

RECONSTRUCTION, ANALYSIS, AND EDITING OF DYNAMICALLY DEFORMING  
3D-SURFACES

Von der  
Carl-Friedrich-Gauß-Fakultät  
der Technischen Universität Carolo-Wilhelmina zu Braunschweig

zur Erlangung des Grades eines  
**Doktor-Ingenieurs (Dr.-Ing.)**

genehmigte Dissertation

von  
Thomas Neumann  
geboren am 12. September 1983  
in Aschersleben

Eingereicht am: 13. Juli 2016

Disputation am: 07. Oktober 2016

1. Referent: Prof. Dr.-Ing. Marcus Magnor

2. Referent: Prof. Dr.-Ing. Christian Theobalt

(2016)



---

## Abstract

Dynamically deforming 3D surfaces play a major role in computer graphics. However, producing time-varying dynamic geometry at ever increasing detail is a time-consuming and costly process, and so a recent trend is to capture geometry data directly from the real world. In the first part of this thesis, I propose novel approaches for this research area. These approaches capture dense dynamic 3D surfaces from multi-camera systems in a particularly robust and accurate way. This provides highly realistic dynamic surface models for phenomena like moving garments and bulging muscles.

However, re-using, editing, or otherwise analyzing dynamic 3D surface data is not yet conveniently possible. To close this gap, the second part of this dissertation develops novel data-driven modeling and animation approaches. I first show a supervised data-driven approach for modeling human muscle deformations that scales to huge datasets and provides fine-scale, anatomically realistic deformations at high quality not attainable by previous methods. I then extend data-driven modeling to the unsupervised case, providing editing tools for a wider set of input data ranging from facial performance capture and full-body motion to muscle and cloth deformation. To this end, I introduce the concepts of sparsity and locality within a mathematical optimization framework. I also explore these concepts for constructing shape-aware functions that are useful for static geometry processing, registration, and localized editing.

---

## Zusammenfassung

Dynamisch deformierbare 3D-Oberflächen spielen in der Computergrafik eine zentrale Rolle. Die Erstellung der für Computergrafik-Anwendungen benötigten, hochaufgelösten und zeitlich veränderlichen Oberflächengeometrien ist allerdings äußerst arbeitsintensiv. Aus dieser Problematik heraus hat sich der Trend entwickelt, Oberflächendaten direkt aus Aufnahmen der echten Welt zu erfassen. Dazu nötige 3D-Rekonstruktionsverfahren werden im ersten Teil der Arbeit entwickelt. Die vorgestellten, neuartigen Verfahren erlauben die Erfassung dynamischer 3D-Oberflächen aus Mehrkamera-Aufnahmen bei hoher Verlässlichkeit und Präzision. Auf diese Weise können detaillierte Oberflächenmodelle von Phänomenen wie in Bewegung befindliche Kleidung oder sich anspannende Muskeln erfasst werden.

Aber auch die Wiederverwendung, Bearbeitung und Analyse derlei gewonnener 3D-Oberflächendaten ist aktuell noch nicht auf eine einfache Art und Weise möglich. Um diese Lücke zu schließen beschäftigt sich der zweite Teil der Arbeit mit der datengetriebenen Modellierung und Animation. Zunächst wird ein Ansatz für das überwachte Lernen menschlicher Muskel-Deformationen vorgestellt. Dieses neuartige Verfahren ermöglicht eine datengetriebene Modellierung mit besonders umfangreichen Datensätzen und liefert anatomisch-realistische Deformationseffekte. Es übertrifft damit die Genauigkeit früherer Methoden. Im nächsten Teil beschäftigt sich die Dissertation mit dem unüberwachten Lernen aus 3D-Oberflächendaten. Es werden neuartige Werkzeuge vorgestellt, die eine weitreichende Menge an Eingabedaten verarbeiten können, von aufgenommenen Gesichtsanimationen über Ganzkörperbewegungen bis hin zu Muskel- und Kleidungsdeformationen. Um diese Anwendungsbreite zu erreichen stützt sich die Arbeit auf die allgemeinen Konzepte der Spärlichkeit und Lokalität und bettet diese in einen mathematischen Optimierungsansatz ein. Abschließend zeigt die vorliegende Arbeit, wie diese Konzepte auch für die Konstruktion von oberflächen-adaptiven Basisfunktionen übertragen werden können. Dadurch können Anwendungen für die Verarbeitung, Registrierung und Bearbeitung statischer Oberflächenmodelle erschlossen werden.

---

## Acknowledgments

I would like to express my sincere thanks to Markus Wacker for his constant support, his unshakeable trust, and for being such an inexhaustibly enthusiastic and energetic person. Markus first encouraged me to start looking into computer graphics research and, from then on, guided me through many interesting research projects. He taught me how to present and explain my work in a clear, inspiring manner and how to approach difficult mathematical problems. Without his guidance and compassion, this work would truly not have been possible. Thank you, Markus.

I am also greatly indebted to my advisor Marcus Magnor for trusting in me, for his friendly support, and for providing invaluable feedback and constructive criticism during each step of this work. Marcus also helped me maneuver the numerous bureaucratic hurdles in order to make this dissertation possible. He gave me the opportunity to work in a wonderful environment at the Computer Graphics Lab at TU Braunschweig for two very productive years.

Very special thanks go to Christian Theobalt. Christian taught me how to position my work in the research community, how to choose the right experiments and the right words to defend my ideas in a paper, and how to prioritize in face of an approaching deadline. In inspiring discussions, Christian shared his immense knowledge about computer vision and computer graphics with me, which often led to exciting and prosperous ideas.

Kiran Varanasi had a major influence on shaping the ideas, methods, and results I present in my publications. Without his technical guidance, the illuminating discussions with him, and his valuable and oftentimes refreshingly critical feedback this work would not have been possible. Kiran has been a very enjoyable person to work with and a great friend. Thank you, Kiran.

During my time as a Ph.D. student, I had the great opportunity to work in three different research groups, where I met many inspiring and bright people. In particular, I would like to mention Stephan Wenger, Lorenz Branz, and Christian Lipski who I thank for the great collaboration on interesting research projects. I also thank Felix Klose, Kai Ruhl, Stefan John, Benjamin Hell, Stefan Guthe and Martin Eisemann for insightful scientific discussions and valuable feedback on my research. Very importantly, I have to thank Carsten Götze for providing and maintaining the computational resources that made it possible for me to process the huge datasets that I captured. I shall also thank Nils Hasler and Kwang In Kim, who provided invaluable methodological and mathematical input during my stay at the Max-Planck-Institute. All the former and present members of the DreMatrix group at HTW Dresden also deserve my gratitude, in particular Jens Friedrich for countless brainstorming sessions and for his help in implementing software for the industrial collaboration projects I took part in. Thanks to Michael Wegner, Jens Friedrich, and Stefanie Gassel for proofreading (naturally, any

---

remaining errors are solely my own responsibility). Moreover, I have to thank all the participants who were willing to wear strange-looking suits so that I could capture them in the multi-camera lab. Further thanks go to Franz Rott and Jochen Süßmuth at adidas for the pleasant and exciting collaboration. Very special thanks go to Georg Freitag for being both a humorous and motivating colleague as well as a sincere and loyal friend.

I am deeply grateful to my family for their support, their understanding, and their encouragement. I thank my father for demonstrating, not only with his dry stone wall in his garden but first and foremost with his kind and patient personality, that there are certain projects that one has to reserve just a little more time for in order to be able to finish them with grace. Finally, I would like to express my sincere gratitude to my partner Theresa for enduring my sometimes grumpy mood during way too long days of work, for chilling me down in phases of stress and pressure, for her friendship, and for her loving support.

# Contents

|  |             |
|--|-------------|
| <b>Preface</b>   | <b>ix</b>   |
| <b>Notation</b>  | <b>xiii</b> |
| <b>1. Introduction</b>   | <b>1</b>    |
| <b>2. Reconstructing Dynamically Deforming 3D Surfaces</b>                           | <b>5</b>    |
| 2.1. Related Work . . . . .  | 6           |
| 2.2. Robust Garment Capturing with a Color-coded Quad Pattern . . . . .              | 14          |
| 2.2.1. Robust Quad Center Detection . . . . .  | 15          |
| 2.2.2. Robust Color Classification . . . . .   | 17          |
| 2.2.3. M-Array Matching . . . . .  | 21          |
| 2.2.4. Results . . . . .   | 21          |
| 2.3. 3D Reconstruction using Random Dot Patterns and Multiview Constraints . . . . . | 24          |
| 2.3.1. Prelude: Stereo Matching using Graph Matching . . . . .                       | 25          |
| 2.3.2. Multiview Constraints . . . . .   | 28          |
| 2.3.3. Optimization with Multiview Constraints . . . . .                             | 30          |
| 2.3.4. Robust Registration across Large Motions . . . . .                            | 35          |
| 2.3.5. Results . . . . .   | 36          |
| 2.4. A first use case: Deformation and Strain Analysis . . . . .                     | 44          |
| 2.4.1. From 3D Reconstructions to Principal Deformations . . . . .                   | 44          |
| 2.4.2. Results . . . . .   | 48          |
| <b>3. Data-Driven Animation of Deforming Muscles</b>                                 | <b>49</b>   |
| 3.1. Related Work . . . . .  | 50          |
| 3.2. Method . . . . .  | 52          |
| 3.2.1. Experiment Setup . . . . .  | 54          |
| 3.2.2. Multiview 3D Reconstruction and Registration . . . . .                        | 56          |
| 3.2.3. Shape Parameterization . . . . .  | 59          |
| 3.2.4. Deformation Learning . . . . .  | 64          |

|  |            |
|--|------------|
| 3.3. Results . . . . .   | 68         |
| 3.4. Discussion . . . . .  | 71         |
| <b>4. Sparse Localized Deformation Components</b>                          | <b>73</b>  |
| 4.1. Related Work . . . . .  | 74         |
| 4.2. Method . . . . .  | 77         |
| 4.2.1. Sparse PCA for mesh sequence decomposition . . . . .                | 78         |
| 4.2.2. Local Support . . . . .   | 80         |
| 4.2.3. Optimization Algorithm . . . . .                                    | 81         |
| 4.3. Results . . . . .   | 85         |
| 4.3.1. Evaluation of Optimization Algorithm . . . . .                      | 85         |
| 4.3.2. Quantitative Evaluation . . . . .                                   | 87         |
| 4.3.3. Facial performances . . . . .                                       | 89         |
| 4.3.4. Muscle deformations . . . . .                                       | 94         |
| 4.3.5. Cloth deformations . . . . .  | 96         |
| 4.3.6. Statistical shape modeling, processing, and visualisation . . . . . | 97         |
| 4.4. Discussion . . . . .  | 98         |
| <b>5. Compressed Manifold Modes for Mesh Processing</b>                    | <b>101</b> |
| 5.1. Related Work . . . . .  | 103        |
| 5.2. Method . . . . .  | 105        |
| 5.2.1. Discretization . . . . .  | 105        |
| 5.2.2. Reformulation using ADMM . . . . .                                  | 106        |
| 5.2.3. Numerical Algorithm . . . . .                                       | 107        |
| 5.2.4. Convergence . . . . .   | 109        |
| 5.3. Results . . . . .   | 109        |
| 5.3.1. Evaluation of the Optimization Algorithm . . . . .                  | 110        |
| 5.3.2. Properties of Compressed Manifold Modes . . . . .                   | 112        |
| 5.3.3. Applications in Shape Matching . . . . .                            | 116        |
| 5.3.4. Localized Editing . . . . .   | 120        |
| 5.4. Discussion . . . . .  | 122        |
| <b>6. Summary and Future Work</b>  | <b>125</b> |
| <b>A. Motion Protocol for Capturing Arm Muscle Deformations</b>            | <b>131</b> |
| <b>Bibliography</b>  | <b>135</b> |

# Preface

This dissertation is largely based on publications that I have presented at and published in peer-reviewed conferences and journals. My dissertation combines these publications under the common theme of reconstructing and editing dynamically deforming 3D surfaces. The text includes figures, plots, data, and text passages from my published work. It complements these with additional results, more in-depth discussions of the limitations as well as more detailed derivations and explanations of the involved math, theory, and algorithms. My advisors Marcus Magnor, Markus Wacker, and Christian Theobalt co-authored all my publications, providing valuable advice and ideas along the way. In the following, I clarify my individual contributions to each chapter of this thesis and relate them to my corresponding publications.

**Chapter 2: Reconstructing Dynamically Deforming 3D Surfaces** describes two different approaches for reconstructing dynamically deforming 3D surfaces. The garment capturing system described in Section 2.2 was the basis for a complete multiview capture system that I developed for adidas AG. It also is the foundation for a shoe last scanning system that I implemented with Jens Friedrich for adidas AG. Furthermore, HTW Dresden actively uses the system for calibrating their multi-camera lab. The system was predominantly developed by me. Jens Friedrich helped with the implementation of the Auto-Detect Feature and assisted in GPU parallelization. Lorenz Rogge programmed the initial version of the pattern generator.

