics (Figure 4.2 (center)). It then introduces those folds into the reconstructed models, using stretch-minimizing deformation, which naturally produces believable fold shapes (Figure 4.2 (right)). A novel space-time deformation framework is used to generate folds which are consistent across time.

Although the goal is to place folds in places where they are observed in real video footage, there is no guarantee of reproducing the exact shape of each fold. That is, the goal is to generate believable wrinkling with an overall appearance similar to that of the original garment, instead of attempting to “measure” the true fold geometry.

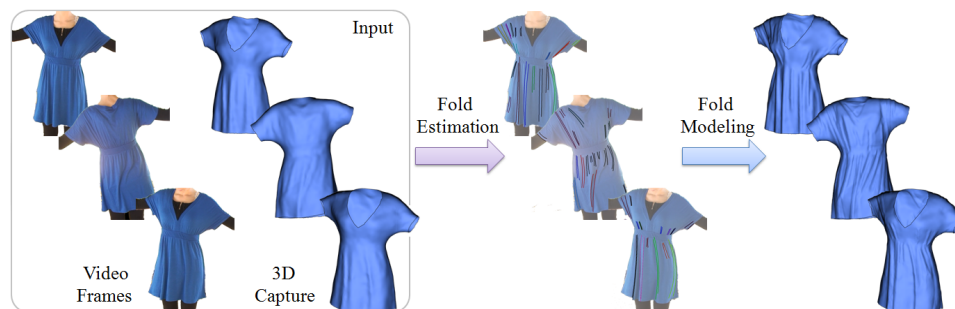


Figure 4.2: Algorithm overview. From left to right: input video, captured smooth geometry, estimated folds, and wrinkled models generated by the space-time deformation approach

4.2 Previous Work

Cloth & Garment Capture: In recent years several methods emerged specifically for capturing garment motion [BPS⁺08, WCF07, SSK⁺05, PH03], as well as for capture of moving humans dressed in loose clothing [VBMP08, dAST⁺08]. Marker-based methods such as the work by White et al. [WCF07] use a custom set of color markers printed on the surface of the cloth, and then use the markers combined with the assumption of low fabric stretch to detect and trace high-resolution

folds. As this approach is restricted to custom made garments, a markerless alternative was recently proposed by Bradley et al. [BPS⁺08]. The method successfully captures off-the-shelf garments correctly reconstructing the low-frequency shape (see Figure 4.1, center), but fails to capture the high-frequency folds.

The template-based methods of Vlasic et al. [VBMP08] and de Aguiar et al. [dAST⁺08] focus on capturing the overall performance of actors rather than concentrating specifically on garments. While these methods also capture an approximation of the clothing the actor is wearing, the geometric details present in the reconstructions of these garments tend to be copied from a high-resolution template scan rather than represent actual high-frequency per-frame details.

Active lighting approaches such as photometric stereo are better suited for capturing fine geometric details. However, such approaches generalize poorly to the full 360° acquisition of deformable models. For example, Hernandez et al. [HVB⁺07] use red, green, and blue lights to estimate surface normals with a photometric stereo approach, making use of the three different color channels in a color camera. Generalizing this approach to a full ring of cameras and light would require at least a dozen light sources that emit light at different frequencies, as well as specialized multi-spectral camera hardware that can distinguish between those light sources. Ahmed et al. [ATD⁺08] use calibrated lighting and multi-view video to capture normal fields, and augment garment geometry templates to include wrinkles and folds. Calibrating the light sources adds complexity to the system. In addition, this technique requires knowledge of the reflectance model at each point on the surface.

Modeling Folds and Wrinkles: The presence of folds is important for garment realism and several approaches exist for generating folds on simulated virtual garments. To model the folds, most methods use a physically-based simulation ap-

proach (see [BMF03] and the references therein), which is time consuming, and requires accurate information about the motion of the character wearing the garment as well as numerous physical parameters for the simulation.

Several authors propose alternative, more efficient, geometric approaches for modeling folds, which work well on tight, high-stretch garments [HBVMT99, CGW⁺07, CMT05]. Hadap et al. [HBVMT99] utilize a user defined fold pattern to generate a bump map representation as a solution. Cutler et al. [CGW⁺07] and Cordier et al. [CMT05] use cloth examples with precomputed, simulated folds to predict new fold geometry. Decaudin et al. [DJW⁺06] suggest a procedural approach for adding folds to loose garments, based on an analysis of likely fold shapes. Their results tend to look very regular and thus somewhat artificial. Like the physics-based methods above, these geometric approaches require accurate information about the proportions and motion of the garment wearer, which might often be unavailable in a capture setup.

Garment folds share some shape similarities with skin wrinkles. To capture dynamic facial wrinkles Bickel et al. [BBA⁺07] define a small number of explicit locations on the surface where wrinkles can occur using face paint and track those over time. This approach cannot be applied for tracking garment folds since folds can occur anywhere on the surface. Since no explicit tracking is possible in the inputs, a space-time deformation is used to ensure temporal coherence.

Adding High-Frequency Motion: Recently, there has been an emergence of several methods [KA08, SZT⁺08, PH08, SB08] that add realism to animation by introducing secondary high-frequency motion, which is not fully physically based, but which increases the realism of the models. The method presented in this chapter can be seen as a continuation of this trend, increasing the realism of garment capture by introducing believable high-frequency folds, without the additional costs of

a full-blown physical simulation or a sophisticated capture setup.

4.3 Background and Overview

To generate realistic looking folds, it is important to take advantage of the unique properties of garments that distinguish them from general deformable geometries. Most fabrics can undergo only very limited stretch [GHF⁺07]. Thus, from a geometry point of view, garments can be seen as piecewise quasi-developable surfaces, where the pieces correspond to the individual garment panels cut out of fabric. This observation imposes very strong constraints on the shapes that garments can form, and specifically on the shape of garment folds. The reader can consult [DJW⁺06] for an overview of the common types of garment folds: sine wave, diamond, and twisted diamond (see Figure 4.3).

Based on the fold classification provided by Decaudin et al. [DJW⁺06], and on our own observations, we note that most folds are formed from generalized cylinders, with roughly sinusoidal cross-sections. This observation applies not only to the regular sine wave folds (Figure 4.3 (left)), but to the diamond/twisted diamond folds as well, as those can be separated into an interior valley region which is nearly flat, and the surrounding fold ridges formed by a union of generalized cylinders (Figure 4.3 (center) and (right)). These generalized cylinders tend to be fairly straight, with low-curvature axes and nearly constant cross-sections. These observations are used to estimate the fold shape and location based on the input video (Figure 4.2 (center)) and to model the folds based on these estimates (Figure 4.2 (right)).

4.3.1 Video Based Fold Estimation

Based on the above discussion of fold shape, as well as some radiometric assumptions, folds are expected to show up in video frames as very specific edge structures robustly identifiable using standard edge detection filters. Specifically, the main assumption is that the fabric of the garment is approximately Lambertian, and that the illumination is diffuse, i.e. uniform across all incident directions. In this chapter I only consider garments without strong texture to avoid ambiguities between shading patterns and texture features.



Figure 4.3: Types of garment folds: sine wave (left), diamond (center), and twisted diamond (right). The diamond and the twisted diamond are formed by four roughly cylindrical ridges each. Top: photos of real folds, bottom: modeling similar fold geometry

Under the assumptions outlined above, the ridges of fold structures show up as bright areas, whereas the valleys are dark, since they receive light from a narrower solid angle. This model for shading of surface features is known as “dark-equals-deep” (e.g. [LB00, GWM⁺08]). As a result of this photometric model, one expects

4.3. Background and Overview

to observe image edges roughly corresponding to iso-lines (i.e. elevation contours) on the fold geometry. Moreover, due to the specific structure of garment folds, as outlined above, each fold component (a generalized cylinder) is observed as a pair of roughly parallel edges, flanking a ridge (bright) or a valley (dark) area (see Figures 4.2 (center) and 4.4). The edges that fit these criteria are called *fold edges*. Section 4.4 details the fold-edge extraction process.

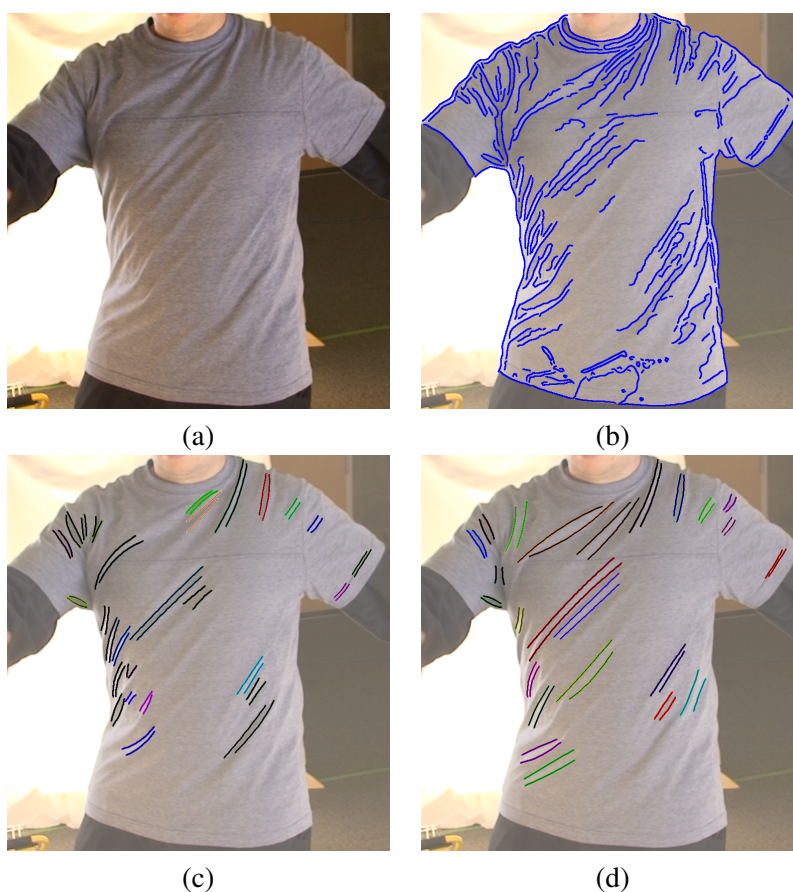


Figure 4.4: Fold edge extraction: input image (a), raw edges (b), and paired fold edges (c) extracted from the image at a single resolution (edges not projected to the garment are discarded). (d) Fold edges after combining extraction results from several image resolutions and filter sizes

Note that these properties are reasonably robust under moderate deviations

from the idealized radiometric model used (e.g. Figure 4.3 (top)). The assumption of an approximately Lambertian material holds well for many common fabrics including cotton and wool, although it may be violated by some synthetic fibers. Fortunately, the “dark-equals-deep” model is only violated for very shiny materials that are very rare in garments. Deviations from the ideal diffuse illumination scenario introduce a directional component that causes the brightest and darkest regions to be somewhat offset from the true ridges and valleys, respectively. In this situation, this approach will produce a fold that is slightly shifted from the true location, but has the correct size and orientation. As such, the overall appearance of the final garment should be very similar to the original. Another note is that uniform diffuse illumination is the preferred lighting setup for movie shots and many capture setups, as it eliminates strong shadows, which would hinder reconstruction (e.g. [BPS⁺08]). So deviations from the idealized setting are expected to be minor.

The pairing requirement imposes a very strong constraint on the detection mechanism, which lets us effectively overcome the simplicity of the photometric model. Furthermore, the edge pairing requirement also helps eliminate detected edges that correspond to texture on the garment rather than to folds. As a result, there are few false positives in the processed data, where non-folds were classified as folds, or where the edge lines did not faithfully reproduce the fold geometry. Even in the case of a false positive, the resulting wrinkled models appear realistic as the placed folds satisfy the typical shape characteristics.

Leveraging Temporal Information: While folds typically persist across a sequence of frames, they can be more recognizable in some than in others. Since edge detection is by its nature discrete, it can potentially miss folds when they are less noticeable. This method overcomes this problem, by utilizing the observation that garment motion is smooth across time. The space-time deformation method,

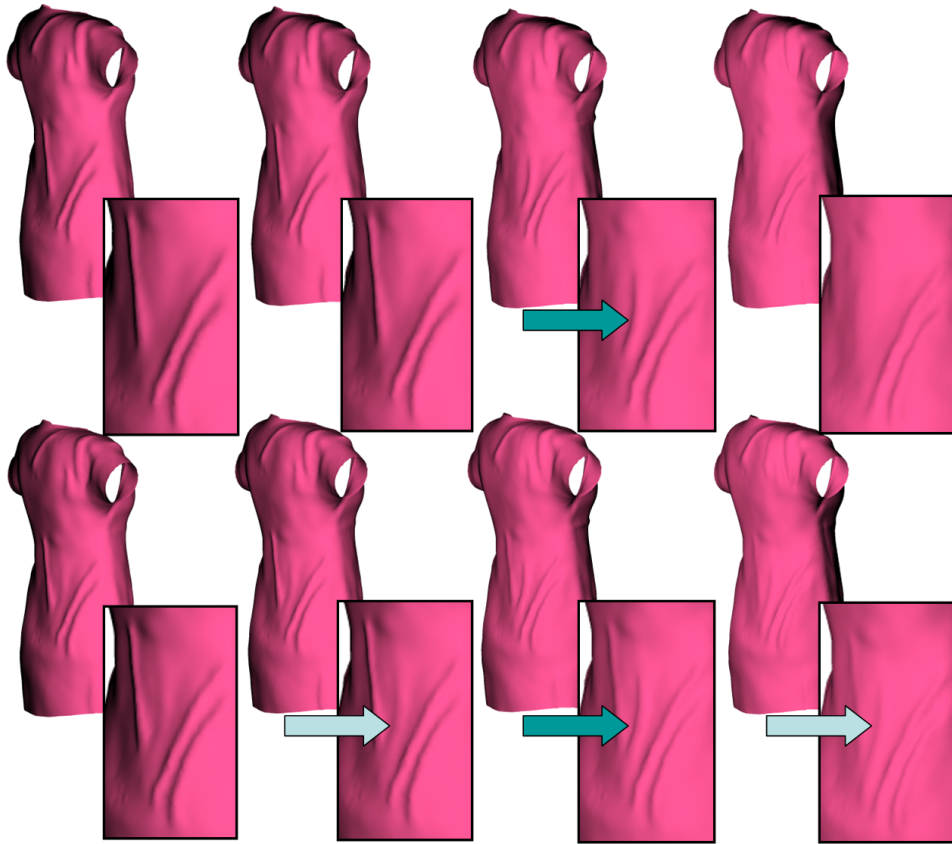


Figure 4.5: In a frame-by-frame deformation setup (top) one fold pops up in a single frame (see arrow) while another vanishes abruptly. Using space-time deformation (bottom) folds both form and disappear gradually

described next, explicitly enforces motion continuity. Thus, if a fold is detected in some, but not all frames of a sequence, the deformation effectively completes it in the frames where it is missing (Figure 4.5). This space-time leveraging allows us to use a conservative set of parameters that avoid false positives when extracting the fold edges.

4.3.2 Fold Modeling

The folds detected in the video footage are incorporated into the geometry of the garments using a novel dynamic deformation setup. Recall that fold edge pairs may correspond to parts of more complex fold structures. In order to capture such structures, a global deformation of a garment shape is performed using all detected pairs of fold edges at once, rather than adding them one by one.

To generate the folds the surface is bent along the edges, using rotation driven deformation. A rotation-driven approach is preferred over the more common positional controls, since folds are characterized predominantly by changes in surface normals, and such changes are more naturally represented by rotations. Moreover, the rotation-based approach removes the need to detect actual depth and shape information from images, which could be quite challenging, as mentioned in the discussion of related work. Instead, rotations are used as inputs to a stretch-minimizing deformation, which computes the positions of the mesh vertices in a way that conforms with the low-stretch property of garments, resulting in realistic looking folds.