A second 3D reconstruction method is described in Section 2.3. The main idea behind it was first presented as a Poster at SIGGRAPH [NWV+12] as:

T. Neumann, M. Wacker, K. Varanasi, C. Theobalt, and M. Magnor. „High Detail Marker based 3D Reconstruction by Enforcing Multiview Constraints“. In: *ACM SIGGRAPH 2012 Posters*. 2012, p. 59.

The poster abstract [NWV+12] was co-written by Kiran Varanasi, Marcus Magnor, Christian Theobalt, and Markus Wacker. The initial idea, implementation, and evaluation are my work. The method is the basis for the 3D reconstruction part in my later publication [NVH+13], but the core of it was never fully described nor fully evaluated in either of these publications since this was not their

focus. After publication of both works, I was able to significantly improve upon [NWV+12]. I plan to publish these new results in the near future.

**Chapter 3: Data-Driven Animation of Deforming Muscles** is based on the publication

T. Neumann, K. Varanasi, N. Hasler, M. Wacker, M. Magnor, and C. Theobalt. „Capture and statistical modeling of arm-muscle deformations“. In: *Computer Graphics Forum (Proc. of Eurographics)* 32.2 (2013), pp. 285–294.

Kiran Varanasi and Christian Theobalt both provided valuable insights, advice, and discussion over the course of the project. Nils Hasler helped developing the theory behind the shape parameterization in Section 3.2.3. Discussions with Stefan John helped me find the correct method for non-rigid registration. Kwang-In Kim provided valuable feedback and code review regarding the kernel regression method described in Section 3.2.4. Kai Ruhl, Michael Stengel, and Maryam Mustafa helped me with the editing and the voice-over of the submission video. The initial draft for parts of the system was sketched during a visit at MPI with Kiran Varanasi, Nils Hasler, and Christian Theobalt; it was further developed in discussions with Markus Wacker and Marcus Magnor. Implementation, recording setup, experiment design, evaluation, and presentation are my work.

**Chapter 4: Sparse Localized Deformation Components** is based on the publication

T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt. „Sparse Localized Deformation Components“. In: *ACM Transactions on Graphics (Proc. SIG-GRAPH Asia)* 32.6 (2013), 179:1–179:10.

Kiran Varanasi and Christian Theobalt both supported me with valuable advice that shaped this project. Stephan Wenger provided valuable insights into sparsity-inducing norms and proximal optimisation methods. Maryam Mustafa helped with the voice-over for the videos. I was responsible for the initial idea, the derivation of the algorithm, its implementation, evaluation, and presentation.

**Chapter 5: Compressed Manifold Modes for Mesh Processing** is based on the publication

T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker. „Compressed Manifold Modes for Mesh Processing“. In: *Computer Graphics Forum (Proc. of Symposium on Geometry Processing SGP)* 33.5 (2014), pp. 35–44.

The paper was written by me and Kiran Varanasi, with valuable feedback from Markus Wacker, Marcus Magnor, and Christian Theobalt. Markus Wacker additionally provided insights into the

---

theoretical background. The initial idea was conceived during a discussion with Kiran Varanasi. The design, development, and implementation of the optimisation algorithm, as well as evaluation and presentation, are my work.

**Additional work.** In addition to these publications, I have co-authored several publications that are loosely related to this dissertation. They may provide additional insight into certain aspects of the presented work or a wider overview of its field of application: An approach for obtaining dense correspondences for high-resolution video data [LLN+10; LLN+12]; an approach for monocular human pose reconstruction for an augmented reality clothing system [RNWM11]; an approach for extending the muscle model presented in Chapter 3 based on Kinect recordings [RNVT13]; a system for classifying soil samples for application in the geosciences [WNEG14]; and a new approach to simultaneously measure muscle activity with EMG, force distribution using a pressure plate, and muscle deformation with the system proposed in Chapter 2, which shall be used for assessment in one-legged balancing tasks [PGF+16].

**Supplementary Videos and Sourcecode.** The DVD attached to this dissertation contains supplementary videos showing many of its results in motion. Furthermore, I open-sourced my implementations of sparse localized deformation components I present in Chapter 4<sup>1</sup>. I also open-sourced a reference implementation of the compressed modes algorithm presented in Chapter 5, as well as all scripts to reproduce the results in this chapter<sup>2</sup>. A copy of these source codes can also be found on the supplementary DVD.

---

<sup>1</sup><https://github.com/tneumann/splocs>

<sup>2</sup><https://github.com/tneumann/cmm>



## Notation

The mathematical notation in this thesis follows conventions from the computer vision and computer graphics literature. Bold upper-case letters are used to denote matrices, for example  $\mathbf{A}$ ,  $\mathbf{\Psi}$ . Bold lower-case letters denote column vectors, for example  $\mathbf{x}$ ,  $\mathbf{\theta}$ . Calligraphic typeface is used to denote sets (e.g.  $\mathcal{I}$ ,  $\mathcal{S}$ ), and the cardinality of a set  $\mathcal{I}$  is denoted as  $|\mathcal{I}|$ . Functions returning a set are also written as calligraphic letters:  $\mathcal{V}(\cdot)$ . Unless stated otherwise, italic letters refer to scalars (e.g.  $x$ ,  $\lambda$ ) or functions (e.g.  $\phi(\cdot)$ ,  $f(\cdot)$ ). Dimensions (size of vectors, sets, matrices) are written in upper-case italic letters, e.g.  $K$ ,  $N$ ; often used like in  $\mathbf{x} \in \mathbb{R}^N$  (which states that  $\mathbf{x}$  is an  $N$ -dimensional column vector of real numbers).

Lower right indices denote entries in a matrix or in a vector:  $\mathbf{A}_{i,j}$  or  $\mathbf{x}_i$ , respectively. Notice that vector and matrix values are bold-faced even when they are indexed, even though  $\mathbf{A}_{i,j}$  and  $\mathbf{x}_i$  would both denote a scalar. Sometimes, a complete row or column has to be extracted from a matrix; this is denoted as  $\mathbf{A}_{k,*}$  or simply  $\mathbf{A}_k$  to extract the  $k$ th row of  $\mathbf{A}$ , and  $\mathbf{A}_{*,k}$  to extract the  $k$ th column.  $\mathbf{x}^\top$ ,  $\mathbf{A}^\top$  denotes transposition of a vector or a matrix, respectively.  $\text{vec}(\mathbf{A})$  vectorizes an  $N \times M$  matrix  $\mathbf{A}$ , introducing a renumbering of the entries from a two-dimensional matrix to a one-dimensional column vector of size  $N \cdot M$ , implemented by stacking its columns on top of one another. The trace of a matrix is denoted as  $\text{Tr}(\cdot)$ ;  $\text{diag}(\mathbf{A})$  yields a column vector with the entries in the diagonal of  $\mathbf{A}$ .  $\mathbf{I}^{(N)}$  denotes the identity matrix of size  $N \times N$ ; often written shortly as  $\mathbf{I}$  unless the size  $N$  cannot be inferred from the context. A vector of ones is written as a bold upright one:  $\mathbf{1}$ ; a vector of zeros as  $\mathbf{0}$ . The signum function  $\text{sgn}(x) = x/|x|$  returns the sign of  $x$ . The vector operations  $\max(\cdot)$  and  $\min(\cdot)$  extract the maximum and minimum element of a vector, respectively. To return the larger value from two given scalars  $a$  and  $b$ , the function  $\max(a, b)$  can be used, and  $\min(a, b)$  to return the smaller of the two values. Unless explicitly stated otherwise, operations like  $\ln$ ,  $\exp$ ,  $\text{sgn}$ , etc. can take a vector or matrix as the parameter, which denotes the application of the operation to all elements, returning a new vector or matrix of the same dimensions as the input parameter.

The  $\ell_p$ -norm of an  $N$ -dimensional vector  $\mathbf{x}$  is denoted as  $\|\mathbf{x}\|_p = \left(\sum_{i=1}^N |\mathbf{x}_i|^p\right)^{1/p}$ . The Frobenius norm of an  $N \times M$  matrix  $\mathbf{A}$  is denoted  $\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^N \sum_{j=1}^M |\mathbf{A}_{i,j}|^2}$ . A common trick is to formulate the squared Frobenius norm using the trace operation:  $\|\mathbf{A}\|_F^2 = \text{Tr}(\mathbf{A}^\top \mathbf{A})$ .

$\mathcal{SO}(3)$  denotes the group of rotations in 3-dimensional Euclidean space, so if  $\mathbf{R}$  is a rotation matrix ( $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$  and  $\det(\mathbf{R}) = 1$ ), then  $\mathbf{R} \in \mathcal{SO}(3)$ . The Lie algebra of  $\mathcal{SO}(3)$  is denoted as  $\mathfrak{so}(3)$ . The set of positive real numbers is denoted as  $\mathbb{R}_+$ , and  $\mathbb{Z}_+$  the set of positive integer numbers.

It is sometimes necessary to introduce a second level of indexing, for example to specify a variation of a variable or when collecting an array of vectors or matrices. In this case, a superscript is used, for example:  $\mathbf{x}^{(1)}$ ,  $\mathbf{x}^{(2)}$ , or  $\mathbf{x}^{(n)}$ . Similarly, quantities that change during an iterative algorithm can indicate the current iteration as  $x^{(k)}$  or  $x^{(0)}$ . The brackets prevent confusion with exponentiation.

# 1. Introduction

The world around us is full of dynamically deforming 3D surfaces: From wrinkling cloth to bending shoe soles, from leaves blowing in the wind to tree trunks bending in a storm, from heavily bulging muscles to the slightest motions of a nostril. Deforming surfaces are spatiotemporal phenomena: Not only do they constitute three-dimensional positions of every point on the surface at a specific point in time, but also their possibly complex motion. The surface evolves, it deforms over time. Nowadays, computer graphics techniques can believably re-create such phenomena based on simulations and renderings. However, digitally (re-)creating deformable surfaces from scratch is a challenging, time-consuming, and costly process, often involving tedious manual work by experienced artists or realistic physical simulation using high-end computers. One possible solution to this problem is to directly reconstruct deformable surfaces from the real world.

Today’s technology can digitize 3D surfaces at high spatial resolution, for example using laser-range scanners or time-of-flight cameras [GTKK13]. But just like a video recording only captures a sequence of 2D photos, without any information of the motion of each pixel, 3D scanners can only output uncorrelated 3D point clouds, providing no information about where each of the points moved between two captured moments. Estimating the motion of the 2D pixels or 3D points is one major open problem in the field of computer vision [Sze10]. The dense estimation of correspondences between pixels across time is commonly called optical flow and enables computers to densely track features in a single video stream over time [BSL+11]. Similarly, pixel correspondences between images from different known camera locations can be used to recover the 3D structure of a visible surface; this is referred to as (multiview) stereo estimation [Sin14; SCD+06]. Combining both approaches enables full spatiotemporal 3D reconstruction of deforming surfaces. Two principal strategies exist for spatiotemporal 3D reconstruction: Marker-based approaches rely on uniquely identifiable markers or specific patterns on the surface, while markerless approaches promise to work in completely unrestrictive settings without any surface preparation. Recent computer vision research almost exclusively concentrates on markerless solutions [MGSHT15]. Such approaches have to rely on the presence of some natural surface texture. But even then, in the general setting, the problem of estimating correspondences within this markerless regime is ill-posed. Thus, researchers have to resort

to additional object-type specific priors and/or rely on regularization [TAS+10]. For example, many so-called full-body performance capture methods rely on a specific template scan of the person who is being captured [AST+08]. While this can produce plausible results, this approach limits applicability and cannot guarantee the correctness of the measured surface. In contrast, marker-based systems are highly accurate, can provide 3D data in real-time, and are robust since markers are designed to be detected reliably [Kra07; Has14; SOS09]. Nowadays, marker-based systems are heavily used in industrial applications. However, none of these techniques offer the perfect solution for every application use case. There is, and possibly always will be, a trade-off between robustness, flexibility, accuracy, and computation speed. This thesis concentrates on robustness and accuracy achieved by marker-based systems. In Chapter 2, I develop new marker-based approaches which allow for the reconstruction of even heavily deforming 3D surfaces such as stretching garments and fast, full-body sports motions, while minimizing preparation times.

3D reconstruction techniques like the ones proposed in this thesis feature increasingly better detail and resolution. Like a recorded video, these dynamic 3D reconstructions can be replayed and re-rendered in their original form. However, analyzing and modifying such data cannot be easily accomplished. Arguably, deformable surface analysis and editing are still in its infancy. The current situation is like having a new ultra-resolution digital camera, but no Photoshop to postprocess the images. Proper tools for analyzing and editing captured data are needed in different application fields. Tools that are specialized for captured 3D data could help discover the underlying factors for observed 3D surface deformation, helping us to better understand the dynamics of deforming surfaces. Computer graphics applications might use such tools to apply artistic modifiers to captured data of deforming surfaces, for example to automatically change facial expressions to convey certain emotions. Or the captured data might be used as a template for quickly instantiating completely different, realistically looking virtual characters.

With sufficient data, statistical analysis of the captured dynamic 3D surfaces becomes possible. This powerful idea is known as data-driven modeling, sometimes also called statistical shape processing. A data-driven method *learns* how the 3D surface behaves in the real world according to the captured data. For example, a data-driven model can learn the variations in human body shape from laser scans of many different people [ASK+05; HSS+09]. Such models generate new virtual characters with body shapes that are not in the input scans but that nevertheless look realistic. A user just has to provide input parameters (labels) such as height, weight, gender, age. The expressiveness of state-of-the-art data-driven models is limited both by the amount of training data recorded and by the number of free input parameters. New capture methods like the one proposed in this thesis ameliorate this situation since they provide fine-scale geometric detail of moving skin surfaces. Compared to

---

static laser scans that usually take a couple of seconds and provide a single pose per scan, the new multiview-based capture systems provide at least 30 scans per second. In effect, this allows capturing much more samples of 3D surfaces, resulting in much larger datasets as compared to static laser scanning. New challenges appear in this setting: The learning approach has to scale to very large datasets, and the approach must be capable of modeling fine-scale surface deformations, phenomena that only become visible at high temporal resolution and which may be related to the input parameters in a non-linear way. In Chapter 3, I tackle this problem in the specific setting of data-driven modeling of human muscles. I present a muscle surface model that automatically learns the complex inter-dependencies between muscle activity of real subjects and their physical constitution, applied external forces, and body pose all from real, captured data. Such a learned model can generate detailed surface deformations based on a sparse set of input parameters, such as pose and external loads. The model is suitable for generating meaningful animations and does not require cumbersome rigging and simulation. While this part of the thesis focuses on muscle deformations and the shoulder-arm-complex, the underlying methodology can work as a general blueprint and as a proof of concept, enabling the application of similar concepts to other body parts, to the whole body, or for even larger datasets.

Statistical shape modeling usually requires capturing both the 3D surface as well as the set of input parameters that generated this 3D surface. For example, the posture must be inferred from a rigged skeleton, and the height of the person has to be recorded alongside the capture process [ASK+05; HSS+09]. Otherwise, the model is not able to learn the relationship between the input parameters and the surface deformation. In some cases, such input parameters cannot be measured conveniently. For example, an artist might want to edit a facial expression by adding a "smile" component to a captured facial performance of an actor. But, arguably, the "amount of smile" cannot be measured objectively. Thus, instead of a supervised data-based modeling approach, we may like to have an unsupervised approach that automatically discovers the latent input parameters. In Chapter 4, I present such an unsupervised data-based modeling method. The core idea is to model surface deformations based on a general and intuitive principle that is based on the observation that deformations usually have a local effect in an isolated region. For example, we humans know that it is possible to move only one of our eyebrows (even if not everybody is able to do so). We also know that the motion of the eyebrow can be performed independently of the motion of the lips (at least after some training). This principle can be cast into an optimization framework based on the formal mathematical concepts of sparsity and geodesic locality. Building upon these principles as well as the recently popular machine learning method of Sparse PCA, I present sparse localized deformation components in Chapter 4 - a powerful tool for artistic editing, data exploration, and statistical shape processing.

During my work on the above-mentioned method, it came to my mind that the successful regime of sparse and localized modeling can have a wider range of applications. I argue that the assumption of sparsity and locality does not only hold for deforming, but also to static surfaces. For example, we humans think of a hand as consisting of 5 fingers and a palm. Thus, a shape processing algorithm should respect this inherent part composition of most everyday objects. This is the idea of compressed manifold modes that I explore in Chapter 5. Equivalently to decomposing deformations (Chapter 4), here the surfaces themselves are decomposed with respect to sparse and localized 3D surface basis functions. Compressed manifold modes provide unique capabilities that can be exploited in geometry processing applications like shape approximation, feature detection, segmentation, and in shape matching. In particular, the shape matching capabilities suggest a huge potential for future applications in 3D reconstruction and registration systems such as the ones presented in Chapter 2 and Chapter 3.

I discuss many of these opportunities and challenges for future work in Chapter 6 and I conclude with an overview of work published by other researchers who build on the results of this thesis.

## 2. Reconstructing Dynamically Deforming 3D Surfaces

To reconstruct dynamically deforming surfaces from the real world, it is necessary to measure the 3D positions and motion of sufficiently many points on the surface. Active 3D scanning systems such as laser scanners, Time-of-Flight sensors, or structured-light scanners directly measure 3D coordinates of surface points, but not their motion and deformation. Motion has to be recovered from the raw data by other means, for example using a tracking or non-rigid registration algorithm. Such tracking is often not possible on measured 3D geometry alone. For example, textiles may stretch while their extrinsic geometry stays completely constant. Such *tangential* deformations can only be inferred when there is a specific texture, pattern, or some form of markers on the surface that can be reliably tracked while the surface is deforming. These limitations are one of the reasons for the development of passive camera-based systems. Passive systems consist of at least two cameras that simultaneously image the dynamic scene. The cameras must first be calibrated so that their position and field-of-view are known. Then, pixel correspondences enable dynamic 3D reconstruction: correspondences between pixels from spatially separate cameras allow reconstruction of 3D coordinates, while correspondences between pixels of successive video frames allow reconstruction of motion. Unfortunately, estimating such correspondences correctly is a computationally challenging problem. In fact, without additional prior information, or without a specific texture on the surface, accurate and robust correspondences cannot be found.

The computer vision community tackles this problem with a multitude of approaches that I will review in Section 2.1. All of them have their strengths and weaknesses as well as specific target application scenarios. Most camera-based 3D reconstruction methods focus on completely markerless setups. They track the natural texture that some surfaces exhibit. Consequently, if the surface does not have such a structured texture, if the resolution of the cameras is too low to image it, if there is fast motion, or if the surface is too smooth or too regular, then such methods often struggle to jointly capture 3D geometry and deformation. Well-engineered priors and regularizers may be able to recover a *plausible* reconstruction result in these cases, but that does not mean that the resulting reconstruction is correct. For application use cases, such as data-driven animation or for engineering applications, this is not acceptable. Precise and reliable measurements are required.

This thesis focuses on the precise reconstruction of highly deforming and fast-moving surfaces, such as textiles, garments, shoes, and skin. I argue that this use case can not rely on markerless approaches, for the above-mentioned reasons. We should not condemn marker-based approaches, but make them more robust, more flexible, and simpler to use. The methods presented in this chapter are a step in this direction.

I propose two methods for reconstructing dynamically deforming 3D surfaces. The first approach, Section 2.2, relies on the idea of jointly inferring the 3D positions and the parameterization of a surface - instead of measuring only coordinates  $(x, y, z)$ . It also measures the intrinsic position of those coordinates on the parameterized domain, e.g.  $(u, v)$ . This implicitly allows recovering motion and deformation of the surface, even for arbitrarily large motion and strong deformation. The approach is specialized for garment and textile surfaces and realized by printing a specific pattern inspired by [SSK+05] onto the textiles. I propose fast and parallelizable image processing algorithms for detecting the printed pattern, enabling robust reconstruction in practical scenarios. This method has been actively applied in the industry by adidas AG.

The second approach focuses on measuring dense dynamic 3D geometry using completely random dot patterns. Randomized patterns are applied very quickly onto surfaces such as textiles, shoes, and even the human skin. However, estimating correspondences between images of different cameras and between successive scans becomes non-trivial since every dot looks exactly the same. In order to solve for these ambiguities, the 3D reconstructions are constrained to be multiview-consistent. To this end, I introduce and formalize the concept of multiview conflicts. These constraints are incorporated into a graph matching framework that is extended to jointly solve for the optimal 3D reconstruction across all stereo pairs of a multi-camera system. I also propose a new shape context based tracking approach that is able to estimate correspondences across very large motion and deformation. This 3D reconstruction method forms the basis for measuring and modeling human arm muscle deformations in Chapter 3. Finally, I show how these dense 3D reconstruction approaches can be used for strain analysis of textiles.

### 2.1. Related Work

Before introducing the 3D reconstruction algorithms, let me first give a broad background on basic techniques used in 3D reconstruction from multiple cameras. I will also review graph matching and consistency-based matching - both are related to the approach presented in Section 2.3.

**Stereo Matching** considers multi-camera reconstruction setups involving only two cameras. Scharstein and Szeliski [SS02] give a general overview of this case. The projective geometry of stereo setups is well-known [HZ04, Part II: Two-View Geometry]. Still unsolved is the problem of finding robust correspondences between multiple camera views. Correspondences are needed in order to triangulate 3D positions (assuming calibrated cameras). To find stereo matches, most approaches integrate information from neighboring pixels. This can be done either locally, by matching image patches [BRR11], by using anisotropic matching cost filters [RHB+11], or by adaptively matching scaled windows [BBH08]. Yet another idea is to seek optimal correspondences so that the photo-consistency of the matches (data term), as well as smoothness of the reconstructed 3D geometry (regularization term), is maximized. This global optimization can be solved jointly for all pixels (or all feature points), for example using belief propagation [FH04; LLN+10] or Markov random fields [KZ01]. These approaches optimize over a discrete label space, where each pixel can be assigned to multiple candidate pixels in the other image, an idea that is related to the graph matching approach presented in Section 2.3. Recent developments hint at a revival of local and semi-global approaches like [Hir05] and focus on improving the data term, e.g. using convolutional neural networks [ŽL15] to learn a robust measure of photo-consistency instead of using classical measures like sum-of-squared differences. My proposed graph matching formulation is completely orthogonal to this development: The proposed algorithm does not require a data term at all and is only equipped with a regularization term. However, in order to break ambiguities, it incorporates information from more than two cameras. Notice that stereo approaches typically operate only in narrow baseline setups (cameras very close to each other), to enable matching based on appearance only. In contrast, the approaches considered here are applicable to wide baseline setups as well.

**Multiview stereo (MVS)** is a natural extension of the stereo scenario, aiming at 3D reconstruction of arbitrary objects from more than two cameras [Sin14]. At the core, these methods still need to solve the image correspondence problem, i.e., finding corresponding pixels, patches, or features across multiple cameras. Seitz et al. [SCD+06] give a taxonomy and evaluation of classical methods for static reconstruction. A straightforward approach to MVS is to fuse depth maps from separate stereo pairs [BBH08; MAW+07], possibly by deciding from multiple depth hypotheses during depth map fusion [CVHC08]. Another classical approach is to directly model the 3D surface to be reconstructed, for example using voxels [KKSS00; SZB+07] or level sets [FK98]. Based on such representations, sophisticated global optimization methods are employed to solve for the 3D reconstruction that best fits the image data [KKBC09; SZB+07; FK98; PKF07]. In recent years, simple point-based representations and local optimization methods turned out to be extremely successful [FP08; SSS08;

GS15], probably thanks to their scalability. Such point-based approaches match small planar 3D patches across different images. Recent work combines this with depth map fusion, e.g. in [GS15]. State-of-the-art MVS methods work best on tens to hundreds of very high-resolution photos, able to faithfully reconstruct static objects. The advent of easy-to-use MVS software [Agi; Aut; Wu11] enables non-experts to perform photo-realistic reconstruction of static scenes. A major problem of MVS reconstructions is their affliction with outliers and frequent missing regions. Jensen et al. [JDV+14] recently observed that these problems do not necessarily occur where photo-consistency fails (e.g. specular highlights). Instead, MVS most often fails in textureless or repetitive regions of the object, where direct matching is completely ambiguous. Subsequent 3D surface interpolation [KH13] can help fill such regions smoothly, but 3D information is ultimately lost and not correctly reconstructed in textureless areas. The multiview conflict method proposed in this chapter is specifically targeted at solving ambiguities in situations where the texture is not unique. While my specific application will be skin and garment capture in scenarios where there are random points on the surface, I nevertheless believe the basic idea can inspire future MVS approaches, since the proposed multiview constraints can be incorporated into optimization frameworks of more general MVS pipelines.

**Spatiotemporal MVS.** Since even static MVS is very memory demanding, general spatiotemporal MVS for dynamic scenes is not well researched. The level set method can be elegantly upgraded to the spatiotemporal setting by jointly formulating tracking and reconstruction using partial differential equations [GM04], but this requires a good initialization. Klose et al. [KLM10] extend the patch-based method by Furukawa and Ponce [FP08] for video data filmed by multiple unsynchronized cameras. Given a set of images, they not only optimize patch position but also their velocity through the scene. Scene flow aims to directly estimate 3D position and 3D motion for each pixel in all given videos [VBC+99; VBZ+10]. Robust scene flow estimation is the holy grail for solving dynamic 3D reconstruction since it directly computes the motion and deformation field for every surface point in the scene that was observed by a camera. For general scenes, this is a truly challenging problem. Right now, only narrow-stereo baselines with small motion between frames work decently [VRS14]. Manual corrections are required so often that tools have been developed so that users can perform such corrections in a manual post-process [REH+15]. Due to the limitations of spatiotemporal MVS, it is currently easier to reconstruct each frame independently. To estimate a spatiotemporally coherent geometry from the individually reconstructed surfaces, non-rigid registration algorithms are applied to estimate the motion of each surface point between successive frames. For example, robust feature detectors are used by Varanasi et al. [VZBH08] for temporally consistent surface mesh registration.