Most recent deformation approaches minimize changes in mean curvature rather than stretch and are thus not suitable for fold modeling (e.g. [SLCO⁺04, LSLCO05]). Existing methods for stretch-minimizing deformation, such as the work by Kilian et al.[KMP07] are non-linear and computationally demanding. Converting them into a setup that solves for deformation across multiple frames at once could be prohibitively expensive.

Hence, to generate realistic dynamic wrinkles a new space-time quasi-isometric deformation method is introduced(Section 4.5). Key to this method is the observation that quasi-isometric deformation implies that as the mesh deforms the triangles shape is preserved. Therefore the goal is to search for as-rigid-as-possible

per-triangle transformations and shared vertex positions that satisfy these transformations. An iterative solution procedure is used which amounts to solving a number of very sparse linear systems, allowing us to efficiently wrinkle very large meshes (>100K triangles) simultaneously across numerous frames. The high mesh resolution is necessary to capture the fine fold geometry.

The final result of the algorithm is a sequence of realistic looking wrinkled meshes with the folds consistent across time (Figure 4.2 (right)).

4.4 Video-Based Fold Edge Extraction

⁶To extract likely folds at each time step, the best front and back views of the garment are selected. Folds are then selected from each of the two still images separately using the photometric model described in Section 4.3.1, by first detecting edges present in the images and then filtering out edges that do not satisfy the fold edge criteria (Figure 4.4).

View Selection: For every frame in the given multiview video input, the best front and back views are automatically selected. As a preprocess, the user selects two mesh vertices, one on the front and one on the back, in a single frame. The method then tracks these vertices across all frames using the consistent mapping between the frame geometries provided by the original capture process. It computes per-frame stable normals at the vertices using a large (eight rings for the examples shown) neighborhood and selects the best, most orthogonal views, based on the dot product of the normals with the camera view vectors.

⁶The work described in this section is not part of this thesis, it has been added for completeness only.

Edge Extraction: The method uses the Canny edge detector [Can86] to extract raw image edges. Recall that fold components are expected to have low curvature in the direction of the fold axis. Since fold edges follow contours of the fold, they should inherit this property. Hence a fold edge can be approximated by a low degree polynomial with bounded curvature. Quadrics are used to fit each raw edge, recursively splitting the edges if the fitting error or the curvature of the quadric along the edge are too high.

Edge Orientation: Fold edges are expected to represent fold contours, and thus have a darker, i.e. deeper side (recall that the “dark-equals-deep” model is used). Thus, after extracting the smooth edges, they are oriented based on the image gradient along the edge. By convention the orientation is selected such that the darker side of the image is to the left of the edge. Edges with no clear orientation are discarded. Edges which exhibit large difference in hue across the edge are discarded as those typically capture texture rather than shape.

Merging: Edge detection often tends to break continuous fold contours into multiple disjoint edges. The continuous contours are recovered by merging edges if they are oriented consistently, their end vertices are close to one another, and the combined edge can still be fitted by a low curvature quadric.

Pairing: The most significant step of the algorithm is the pairing, which detects parallel edges of approximately the same length, with opposite orientation and a relatively short distance between them. (Figure 4.4(c)). As noted earlier, garment folds are formed by a union of generalized cylinders, thus opposite contours on these cylinders are expected to be roughly parallel and relatively narrow. Therefore, pairs of edges representing such contours should satisfy the criteria of

4.5. Space-Time Deformation

parallelism, and opposite orientation and closeness. All edges that have no pairing are discarded, as those are not likely to represent or form realistic folds. The parallelism test between two edges is performed by comparing tangents between closest points along the quadric curves. Since folds are expected to be relatively narrow, the average and maximal distance between the paired edges is constrained.

The pairing process must address potential pairing ambiguities, as some edges may have multiple pairing choices. To resolve those, the pairings are prioritized by their likelihood of being real and significant folds. On real garments there are more ridge than valley folds. This observation leads us to prefer ridge pairs, namely ones where the area between the edges is brighter, i.e. the edges are to the right of one another. Pairs with longer edges are also given priority, as those indicate more influential folds.

Multi-Resolution Processing: To maximize the number of folds detected, the detection and filtering steps are done on several image resolutions and with different Gaussian filter sizes in the Canny edge detector. The results are then combined together, creating a union of edge pairs (Figure 4.4(d)). When pairs from different layers overlap, the best one is selected using the same criteria as for pairing. The resulting set of pairs represents believable folds in the input images, and is used as input for wrinkling the garment surface.

4.5 Space-Time Deformation

This section discusses the space-time deformation method for smoothly wrinkling the garment surface over time, using the folds estimated from the video sequence.



Figure 4.6: Deformation: (left) anchor triangles (fold edges projected to input garment); (center) vertex positioning result; (right) garment after stretch reduction. The coloring (center, right) shows per-triangle stretch. The iterations reduce the maximal per-triangle stretch from 8% to 4%

4.5.1 Control Mechanism

The deformation is controlled by the fold edges extracted from the individual frames, which indicate where the sides of the newly formed folds should be. The surface is wrinkled by rotating the corresponding, *anchor* triangles counterclockwise around the edges. This rotation is consistent with the assumption that the left, darker, side of each fold is expected to be deeper than the right one, i.e. closer to the body of the wearer. To locate the anchor triangles the computed fold edges are projected onto the corresponding garment frames, using the available camera calibration (Figure 4.6 (left)). Fold edge pairs where one of the edges projects outside the garment or onto its silhouette are discarded. Several models for setting the rotation angle have been tested, including using the gradient along the edge as a measure of the amount of rotation, but found no correlation between the two. In empirical testing the best results were achieved when setting the rotation angle to be inversely proportional to the width of the fold edge pairs with the angle varying from 60° for narrow folds to 30° for wide ones.

4.5.2 Iterative Space-Time Deformation

To obtain smooth garment motion across time, the method solves for the new shape of the mesh across multiple time frames. Quasi-isometric deformation implies preserving the shape of the triangles during deformation, while allowing the dihedral angles between them to change. Solving simultaneously for as-rigid-as-possible per-triangle transformations and shared vertex positions that satisfy these transformations, leads to a non-linear formulation, which would be prohibitive to optimize in space-time. Therefore, similar to [LSLCO05, SP04], the formulation decouples the rotation computation from the vertex positioning. First, optimal triangle rotations for all triangles are computed based on the input rotations across all frames. Then the method computes vertex positions consistent with the rotations, obtaining wrinkled meshes that satisfy most of the requirements, but which can exhibit non-negligible stretch near the newly formed folds (Figure 4.6 (center)). Therefore the rotations and the positions are iteratively updated to reduce the stretch to an acceptable level (Figure 4.6 (right)). The stretch reduction can be accomplished using rotations computed per-triangle, without a need for solving yet another global system to compute them. In other words, the per-triangle update combined with the vertex repositioning satisfies the temporal and spatial requirements imposed on the folds, without explicitly enforcing them. The following sections describe the three steps in more detail.

4.5.3 Space-Time Rotation

Given the specified *anchor triangles* and the corresponding rotation angles for each time frame, the method solves for the rotations of the mesh triangles in multiple frames simultaneously. Recall that since this is a space-time captured mesh, all frames have the same connectivity, thus it is possible to explicitly compute the

4.5. Space-Time Deformation

rotations for all frames at once. The space-time solution is key to making the rotations and hence the final deformation consistent across time, enforcing smooth garment motion (Figure 4.5).

For the folds to look realistic, in addition to smooth change across time, rotations have to satisfy two additional requirements. First, the garments are expected to be smooth, thus adjacent triangles should rotate in roughly the same way. Second, the folds are expected to be fairly local, and thus triangles spatially or temporally distant from the anchors should rotate very little, if at all.

To propagate the rotations across the meshes the per-triangle rotations are expressed as convex combinations of anchor rotations and the identity rotation. The inclusion of the identity mitigates the anchor influence away from the folds. The challenge is to compute the per-triangle, convex combination blending weights w_i^t , where i goes over the mesh triangles and t goes over the time steps. Each weight is a vector with entries corresponding to the anchors. The sum of the weights is less or equal to one, with the weight assigned to the identity rotation completing the sum to be exactly one, thus ensuring the convex combination property. The weights are computed by minimizing the following quadratic functional,

$$\begin{aligned} \arg \min_w \quad & c_s \sum_{t,e=(i,j)} \|w_i^t - w_j^t\|^2 + c_t \sum_{i,t} \|w_i^t - w_i^{t-1}\|^2 + \\ & c_i \sum_{(i,t) \notin A} \|w_i^t\|^2 + c_a \sum_{(i,t) \in A} \|w_i^t - I_i^t\|^2, \end{aligned} \tag{4.1}$$

where $e = (i, j)$ are the edges shared by triangles i and j , A is the set of anchor triangles, and I_i^t is a vector with one in the entry that corresponds to the anchor (i, t) and zero everywhere else. The first two terms of the functional ensure rotation continuity across space and time. The third term provides for rotation mitigation

away from the anchors, and the last term constrains the anchor rotations. The coefficients which control the influence of each individual term were set to $c_s = 0.9, c_t = 0.5, c_i = 0.1, c_a = 10$ in all the examples. The input meshes are fairly uniform, thus the formulation does not weigh triangles or edges differently based on their area or length. Having uniform weights improves the conditioning of the linear system solved, and thus speeds up convergence. To perform the actual blending, given the weights, I extend the method presented in chapter 2 [PJS06] , and use the transformation algebra presented by Alexa [Ale02].

Numerical Solution: Minimizing the above functional amounts to solving a linear system with multiple right hand sides, corresponding to the individual entries in the weight vectors. The matrix is positive definite and very sparse (approximately six non-zero entries per row), thus I used a conjugate gradient solver to compute the solutions. Since very accurate weights are not needed, I used a fairly lax convergence tolerance of 10^{-3} , resulting in the solver converging in under ten iterations. The temporal influence of anchors reduces significantly over time, thus rather than solving for all time steps at once a staggered approach is used, significantly reducing both memory footprint and runtime. The method solves for overlapping sequences of ten frames at a time, with a time shift of seven frames between them and blend the rotations obtained in the three overlapping frames, to maintain rotation smoothness. The rotations generated with this staggered approach are practically identical to those generated using a global solution.

4.5.4 Vertex Positioning

Given the new rotations, the vertex positions are computed by solving a least-squares system, which aims to preserve the computed rotations as the per-triangle transformation gradients. The vertices of a given triangle are labeled v_1, v_2 and v_3 .

A virtual vertex v_4 is added by offsetting v_1 by the triangle normal thus defining a local triangle frame $V_i^t = (v_4 - v_1, v_4 - v_2, v_4 - v_3)$. To obtain the vertex positions, the following quadratic functional is minimized [SP04],

$$\arg \min_{\tilde{v}} \sum_{i,t} \|\tilde{V}_i^t (V_i^t)^{-1} - R_i^t\|_F^2, \quad (4.2)$$

where \tilde{v} are the new positions of the vertices, V_i^t and \tilde{V}_i^t the local frames before and after the deformation, and R_i^t are the previously calculated triangle rotations. Since this objective functional has no temporal component it can be minimized independently for each frame. A direct solver (SuperLU [DEG⁺99]) is used to solve the corresponding linear system, fixing one of the mesh vertices to remove the redundant translational degrees of freedom.

4.5.5 Stretch Reduction

After the positioning step, the deformed per-frame meshes satisfy the anchor rotations (in a least-squares sense) and are consistent across time. The overall stretch of the output meshes, compared to the inputs, is typically very low, as the rotation step bends the surface along feasible fold-lines while the vertex repositioning step attempts to preserve the rotation-only gradients. A typical example shown in Figure 4.6 (center) has L_2 stretch [SSGH01] of 1.00036 (the optimum is one). However near the folds the outputs often exhibit local stretch which is higher than what many fabrics can tolerate, and one that can interfere with the visual consistency of the folds across time.

In the two step wrinkling procedure (eqs.4.1 and 4.2), the stretch in the mesh is introduced by the latter step that reconciles the position of vertices across adjacent triangles. To reduce stretch to an acceptable level, the algorithm searches for new

rotation-only gradients which are more consistent across adjacent triangles, and thus lead to less triangle stretch during vertex repositioning. One way to obtain new rotations would be to formulate and solve a new space-time rotation optimization problem with the set of spatial and temporal requirements listed earlier. However, in practice such a global solution is unnecessary.

The rotations implied by the difference between the triangle normals before the deformation n and after \tilde{n} are close enough to the input rotations to maintain the desired properties. At the same time, these rotations are more feasible to satisfy in Equation 4.2, leading to a smaller minimum and thus less stretch. We therefore simply use them as the new per-triangle rotations R_i^t (rotation axis is $n \times \tilde{n}$ and the angle is $\arccos(n \cdot \tilde{n})$) and reapply the vertex positioning procedure. A possible alternative is to compute R_i^t as the rotational component in polar decomposition of $\tilde{V}_i^t (V_i^t)^{-1}$. This is however computationally more expensive, and they produced nearly identical results. One or two iterations of stretch reduction are generally sufficient to reduce the maximal per-triangle stretch to under 5%, at which point it becomes visually unnoticeable. In the example in Figure 4.6 (right) the maximal per-triangle stretch in the final model is 4%, and the L_2 stretch is 1.00017.

4.6 Results

The method was tested on a variety of garments generated by three state-of-the-art capture techniques [BPS⁺08, VBMP08, dAST⁺08]. Figures 4.1 through 4.9 demonstrate the method’s behavior on five diverse garment models captured by Bradley et al. [BPS⁺08] using a sixteen camera setup : two T-shirts, two dresses and a jacket. In all the examples the method adds believable folds to the models. As demonstrated in the attached video, the generated folds change smoothly across time, emulating realistic fold behavior.

The method was also tested on data from Vlastic et al. [VBMP08] and de Aguiar et al. [dAST⁺08] (Figure 4.11). Since de Aguiar et al. preserve fine details from the template throughout the sequence (Figure 4.11 (left,middle)) we smooth those out prior to applying the dynamic wrinkling algorithm. Notice how the new folds vary across time, while the original ones remain fixed (see video for more examples). The video footage used by these two methods is of lower resolution than that of Bradley et al., and therefore the number of folds visible in the footage and captured by the method is lower. Since both methods heavily rely on silhouettes, they pay less attention to lighting, thus providing conditions that are far from optimal for the approach. Despite these conditions, the method manages to extract sufficient fold information to generate believable dynamic folds. Both setups use only eight cameras, raising a concern of temporal consistency when the fold extraction switches between different camera views. Nevertheless, the technique reconstructs realistic folds even in these sub-optimal conditions with no noticeable artifacts. In situations where the camera views are too far apart, temporal inconsistencies can occur.

Runtimes: The most time consuming step of the method is the space-time rotation computation. For a 100K triangle mesh there are typically on the order of 2000 anchors per-frame, so computing the weights for a block of 10 frames requires solving a linear system of approximately a million variables 2000 times for each frame. This takes about one minute per frame on a 27 Intel Xeon 3GHz CPU cluster. The rest of the computation is done on a single CPU and takes about two minutes per frame which are split roughly equally between edge detection and vertex positioning (including the stretch reduction).

4.7. Conclusions

Parameters: The estimation of fold edges requires a few parameters that depend on the garment fabric and cut. The parameters are maximal distance and angle between paired fold edges, and Canny edge detector radii. These are set by the user once per garment. In all the examples the angle was set to 45° , the other parameters are listed in Table 4.1. In the future it might be possible to learn those from fabric parameters or example inputs.

	Pink Dress	Blue Dress	T-Shirts	Red Jacket	Dancer and capoeira kick
Maximal distance	∞	100	100	100	12
Canny radii	1.5, 2.5, 3.5	1, 2, 3	1, 2, 3	0.5, 2, 4	0.5, 1, 1.5

Table 4.1: Fold edge parameters

4.7 Conclusions

This is the first method for augmenting the realism of captured garments by introducing believable dynamic wrinkles. As demonstrated, the method works robustly on numerous input examples, generated by different capture methodologies.