In Section 2.3.4, I present a robust feature descriptor for non-rigid matching which is adopted for non-rigid mesh registration in Chapter 3.

**Optical marker-based motion capture** is well-established, for example in the entertainment industry for animating virtual characters, in biomedicine for assessing pathological conditions using gait analysis, or in sports sciences for monitoring the motion of athletes. Motion capture records only the approximate skeletal motion of (human) bodies [Has14]. Park and Hodgins [PH06] try to overcome this limitation by using up to 350 markers to reconstruct human skin deformation. Their setup requires sticking many small markers (3 mm in diameter) onto the skin, which requires a lot of manual effort. The commercial motion capture software from Vicon that was used by Park and Hodgins [PH06] was not able to handle so many identical markers and failed to track them so that their 3D reconstruction algorithm required a special semi-automatic correction scheme. However, their results are very accurate. While current research concentrates on markerless motion capture [SBB10; HRT+09; GSD+09; SHG+11; EAJ+15; RRR+16], I argue that there is still room for improving optical marker-based *surface* capture: Instead of using spherical markers that have to be glued to the surface, I propose a more flexible setup that just needs a pattern of quads or random dots which can be very easily applied to most surfaces. I also show a new way to solve the correspondence problem in a robust way, allowing spatiotemporally consistent 3D reconstruction with thousands of identical markers.

**Performance Capture.** While motion capture only measures skeletal motion, performance capture also reconstructs detailed spatiotemporal geometry and surface texture of a moving subject - without any markers [TAS+10]. Some approaches use silhouettes [SMP03] and combine them with stereo cues [SH07] for capturing the shape of the subject. In performance capture, one can assume that a specific (human) subject is being captured, so one can resort to additional priors. For example, many approaches use a template mesh for cross-parameterization of the 3D surfaces extracted at different time frames. A prominent example is the work by Aguiar et al. [AST+08] who use a laser scan of the specific person to be captured as a prior mesh surface. This mesh surface is then deformed according to silhouettes and feature points in the input videos. Surface deformation can be modeled either using a coarse mesh [AST+08] or a rigged mesh with underlying kinematic skeleton [VBMP08; GSD+09]. Fine-scale surface details can be added on top based on shape-from-shading [PZB+09; WVL+11; WSVT13]. This gives believable folds and wrinkles but fails to correctly capture tangential surface shifting. The coarse initial reconstructions can also be refined by optimizing for multiview photo-consistency [SH07; AST+08; RDHT14]. However, this only works correctly

in areas with sufficient surface texture. The present work lifts the requirement of markerless-ness and trades the flexibility of markerless approaches against higher robustness, focusing on obtaining very precise reconstruction quality under large motion and deformation. Even if this requires a special pattern on the surface, the two approaches are reasonably flexible and the patterns are very easily applicable. Most importantly, the approaches presented here obtain true tangential surface deformations. Furthermore, they robustly deliver spatiotemporally consistent matches across large motion. The idea of cross-parameterization will be used for capturing and registering muscle deformations in Chapter 3.

**Garment Capturing** methods reconstruct the dynamic geometry of moving and deforming cloth from (multiview) video. Early approaches captured a single sheet of cloth by tracking SIFT features [PH03], by estimating the motion of the cloth using optic flow [SM04], or using physically based analysis-by-synthesis [HAR+06]. This required textiles with highly detailed texture. Guskov et al. [GKB03] printed a special pattern onto the textile: small colored quads that are easily detected and identified. Scholz et al. [SSK+05] adopted M-Array patterns [MOC+98]. An M-array consists of color-coded dots whose  $3 \times 3$  neighborhood allows unique identification across time and across different camera views. This enables both reconstruction and tracking, allowing Scholz et al. [SSK+05] to capture a complete garment for the first time. White et al. [WCF07] analyze different printed pattern designs and propose to use triangular markers as well as more colors as Scholz et al. [SSK+05]. Both approaches require calibration of the pattern colors to enable correct color classification of the markers. In my experience, this color classification fails far too often in practical situations. Section 2.2.2 describes a more robust color classification approach which does not require a calibration step. While the proposed method builds upon the work by Scholz et al. [SSK+05], it still allows the same high resolution as the follow-up work by White et al. [WCF07]: markers are approximately  $7 \times 7$  pixels in size. Markers are extracted quickly with a novel parallelized detection method presented in Section 2.2.1. Section 2.3 then introduces a method that allows capturing garments (and other surfaces) but with a completely randomized pattern instead of a color-coded one.

But the printed pattern is not always required. Bradley et al. [BPS+08] demonstrated the first system for capturing full garments without any markers. The system performs multiview stereo reconstruction [BBH08] at each time step, followed by template registration on the basis of a cross-parameterization algorithm. However, the garment itself does not provide sufficient visual cues for accurate tracking. Therefore Bradley et al. have to rely both on purely geometric features and on strong regularizers, which disallows stretching of the garment. The results seem to contain slight drift and, suspiciously, do not feature any fast or complex motions. A very different approach employs

photometric stereo for detailed reconstruction of garments, up to the yarn level [BHV+11]. But this capture setup requires accurately placed light sources, hinders  $360^\circ$  reconstruction, and suffers from the same tracking problems as other markerless approaches. As noted by Miguel et al. [MBT+12], if precise measurements of garment deformation are needed it is ultimately necessary to apply a pattern onto the cloth.

**Photogrammetry** comprises methods of measurement and analysis of 3D coordinates from image or laser scanners [Kra07]. The focus of photogrammetry is on high precision in 3D reconstruction. For measuring dense geometry, some photogrammetry methods rely on regular or random patterns of (projected or printed) markers [Kra07]. A classic algorithm to 3D-reconstruct geometry from such patterns is the method of intersection of epipolar lines by Maas [Maa92]. Maas reduces the number of matching candidates by checking multiview consistency within three camera views. Without further heuristics, only non-ambiguous multiview-consistent matches can be 3D-reconstructed with this method. Therefore, this strategy misses all ambiguous points, requires all points to be visible in all cameras, and is very sensitive both to false detections and to slightly imprecise calibration. An extension approximates the algorithm for more than three views using a voting procedure [Maa97], but is still restricted to multiview consistency alone. In contrast, the algorithm presented in Section 2.3 does not suffer from these limitations: It reduces the matching ambiguities by incorporating both the neighborhood structure (via spatial regularization) and multiview consistency in a joint optimization framework.

Another related technique in photogrammetry is particle tracking velocimetry (PTV) for reconstructing fluid and gas flows. PTV extracts 2D or 3D velocity fields from images showing individual seed particles (that have been immersed in fluid or gas and can be easily detected in images). Such methods can resolve outliers and ambiguous triangulations in the temporal domain, since the flow field can be assumed to have low divergence [RGPS05; JWZ13], similar to variational optical flow algorithms [BSL+11]. However, the temporal smoothness assumption is invalid for the dynamically deforming 3D surfaces considered in this chapter, which is why PTV methods cannot be employed.

**Digital Image Correlation** is a technique developed in material sciences for deformation analysis in mechanical testing. This technique is usually based on relatively simple stereo reconstruction techniques such as sub-pixel block matching and sum-of-squared-differences cross-correlation. Engineers using the technique are required to spray 3D surfaces with a speckle (noise-like) pattern. This pattern allows for very reliable and extremely accurate estimation of correspondences between stereo cameras. The book by Schreier et al. [SOS09] provides a comprehensive literature survey as well as

an introduction to the mathematical foundations of the involved stereo reconstruction algorithms. Nowadays, digital image correlation systems are commercially available, for example the ARAMIS system made by GOM mbH. These deformation analysis systems are used in lab setups where the capture volume can be controlled and the motion of the 3D surface is minimal. State-of-the-art industrial systems are limited to stereo-setups. The calibration of multiple stereo setups into a common coordinate system is possible but then individual stereo reconstructions have to be combined and blended to obtain a faithful  $360^\circ$  reconstruction. In contrast, the 3D reconstruction methods presented in this chapter allow for large motion, large baseline between cameras, are flexible in their camera setups and do not require compositing separate stereo reconstructions since they exploit all cameras jointly for reconstruction.

**Graph Matching** aims to find correspondences between the nodes of two graphs, such that nodes of the first graph can be mapped to one node in the second graph. In finding such a mapping, graph matching algorithms respect node-to-node similarities as well as the edge relationships of both graphs. Applications range from pattern recognition to machine learning [CFSV04; FPV14]. In computer vision, graph matching can be used to match feature points between images. The feature points become graph nodes, the spatial relationships between features become edges. Graph matching optimizes for the best feature correspondences such that feature point similarities (*unary terms*) as well as geometric relationships (*pairwise terms*) are preserved. Graph matching can be seen as an optimization problem, more precisely as an integer quadratic program (IQP). Unfortunately, it is NP-hard in general [BCPP98]. Many methods circumvent this problem by relaxing the constraints of the IQP, which lifts some of the combinatoric complexity of this problem. A myriad of relaxation methods exist, using spectral techniques [LH05] or gradient-descent-like algorithms [GR96; WW04], employing probabilistic interpretations [ZS08] or adopting the Frank-Wolfe algorithm [LHS09; VCL+15], dual decomposition [TKR13], semidefinite programming [HG13; KKBL15], and random walks [CLL10]. I will formulate graph matching for stereo matching in Section 2.3.1. The formulation differs from the feature matching setting: The calibrated multiview setting allows for directly respecting epipolar stereo constraints. Section 2.3.2 then introduces a novel multiview extension of graph matching. It optimizes jointly over all pairwise stereo matches and incorporates multiview constraints that enforce consistent matching across multiple cameras. For optimization of the matching objective, I adopt the classical Graduated Assignment (GA) method of Gold and Rangarajan [GR96]. I propose a novel derivation of GA which provides a clear theoretical foundation of the classical GA. It also enables the inclusion of the novel multiview constraints into the optimization task.

**Consistent Feature Matching** estimates correspondences between more than two images jointly. The idea is to identify correspondences between features (points, edges, or individual pixels) such that these correspondences are *cyclically consistent*: starting from a feature and following the correspondences should eventually lead back to the starting feature. In other words, correspondences should respect *loop constraints*. Researchers recognized quickly that respecting such constraints can be very beneficial for robust MVS [BTZ96; STG03; ZKP10]. For example, Ferrari et al. [FTG03] establish feature correspondences in pairwise views, integrate pairwise matches to feature tracks, and then filter inconsistent feature tracks. In a similar framework, Yao and Cham [YC07] iteratively add reliable matches by looking at triplets of images. These works do not express loop consistency within a global optimization framework, e.g. estimating correspondences and checking consistency are performed independently. Sellent et al. [SRM12] enforce loop-consistency both for feature matching and for optic flow computation, but only across three images. To obtain three-view consistent feature matches they solve the classical three-matching procedure (basically an extension of bipartite matching). Spatial regularization is only included in their global optic flow formulation, which in turn does not model consistency directly as an optimization constraint. Yan et al. [YWZ+15] match multiview features in an uncalibrated scenario, determining calibration parameters (e.g. pairwise transformation or fundamental matrix) and the unknown correspondences jointly, but without using spatial regularization as used in graph matching. A recent line of works directly optimizes for consistent feature matching across multiple images. These approaches are initialized by matching all image combinations pairwise and independently, for example using nearest neighbor or graph matching, which gives a permutation matrix for each image pair. From this, Yan et al. [YLL+14] continuously optimize for loop consistency, gradually improving the consistency and graph matching score by optimally selecting permutations over three images at a time. Huang and Guibas [HG13] stack all the given initial permutation matrices into a large *map collection matrix*. They show that a matching is cycle-consistent when this matrix is low-rank and positive semidefinite. Optimizing for such a low-rank matrix given an initial, inconsistent matching can be done exactly using semidefinite programming [HG13; CGH14] or approximately either using eigendecomposition [PKS13] or nuclear norm minimization [ZZD15]. None of these approaches simultaneously optimize for spatial regularization and consistency over all image pairs at the same time. I suppose the reason is that expressing cycle-consistency directly requires a huge set of constraints - one would need to enumerate all possible loops through the images. Section 2.3 will demonstrate that when we concentrate on *multiview conflicts* instead of *loop constraints*, it is indeed possible to express this consistency globally within a joint graph matching framework. The proposed method thus differs

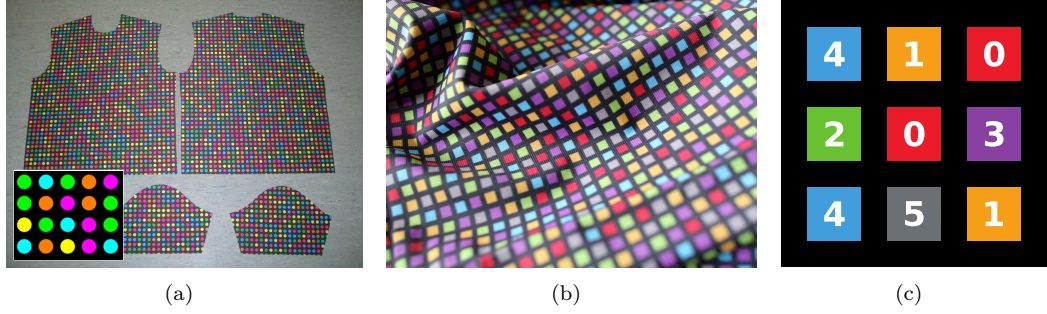


Figure 2.1.: (a) The color coded circle pattern used for garment reconstruction by Scholz et al. [SSK+05]. The inset shows the template that is printed onto the textile. Image reproduced from [SSK+05]. (b) Proposed quad pattern as printed on a sheet of fabric. In contrast to the circle pattern, the quad pattern allows more markers per surface area and lends itself to faster image processing algorithms. (c) Each  $3 \times 3$  neighborhood forms a unique M-Array code: Each color has a predefined index, for example red = 0 and blue = 4, visualized here as numbers on top of the quads. In this example the M-Array code ID of the central red quad is "410 203 451", assuming the color codes are read out in a row-major order. The pattern is generated such that this ID is unique across the whole garment.

from the related work since it optimizes spatial regularization and multiview consistency jointly, thus enabling robust matching of thousands of marker dots.

## 2.2. Robust Garment Capturing with a Color-coded Quad Pattern

This section presents a garment capture system capable of reconstructing moving cloth surfaces from multi-camera recordings. The system can be seen as a further development of the approach presented by Scholz et al. [SSK+05]. To enable 3D reconstruction, the textile is printed with a pattern specifically constructed to facilitate dense capturing of highly deforming garments. Just like [SSK+05], the pattern is based on M-Array color codes [MOC+98]. Such a scheme directly encodes the parameterization of surface points by the colors of the  $3 \times 3$  neighboring points. In other words, every point can be uniquely assigned to a  $(u, v)$  position on the texture printed onto the garment, as visualized in Figure 2.1(c).

In contrast to [SSK+05], instead of circles, I recommend quads as main primitives, Figure 2.1. This helps in better segmenting the quads, lends itself to fast image processing algorithms, provides more color information in the same area, and allows for printing slightly smaller markers at the same camera resolution, thus increasing the possible capture resolution. As observed by [SSK+05], the size of the textile is limited by the number of colors used, because only a limited amount of unique  $3 \times 3$  codes is available. The proposed system adopts the pattern generation algorithm from Morano

et al. [MOC+98]. In addition to the rotational and shear symmetry checks suggested by Scholz et al. [SSK+05], we can further improve results by preventing the generation of neighboring quads that have the same color. This helps in later segmentation steps. The pattern generator is able to build a pattern of  $333 \times 249$  quads from 6 colors.<sup>1</sup> The typical size of one quad on a textile produced for the lower limbs (e.g. for capturing trousers or tights) is  $4 \text{ mm} \times 4 \text{ mm}$  with separating lines of  $2 \text{ mm}$  thickness.<sup>2</sup> A  $333 \times 249$  quad pattern can fill a  $2 \text{ m} \times 1.5 \text{ m}$  sheet of textile. The sheet can be used to manufacture any garment, for example trousers, jerseys, T-shirts, and even shoes.

To reconstruct a textile with the quad pattern based on multi-camera recordings, the following pipeline steps are performed for robust detection. First, the locations of every quad in each image are detected. Compared to previous work [Neu09; SSK+05] the pipeline works also for very small quads, is able to obtain high-resolution reconstructions and lends itself to parallel implementation on GPUs. Next, each located quad is classified, converting the RGB color as observed in the image to a known color of the pattern (for example to *red*, *green*, etc.). The next step matches the color-classified quads to the known M-Array quad pattern. I propose a novel color-classification routine that increases the number of matched quads as compared to [SSK+05]. The M-Array matching gives each quad in each image a unique identification. Matching the identification across different cameras then allows 3D-reconstruction of the quad locations by triangulating corresponding quads that have been detected in at least two cameras. The unique identification of the quads also allows tracking them across large distances, without constraints on the amount of motion and with perfect recovery after temporary occlusion of a quad - the quad is simply re-detected and uniquely re-identified whenever and wherever it becomes visible.

### 2.2.1. Robust Quad Center Detection

The first step of the pipeline detects the center locations of each quad, as summarized in Figure 2.2. The quad center lies inside the uni-colored quad area, separated by lines. Finding the lines enables segmentation of the quad centers. But line-like structures occur everywhere in the image, from small textile strings to larger structures like the arm of a person. Thus, it is essential to detect only line-like structures with a specific size in the image, in other words, perform detection at a specific *scale*. This

---

<sup>1</sup>In theory, even larger patterns are possible with 6 colors, but the theoretical maximum cannot be easily derived. The pattern generator works in an iterative and random fashion and is therefore not able to generate a fully conforming pattern at every run, requiring a restart.

<sup>2</sup>So we can reconstruct a 3D point every 6 mm. In comparison, Scholz et al. [SSK+05] use 20 mm distance between circles. However, this comparison is ambiguous as 3D resolution depends on image resolution and distance of cameras from the garment. A better way is to compare the size of the imaged markers in the images. The proposed quad pattern pipeline allows quads down to  $7 \times 7$  pixels across. From visual inspection of [SSK+05, Fig. 7], circles seem to be about  $12 \times 12$  pixels wide. This means the proposed quad pattern pipeline offers higher 3D resolution given the same multi-camera setup.

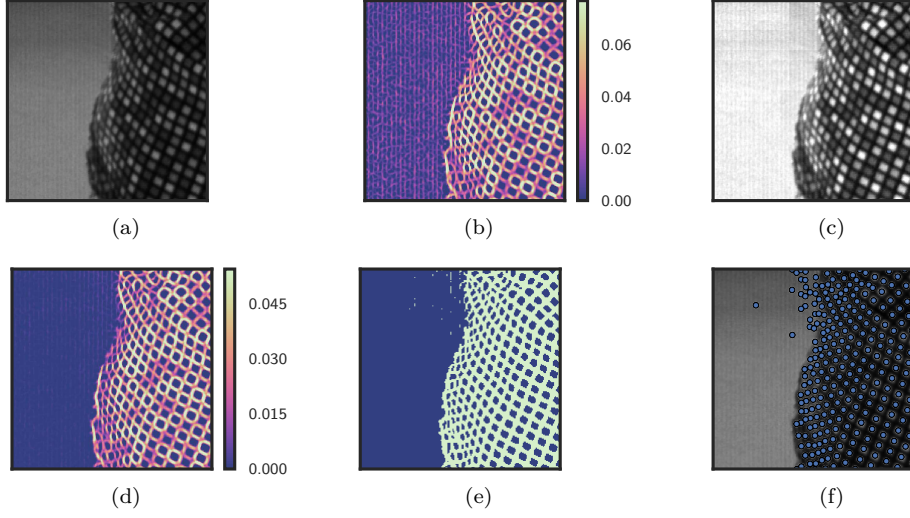


Figure 2.2.: Image filtering pipeline to obtain quad centers. (a) input image; (b) ridgeness image given by computing Eq. (2.3) at each pixel, here at scale  $t = 0.8$ . This amplifies line-like structures. A sliding-window maximum filter gives the locally-normalized brightness of each pixel, shown in (c), which is used to get rid of artifacts in the white background in (d). Binary thresholding is performed, (e), followed by connected component analysis, which gives the quad center locations in the image, shown in (f) overlaid on top of the input image.

scale depends on the size of the printed lines, the distance of the textile to the camera, and camera resolution. Scale-agnostic detection can be performed within the framework of scale space theory [Lin07; Lin14] as follows. Given a gray-scale image  $I(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}_+$ , its scale space representation  $L(x, y, t) : \mathbb{R}^2 \times \mathbb{R}_+ \rightarrow \mathbb{R}_+$  is constructed by means of convolution with a Gaussian kernel of variance  $\sigma = \sqrt{t}$ ,

$$L(x, y, t) = \int_u \int_v \frac{1}{2\pi t} \exp\left(-\frac{u^2 + v^2}{2t}\right) I(x - u, y - v) du dv \quad . \quad (2.1)$$

$t$  is the so-called scale parameter. Specific image structures can be detected in scale space based on spatial derivatives, commonly denoted in the literature as  $L_{x^a y^b}(x, y, t) = \frac{\delta L(x, y, t)}{\delta x^a y^b}$ , with  $(a, b) \in \mathbb{N}$  giving the order of the derivative in each direction [Lin07].

To distinguish quad centers from surrounding lines, the scale-adapted Hessian [MS04] is computed at each pixel,

$$H(x, y) = t \cdot \begin{bmatrix} L_{xx}(x, y, t) & L_{xy}(x, y, t) \\ L_{xy}(x, y, t) & L_{yy}(x, y, t) \end{bmatrix} \quad , \quad (2.2)$$

where  $L_{xx}, L_{yy}, L_{xy}$  are the second-order spatial partial derivatives. The scale parameter  $t$  acts as a normalization factor and ensures that values are comparable across scales. A larger  $t$  generally blurs the image more, and this blurring decreases the average response of the spatial derivatives, thereby reducing the range of values in Eq. (2.2). In our case, whenever the scale  $t$  is changed (e.g. tuned to the specific width of the lines by the user), this would interfere with successive steps in the pipeline (e.g. thresholding). Scale normalization prevents this.

To detect if a specific pixel is part of a line, we can look at the curvature around that pixel. The curvature in scale space is computed by the largest eigenvalue of the Hessian

$$\lambda = \frac{1}{2}t \left( L_{xx} + L_{yy} + \sqrt{(L_{xx} - L_{yy})^2 + 4L_{xy}^2} \right) . \quad (2.3)$$

This measure successfully amplifies dark ridges (lines), blobs, and crossing points. It is computed for every pixel, Figure 2.2(b).

**Local filter normalization (LN).** The ridgeness measure Eq. (2.3) unfortunately also responds to bright image structures. This can be seen on the left-hand side in Figure 2.2(b), where lots of structure is detected in the slightly noisy background. These structures are then detected as quad centers, which in turn are passed down the pipeline to be matched to the M-Array pattern, only to be detected as false positives. These unnecessary checks decrease runtime speed, so we need to filter such too-bright image structures early on. This is done by computing the maximal brightness of the neighborhood around every pixel, an operation that is implemented using a sliding-window maximum filter across the image. Dividing the original image greyscale values by this filter response gives the locally-normalized image with brightness  $I_l(x, y) \in [0, 1]$ , Figure 2.2(c). To attenuate bright regions, the normalized brightness  $I_l$  is combined with the ridgeness measure of Eq. (2.3) to obtain  $R(x, y) = (1 - I_l(x, y)) \cdot \lambda(x, y, t)$ . Figure 2.2(d) shows the resulting image: lines that were too bright have been successfully filtered. Finally, a binary image can be obtained by thresholding  $R$  with a fixed threshold, Figure 2.2(e). Then, connected components are labeled in the binary image using an efficient GPU implementation [HLP10]. The centers of these connected components are the desired quad centers, Figure 2.2(f), which can next be matched to the textile pattern.