This algorithm does not aim at exact reproduction of fold shapes observed in the video footage, but rather to discover a large enough number of folds to enhance the appearance of the garments. The space-time fold modeling technique has a built-in safe-guard against false negatives (folds that are not discovered, but exist in the data): to miss a fold is to fail to detect it not in one, but in ten consecutive frames which happens very infrequently. In the rare cases when a false positive fold occurs (a fold that is discovered but it does not exist in the real data) the resulting wrinkled models appear realistic as the placed fold satisfies the typical shape characteristics.

It might be possible to increase fold capture accuracy by combining the deformation approach with a more sophisticated photometric model using information

4.7. Conclusions

from multiple cameras simultaneously.

Limitations: This method operates directly on meshes, and thus the size of the folds it can capture is bounded from below by the mesh resolution. Since the results were already generated using meshes with around 100K triangles, increasing the resolution further is not really feasible. One alternative could be to combine geometric deformation with normal space processing, developing a mixed geometry and bump map modeling approach.

Another limitation is derived from reliance of the fold detection algorithm on edge detection, affecting the method's usability when garments have patterned textures. Similarly, garments with pockets and very visible seams might produce artifacts as those are likely to be detected as folds, for instance in the case of the jacket shown in Figure 4.9, the front zipper is detected as fold. But the resulting "artifact" enhances in this case the result.

4.7. Conclusions



Figure 4.7: Wrinkling dress captured by [BPS⁺08], (top) video input, (middle) captured geometry, (bottom) results

4.7. Conclusions



Figure 4.8: Wrinkling dress captured by [BPS⁺08], (top) video input, (middle) captured geometry, (bottom) results

4.7. Conclusions



Figure 4.9: Wrinkling a t-shirt captured by [BPS⁺08], (top) video input, (middle) captured geometry, (bottom) results



Figure 4.10: Wrinkling a jacket captured by [BPS⁺08], (top) video input, (middle) captured geometry, (bottom) results



Figure 4.11: Wrinkling outfits captured by [dAST⁺08]: (top) video input, (middle) captured geometry, (bottom) results. The captured geometry contains template folds which remain constant throughout the sequence. In contrast, the folds in the outputs change dynamically following the character's motion

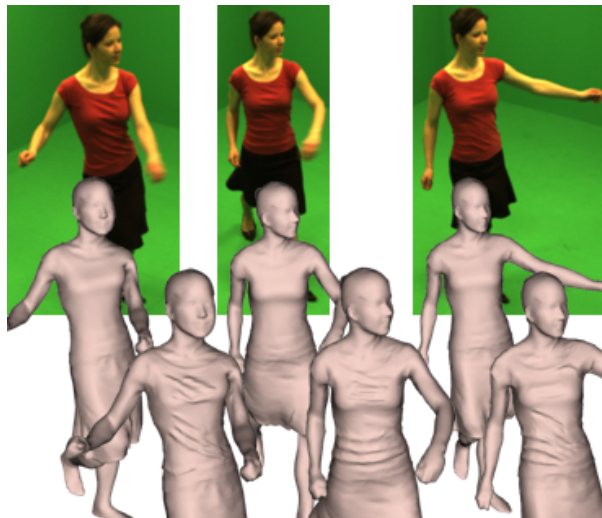


Figure 4.12: Wrinkling outfits captured by [VBMP08]: (top) video input, (middle) captured geometry, (bottom) results.

Bibliography

- [Ale02] Marc Alexa. Linear combination of transformations. *ACM Trans. Graph.*, 21(3):380–387, 2002.
- [ATD⁺08] Naveed Ahmed, Christian Theobalt, Petar Dobrev, Hans-Peter Seidel, and Sebastian Thrun. Robust fusion of dynamic shape and normal capture for high-quality reconstruction of time-varying geometry. In *CVPR 2008*, 2008.
- [BBA⁺07] B. Bickel, M. Botsch, R. Angst, W. Matusik, M. Otaduy, H. Pfister, and M. Gross. Multi-scale capture of facial geometry and motion. *ACM Trans. Graph.*, 26(3):33, 2007.
- [BMF03] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA '03*, pages 28–36, 2003.
- [BPS⁺08] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Trans. Graph.*, 27(3):99, 2008.
- [Can86] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, 1986.
- [CGW⁺07] Lawrence D. Cutler, Reid Gershbein, Xiaohuan Corina Wang, Cassidy Curtis, Erwan Maignet, Luca Prasso, and Peter Farson. An art-

- directed wrinkle system for CG character clothing and skin. *Graphical Models*, 69(5-6):219–230, 2007.
- [CMT05] Frederic Cordier and Nadia Magnenat-Thalmann. A data-driven approach for real-time clothes simulation. *Computer Graphics Forum*, 24(2):173–183, 2005.
- [dAST⁺08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98, 2008.
- [DEG⁺99] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. *SIAM J. Matrix Analysis and Applications*, 20(3):720–755, 1999.
- [DJW⁺06] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual garments: a fully geometric approach for clothing design. *Computer Graphics Forum*, 25(3):625–634, 2006.
- [GHF⁺07] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient simulation of inextensible cloth. *ACM Trans. Graph.*, 26(3):49, 2007.
- [GWM⁺08] Mashhuda Glencross, Gregory J. Ward, Francho Melendez, Caroline Jay, Jun Liu, and Roger Hubbard. A perceptually validated model for surface depth hallucination. *ACM Trans. Graph.*, 27(3):1–8, 2008.

- [HBVMT99] Sunil Hadap, Endre Bangerter, Pascal Volino, and Nadia Magnenat-Thalmann. Animating wrinkles on clothes. In *VIS '99*, pages 175–182, 1999.
- [HVB⁺07] C. Hernández, G. Vogiatzis, G. Brostow, B. Stenger, and R. Cipolla. Non-rigid photometric stereo with colored lights. In *Proc. ICCV*, 2007.
- [KA08] Michael Kass and John Anderson. Animating oscillatory motion with overlap: wiggly splines. *ACM Trans. Graph.*, 27(3):28, 2008.
- [KMP07] Martin Kilian, Niloy J. Mitra, and Helmut Pottmann. Geometric modeling in shape space. *ACM Trans. Graph.*, 26(3):64, 2007.
- [LB00] Michael S Langer and Heinrich H Bülthoff. Depth discrimination from shading under diffuse lighting. *Perception*, 29:649–660, 2000.
- [LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph.*, 24:479–487, 2005.
- [PH03] D. Pritchard and W. Heidrich. Cloth motion capture. In *Proc. Eurographics*, pages 263–271, 2003.
- [PH08] Sang Il Park and Jessica Hodgins. Data-driven modeling of skin and muscle deformation. *ACM Trans. Graph.*, 27(3):96, 2008.
- [PJS06] Tiberiu Popa, Dan Julius, and Alla Sheffer. Material-aware mesh deformations. In *SMI '06*, 2006.
- [SB08] Hagit Schechter and Robert Bridson. Evolving sub-grid turbulence for smoke animation. In *Prof. SCA*, 2008.

- [SLCO⁺04] Olga Sorkine, Yaron Lipman, Daniel Cohen-Or, Marc Alexa, Christian Rössl, and Hans-Peter Seidel. Laplacian surface editing. In *Symposium on Geometry Processing*, pages 179–188, 2004.
- [SP04] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.
- [SSGH01] Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *SIGGRAPH*, pages 409–416, 2001.
- [SSK⁺05] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor. Garment motion capture using color-coded patterns. In *Proc. Eurographics*, pages 439–448, 2005.
- [SZT⁺08] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Example-based dynamic skinning in real time. *ACM Trans. Graph.*, 27(3):29, 2008.
- [VBMP08] Daniel Vlastic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):97, 2008.
- [WCF07] Ryan White, Keenan Crane, and David Forsyth. Capturing and animating occluded cloth. *ACM Trans. Graph.*, 26(3):34, 2007.

Chapter 5

Globally Consistent Space-Time Reconstruction

5.1 Introduction

⁷ The combination of geometry scanning and capture techniques with surface reconstruction methods provides a powerful and convenient way for creating virtual replicas of real-world objects. Until recently, both scanning and reconstruction methods focused on processing of static shapes. Over the past few years, however, scanning methods for dynamic geometries have evolved as both experimental research systems and commercial products. These systems are capable of providing raw capture data, typically point clouds, for dynamically deforming shapes. The available surface reconstruction methods have so far lagged behind this development, as the reconstruction of topologically consistent time-varying meshes from such point clouds is still an unsolved problem for general datasets.

Such reconstruction of geometric shapes from space-time point clouds presents two important and interconnected challenges: correct reconstruction of per-frame 3D geometry and the establishment of consistent correspondences between these per-frame geometries. While it is possible to reconstruct the geometry in each frame independently using a variety of existing techniques, a significantly bet-

⁷A version of this chapter will be submitted for publication. Popa, T. and South-Dickinson, I. and Bradley, D. and Sheffer, A. and Heidrich, W.

ter reconstruction is possible by accumulating information over time, leveraging knowledge on the temporal behavior of the captured object, e.g. [PG, WJH⁺07, SAL⁺08, BPS⁺08].

A key factor in the ability to accumulate geometric information across frames is a set of assumptions that can be made about object behavior. This has led to the development of domain-specific reconstruction methods targeting narrow classes of shapes such as piece-wise rigid objects [PG], garments [BPS⁺08], or faces [BBA⁺07, ZSCS04]. Such methods facilitate high quality space-time reconstruction but are not applicable to wider sets of objects.

This chapter addresses a much wider, yet sufficiently well-defined class of objects. An important observation is that for most typically scanned objects the changes in the object's shape are very gradual both in terms of Euclidean coordinates and in terms of intrinsic shape. This observation is true for any articulated shape, for humans or animals, garments, and many other objects in our everyday surrounding. The *gradual change* assumption effectively implies that no discrete changes which drastically affect the intrinsic shape, such as a change in the object genus, are possible. So for instance, when the fingers of a human hand touch as shown in Figure 5.1 (top), the correct interpretation, and reconstruction, is two surfaces coming into contact and not a genus change. Thus, a key observation is that since the change is gradual, the cross-parameterization, or correspondence, between consecutive frames should exhibit very little stretch, as the intrinsic surface distances change very little with a small change in time. Note that the gradual change assumption does not discount the possibility of large changes in the shape across time. This observation forms the backbone of the method.

Another key design decision made by different reconstruction techniques is the degree of reliance on individual frame information when assembling the data across time. Many methods, e.g. [dAST⁺08, VBMP08], use the per-frame data as

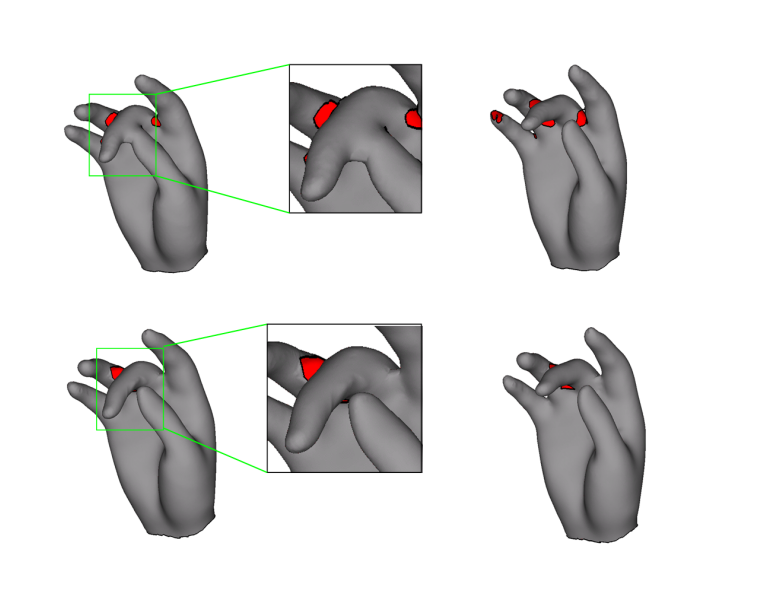


Figure 5.1: Two frames of the hand sequence. Top: initial reconstruction. Bottom: final reconstruction with correct topology

a fairly loose guide for the reconstruction, thus overcoming noise or inconsistencies in the data, but also losing fine details. On the other end of the spectrum, Bradley et al. [BPS⁺08] reconstruct each frame individually, capturing all the geometric details present, and then find correspondences between them. This approach fails if the per-frame reconstruction produces inconsistent frame connectivity or erroneous geometry. The method presented here combines the advantages of both types of methods, capturing fine details while recovering from sub-optimal local decisions. To this end it starts by reconstructing individual frame geometry. Then it proceeds to assemble the individual frames into a consistent space-time frame sequence, simultaneously completing missing information across time and correcting local inconsistencies. This is the first method to develop such an explicit temporal correction mechanism.

As part of the solution, an explicit bijective mapping between any pair of con-

secutive frames is computed, thus establishing a global parameterization as well as a common connectivity for all the frames. While space-time accumulation of geometric information can be achieved without establishing explicit correspondences between the per-frame geometries [SAL⁺08], correspondence or cross-parameterization between per-frame geometries is necessary for explicitly capturing the object's motion across time as well as for such basic geometry processing tasks as texture mapping.

Computing this map usually consists of two steps: first computing a rough initial registration or partial-registration of the frames followed by an optimization step that improves upon the initial registration to obtain a complete and accurate map [PG, WJH⁺07, WAO⁺09, ATR⁺08]. Some methods use geometric proximity for the initial registration [PG, WAO⁺09], but this may lead to poor initial registration. Other methods use either 2D or 3D feature matching techniques [Low99, HSKK01] to acquire an initial registration [VZBH08]. These feature correspondences are typically sparse and non-uniformly distributed in the image or model space respectively. Since the motion of most objects is gradual and continuous I use optical flow [Bou02] to obtain an initial set of correspondences between frames. Optical flow has the advantage that it provides a more uniform distribution of the correspondence points which makes the reconstruction more robust.

As demonstrated by the results, this method can handle a wider variety of models than previous techniques, better preserving the input geometric details, robustly handling noise and inconsistencies in the data, while preserving important properties of the object motion across time.

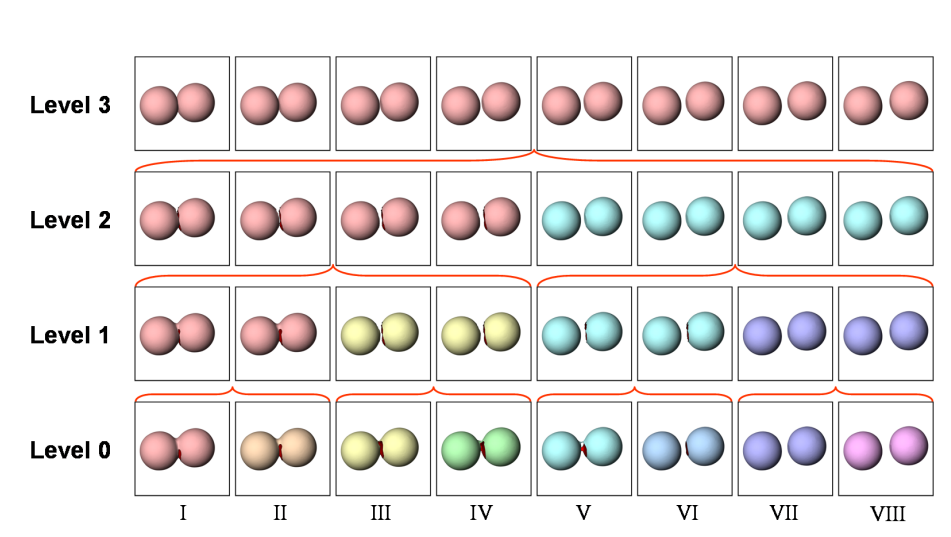


Figure 5.2: Hierarchical assembly of a consistent frame sequence. Bottom to top sequences of length 1,2,4 and 8 respectively. Note the correction step applied to frames III and IV, V and VI, and again to the first two sequences in layer two.