### 2.2.2. Robust Color Classification

In the previous step, quad centers were detected and located in images showing the textile pattern. Next, the color code of each quad must be determined. This step converts the observed RGB color to a known color class (for example to *red*, *green*, etc.). After this color classification step, each quad

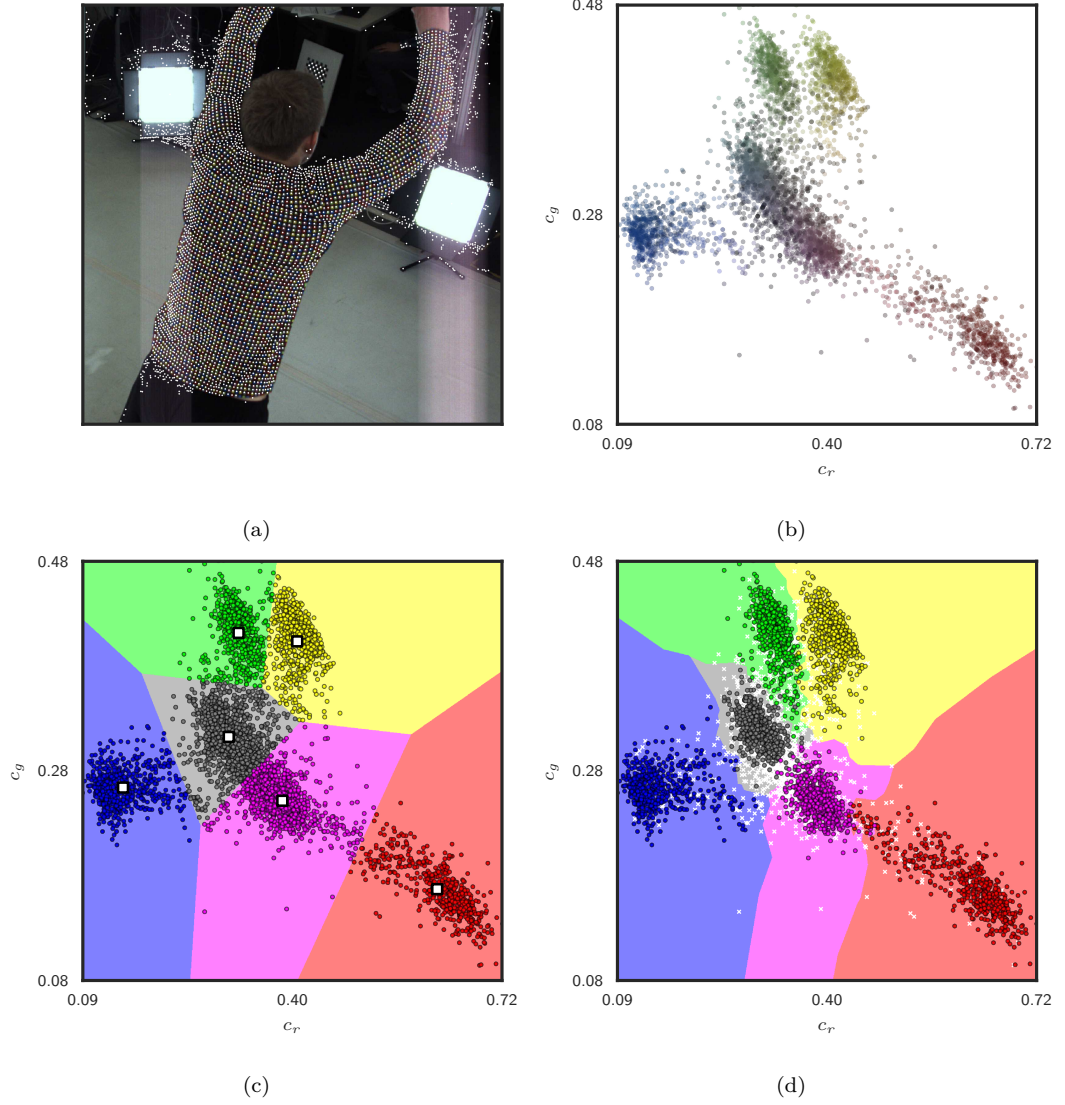


Figure 2.3.: Robust color classification pipeline. (a) Input image showing detected quad centers as white dots. (b) The RGB color of each quad center is converted to RG-chromaticity color space. Plotting the colors in this space reveals distinct color clusters, for example a green and yellow cluster. (c) Each quad center color is initially classified based on the proximity to the cluster centers. K-means finds these cluster locations (white squares). The background here shows the classification regions, corresponding to the Voronoi regions of the cluster centers. (d) After color classification refinement, the classification regions are much more complex. This step also sorts out quad centers that are not part of the M-array pattern (white crosses).

is uniquely identified by matching the color codes of its  $3 \times 3$  neighborhood to the known M-Array pattern.

Unfortunately, the RGB colors in an image can be very dissimilar to the colors in the template pattern. This color distortion has numerous reasons: color inaccuracies of the textile printer, unpredictable color attenuation during printing on textile, inaccurate, and noisy image sensors. Even within the same image, colors are heavily influenced by environment lighting, shadows, or even by indirect lighting. These conditions considerably complicate automatic color classification.

To mitigate some of these problems, a color space that is invariant to illumination changes is used. The *RG-Chromaticity* turned out to work well [GGWG12]. Its two dimensions  $(c_r, c_g)$  are computed from an  $(r, g, b)$  tuple of the red, green, and blue component of the input pixel color by simple normalization

$$c_r = \frac{r}{r + g + b} \quad , \quad c_g = \frac{g}{r + g + b} \quad .$$

Classification inside this color space improves accuracy but still produces many falsely matched colors in practice. The proposed algorithm can automatically adopt observed colors by a *learning* procedure. The algorithm analyzes the distribution of colors. The color distribution constitutes distinct and separate color clusters, an observation illustrated in Figure 2.3(b), where the colors of all detected quad centers are plotted in RG-chromaticity color space. The clearly visible clusters can be discovered automatically with K-means. The number of clusters  $K$  to detect is set to the known number of colors of the pattern template.

Next, each of the  $K$  discovered cluster centers must be assigned exactly one known template color. Given the cluster center colors  $\mathbf{a}^{(i)} \in \mathbb{R}^2$ ,  $i = 1, \dots, K$  and the template colors  $\mathbf{b}^{(j)} \in \mathbb{R}^2$ ,  $j = 1, \dots, K$ ,  $\mathbf{a}^{(i)}$  and  $\mathbf{b}^{(j)}$  being RG-chromaticity colors, we aim to find an optimal assignment of cluster color to template color by solving

$$\underset{\mathbf{X} \in \mathcal{S}_K}{\text{minimize}} \quad \sum_{i=1}^K \sum_{j=1}^K \mathbf{X}_{i,j} \left\| \mathbf{a}^{(i)} - \mathbf{b}^{(j)} \right\|_2 \quad . \quad (2.4)$$

The optimal solution  $\mathbf{X}^*$  must be a unique one-to-one assignment and is thus constrained to the set  $\mathcal{S}_K$  of  $K \times K$  permutation matrices. The optimization problem of Eq. (2.4) can be interpreted as a maximum-weight bipartite matching problem, which in turn can be solved efficiently [Gal86]. The resulting optimal cluster assignment facilitates classification of all quads in the image: every quad is assigned the color of its nearest cluster center, Figure 2.3(c).

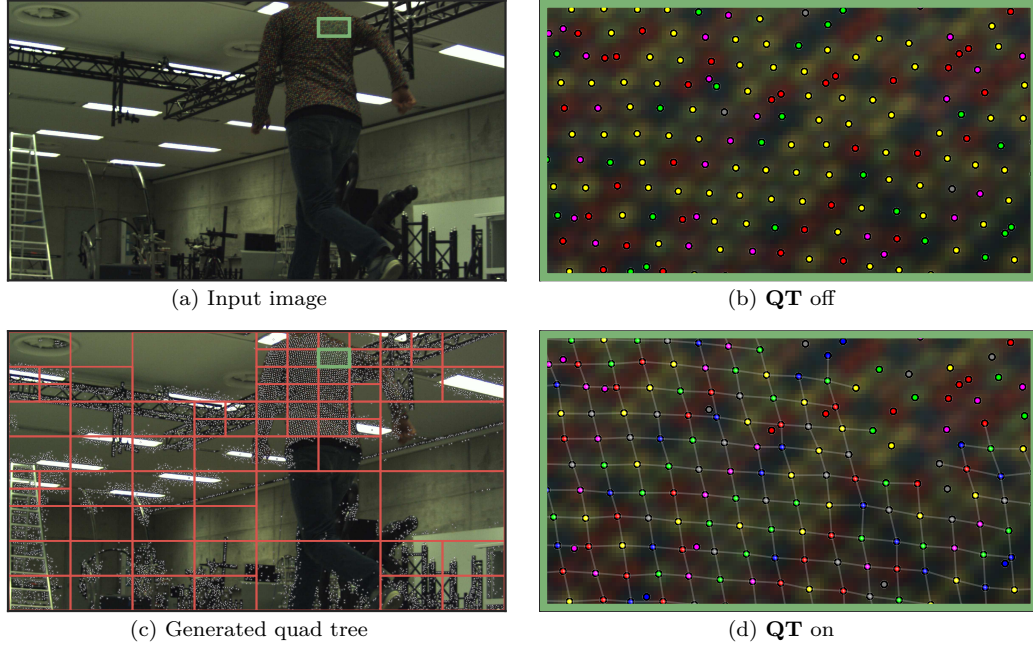


Figure 2.4.: Robust color classification result. (a) Input image showing bad white balance, very small quad pattern and lots of background clutter. (b) Color classification result, zoomed in on marked area. The quad centers are well detected in the blurry image, but the color classifier is not able to adapt to the minute differences between green, yellow, and grey quads. For example, all green quad centers are classified as yellow. The color classifier is disturbed by many quad center detections in the background. (c) From all detected quad corner hypotheses (white points), a quad tree is built (red lines). The color classification and adaptation is then run in each leaf cell. (d) Color classification result with the proposed quad tree extension: The quad centers are classified correctly, making it possible to match them to the M-Array pattern, as indicated by white lines.

**Local color-classification in a quad tree (QT).** The above-described technique works astonishingly well in practice but has limitations when dealing with images showing cluttered backgrounds or bad lighting, Figure 2.4(a). In such cases, many putative quad centers are accidentally detected in the background even though they are not actually on the textile. These false detections do not pose a problem in principle since the chance of some background quads coincidentally forming a correct M-array  $3 \times 3$  color code is extremely unlikely. But the presence of many outliers distorts the color distribution discussed previously. It breaks the K-Means clustering step described above and thus deteriorates correct color classification even in the foreground, Figure 2.4(b). The proposed algorithm circumvents this by adding a step that partitions the image into small sub-regions using a quad-tree data structure, Figure 2.4(c). The partitioning approximately separates background regions from

foreground regions, thereby allowing correct color classification in foreground regions, Figure 2.4(d). After this step, each quad is equipped with a hypothetical color class (e.g. *red* or *green*). The next step is to uniquely identify quad centers by matching them to the M-Array codes.

### 2.2.3. M-Array Matching

The quad centers are now matched to the M-Array codes by checking the classified color of their  $(3 \times 3)$  neighbors. For this step, an algorithm first presented in [SSK+05] is used. For this algorithm to succeed, quad centers are required to be correctly color-classified, otherwise they cannot be matched to the M-Array. Yet, unmatched quads may have simply been falsely color-classified in the previous step. For example, a yellow quad cannot be matched when it was mistakenly classified as *green* because this misclassification invalidates its  $3 \times 3$  M-Array code.

**Local color re-classification (LR).** I propose to re-label the color of such *unmatched* quads by examining nearby *matched* quads. These matched quads are very likely to be color-classified correctly, because their color was confirmed by the M-Array and their  $3 \times 3$  neighborhood. Matched quads are treated as positive examples and unmatched quads are re-classified using nearest neighbor classification with the matched quads acting as training set. After this re-classification, the M-Array matching of [SSK+05] is repeated with the corrected color classification. Matching and re-classification are iteratively repeated until no new quads can be added. This nearest-neighbor classification strategy allows adapting to complex non-linear, non-convex color class separation boundaries, Figure 2.3(d). This improvement ultimately yields more correctly matched quads to be used for 3D reconstruction.

**Auto-Detection of scale (AD).** The detection pipeline described so far requires tuning the scale parameter  $t$  to the specific dataset. A simple technique can completely remove this tuning step: The complete pipeline is run through different values of  $t$ , in steps of 0.3 from 0.6 to 5.0. This gives, for each scale  $t$ , a set of locations of the quads in the pattern. To merge the results of the same quad located at different scale  $t$ , the geometric median of their 2D locations is selected. In contrast to the geometric mean, the geometric median is robust to spurious outliers. The auto-detection step increases runtime but allows detecting more quads. Enabling this feature is always a tradeoff between runtime speed and accuracy.

### 2.2.4. Results

Table 2.1 summarizes results from the 3D reconstruction of three different datasets. Each dataset was processed in 5 separate runs. For each run, different algorithm extensions were disabled to test




|  | run | Algorithm Options |    |    |    | Results    | Computation Time |         |
|--|-----|-------------------|----|----|----|------------|------------------|---------|
|  |     | QT                | LR | LN | AD | Num. Quads | per image        | Full    |
| <br>jumpthrow | 0   | -                 | -  | ✓  | -  | 22 649     | 168 ms           | 0m 37s  |
|  | 1   | ✓                 | -  | ✓  | -  | 245 794    | 125 ms           | 0m 24s  |
|  | 2   | ✓                 | ✓  | -  | -  | 267 090    | 1974 ms          | 5m 17s  |
|  | 3   | ✓                 | ✓  | ✓  | -  | 267 637    | 252 ms           | 0m 44s  |
|  | 4   | ✓                 | ✓  | ✓  | ✓  | 286 756    | 1187 ms          | 3m 12s  |
| <br>sock      | 0   | -                 | -  | ✓  | -  | 97 787     | 88 ms            | 0m 34s  |
|  | 1   | ✓                 | -  | ✓  | -  | 139 104    | 79 ms            | 0m 15s  |
|  | 2   | ✓                 | ✓  | -  | -  | 141 092    | 1191 ms          | 3m 4s   |
|  | 3   | ✓                 | ✓  | ✓  | -  | 144 554    | 145 ms           | 0m 25s  |
|  | 4   | ✓                 | ✓  | ✓  | ✓  | 154 674    | 812 ms           | 2m 6s   |
| <br>torrero   | 0   | -                 | -  | ✓  | -  | 59         | 403 ms           | 6m 2s   |
|  | 1   | ✓                 | -  | ✓  | -  | 2 278 158  | 108 ms           | 5m 53s  |
|  | 2   | ✓                 | ✓  | -  | -  | 2 449 422  | 1230 ms          | 13m 58s |
|  | 3   | ✓                 | ✓  | ✓  | -  | 2 456 458  | 233 ms           | 6m 21s  |
|  | 4   | ✓                 | ✓  | ✓  | ✓  | 2 516 449  | 1135 ms          | 11m 15s |

Table 2.1.: Algorithm performance for different datasets. To test the influence of individual contributions, different runs through the same dataset were performed, with each run enabling or disabling certain contributions presented in this thesis. Run 3 and 4 had all proposed contributions enabled, achieving the highest number of 3D reconstructed quads. See text for details.

if the described contributions yield better 3D reconstruction results. The following contributions were tested. **QT**: use a quad tree (leaf size 50) to perform color classification locally, as explained in Section 2.2.2 and visualized in Figure 2.4(a); **LR**: local color re-classification as described in Section 2.2.3 and visualized in Figure 2.3(d); **LN**: local filter normalization using a maximum filter, described in Section 2.2.1 and visualized in Figure 2.2(c); **AD**: auto-detection of scale parameter  $t$ .