5.2 Related Work

The main challenge faced by dynamic reconstruction is to utilize the temporal component, accumulating information over time, to correctly complete locally missing data [RHHL02]. Existing space-time reconstruction techniques can be loosely classified based on the assumptions they make about temporal behavior of the scanned objects, as well as based on the way they process the available per-frame data.

Many of the methods target a narrow class of objects such as garments [BPS⁺08] or faces [ZSCS04, BBA⁺07]. Some of the earlier methods used dense sets of custom markers on the scanned objects as an aid for establishing temporal correspondences and data accumulation [SSK⁺05, WCF07, BBA⁺07]. Furukawa and Ponce [FP08, FP09] developed a 3D tracking technique to reconstruct the geometry and motion of garments and faces. They use the first frame as a template and

they advance this geometry in time. Therefore, the first frame must have the correct topology of the captured object.

More recently, Bradley et al. [BPS⁺08] provided a markerless solution for garment capture utilizing unique garment motion characteristics. For objects that do not exhibit significant stretch under deformation, [HAWG08, TBW⁺09] propose methods for isometric pose registration. Another class of specialized methods focuses on piece-wise rigid shapes [PG, CZ].

Much attention had been paid to capture of human motion [ACP02, ACP03, dAZT⁺07, AG04]. Several techniques [PH06, PH08, ASK⁺05] use markers placed at known locations on the body together with skeletal structure knowledge to facilitate the reconstruction. In the absence of markers [ATR⁺08] use SIFT feature matching [Low99] to generate a sparse set of correspondences between frames. Then they extend these correspondences to the entire mesh. However, in their setup they do not have missing geometry so no correction/completion mechanism.

De Aguiar [dAST⁺08] and Vlasic [VBMP08] avoid the need for special markers by using an articulated template of the human being scanned and a user-provided correspondence between the template and the first frame in the capture. The space-time reconstruction is generated by automatically fitting the template to the rest of the input frames. This approach can potentially be extended to a wider class of shapes if a template is available. However, constructing a template and establishing the correspondence between it and the first frame can be non-trivial. More importantly, both methods use the input per-frame data as a loose guide for deforming the template, and thus tend to smooth out fine geometric features present in the data. Varanasi et al [VZBH08] propose a more general method to compute dense motion flow between frames using a mesh evolution technique. Their method focuses on recovering the motion flow and does not complete missing geometry.

The aim is for a more general method applicable to a large set of objects, in-

cluding all the classes listed above, using the basic assumption of gradual change. The main goal is to use the global spatial-temporal geometric space to coherently and robustly complete and correct geometry locally.

Several methods approach the spatial-temporal capture problem in a global way. [SAL⁺08, GILM07, MG04] propose volumetric-based space-time techniques for the reconstruction of moving and deforming objects from point data. Their methods are based on the reconstruction of a 4D manifold represented by the data. These methods work well to complete the missing geometry leveraging temporal information, but they do not produce correspondences between frames, greatly limiting the number of applications. [SWG08] employs a volumetric space-time approach to reconstruct the geometry. Their method uses the first reconstructed frame as a template and it computes the motion of the object by advancing this template frame through time.

Mitra et al [MFO⁺07] propose a method for registering arbitrarily deformable models. Since their algorithm is geared toward the registration problem it does not fill in missing geometry. Wand et al [WAO⁺09] developed a framework for global, topology-aware space-time reconstruction. Given the geometry reconstructed in every frame, they use an optimization framework to improve the geometry and compute the motion of the object. Their optimization framework does not restrict the final position of the points be on the reconstructed geometry, therefore the final reconstruction may be different from the initial per-frame reconstruction.

5.3 Overview

The goal of this method is to reconstruct a *consistent frame sequence* from a given space-time point cloud using precomputed optical flow correspondences as an aid. A consistent frame sequence is defined as a sequence of meshes with the same

connectivity and with per-frame vertex positions that satisfy the gradual change assumptions of small spatial and intrinsic changes in the mesh. The shared connectivity explicitly defines a cross-parameterization or mapping between all pairs of frames in the sequence. When performing the reconstruction one of the objectives is to maximize the use of per-frame geometric information. To this end the method starts by independently reconstructing per-frame meshes from the input point clouds using a state of the art technique [BBH08]. Then the per-frame meshes are assembled into a consistent frame sequence while maintaining as much as possible the per-frame geometry.

In order to maximize the use of available geometric information this method propagates the geometric information available in every frame to the entire sequence. At each step, a hierarchical sequence assembly procedure combines pairs of consecutive consistent frame sub-sequences, possibly of length one, into a single consistent sequence (Figure 5.2), merging the geometric information available in both. At the bottom level, zero, individual frames are paired. At level one sub-sequences of length two are paired, etc. When selecting pairs of sequences to process, each sequence is used in only one pairing, avoiding redundancy. Thus at each level the number of sequences shrinks by a factor of two.

Two sequences are merged by combining the last frame of the first sub-sequence referred to as *source mesh* and the first frame of the second sub-sequence referred to as *target mesh*. While this operation is done locally between two consecutive frames, each of the two sub-sequences have already accumulated the consensus geometry of a larger frame interval, propagating the geometric information across multiple frames, thus improving the initial per-frame reconstruction. For example, in the sphere sequence in figure 5.2, the local per-frame reconstruction of both frames four and five have missing geometry and incorrect topology. However, when they are combined on level two, they both already have correct topology, and

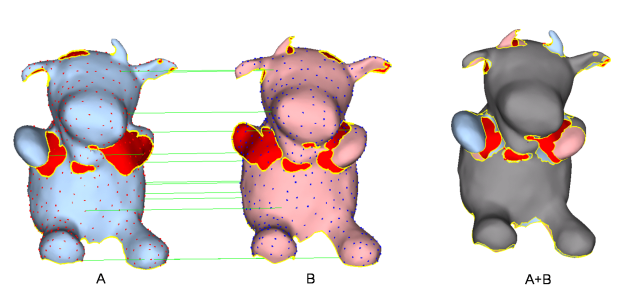


Figure 5.3: Combining consecutive frames: (left-center) sparse correspondence from the optical flow; (right) consensus geometry of the frames. Common geometry is shown in gray and holes are highlighted in red

the complete sphere geometry that exists only in frame seven is already propagated to frame five. And when frame four and five are combined, that geometry is further propagated through the entire sequence.

Pairwise Sequence Combination: The combination of two consecutive sequences starts by computing a cross-parameterization between the source and target meshes. Directly computing a global parameterization [KS05] between two incomplete meshes can be quite challenging, and may not even be possible if the meshes have, for instance, different genus. Global cross-parameterization methods usually parameterize both meshes onto a common base-mesh. Disjoint patches on each model are parameterized onto common faces of the base-mesh, thus providing a map between the source and the target. All three meshes: source, target and base must be topologically compatible in order for these cross-parameterization methods to work. However, as illustrated in figures 5.1 and 5.3, sequences often contain meshes that are not topologically compatible therefore it is not possible to use these methods. These incompatibilities are typically the result of incorrect per-frame reconstruction, as the per-frame data is often ambiguous. A robust space-time reconstruction method should ideally detect and correct such per-frame errors.

A local parameterization technique is used as a stepping stone toward a global cross-parameterization. While directly computing a global cross-parameterization between topologically incompatible meshes may not be possible, a local parameterization method that maps disk-shaped patches on one model to disk-shaped patches on the other model can be used to cross-parameterize incomplete and topologically inconsistent models without any special processing. These disk-shaped patches can be easily cross-parameterized by flattening and aligning them both onto the 2D plane. By performing this local cross-parameterization repeatedly on a set of patch pairs that covers the entire source and target models respectively, it is possible to obtain a global cross-parameterization from the source to the target mesh.

This cross-parameterization is expected to satisfy the gradual change assumptions. Therefore, any local violation of these is typically indicative of *geometric inconsistencies* between the two sequences being processed. For instance, the initial geometry reconstruction of the source frame in figure 5.8 incorrectly connects the two spheres due to their spatial proximity. These inconsistencies are thus detected and fixed in order to combine the frames.

Another important goal of the method is to improve the initial per-frame geometry using geometric information from other frames. Each of the combined sequences can contain geometric information not available in the other. For instance, in figure 5.3(left) the right arm of the hand-puppet is only available in the source mesh while the left arm is only available in the target mesh. Therefore the resulting meshes contain the union of the geometric information in the two frames as illustrated in Figure 5.3(right). Finally, a common connectivity is established for the merged subsequences. The combination algorithm consists of three main steps:

- **Cross-Parameterization:** Using a local parameterization technique the method

first computes a cross-parameterization between the source and target meshes, striving to minimize both intrinsic change (stretch) and spatial change (vertex movement). To initialize the cross-parameterization computation the algorithm requires a sparse set of initial correspondences, called **anchors**, **between the two frames. These are computed using 2D optical flow from the video footage (figure 5.3(left)), (Section 5.4).**

- **Analysis and Correction: The gradual change assumption implies small spatial and intrinsic changes between consecutive frames which in practice means that the mapping from source to target should exhibit low stretch and the vertices should move consistently with their neighboring vertices as well as their neighboring anchors. Vertices that do not obey the gradual change assumption are identified and removed from the model. This will generate holes in the mesh that will later be filled, if possible, by the completion step. This is a key step in the algorithm that corrects geometric inconsistencies and propagate them throughout the sequence. (Section 5.4.3**
- **Completion and Connectivity Combination: The geometric information in the source and target frames is merged as illustrated in Figure 5.3(right) and a common connectivity for both frames is obtained. (Section 5.6)**

Note that using this hierarchical sequence assembly approach, the completion mechanism can propagate geometric information across any number of frames, as the incremental completion will propagate the information up through the hierarchy. This is demonstrated in Figure 5.2, where the complete geometry for the right ball is available in only one frame and is nevertheless successfully propagated across the entire sequence.

This combination algorithm results in a consistent frame sequence that approximates as much as possible the geometry of the two input sequences. The following sections discuss the major steps of the combination procedure.

5.4 Cross-Parameterization

The goal of this step is to compute a low distortion map between the source and target meshes. This map will only be a partial map since vertices in the source mesh that correspond to holes in the target mesh will have no correspondence. Computing such parameterization typically requires some sparse set of initial correspondences [BPS⁺08, KS05, ASK⁺05]. Without any correspondences, geometric proximity is the only way to initialize such a parameterization, but this typically leads to poor initial maps that are difficult and time-consuming to improve. Since this is a video-based capture system, standard optical flow methods [Bou02] can be used to obtain a sparse set of correspondences (Section 5.4.1). One advantage of optical flow based correspondence over other types of feature tracking techniques [Low99] is that it provides a fairly uniform distribution of the correspondence points.

A low-distortion consistent parameterization is computed in two steps. First, a *local parameterization* technique is used to efficiently compute an initial low-stretch parameterization (Section 5.4.2). This local parameterization provides a good initial guess for a global parameterization, but it may contain local distortions (Figure 5.7 (left)). Therefore the mapping is further optimized using a *relaxation* technique (Section 5.4.3), to provide a final low-stretch map. Areas that still have local high stretch are indicative of inconsistencies between the processed frames. Geometry in these areas is further analyzed and corrected as described in Section 5.5.

5.4. Cross-Parameterization

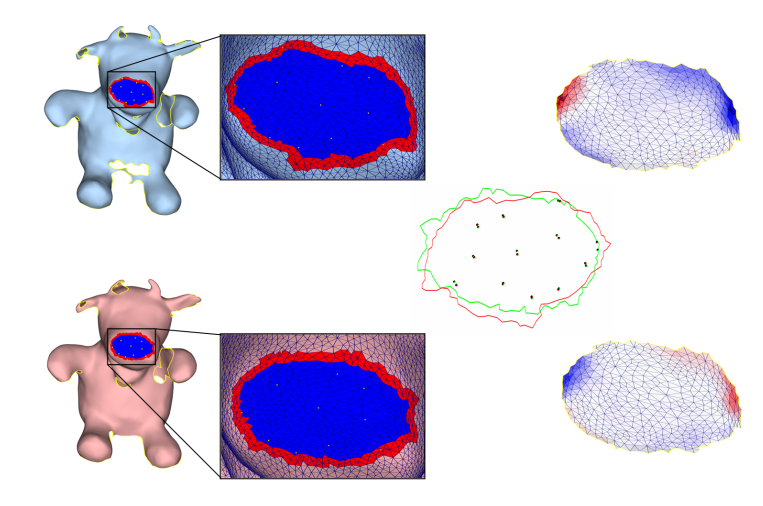


Figure 5.4: Local patch parameterization: (left) matching patches with anchors highlighted and ignored triangles shown in red; (center) aligned planar parameterizations; (right) combined parameterization visualizing stretch. The maximal stretch (red) is 1.05 and the minimal (blue) is 0.95 (ideal stretch is one)

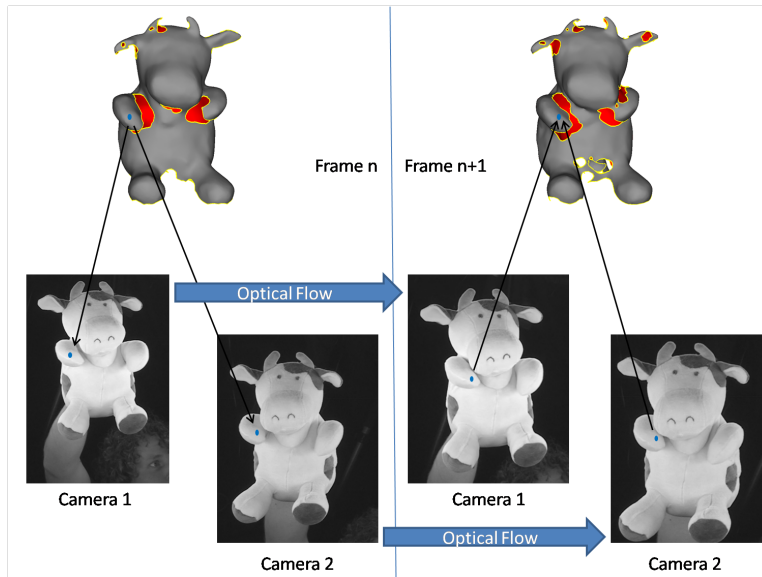


Figure 5.5: Finding point correspondences between consecutive frames using calibrated cameras and optical flow.

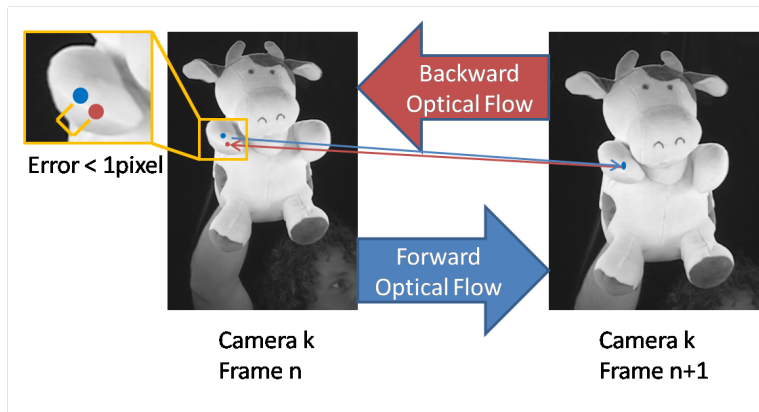


Figure 5.6: Optical flow pruning in 2D

5.4.1 Optical Flow Tracking

To generate the set of sparse correspondences for two consecutive meshes, the method uses the optical flow [Bou02] between frames in all camera views. Each vertex of the mesh is first projected onto the screen space of two consecutive cameras that see the vertex. Then these 2D points are slid forward in time to the next frame using the 2D optical flow in camera space. The slid points are projected back onto the model at the next frame, thus obtaining the correspondence for the vertex in that frame. This process is illustrated in Figure 5.5 Due to errors in the optical flow, not all correspondences are reliable. Unlike other methods such as [FP09] that rely on a dense, accurate optical flow to track the motion of the vertices through time, the cross-parameterization method used here is more robust requiring only few optical flow correspondences between the source and target meshes, roughly on the order of only three percent of the number of vertices of the mesh. Therefore, the selection of optical flow matches can be very conservative, keeping only reliable optical flow correspondences. To eliminate unreliable correspondences two levels of pruning are performed. First level is in 2D and it checks the

5.4. Cross-Parameterization

optical-flow in camera space for consistency as illustrated in Figure 5.6. Every point is slided forward and then backward in time with the expectation that, if the optical flow is accurate, it should map back to the same point. All points that do not map back within one pixel are discarded. The second level operates on the 3D vertices of the reconstructed mesh. Each vertex is slided forward and then backward using the optical flow information and camera calibration as illustrated in Figure 5.5. The expectation is that the original point and the mapped point should have the same 3D location. All vertices that do not map within an epsilon distance from their original position are pruned. The remaining set of reliable optical flow points are the anchors used to perform the local cross-parameterization. Even after pruning, the optical flow matches may still be slightly inaccurate. However, in this setting they are used only as an initial guide for parameterization, and thus incorrect matches will only have a minor impact on the results.

Depending on the motion speed and the natural texture of the object, the anchors may be distributed unevenly over the model surface. There could be a high concentration of anchors in areas where the object has a rich natural texture or a low concentration of anchors in areas where, for instance, the motion from one frame to the next is too fast. The local parameterization works optimally if the anchors are distributed roughly uniformly, therefore the anchors are re-distributed in order to improve the final results. In areas with a high distribution of anchors their density can be reduced by removing some of the redundant anchors. In areas with low distribution of anchors the anchor set can be refined as described in section 5.4.4. Using these anchors the method then proceeds to compute the cross-parameterization between the consecutive frames.

5.4.2 Local Patch-Based Parameterization

A local patch-based parameterization technique is used as a stepping stone toward computing a global parameterization. As noted by Bradley et al. [BPS⁺08], if two near-isometric patches are mapped to the same domain using stretch-minimizing parameterization, then their maps are likely to be identical, or in other words, using the map from one patch to the common domain and then the inverse map from the common domain to the second patch should lead to a near-isometric parameterization between the patches (Figure 5.4). Based on this observation patches are grown on both models centered around anchors and find correspondences by parameterizing the patches into the plane and aligning them using the anchors. These patches are grown independently and will overlap. Therefore, most vertices will typically be located in multiple patches. Using these local cross-parameterizations between patches, a global cross-parameterization is established between the meshes by selecting for each vertex only a single patch. The remainder of this section presents the steps of the algorithm in more detail.

Patch Growth: Anchor points on the source mesh have already a corresponding point on the target mesh, thus first patches are grown simultaneously from all the source anchors and their corresponding points on the target while preserving patch compactness and disk topology. The growth process terminates once the patches contain three to five matching anchors (Figure 5.4(left)). While three anchors are sufficient for subsequent processing, the parameterization is more robust to noise in the optical flow when the number of anchors per patch is higher. If the patches grow beyond a certain radius but not enough anchors are found, the growth is aborted and the patch is discarded. Large patches are avoided because they may exhibit significant cross-parameterization stretch.

Patches grown only from anchors may not always cover the entire model result-

ing in unmapped vertices. This could happen in areas with no anchors, typically near boundaries. To improve the coverage a second iteration of patch growth is performed, but this time the patches are grown from the remaining uncovered regular vertices. The challenge here is that regular source mesh vertices do not a priori have a natural match on the target mesh. Therefore, when growing a patch from a regular vertex, the patch is grown on the source mesh until an anchor is encountered, and then a patch is grown on the target mesh around the matching anchor until its radius is equal to the one of the first patch. From here the patch is grown as before. Vertices that still remain uncovered at the end of this process typically indicate source regions that lack matching geometry on the target. Therefore they do not have a mapping and are ignored in the cross-parameterization step.

Patch Cross-Parameterization: The two matching patches are first parameterized independently in the plane. ABF++ [SLMB05, Gra03] is used for computing the planar parameterizations as it provides a reasonable trade off between minimizing stretch and efficiency. The two parameterizations are then overlapped in the 2D plane using an affine transformation that aligns matching anchors in a least-squares sense (Figure 5.4 (center)).

After the alignment the parameterization typically has the correct low stretch in the interior of the patches, close to the anchors. However since the patches are not perfectly aligned and may have fairly different boundary shape, the distortion is larger closer to the boundaries (Figure 5.4 (right)). Hence, for parameterization purposes the mapping in the boundary region (highlighted in red in figure 5.4 (center)) is ignored.

Patch Selection: The majority of mesh vertices are likely to be covered by more than one patch. Each patch map may indicate a slightly different position. This happens because the cross-parameterized patches are not strictly isometric. In order to compute a consistent low-stretch, global cross-parameterization from the

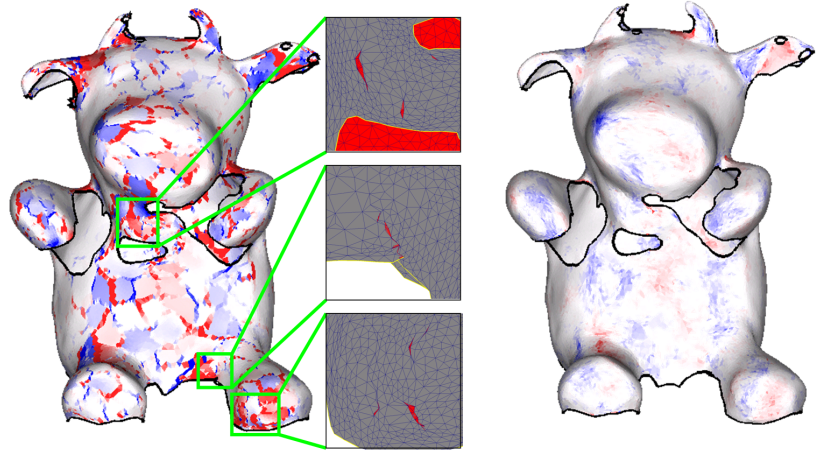


Figure 5.7: Relaxation: (left) Initial global parameterization with stretch and flipped triangles most prominent when mapping switches patches; (right) map stretch after six iterations of local relaxation

set of local patch-based parameterizations the maps of overlapping patches need to be reconciled. This is achieved by selecting for each vertex the patch that provides its best mapping. Since lower stretch usually indicates a more accurate map, one way to select the best patch is by checking the overall stretch of all patches that contain the vertex and select the one with least overall stretch. However, as mentioned above, the stretch occurs mostly near boundaries, therefore a better choice is to select the patch in which the vertex is closest to the center. This step creates a global cross-parameterization between the source and target meshes. The cross-parameterization is further improved using a local relaxation technique.

5.4.3 Local Relaxation

The mapping computed by this cross-parameterization is usually fairly close, distance wise, to the target minimal-stretch mapping, where one exists. At the same time, locally, the quality of the mapping can be arbitrarily bad, including the ex-

treme case of flipped triangles (Figure 5.7 (center)). This tends to happen near patch boundaries where neighboring vertices may be included in different patches. Since patches are created independently and may overlap, a triangle that has vertices in two adjacent patches may be flipped (Figure 5.7). An optimization procedure is needed that simultaneously improves the stretch of the map and detects and fixes flipped triangles. Choosing a stretch minimizing functional such as the one in Schreiner et al. [SAPH04] will not resolve the extreme cases of flipped triangles, because the stretch minimizing functional often gets stuck in a bad local minima. This particular functional is also quite time consuming, with Schreiner et al. listing runtimes of an hour or more.

Since the change in the intrinsic shape from one frame to the next is small, a shape preserving optimization functional will be consistent to a stretch minimizing functional. Thus, rather than directly minimizing stretch it is better to use Laplacian coordinates as the optimizing functional [LSA⁺05]. Evaluating the Laplacian is significantly faster than evaluating stretch, but more importantly the Laplacian functional tends to detect flipped triangles, and optimizing it eliminates them. Although there is no guarantee that all flipped triangles are detected and fixed, in my experience it does work in the vast majority of cases.

The minimizing problem is the following:

$$\min_{\tilde{v}} \|L\tilde{v} - Lv\|_2^2, s.t. \tilde{v} \in B \quad (5.1)$$

where L is the Laplacian operator over the source mesh, v are the original positions on the source mesh, \tilde{v} are the mapped positions on the target mesh B . Without the constraints, the obvious solution is the source mesh: $\tilde{v} = v$. By constraining the solution space to the target mesh, the initial, per-frame reconstructed geometry of the target mesh is preserved.

Due to the nature of the constraints, it is not easy to formulate 5.1 as a global optimization problem. Also, the local cross-parameterization mapping provides a good initial guess for the optimization problem with the distortion concentrated only in a few places in the mesh. These characteristics make this problem particularly suitable for a local relaxation approach where vertices are iteratively optimized by perturbing their location in a small neighborhood.

More specifically, a local random-walk search mechanism is used. All the source mesh vertices are first sorted in decreasing order of their error (eq. 5.1). Next, for each vertex, a random direction is selected and the vertex is moved along this direction, optimizing for its Laplacian vector. For each vertex, around a hundred local random walks are performed. Once all vertices are processed, the relaxation procedure is repeated. Typically only a few global iterations (five to six) of Laplacian relaxation are sufficient to reduce stretch across much of the mesh to acceptable levels (Figures 5.7 (right)). Once the flipped triangles are corrected, it is also possible to run the stretch-optimization technique, but in most cases there is no noticeable difference between optimizing stretch and optimizing for the Laplacian vector.

5.4.4 Anchor Set Refinement

The set of anchors obtained from the optical flow procedure described in section 5.4.1 can be too sparse in some areas. This can happen if, for instance, the motion in a particular frame is too fast and the video images have motion blur. As a consequence, the areas with sparse anchors contain large patches. Large patches tend to have more significant misalignment errors and that generally leads to larger cross-parameterization stretch.

To avoid this situation, the initial anchor set is refined to provide a more dense

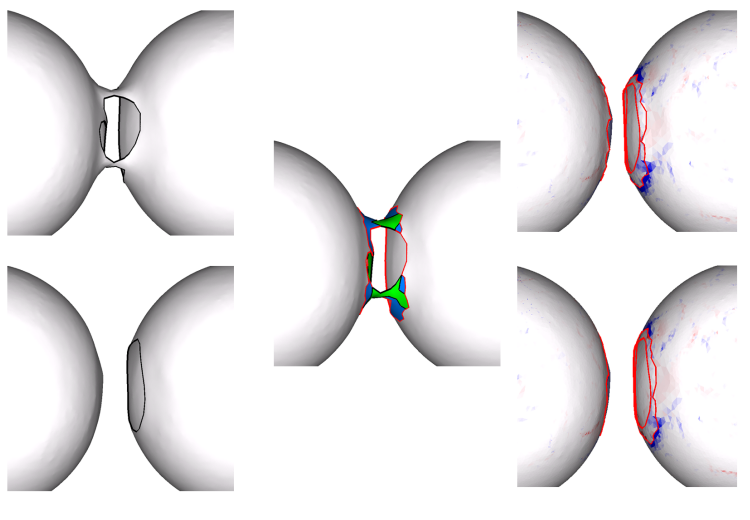


Figure 5.8: Correction: (left) initial geometry of frames 5 and 6 of the sphere sequence in 5.2; (center) problem regions: high stretch regions (blue) and regions with no mapping (green); final result with geometry completion. The color represents stretch varying from 0.9 (blue) to 1.1 (red)

and uniformly sampled set of anchors. A naive method to add anchors is to select some vertices in the areas with sparse anchors on the source mesh and find their correspondences on the target mesh using a closest point search. Unfortunately these anchors will have a poor mapping, because the two frames are in different poses. However, if the source mesh could be only roughly registered in the same pose as the next frame, a closest point search will provide a more reliable map for the new anchors.

The refinement algorithm first runs the cross-parameterization step with the initial set of anchors, limiting the maximum patch size. Limiting the maximum size of a patch results in vertices that do not belong to any patch and as a consequence they will be unmapped. Using the local Laplacian coordinates from the source mesh, the unmapped vertices are positioned in the target frame using the mapped vertices as constraints to the Laplacian system [LSA⁺05]. After this step,

the source mesh is registered in the target frame. Additional reliable anchors are now added by simply using a closest point search. Unmapped vertices that have a nearly zero distance to the target mesh are added to the anchor set.

The cross-parameterization is then re-computed using this refined anchor set for a better mapping. At the end of this process there are typically sufficient anchors to compute a reliable, low-stretch map when no inconsistencies exist.

5.5 Analysis and Correction

The cross-parameterization step computes a map between the source and target meshes. Under the gradual change assumption, mapped triangles should exhibit low stretch and mapped vertices should have a motion consistent with their neighboring vertices as well as their neighboring anchors. If either of these conditions is violated it means that locally the cross-parameterization map is inconsistent. These inconsistencies generally occur as a result of a topological change (figure 5.8), but they can also occur if the reconstructed per-frame geometry of one of the frames is wrong. This simple observation provides a powerful mechanism to detect and subsequently fix geometric inconsistencies only by measuring triangle stretch and vertex motion and it is a key step of the method.

The cross-parameterization map is checked for inconsistencies that violate the gradual change assumption by analyzing triangle stretch and vertex motion of the cross-parameterization map. Vertices belonging to triangles that exhibit high-stretch, and vertices whose motion is much larger compared to their neighboring vertices and anchors are flagged as inconsistent and clustered into regions. It is not known a priori whether the inconsistencies are caused by problems with the geometry in the source or target regions, but at least one of the source and target regions has incorrect geometry. Therefore, the algorithm needs to identify which, if any, of

the source or target regions is consistent with the gradual change assumptions and which isn't. For each inconsistent region the algorithm executes two scenarios and in the end it picks the best out of the two. First the inconsistent region in the source mesh is deleted and replaced with a patch from the target mesh, using the completion mechanism described in the next section. A test is performed to see whether this new geometry pasted from the target obeys the gradual change assumptions. Second the inconsistent region on the target mesh is deleted and replaced with a patch from the source mesh. A test is performed to check if it satisfies the gradual change criteria. The test of compliance with the gradual change consists of two stretch criteria. To pass the test, a patch must have an overall stretch between 0.9 and 1.1, and no more than 15% of the area of the patch should have stretch below 0.9 or above 1.1

There are three possible outcomes. The first, and most frequent one, is that only the geometry of one of the meshes satisfies the gradual change criteria. In this case the one that satisfies the gradual change criteria is selected as the consistent geometry for both frames, with the completion result propagated across the relevant frame sequence. The second case is when both completions violate the gradual change assumption, indicating that neither is acceptable. In this case the inconsistent region is deleted from both frame sequences. The third case is when both completions lead to a consistent result. In this case, for simplicity, the region from the source is selected.

5.6 Geometry Completion and Connectivity Combination

The goal of the pairwise sequence combination step is to compute two compatible meshes, namely meshes with the same connectivity, for the source and target

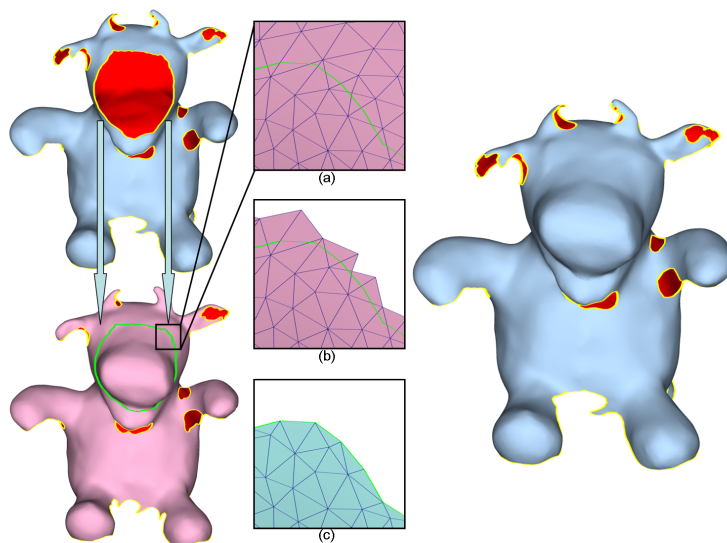


Figure 5.9: Completion: (left) Mapping the boundary; (right) completing the geometry. Missing geometry highlighted in red

frames that reflect their *consensus geometry*. For instance, even though the original source mesh is missing the left arm (Figure 5.3), the resulting completed source mesh should acquire the geometry of the arm by extracting it from the target mesh and paste it into the source mesh. Similarly for the right arm which is missing in the target mesh.