The datasets show very different deformed textiles: a loose tricot on an athlete performing a typical handball motion ("jumpthrow"); a tight-fit sleeve around the lower leg ("sock"); and a large sheet of textile, wiggled to create dynamic folds and wrinkles ("torrero"). The datasets "jumpthrow" and "sock" were captured with 16 cameras and  $1600 \times 1200$  pixels resolution running at 30 Hz. The "torrero" dataset was captured with 12 cameras and  $1920 \times 1080$  pixels resolution running at 100 Hz. Figure 2.5 visualizes the result for one frame of the "torrero" dataset.

In Table 2.1 we see that for **run 0**, without **QT** and without **LR**, the algorithm reconstructs a very low number of quads. With QT turned on, in **run 1**, we achieve better results and even faster computation<sup>3</sup>. Results of **run 2** and **run 3** both have **LR** turned on, which yields even more 3D-reconstructed quads across all datasets. However, in **run 2** we can see that turning off the **LF**

<sup>3</sup>The speedup is in the K-Means routine which now works on many small sets of marker hypotheses (one for each quad tree leaf node). This converges quicker than running the K-Means on the full set of detected quads.

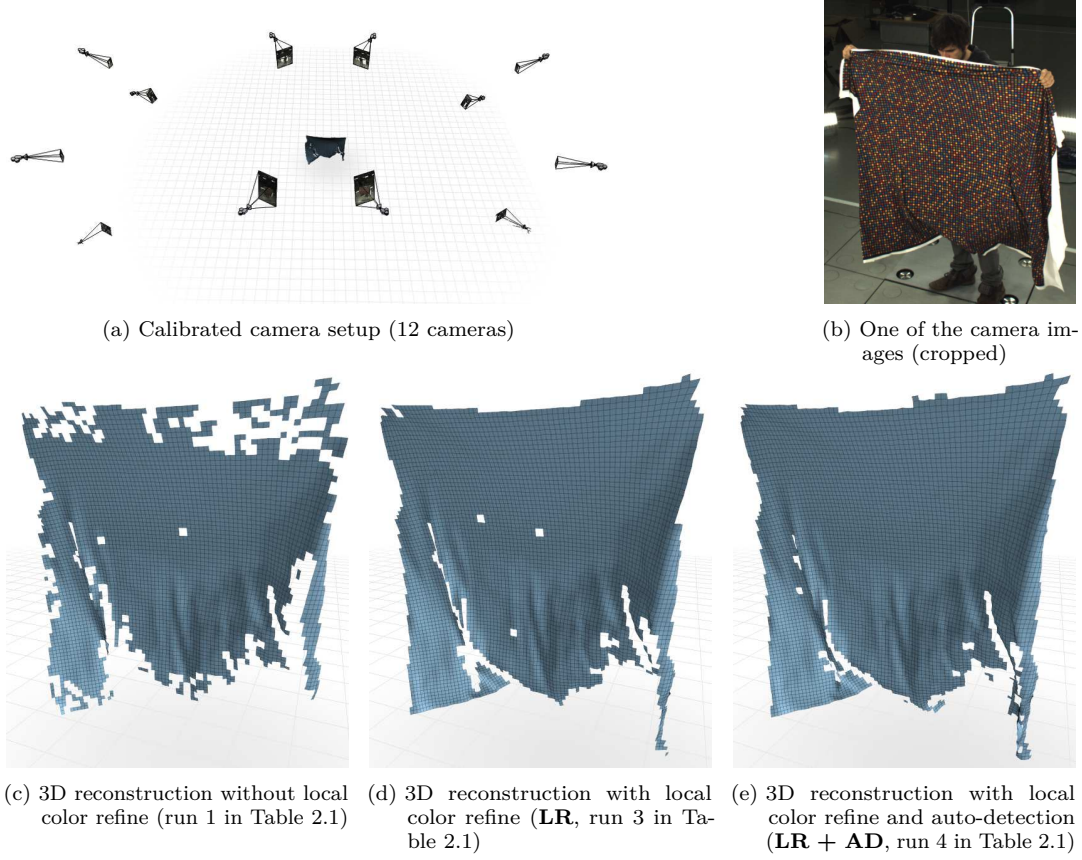


Figure 2.5.: Results visualized for one frame from the "torrero" dataset. Between (c) and (d) we can see that many more quads are 3D-reconstructed correctly if the proposed local re-classification method is used. (e) shows additional improvement due to the proposed auto-detection (**AD**). Run 0 in Table 2.1 without **QT** is not shown as it was not able to reconstruct a single quad.

(local filter normalization) results in very long runtimes since more false positive quads are extracted, Figure 2.2(c). **Run 3** has all proposed features enabled, yielding solid results. **Run 4** can improve upon that using the auto-detection of the scale parameter (**AD**), albeit at longer computation times.

**A note on runtime.** 3D reconstruction was performed on an Intel Core i7-3770 with 16GB RAM and NVIDIA GTX650 GPU running on Ubuntu 14.04. The implementation uses C++, OpenCV and CUDA 6.0. At the time of publication of this thesis, this setup is already outdated. But the implementation scales to an arbitrary number of CPU cores, each CPU thread working on a single image. The GPU performs the image filtering operations, Section 2.2.1. The quad detection algorithm

can deal with an arbitrary number of GPUs. The "per image" times in Table 2.1 represent the average time taken for the quad detection pipeline, for a single image, on a single CPU core with a single GPU device. On the tested machine, the GPU operations typically took between 10 – 20 ms. If we want to scale up operations, a guess would be to add one GPU per 10 additional cores. Additional fine-tuning and optimization of the kernel code can yield an additional speedup. This optimized pipeline can be optimized for and deployed on a modern multi-CPU, multi-GPU system. With such a fast system and a limited amount of 2 – 6 cameras running at 30 Hz I believe that the 3D reconstruction can be done in realtime.

### 2.3. 3D Reconstruction using Random Dot Patterns and Multiview Constraints

Printing a regular (quad-) pattern on surfaces is impractical for objects like human skin or shoes. On the other hand, some of these surfaces have an intrinsic texture that can be tracked, e.g. pores on a face [BHB+11]. However, texture-based tracking becomes impossible if surfaces lack such a structured texture, if there is fast motion, or if the resolution of the cameras is too low. These are practically relevant scenarios where these kinds of methods fail. In those cases, preparing the surface with a specific texture or pattern cannot be avoided. Therefore, such a preparation step should be as simple and flexible as possible. I propose a method that requires only randomly placed dots on the surface, as shown in Figure 2.6.

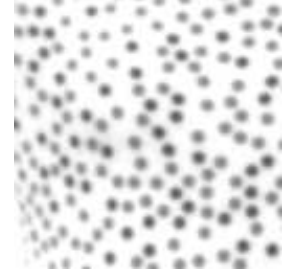


Figure 2.6.: The random dot pattern

Applying a pattern of random dots is easy, for example by sprinkling them onto the surface using a pen. Reconstructing the 3D position of these dots requires that the dots can be identified across different camera images. However, all marker dots look almost identical in the images, making this matching task highly ambiguous. An optimization method can break those ambiguities by finding a matching that best respects the relation of the neighboring dots. This leads to graph matching approaches that are, however, suited for matching between two cameras only [Tre13, Ch. 5: "Correspondence Problems"]. The proposed method improves significantly upon the state of the art by solving for matches between all cameras simultaneously. This effectively produces a 3D reconstruction of the dots that is also maximally consistent.

### 2.3.1. Prelude: Stereo Matching using Graph Matching

Graph Matching can be used to match marker dots between two cameras,  $c_1$  and  $c_2$ . I now formalize this two-camera setting to set the stage for the multi-camera setting discussed later. Let  $\mathcal{S}$  be the set of cameras,  $\mathcal{S} = \{c_1, c_2\}$ . The marker dots can be efficiently detected in the images as multi-scale extrema in the image pyramid, for example using the determinant of the Hessian approach [Lin94; MS02; MS04; BTV06]. Those locations are stereo-rectified [FTV00], a process that transforms epipolar lines to lie parallel to the  $x$ -axis so that corresponding points will have approximately the same  $y$ -coordinate. This greatly simplifies the upcoming equations. Let  $\mathcal{P}^{(c_1)} = \{\mathbf{p}_i^{(c_1)} \in \mathbb{R}^2 \mid i \in 1, \dots, P^{(c_1)}\}$  and  $\mathcal{P}^{(c_2)} = \{\mathbf{p}_j^{(c_2)} \in \mathbb{R}^2 \mid j \in 1, \dots, P^{(c_2)}\}$  be those stereo-rectified dot locations to be matched between the first and the second image, respectively, where  $P^{(c_1)}$  and  $P^{(c_2)}$  give the number of points in each image. The usual graph matching formulation allows every point in  $\mathcal{P}^{(c_1)}$  to match any other point in  $\mathcal{P}^{(c_2)}$ . But in stereo matching, only few matches respect the epipolar constraint<sup>4</sup>, and we are only interested in those valid matches. Hence,  $m_{\iota(i,j)}^{(c_1,c_2)} = (\mathbf{p}_i^{(c_1)}, \mathbf{p}_j^{(c_2)})$  denote a match between point  $\mathbf{p}_i^{(c_1)}$  and  $\mathbf{p}_j^{(c_2)}$ . The index mapping  $\iota : (\mathbb{N}, \mathbb{N}) \rightarrow \mathbb{N}$  uniquely numbers each match, enabling us to write the set of all possible matches as  $\mathcal{M}^{(c_1,c_2)} = \{m_n^{(c_1,c_2)} \mid n = \iota(i,j), n \in 1, \dots, M^{(c_1,c_2)}\}$ . The actual implementation of  $\iota$  does not affect the algorithm, it just provides us with a one-dimensional numbering of the matches.

To enable partial matching, dummy matches are included in  $\mathcal{M}^{(c_1,c_2)}$ : for every point  $\mathbf{p}_i^{(c_1)} \in \mathcal{P}^{(c_1)}$  there is an empty match  $(\mathbf{p}_i^{(c_1)}, \emptyset)$ , and analogously for the points in the second camera,  $(\emptyset, \mathbf{p}_j^{(c_2)})$ . This explicit handling of partial matching using dummy matches is unique to our method. In fact, partial matching is frequently overlooked in state-of-the-art graph matching methods, most works assume that at least one of the point sets can be completely matched [CLL10; LH05; LL11; GR96; KKBL15; LHS09; ZS08]. Only Torresani et al. [TKR13] add an occlusion term but within a completely different optimization framework.

From here on, notation is shortened by denoting a variable related to the stereo pair  $(c_1, c_2)$  with a bar, e.g.  $\bar{m}_n = m_n^{(c_1,c_2)}$  and  $\bar{\mathcal{M}} = \mathcal{M}^{(c_1,c_2)}$ . Every match  $\bar{m}_n$  corresponds to a reconstructed 3D point  $\bar{\mathbf{r}}_n = \mathbf{r}_{\iota(i,j)}^{(c_1,c_2)} \in \mathbb{R}^3$ , obtained by triangulation from the given 2D points of the match,  $\mathbf{p}_i^{(c_1)}$  and  $\mathbf{p}_j^{(c_2)}$ . Notice that the set  $\bar{\mathcal{M}}$  contains all potential matches so it contains all the ambiguities even if epipolar constraints are properly respected.

The match ambiguities can be partly broken by introducing a regularization energy between related matches. Consider two matches  $\bar{m}_{\iota(i,j)} = (\mathbf{p}_i^{(c_1)}, \mathbf{p}_j^{(c_2)})$  and  $\bar{m}_{\iota(a,b)} = (\mathbf{p}_a^{(c_1)}, \mathbf{p}_b^{(c_2)})$ , as shown in

<sup>4</sup>We can simply check the reprojection error of every possible match. If the error is above a user-defined threshold, we can omit the corresponding stereo match. The reprojection threshold is related to the quality of the calibration. Better calibration allows for lower values of this threshold.