An equivalent way to state this problem is to say that one mesh connectivity is needed with two sets of vertex positions: one corresponding to the source frame k and one corresponding to the target frame $k + 1$. This is called a *union mesh*. This union mesh is created in two steps. It starts from the geometry which is common to both meshes (figure 5.3 (right) highlighted in gray) using the connectivity of the source mesh. This partial union mesh already has two sets of vertex positions: one from the original source mesh at frame k and the second one at frame $k + 1$ from the mapping onto the target. This mesh will have holes in the areas where the

5.6. Geometry Completion and Connectivity Combination

geometry is only available in one of the source or target meshes. Therefore, these holes need to be filled with surface patches available either only in the source mesh such as the left arm of the hand-puppet highlighted in blue in figure 5.3, or only in the target mesh such as the right arm highlighted in pink in figure 5.3.

Geometric patches that exist only in the source mesh can be easily identified as regions where source vertices have no mapping on the target mesh. For these vertices only their position in the source frame k is known therefore all is left to compute are their positions in the target frame $(k + 1)$. Using the local Laplacian coordinates from the source mesh, vertices with no mapping onto the target mesh are positioned in the target frame $(k + 1)$ solving the Laplacian system [LSA⁺05] using the mapped vertices as constraints.

Geometric patches that exist only in the target mesh present an additional challenge. Since the connectivity of the union mesh is derived from the source mesh and the connectivity of these patches is derived from the target mesh, the connectivity of each patch needs to be seamlessly merged with the connectivity of the union mesh. Effectively, for each patch a boundary on the union mesh is mapped onto the target mesh (figure 5.9 (left)). The goal is to use the geometry of the target mesh at frame $k + 1$ located inside the boundary to complete the geometry in the corresponding region on the union mesh [KS05, Lev03] at frame k . A patch on the target mesh that can seamlessly connect the new geometry to the union mesh needs to be created. To achieve this goal the target region is extended to include all the triangles intersected by the boundary mapping (Figure 5.9(b)), providing a sequence of edges that will be used to glue the new geometry to the union mesh (Figure 5.9(b) - green edges). The completion region is extended to contain the triangles the edge sequence intersects and parameterize it in the plane. A standard planar mesh generation is now used to connect the mapped edges to the vertices in the completion region (Figure 5.9(c)) thus merging the two connectivities. Finally

the positions of the patch vertices at frame k are computed as before using the the Laplacian coordinates computed from the vertex positions at frame $k + 1$ (Figure 5.9(d)).

Propagation: If the processed frame sequence contains more than one frame, the completion needs to be propagated across the sequence. Since all the frames in the sequence are compatible, the connectivity of the union mesh is used for the entire sequence. However, the vertex positions for all the completed holes are recomputed for each frame in the sequence using the same Laplacian mechanism.

After all these steps are performed a one-to-one mapping between all frames is established in the combined sequence. This means that the two subsequences are effectively merged into one consistent frame sequence as discussed in Section 5.3.

5.7 Discussion and Results

This reconstruction method has been run on several real and synthetic datasets (Figure 5.2 and Figures 5.10- 5.13). The *spheres* and the *hand* dataset have been created synthetically. In order to generate realistic datasets, self-occlusion was “simulated” by placing virtual cameras in the scene, hence even the synthetic datasets have missing geometry. The *hand-puppet*, *T-Shirt* and *two hands* datasets are captures of real objects. The *hand-puppet* and *two hands* were captured using a 16 cameras ring setup, and the geometry of each individual frame has been reconstructed using [BBH08]. The *T-Shirt* dataset has been taken from [BPS⁺08].

For efficiency all reconstructed per-frame meshes have been simplified to around 25k vertices and on average about 600 optical flow points correspondences on the surface, or around 2% of the size of the mesh, were used. Since the algorithm is highly paralellizable, it was run on a cluster of 27 Intel Xeon 3Ghz CPUs. The run-time is split roughly among three operations: initial patch-based parameteriza-

tion, local relaxation and completion. The initial patch-based parameterization and local relaxation they each take about 5 – 7 CPU minutes per frame. Depending on the model around 600 – 900 patches were created on each model and 6 local relaxation iterations were performed. Timing the completion stage is more difficult because the run-time depends on the amount of missing geometry as well as on the hierarchy level. For instance, at the bottom level, the completion process is almost instantaneous, while at the top level where the geometry is propagated to nearly all the frames in the sequence, this process may take up to 5 – 10 minutes. The result figures show the original geometry reconstructed at every frame in gray and the reconstructed geometry in beige. The reconstructed geometry are rendered with circles to follow the cross-parameterization of the frames. The missing geometry is highlighted in red.

5.7.1 Results

The *spheres* dataset is a short 8 frame sequence that illustrates how the analysis mechanism separates the spheres that have been glued together by the per-frame reconstruction, thus restoring the correct topology of the object in all frames. The method does not require that the topology of the reconstructed object is correct in any given frame. The correct topology is inferred from the data from all the frames.

The *hand* dataset is a 64 frame dataset that illustrates both the topology correction and the completion mechanisms. The geometry between the fingers is particularly difficult to reconstruct. The completion mechanism accumulates the available geometry in all frames. The final result, however, will still have missing data in areas where no frame had any data.

The dog *hand-puppet* dataset has 36 frames and shows how the completion algorithm works on real data. As the dog's paws are brought in, the chest of the



Figure 5.10: Capture results for a hand model. Input meshes (top), reconstructed meshes (bottom)

dog has a lot of missing geometry that gets completed with geometry from different frames.

The *two-hands* dataset has 8 frames and shows multiple topological corrections. The topology in the first two frames gets corrected using information from the other frames.

The *T-Shirt* sequence from [BPS⁺08] has 64 frames and it illustrates how the system can be successfully applied to garments. The geometry accumulation mechanism can replace most of the missing geometry without using a template.

5.7.2 Comparisons with Other Methods

Methods that use the template to fill in the missing geometry such as [BPS⁺08] may exhibit artifacts if they are not well registered to the frame geometry. As can be seen in figure 5.14 the geometry looks unnatural near boundaries. Particularly the sleeve boundaries are vertical and the torso boundary is pulled down significantly. The reconstruction that does not use a template, fills in most of the missing geometry, except for a narrow region under the arm where there is not enough

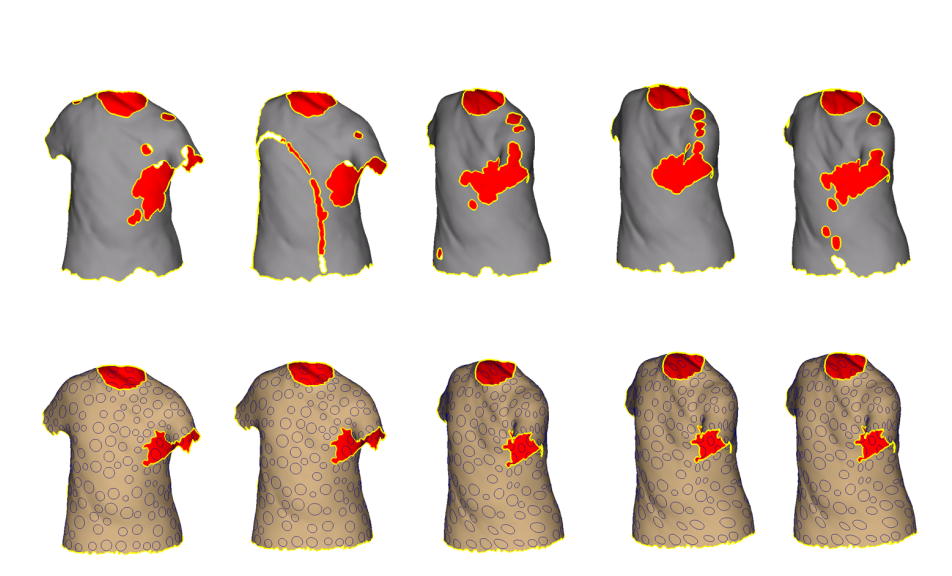


Figure 5.11: Capture results for a T-Shirt model. Input meshes (top), reconstructed meshes (bottom), Missing geometry highlighted in red

information available.

In contrast to tracking methods such as the method of Furukawa and Ponce [FP08, FP09], this algorithm does not need to rely on any frame to extract the topology of the object. Figure 5.2 illustrates how the two incorrectly glued spheres by the per-frame reconstruction are split, thus restoring the correct topology.

Most optimization methods such as Wand et al. [WAO⁺09] perform the optimization in Euclidean coordinates and as a result the final position of the model in a frame does not necessarily fit the initial geometry, resulting in an incorrect registration with the initial geometry. In contrast to these methods the relaxation is performed on the surface of the target geometry, therefore, the final geometry will conform to the target.

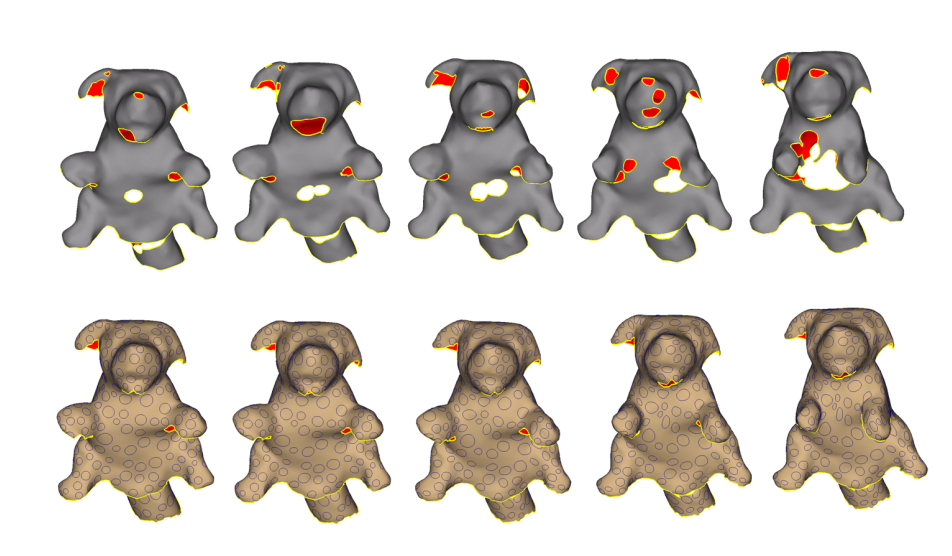


Figure 5.12: Capture results for a hand-puppet model. Input meshes (top), reconstructed meshes (bottom). Missing geometry highlighted in red

5.8 Conclusions, Limitations and Future Work

This chapter presented a method for reconstructing a consistent frame sequence from a sequence of point clouds captured using multiple video streams. This method uses optical flow to guide a local-parameterization-based cross-parameterization method. This approach enables us to accumulate geometric information from all frames consistently using a novel correction and completion mechanism based on stretch analysis of the pairwise mapping. The method does not need a template and it does not require that a frame geometry with correct topology exists anywhere in the sequence. The local relaxation technique optimizes the mapping by restricting the motion of the vertices to the geometry of the next frame. This effectively ensures that the final reconstructed model in a given frame is registered to the initial geometry of that frame.

The relaxation technique optimizes the mapping by moving vertices from the

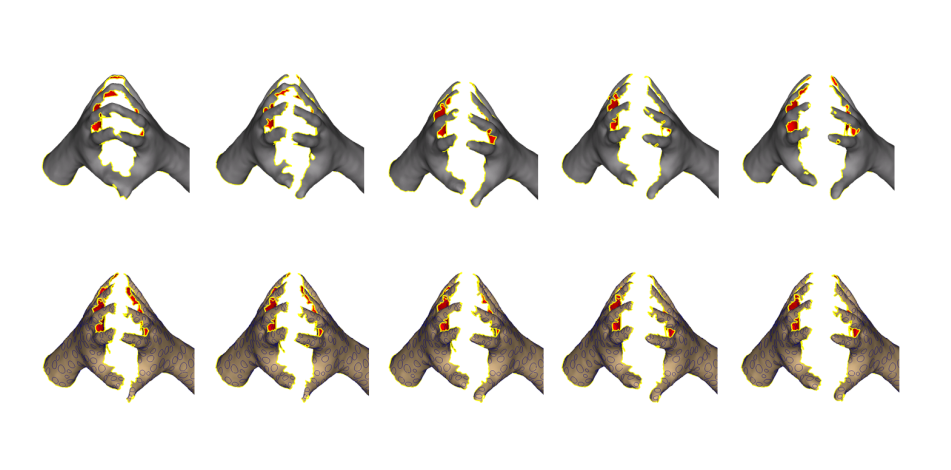


Figure 5.13: Capture results for a real two-hand sequence. Input meshes (top), reconstructed meshes (bottom). Missing geometry highlighted in red

source mesh onto the destination mesh. However, in the presence of holes, the relaxation technique can get stuck when for instance the vertex hits the boundary of a hole on the target mesh, thus yielding a sub-optimal map. This could be improved by temporarily completing the geometry artificially using a smooth membrane [BPS⁺08]. This would give more degrees of freedom to the local optimization procedure yielding a better overall result.

The smooth and gradual quality of motion can be further exploited to improve the initial correspondence. In the higher levels of the hierarchy where a history of the motion is embedded in the sub-sequences, it is possible to use the known trajectory to improve the accuracy of the optical flow vertices, yielding a better cross-parameterization.

Another area of improvement is the managing of the completion strategy in inconsistent regions when both completions are possible. Rather than choosing arbitrarily one of them, a more optimal solution would be to carry them both up the hierarchy and check in a more global way which completion is better.

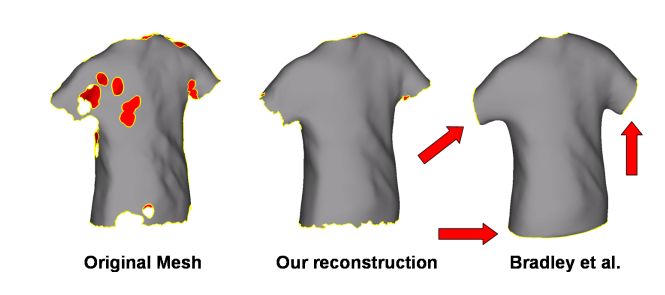


Figure 5.14: Comparison to Bradley et al. Due to misalignment between the frames and the template their method exhibits artifacts around the boundaries

Bibliography

- [ACP02] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 612–619, New York, NY, USA, 2002. ACM.
- [ACP03] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Papers*, pages 587–594, New York, NY, USA, 2003. ACM.
- [AG04] Nizam Anuar and Igor Guskov. Extracting animated meshes with adaptive motion estimation. In *Vision, Modeling, and Visualization*, pages 63–71, 2004.
- [ASK⁺05] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, 2005.
- [ATR⁺08] N. Ahmed, C. Theobalt, C. Roessl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parameterization-free animation reconstruction from video. In *Proc. CVPR*, 2008.
- [BBA⁺07] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale

- capture of facial geometry and motion. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 33, 2007.
- [BBH08] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. CVPR*, 2008.
- [Bou02] Jean Y. Bouguet. Pyramidal implementation of the lucas kanade feature tracker: Description of the algorithm, 2002.
- [BPS⁺08] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Trans. Graph.*, 27(3):99, 2008.
- [CZ] Will Chang and Matthias Zwicker. Automatic registration of articulated shapes. *Computer Graphics Forum (Proc. of SGP '08)*.
- [dAST⁺08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98, 2008.
- [dAZT⁺07] Edilson de Aguiar, Rhaleb Zayer, Christian Theobalt, Magnus Magnor, and Hans-Peter Seidel. Video-driven animation of human body scans. *3DTV*, pages 1–4, 2007.
- [FP08] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture from synchronized video streams. In *Proc. CVPR*, 2008.
- [FP09] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture for human faces. In *Proc. CVPR*, 2009.

- [GILM07] B. Goldlucke, I. Ihrke, C. Linz, and M. Magnor. Weighted minimal hypersurface reconstruction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(7):1194–1208, July 2007.
- [Gra03] Graphite, 2003. <http://www.loria.fr/~levy/Graphite/index.html>.
- [HAWG08] Q.-X. Huang, B. Adams, M. Wiche, and L.-J. Guibas. Non-rigid registration under isometric deformations. *Proc. 6th Eurographics Symposium on Geometry Processing*, 2008.
- [HSKK01] Masaki Hilaga, Yoshihisa Shinagawa, Taku Kohmura, and Toshiyasu L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 203–212, New York, NY, USA, 2001. ACM.
- [KS05] Vladislav Kraevoy and Alla Sheffer. Template-based mesh completion. In *SGP '05: Proceedings of the third Eurographics symposium on Geometry processing*, page 13, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.
- [Lev03] Bruno Levy. Dual domain extrapolation. *ACM Transactions on Graphics (SIGGRAPH conference proceedings)*, 22(3):364–369, Jul 2003. SIGGRAPH 2003 Session : Parameterization.
- [Low99] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.
- [LSA⁺05] Yaron Lipman, Olga Sorkine, Marc Alexa, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Laplacian framework

- for interactive mesh editing. *International Journal of Shape Modeling (IJSM)*, 11(1):43–61, 2005.
- [MFO⁺07] N. J. Mitra, S. Flory, M. Ovsjanikov, N. Gelfand, L. Guibas, and H. Pottmann. Dynamic geometry registration. In *Symposium on Geometry Processing*, pages 173–182, 2007.
- [MG04] Marcus Magnor and Bastian Goldlucke. Spacetime-coherent geometry reconstruction from multiple video streams. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 365–372, Washington, DC, USA, 2004. IEEE Computer Society.
- [PG] Yuri Pekelny and Craig Gotsman. Range scan registration using reduced deformable models. *Computer Graphics Forum (Proceedings of Eurographics 2008)*.
- [PH06] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, 25(3):881–889, 2006.
- [PH08] Sang Il Park and Jessica Hodgins. Data-driven modeling of skin and muscle deformation. *ACM Trans. Graph.*, 27(3):96, 2008.
- [RHHL02] Szymon Rusinkiewicz, Olaf A. Hall-Holt, and Marc Levoy. Real-time 3d model acquisition. In *Siggraph*, pages 438–446. ACM, 2002.
- [SAL⁺08] Andrei Sharf, Dan A. Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. Space-time surface reconstruction using incompressible flow. *ACM Trans. Graph.*, 2008.

- [SAPH04] J. Schreiner, A. Asirvatham, E. Praun, and H. Hoppe. Inter-surface mapping. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 870–877, 2004.
- [SLMB05] Alla Sheffer, Bruno Lévy, Maxim Mogilnitsky, and Alexander Bogomyakov. Abf++: fast and robust angle based flattening. *ACM Trans. Graph.*, 24(2):311–330, 2005.
- [SSK⁺05] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor. Garment motion capture using color-coded patterns. In *Proc. Eurographics*, pages 439–448, 2005.
- [SWG08] Jochen Sussmuth, Marco Winter, and Gunther Greiner. Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum*, 27(5):1469–1476, 2008.
- [TBW⁺09] Art Tevs, Martin Bokeloh, Michael Wand, Andres Schilling, and Hans-Peter Seidel. Isometric registration of ambiguous and partial data. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009)*, page to appear, Miami Beach, Florida, USA, 2009. IEEE Computer Society.
- [VBMP08] Daniel Vlastic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):97, 2008.
- [VZBH08] Kiran Varanasi, Andrei Zaharescu, Edmond Boyer, and Radu P. Horaud. Temporal surface tracking using mesh evolution. In *Proceedings of the Tenth European Conference on Computer Vision*, volume Part

II of *LNCS*, pages 30–43, Marseille, France, October 2008. Springer-Verlag.

- [WAO⁺09] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Trans. Graph.*, 28(2):1–15, 2009.
- [WCF07] R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 34, 2007.
- [WJH⁺07] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. SGP*, pages 49–58, 2007.
- [ZSCS04] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Space-time faces: high resolution capture for modeling and animation. *ACM Trans. Graphics (Proc. SIGGRAPH)*, pages 548–558, 2004.

Chapter 6

Conclusions

This chapter presents in Section 6.1 a summary of the contributions of this thesis and in Section 6.2.1 suggestions for future work, extending the material presented in this thesis.

6.1 Contributions

Modeling and acquisition of deformable objects are important fundamental problems that have wide spread applications in computer graphics, engineering and medical fields. In engineering, for instance, acquisition of deformable objects methods can be used to study the behavior of complex surfaces under stress, such as the wings of an airplane, and can therefore validate physical theories and simulations. In the medical field they can be used to conveniently visualize the mechanics of human organs such as the heart. In game and movie industries they can be used as a substitute for expensive simulations or time-consuming animation sequences.

This thesis explores the problem of creating digital deformable 3D objects using *geometric deformation* and *acquisition* techniques. Geometric deformation techniques [YZX⁺04, SK04a, LSLCO05, ZRKS05, BPGK06, NISA07] allow even novice users to create physically plausible and esthetically pleasing results using intuitive controls placed directly on the surface of the model. In contrast, acquisition techniques [WJH⁺07, WCF07, VBMP08, dAST⁺08, WAO⁺09] can be used to directly capture the dynamic behavior of more complex objects such

as cloth, that would be difficult to model otherwise. Acquisition and deformation techniques are often integrated together. Several acquisition methods use geometric deformation techniques to reconstruct the motion of the geometry through time [PG, VBMP08, dAST⁺08]. And there are deformation methods that can learn the deformation behavior from an existing sequence [SP04].

More specifically, this thesis presents a novel method for geometric deformation (chapter 2), a system for capturing the geometry of moving garments (chapter 3), a method for augmenting the capture data with high-frequency data-driven folds that increase the realism of the captured sequence (chapter 4) and a system for capturing more general deformable objects (chapter 5). The following sections summarize the individual contributions.

6.1.1 Chapter 2 - Material-aware Mesh Deformation

Chapter 2 presents a new approach to model deformation that incorporates anisotropic, non-uniform materials into the geometric deformation framework. This method combines the efficiency, generality and control of geometric methods with material awareness previously found only in physics-based and skeleton-based methods. Using this tool, a novice user can generate realistic looking, material-aware deformations at interactive rates relying only on a sparse set of control points. The material properties can be specified by the user using a simple and intuitive interface or can be inferred from a set of sample poses. This method also allows for a combination of the two where the user can override the data-driven material properties in specific areas. The resulting deformations are as-rigid-as-possible, subject to the material stiffness and the user defined transformations, and therefore preserve the geometric detail of the object. The formulation is simple and efficient. It requires solving only two sparse linear systems, and thus works at interactive rates

for models up to 80k vertices.

A popular modeling paradigm used by several deformation methods, including this one, is to specify rotational and positional constraints on a set of anchors on the surface [YZX⁺04, LSLCO05]. This enables the user to have simple intuitive control over the deformation. An even simpler interface would be one where the user specifies only positional constraints for the anchors. However, methods that rely on positional constraints alone typically require a non-linear formulation [SK04b, BPGK06] which makes them less efficient. This method also provides an interface that uses only positional constraints, which is a more intuitive interface for the user. Using the positional constraints specified, the method automatically computes local rotations that conserve global criteria such as total volume or total area without increasing the asymptotic complexity of the algorithm. The choice of global preservation criteria adds an additional level of flexibility that provides the user with more control over the final result.

6.1.2 Chapter 3 - Markerless Garment Capture

Chapter 3 presents a method for markerless capture (or acquisition) of full garments. Since this method does not require special processing of the captured garments one could rapidly capture their dynamic behavior. Unlike physical simulations of garment that would require material-specific parameters, this garment capture method is independent of the material of the garment and therefore one could easily capture garments made out of a variety of different materials including cotton, fleece, Gore-Tex and nylon. The method was applied to a large and diverse set of garments such as T-Shirts, dresses and jackets.

The pipeline can be divided into two stages. An *acquisition* stage that reconstructs the complex, changing structure of garments during human motion in every

frame, and a *geometric processing* stage that performs geometric completion and computes the motion of the garment through time. The initial geometries provided by the acquisition and reconstruction stage typically have large areas of missing data due to self occlusion. Since these frames are reconstructed independently the motion of the garment is unknown. Reconstructing the motion of the garment is equivalent to finding a cross-parameterization between the geometries at different time frames. Computing the motion of the garment is critical for many applications such as texture mapping that require finding the location of every point on the garment in each individual frame.

The geometric processing stage performs both consistent cross-parameterization of the frames in the sequence and geometry completion. It leverages fabric incompressibility and garment topology to generate realistic garment geometry and motion faithful to the input. Thus a dense correspondences between the meshes in the sequences is obtained. The completion step allows us to fill in the missing geometry from a template. The system needs to perform many operations on large sequences of hundreds of frames. This can be a time consuming task. However, the pipeline is parallelizable, thus enabling us to efficiently process a large number of frames.

6.1.3 Chapter 4 - Wrinkling Captured Garments Using Space-time Data-driven Deformation

One of the defining characteristics of cloth under motion is the presence of dynamic complex folds. Although they are highly noticeable, these folds tend to be very shallow, making them difficult to capture and separate from noise. Thus, on the one hand the appearance of folds consistent with the garment motion are very important for the realism of the captured garments. But on the other hand, due to their specific

structure, folds are difficult to capture. Hence, this thesis proposes a hybrid two pass *capture* and *modeling* solution to ensure the presence of folds in the captured output.

Chapter 4 presents for the first time a method for modeling data-driven, dynamic fine-folds into a captured sequence. First the folds are detected in the video sequences and then apply them to the existing captured geometries using a novel space-time deformation scheme. As demonstrated, this method works robustly on numerous input examples that are generated by very different capture methodologies. This algorithm does not aim at exact reproduction of the fold shapes observed in the video footage, instead it generates a large network of believable folds which greatly enhance the realism of the capture.

6.1.4 Chapter 5 - Globally Consistent Space-Time Reconstruction

Chapter 5 presents a general surface reconstruction method that can be applied to a wide range of objects. The foundation of this method is the observation that the shape of an object in motion *changes gradually* and therefore the cross-parameterization (correspondence) between consecutive frames should exhibit very little stretch, as the intrinsic surface distances change very little with a small change in time. This observation forms a strong yet very general prior.

The method starts by reconstructing individual frame geometry. Then it proceeds to assemble the individual frames into a consistent space-time frame sequence, preserving the correct topology of the captured objects and simultaneously completing missing information across time and correcting local inconsistencies. By solving the cross-parameterization and completion problems simultaneously in a global setup, it is possible to better preserve the geometric detail and more robustly handle noise and inconsistencies in the data without the need of a pre-

computed template geometry or other stronger priors.

6.2 Future Work

Despite the many recent advances in the acquisition of time varying surfaces, there are still many challenges ahead and many more applications to address. The following sections briefly address some possible new directions.

6.2.1 Quantitative Analysis and Validation

Most current methods for dynamic surface reconstruction provide visual evaluations of their methods by showing images or video of the real and captured data. This qualitative analysis is often sufficient for applications in film and game industries where visual quality is the ultimate goal, but it is not appropriate for many applications in CAD and other engineering disciplines where the accuracy of the measurement and formal error analysis are important. For material engineers, for instance, capturing the accurate behavior of a material under various types of stress can be a very useful tool in validating theoretical physical models and simulations. Therefore, methods for quantitative evaluation of reconstruction methods should be developed.

For static 3D reconstruction methods there are standard, industry-adopted, methodologies to measure the error as well a growing database of testing models. Midlbury database [SCD⁺06], for instance, provides benchmarking tools for stereographic reconstruction for 3D models based on several criteria such as efficiency and accuracy. However, there are no such tools for 3D dynamic surface reconstruction. As techniques for dynamic 3D reconstruction continue to emerge, the need will increase for standard qualitative ways of comparing the results.

Such quantitative error analysis for dynamic surface reconstruction can be challenging since typically there is no ground truth. Assuming that one would like to evaluate the accuracy of a video based 3D capture system, one could perhaps simulate ground truth by rendering a 3D model into a virtual environment from a number of synthetic video cameras and use those images as input. Using light-stage technology [DWT⁺02, CEJ⁺06] and a high-quality renderer it could be possible to simulate very accurately the lighting condition and the images of the real setup. These images could be fed through the capturing pipeline and the existing result could be compared on a frame-by-frame basis to the original. Using this approach, essentially all aspects of the capture system could be tested systematically to determine error bounds and error correlations. This approach would be relatively easy to try as it does not require any expensive hardware setup.

6.2.2 Real-time Dynamic Surface Reconstruction

One of the biggest challenges of 3D dynamic surface reconstruction is accurate real-time reconstruction. This can be particularly useful in the medical field for diagnosis as well as during surgeries or other interventions.

Although there is some work in real-time reconstruction of deformable surfaces such as real-time scanning devices [ZH06], or depth cameras [Inc09b], these devices are currently either limited to low resolution data, or do not reconstruct simultaneously both the geometry and the motion of the object, or they exhibit a narrow capture space.

An important observation that might facilitate a more efficient dynamic surface reconstruction techniques is that the motion of a dynamic object is generally faster than the change in the intrinsic shape of an object. Several methods for synthesising human motion used this observation to decouple the articulated motion of a char-

6.2. Future Work

acter from the high-frequency local detail [ASK⁺05]. This decoupling technique could be also used for dynamic surface reconstruction.

A step toward real-time dynamic surface reconstruction is a hybrid system that can combine a fast motion capture system that acquires the fast changing pose of a character with a surface detail reconstruction system that acquires the slower changing local shape. The fast motion capture can track the motion in every frame while the surface detail reconstruction module would correct surface details only in a subset of the frames for a speed-quality trade-off. With the emerging of mainstream low cost optical systems such as the new XBox [Inc09a] which can track the pose of a real life character in real time, this type of approach can be much easily realized than before.

Bibliography

- [ASK⁺05] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416, 2005.
- [BPGK06] Mario Botsch, Mark Pauly, Markus Gross, and Leif Kobbelt. Primo: coupled prisms for intuitive surface modeling. In *SGP '06: Proceedings of the fourth Eurographics symposium on Geometry processing*, pages 11–20, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
- [CEJ⁺06] Charles-Félix Chabert, Per Einarsson, Andrew Jones, Bruce Lamond, Wan-Chun Ma, Sebastian Sylwan, Tim Hawkins, and Paul Debevec. Relighting human locomotion with flowed reflectance fields. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 76, New York, NY, USA, 2006. ACM.
- [dAST⁺08] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Trans. Graph.*, 27(3):98, 2008.
- [DWT⁺02] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. A lighting reproduction approach to live-action compositing. In *SIGGRAPH '02: Proceedings of the 29th*

- annual conference on Computer graphics and interactive techniques*, pages 547–556, New York, NY, USA, 2002. ACM.
- [Inc09a] Microsoft Inc. Project natal. <http://www.xbox.com/en-ca/live/projectnatal/>, 2009.
- [Inc09b] Ppoint Grey Inc. Point grey. <http://www.ptgrey.com>, 2009.
- [LSLCO05] Yaron Lipman, Olga Sorkine, David Levin, and Daniel Cohen-Or. Linear rotation-invariant coordinates for meshes. In *Proceedings of ACM SIGGRAPH 2005*, pages 479–487. ACM Press, 2005.
- [NISA07] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. FiberMesh: Designing freeform surfaces with 3D curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 26(3):article no. 41, 2007.
- [PG] Yuri Pekelny and Craig Gotsman. Range scan registration using reduced deformable models. *Computer Graphics Forum (Proceedings of Eurographics 2008)*.
- [SCD⁺06] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR '06: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society.
- [SK04a] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT '04: Proceedings of the 3D Data*

- Processing, Visualization, and Transmission, 2nd International Symposium*, pages 68–75, Washington, DC, USA, 2004. IEEE Computer Society.
- [SK04b] Alla Sheffer and Vladislav Kraevoy. Pyramid coordinates for morphing and deformation. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 68–75, Washington, DC, USA, 2004. IEEE Computer Society.
- [SP04] Robert W. Sumner and Jovan Popovic. Deformation transfer for triangle meshes. *ACM Trans. Graph.*, 23(3):399–405, 2004.
- [VBMP08] Daniel Vlastic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):97, 2008.
- [WAO⁺09] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Trans. Graph.*, 28(2):1–15, 2009.
- [WCF07] R. White, K. Crane, and D. Forsyth. Capturing and animating occluded cloth. *ACM Trans. Graphics (Proc. SIGGRAPH)*, page 34, 2007.
- [WJH⁺07] M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. SGP*, pages 49–58, 2007.

- [YZX⁺04] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Mesh editing with poisson-based gradient field manipulation. *ACM Trans. Graph.*, 23(3):644–651, 2004.
- [ZH06] Song Zhang and Peisen S. Huang. High-resolution, real-time three-dimensional shape measurement. *Optical Engineering*, 45(12):123601, 2006.
- [ZRKS05] Rhaleb Zayer, Christian Rössl, Zachi Karni, and Hans-Peter Seidel. Harmonic guidance for surface deformation. In *Computer Graphics Forum, Proceedings of Eurographics 2005*, pages 601–609, Dublin, Ireland, 2005. Eurographics, Blackwell.

Appendix A

Linear Combinations of Transformations

Geometric transformations are typically represented as square matrices. Matrix multiplication is used to compose and apply the transformations. This representation has two key shortcomings:

- Rotation transformations cannot be interpolated by interpolating the matrix elements.
- Matrix multiplication is not commutative.

Both of these properties are crucial for us in order to propagate and later decompose anchor triangles.

To deal with these issues a number of interpolation methods for rotations have been developed over the years. When dealing with rotations there are three desired properties: torque-minimization, constant speed, and commutativity. Currently no interpolation method exhibiting all three properties exists. SLERP [Sho85] exhibits constant speed and minimal-torque. LERP, popularized by Casey Muratori, is commutative and minimal-torque. The exponential map interpolation [Gra98] is commutative and constant speed.

Alexa [Ale02] extended the exponential map interpolation method into a commutative algebra of (almost) general transformations that supports matrix blending

and interpolation. This algebra is commutative and interpolates transformations with constant speed. Furthermore, it is not limited to rotations only, thus simplifying the task of dealing with combinations of rotations and scales.

We have chosen to use the later method since it answers both our requirements. We now give a brief overview of the blending operators defined in this algebra.

By defining two new operations denoted by \oplus and by \odot (corresponding to matrix addition and scalar multiplication), the blending of transformations T_a with weights ω_a becomes $\oplus \omega_a \odot T_a$.

The two operators are based on matrix *exp* and *log* operators defined as follows:

$$\exp(A) = \sum_{k=0}^{\infty} \frac{A^k}{k!},$$

$$A = \log(X) \Leftrightarrow \exp(A) = X.$$

Alexa [Ale02] shows that this sum is well defined and closed for 3x3 rotation matrices and non-uniform scales under some minimal conditions. The blending formula is defined as follows:

$$\bigoplus_{a=1}^k \omega_i^a \odot T_a = \exp\left(\sum_{a=1}^k \omega_i^a \log(T_a)\right).$$

More details as well as numerical methods to compute these operations are found in [Ale02].

Bibliography

- [Ale02] Marc Alexa. Linear combination of transformations. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 380–387, New York, NY, USA, 2002. ACM Press.
- [Gra98] F. Sebastin Grassia. Practical parameterization of rotations using the exponential map. *J. Graph. Tools*, 3(3):29–48, 1998.
- [Sho85] Ken Shoemake. Animating rotation with quaternion curves. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, New York, NY, USA, 1985. ACM Press.

Appendix B

Markerless Garment Capture - Boundary Ellipse Fitting

Chapter 3 presents a method for markerless garment capture. It reconstructs the geometry of the moving garment from a set of video sequences. First, the acquisition and reconstruction methods generate per-frame meshes that capture the complex geometry of garments during motion. Next, the geometric processing stage performs both consistent cross-parameterization of the frames in the sequence as well as geometric completion from a template. The consistent parameterization step relies on a sparse set of off-surface anchors as initial correspondences. These off-surface anchors in our setup correspond to the natural boundaries of the processed garment. For example, for a T-Shirt the natural boundaries are the boundaries around the neck, sleeves and waist. These off-surface anchors are placed at the centers of a natural boundary at some constant distance away from it. Figure B.1 shows the motion of the off-surface anchors for a short sequence. This Appendix describes in detail our technique to determine the center of the natural boundary by tracking the motion of the boundary using an ellipse fitting technique. This can be done automatically for most cases (Section B.1). For the few cases where the automatic algorithm may fail a semi-automatic algorithm was developed (Section B.2). A short discussion about more general methods to track boundaries is presented in Section B.3.

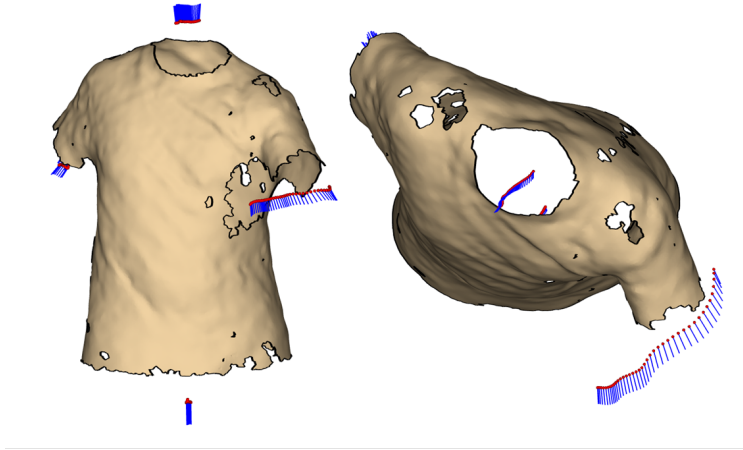


Figure B.1: Motion of the off-surface anchors

B.1 Automatic Ellipse Fitting

Since most garment boundaries are elliptical, the centers of the natural boundaries are determined by finding the ellipse that best fits the boundary as illustrated in figure B.2. This problem is challenging because the boundaries of the reconstructed geometry of a given frame can be incomplete. For instance, in the case of the sleeve boundary of the T-Shirt in Figure B.2, the entire under-arm area is missing due to self occlusion. The reconstruction at the extremities of the garment is the least reliable, therefore the geometric boundary will not match our expectation of the true garment boundary and can have arbitrary shape. Naively choosing the ellipse that best fits all points on the boundary in the least-squares sense would not work. Another solution would be to select an ellipse that fits the most points on the boundary. Although this solution would likely perform better than the first, it is still not robust enough and would still fail in the cases where large chunks are missing. For example, in figure B.3, the red ellipse provides the best fit the left sleeve boundary, but the green ellipse fits the most points.

To facilitate a better fitting, one observation is that the motion of the garment is gradual and continuous therefore, the boundary ellipses should remain close between consecutive frames. Hence, it is possible to use the ellipse from the previous frame to guide the search for an ellipse in the current frame. Based on this observation this appendix presents a robust algorithm that matches an ellipse only to the appropriate part of the boundary. For the first frame, the method computes a set of ellipse candidates (section x) and the user selects from it the ellipse that best fits the real boundary. The user assistance is necessary since the algorithm does not know which points are part of the “real” boundary and which ones are not. In most cases this is the only user intervention required for an entire sequence and it typically only takes a few seconds. For each subsequent frame the algorithm first generates a set of ellipse candidates that fit parts of the boundary. From this set the algorithm prunes the ellipses which are very different in *location*, *size* and *shape* from the ellipse fitted to the previous frame. After the pruning process, all remaining ellipse candidates should be reasonable selections. Among these ellipses the method chooses the one that fit the most points on the boundary. The following section describe these steps in more detail.

B.1.1 Ellipse Candidate List

Ellipses are a special type of conic section and are strictly determined by five planar non-degenerate points. A non-degenerate set of points is a set of points where no three points from the are collinear. Therefore, one way to generate a candidate list is to find the set of ellipses that interpolate all non-degenerate five-tuples of points. This means that on a boundary of n points one would need to check $\binom{n}{5}$ ellipses in order to find the best match, which for our data is typically on the order of 10^9 . Searching through these many ellipses is impractical, therefore a randomized ap-

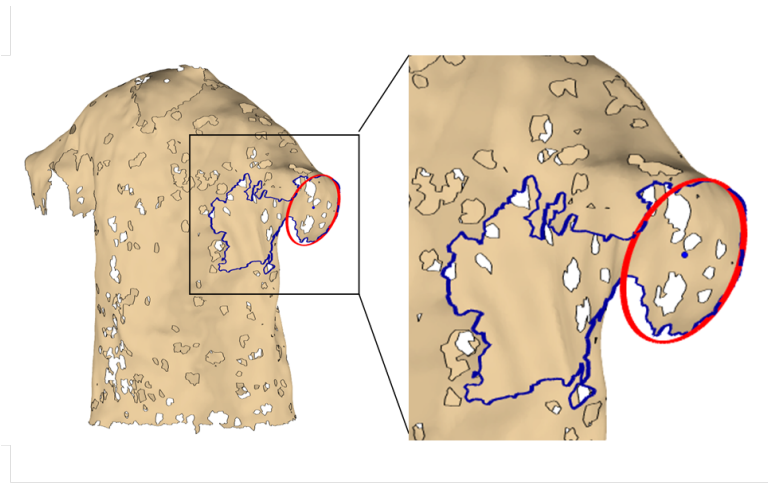


Figure B.2: Ellipse that fits the garment sleeve boundary (red)

proach is used. A variant of the RANSAC algorithm[Fischler and Bolles 1981] is used and select random tuples of points to obtain our candidate list. Although randomly selecting five points for each ellipse is in theory enough, due to degeneracies in the data such as points being very close to one another or collinear, it is more robust to generate ellipses by selecting one extra point. This tends to be sufficient to eliminate degeneracies while slightly increasing the computational cost. Given 6 randomly selected boundary points, we compute the ellipse candidate by first projecting all points on their best fitting plane and then the WildMagic library is used to analytically compute the best fit ellipse. Typically between 50k-100k RANSAC iterations are needed to prepare a good list of candidates, which is several order of magnitude less than the exhaustive approach.

B.1.2 Pruning Criteria

First, already when we random points are selected on the boundary to build the ellipse candidate list, only sample points which are within a radius R of the center

B.1. Automatic Ellipse Fitting

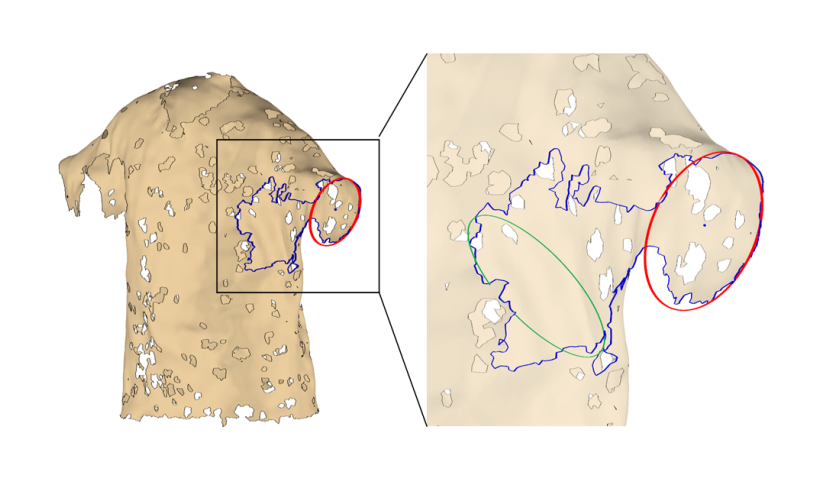


Figure B.3: TShirt boundary. Green ellipse: the ellipse that fits the most points on the *geometric boundary*. Red ellipse: ellipse that best fits the *garment boundary*

of the reference ellipse are used. The radius R was chosen to be twice the largest of the radii of the reference ellipse. This dramatically reduces the space of candidates especially in the cases where large chunks of the real boundary are missing and hence the geometric boundary is very long. Second, several pruning criteria are used to remove ellipses from the list that are too far away from the *reference* ellipse - that is, the ellipse selected in the previous frame. Since the motion and shape change of the garments are gradual, the ellipses that are significantly different in *location*, *orientation* and *size* from the reference ellipse are pruned. For location pruning, we enforce the center of the ellipse to be within epsilon distance from the center of the reference ellipse. This epsilon depends in general on the speed of the motion. For all our examples an epsilon of 10cm is used. For orientation pruning, the normal of the supporting plane of the ellipse cannot differ by more than 30 degrees. For size pruning both the radii and perimeter of the ellipse has to be within an epsilon range from those of reference ellipse.

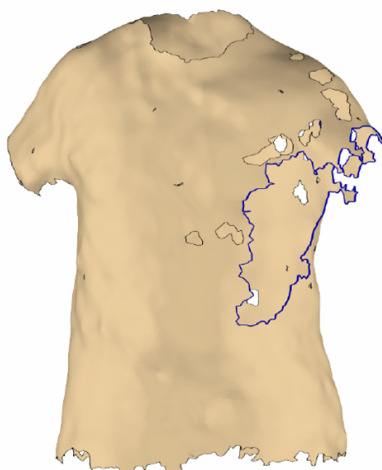


Figure B.4: Example of extreme degenerate boundary where most garment sleeve boundary is missing

B.2 Semi-automatic Off-surface Anchor Placement

Occasionally there are frames where a boundary is so degenerate (e.g. Figure B.4 due to occlusion and the relatively small number of cameras, that the real garment boundary is barely identifiable even for a human eye. In such cases the candidate list after pruning can be empty and the algorithm would not find any good candidates. When this happens, the algorithm “skips” and process the following frames keeping as a reference the last good ellipse found. If this phenomenon happens in a small number of frames, the algorithm recovers by computing the ellipses for the missing frames using interpolation. However if there is a block of many frames that have such extreme degeneracies the algorithm may not recover. For sequences where the automatic anchor placement fails for a larger number of consecutive frames, a semi-automatic alternative was developed, where for a number of key-frames, the user chooses an ellipse for each boundary from a selection of high-confidence ellipses determined by the RANSAC algorithm, very much like

the procedure in the first frame of the automatic case. The off-surface anchors are then computed by interpolation using these key frames. The required density of key-frames depends on the speed of motion; for fast motions, a key-frame approximately every 30 frames (1/2 second) yields good results.

B.3 More General Boundary Tracking

The assumption of elliptical boundary does not always hold. Some types of garments such as dresses may have non-elliptical boundaries especially at the neck. The sleeve boundaries, even if generally elliptical in their rest pose may become non-elliptical due to gravity forces. Although the current boundary tracking method is robust enough for many of such cases, an interesting line of future work is to explore more generic fitting techniques that do not rely on a specific boundary shape. One idea is to fit a generic spline to the boundary of the previous frame and use that spline instead of an ellipse to find the boundary of the current frame.