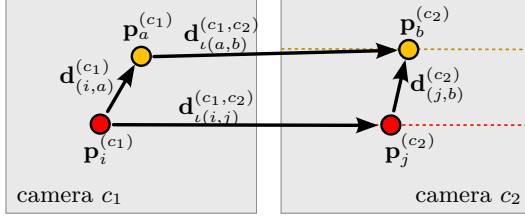


Figure 2.7.: Pairwise potentials between two stereo matches  $m_{\iota(i,j)}^{(c_1,c_2)}$  and  $m_{\iota(a,b)}^{(c_1,c_2)}$ . The two matches fulfil the epipolar constraint:  $\mathbf{p}_j^{(c_2)}$  lies on the (red dashed) epipolar line of  $\mathbf{p}_i^{(c_1)}$ . Both matches are compatible if their flow vectors  $\mathbf{d}_{\iota(i,j)}^{(c_1,c_2)}$  and  $\mathbf{d}_{\iota(a,b)}^{(c_1,c_2)}$  are similar. This is expressed in Eq. (2.6).

Figure 2.7. Both matches constitute a flow vector  $\mathbf{d}_{\iota(i,j)}^{(c_1,c_2)} = \mathbf{p}_j^{(c_2)} - \mathbf{p}_i^{(c_1)}$  and  $\mathbf{d}_{\iota(a,b)}^{(c_1,c_2)} = \mathbf{p}_b^{(c_2)} - \mathbf{p}_a^{(c_1)}$ . Since both cameras are stereo-rectified, the magnitude of this flow vector is essentially the disparity of the match. This value can be set in relation to the proximity of the points which can be computed from the length of the vector  $\mathbf{d}_{(i,a)}^{(c_1)} = \mathbf{p}_a^{(c_1)} - \mathbf{p}_i^{(c_1)}$  in camera  $c_1$  and  $\mathbf{d}_{(j,b)}^{(c_2)} = \mathbf{p}_b^{(c_2)} - \mathbf{p}_j^{(c_2)}$  in camera  $c_2$ . If both vectors are short, and hence if the involved dots are close to each other in both camera images, then the flow vectors (thus, their disparities) should be similar in most cases. This smoothness assumption is common in stereo matching [SS02] and, roughly translated, it means that we expect the reconstructed 3D surface to be smooth. To measure the degree of smoothness between two matches we define an *affinity* measure between them

$$\bar{\alpha}(i, j, a, b) = \exp \left( - \frac{\left\| \mathbf{d}_{ij}^{(c_1,c_2)} - \mathbf{d}_{ab}^{(c_1,c_2)} \right\|_2}{\lambda \left( \left\| \mathbf{d}_{ia}^{(c_1)} \right\|_2 + \left\| \mathbf{d}_{jb}^{(c_2)} \right\|_2 \right)} \right), \text{ with } \lambda \in \mathbb{R}_+ . \quad (2.5)$$

Similar affinity terms are used in related point matching approaches, e.g. [CLL10]. The user-defined parameter  $\lambda$  dictates the strength of the regularization. For common values of  $\lambda$ , the affinity term quickly goes to zero as the denominator becomes small, which is a desired effect for matches far apart from each other (we do not want matches to influence each other when they are in distant parts of the image). We can prevent evaluating and storing the small affinity values arising in such cases by checking if the involved points are both inside each other's closest neighborhood  $\mathcal{N}$  in the same camera image, similar to [TKR13]. This idea is respected when all affinity values are collected into an affinity matrix:

$$\bar{\mathbf{A}}_{\iota(i,j),\iota(a,b)} = \begin{cases} \bar{\omega}(i, j) & \text{if } i = a \wedge j = b , \\ 0 & \text{if } \mathbf{p}_a^{(c_1)} \notin \mathcal{N}(\mathbf{p}_i^{(c_1)}) \vee \mathbf{p}_b^{(c_2)} \notin \mathcal{N}(\mathbf{p}_j^{(c_2)}) , \\ \bar{\alpha}(i, j, a, b) & \text{otherwise} . \end{cases} \quad (2.6)$$

The first case covers compatibility between a match and itself (so-called unary potentials). This data term can include a feature point similarity,

$$\bar{\omega}(i, j) = \begin{cases} \omega_{\text{dummy}} & \text{if } \mathbf{p}_i^{(c_1)} = \emptyset \vee \mathbf{p}_j^{(c_2)} = \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (2.7)$$

The tuning parameter  $\omega_{\text{dummy}}$  controls the strength of the data term for dummy matches. Increasing this parameter will effectively filter out bad matches, since dummy matches will seem like better matches to the optimizer. For non-dummy matches, the data term is zero because all random dots are equally similar. For these cases, there is no data term, only a smoothness term. Notice that the neighborhood check (second case in Eq. (2.7)) makes the affinity matrix  $\mathbf{A}_{i(i,j), i(a,b)}^{(c_1, c_2)}$  very sparse (depending on the predefined size of the neighborhood  $\mathcal{N}$ ), a fact that the proposed optimization method can make effective use of.

Based on the affinity, the proposed approach finds an optimal matching configuration, expressed as an assignment vector  $\mathbf{x}^{(c_1, c_2)} = \bar{\mathbf{x}} \in \{0, 1\}^{\bar{M}}$ , where  $\bar{x}_{i(i,j)} = 1$  when match  $\bar{m}_{i(i,j)}$  is part of the valid 3D reconstruction, and  $\bar{x}_{i(i,j)} = 0$  if not. The best consistent assignment with respect to the affinity measure can be found by solving a constrained quadratic integer program

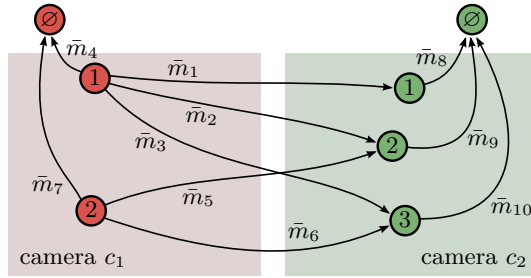
$$\underset{\bar{\mathbf{x}} \in \bar{\Pi}}{\text{maximize}} \quad \bar{\mathbf{x}}^\top \bar{\mathbf{A}} \bar{\mathbf{x}} \quad (2.8)$$

The constraint set  $\bar{\Pi} = \Pi^{(c_1, c_2)}$  enforces a binary one-to-one matching so that one point from  $\mathcal{P}^{(c_1)}$  can match at most one point in  $\mathcal{P}^{(c_2)}$ , and vice versa. We define  $\bar{\Pi}$  using a constraint matrix  $\mathbf{C}^{(c_1, c_2)} = \bar{\mathbf{C}} \in [0, 1]^{(P^{(c_1)} + P^{(c_2)}) \times \bar{M}}$  such that

$$\Pi^{(c_1, c_2)} = \left\{ \bar{\mathbf{x}} \in \{0, 1\}^{\bar{M}} \mid \bar{\mathbf{C}} \bar{\mathbf{x}} = \mathbf{1} \right\} \quad (2.9)$$

Let us now define the constraint matrix  $\bar{\mathbf{C}}$ , first by using an example, then for the general case. The example is illustrated on the right where 2 points in  $c_1$  shall be matched to 3 points in  $c_2$ . The arrows denote valid matches with respect to the epipolar constraint, including dummy matches (arrows to  $\emptyset$  and  $\emptyset$ ).

One can see that point  $\mathbf{p}_1^{(c_1)}$  (1) is used in 4 of the possible matches, i.e.  $\bar{m}_1 = (p_1^{(c_1)}, p_1^{(c_2)})$ ,  $\bar{m}_2 = (p_1^{(c_1)}, p_2^{(c_2)})$ ,  $\bar{m}_3 = (p_1^{(c_1)}, p_3^{(c_2)})$ , and the



dummy match  $\bar{m}_4 = (p_1^{(c_1)}, \emptyset)$ . All these matches are in one-to-one conflict - they cannot be true simultaneously. So the assignment values of these matches are required to fulfil the condition  $\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4 \stackrel{!}{=} 1$ . Similarly, we have  $\bar{x}_5 + \bar{x}_6 + \bar{x}_7 \stackrel{!}{=} 1$  for point  $\mathbf{p}_2^{(c_1)}$  (2). This is expressed in the first two rows of the constraint matrix  $\bar{\mathbf{C}}$ :

$$\begin{array}{c}
 \text{matches from } \mathbf{p}_1^{(c_1)} \text{ (1)} \quad \text{matches from } \mathbf{p}_2^{(c_1)} \text{ (2)} \quad \text{matches irrelevant in } c_1 \\
 \begin{array}{c} \bar{m}_1 \quad \bar{m}_2 \quad \bar{m}_3 \quad \bar{m}_4 \quad \bar{m}_5 \quad \bar{m}_6 \quad \bar{m}_7 \quad \bar{m}_8 \quad \bar{m}_9 \quad \bar{m}_{10} \end{array} \\
 \text{row 1:} \quad \left[ \begin{array}{cccc|ccc|ccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right] \\
 \text{row 2:} \quad \left[ \begin{array}{cccc|ccc|ccc} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{array} \right] . \quad (2.10)
 \end{array}$$

The remaining rows of the constraint matrix express the one-to-one constraints of points in camera  $c_2$ , e.g. for  $\mathbf{p}_3^{(c_2)}$  (3) it is required that  $\bar{x}_3 + \bar{x}_6 + \bar{x}_{10} \stackrel{!}{=} 1$ . The last three rows are thus

$$\begin{array}{c}
 \bar{m}_1 \quad \bar{m}_2 \quad \bar{m}_3 \quad \bar{m}_4 \quad \bar{m}_5 \quad \bar{m}_6 \quad \bar{m}_7 \quad \bar{m}_8 \quad \bar{m}_9 \quad \bar{m}_{10} \\
 \text{row 3:} \quad \left[ \begin{array}{cccc|ccc|cc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \right] \\
 \text{row 4:} \quad \left[ \begin{array}{cccc|ccc|cc} 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \end{array} \right] \\
 \text{row 5:} \quad \left[ \begin{array}{cccc|ccc|cc} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{array} \right] . \quad (2.11)
 \end{array}$$

In general, the constraint matrix  $\bar{\mathbf{C}}$  is of size  $(P^{(c_1)} + P^{(c_2)}) \times \bar{M}$ . There is one row for each point (from both cameras), and each column corresponds to one of the matches in  $\bar{\mathcal{M}}$ . In each row, the entries are set to 1 for columns corresponding to matches coming from this rows' point.

Interestingly, if the points are interpreted as graph nodes, and the matches as undirected edges, then the constraint matrix equals the incidence matrix of the graph of matches, but without the rows corresponding to dummy nodes. In the literature [GR96; LH05; LHS09; CLL10; LFF+14; KKBL15], Eq. (2.9) is usually expressed by means of an assignment matrix  $\bar{\mathbf{X}} \in \{0, 1\}^{P \times P}$  instead of an assignment vector<sup>5</sup>. This constrains the assignment matrix to be a binary permutation matrix. However, the representation of Eq. (2.9) is more flexible and better promotes the inclusion of interactions between many cameras, as I shall show next.

### 2.3.2. Multiview Constraints

Until now, we only considered 3D reconstruction with two cameras. How can we integrate multiple camera views? A trivial solution would be: solve several stereo correspondence problems for all available pairs of cameras; then merge the stereo reconstructions into a common world coordinate frame.

<sup>5</sup>Usually, this requires introducing artificial slack variables to ensure  $P = P^{(c_1)} = P^{(c_2)}$ .

Experimental results indicate that this obvious approach fails to provide convincing reconstructions in practice, presumably because it does not take all available information from all cameras into account.

I therefore propose to unite the distinct stereo matching instances into one single optimization procedure. This approach finds the optimal, complete 3D reconstruction from all cameras. To this end, the distinct stereo matching instances are first assembled together into an optimization problem of the form of Eq. (2.8). Let  $\mathcal{S}$  be the set of camera pairs. For example, a 3-camera system generates the pairs  $\mathcal{S}^{(c_1, c_2, c_3)} = \{(c_1, c_2), (c_1, c_3), (c_2, c_3)\}$ . Every pair has a separate affinity matrix  $\mathbf{A}^{(c_d, c_e)}$ , a separate constraint matrix  $\mathbf{C}^{(c_d, c_e)}$  and a separate assignment vector  $\mathbf{x}^{(c_d, c_e)}$ , with  $(c_d, c_e) \in \mathcal{S}$ . The assignment vectors  $\mathbf{x}^{(c_d, c_e)}$  are combined by stacking them on top of each other,  $\mathbf{x}^\top = ((\mathbf{x}^{(c_1, c_2)})^\top, \dots, (\mathbf{x}^{(c_d, c_e)})^\top)$ . The affinity matrices  $\mathbf{A}^{(c_d, c_e)}$  are filled as in Eq. (2.6) and combined in blocks to form a global affinity matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}^{(c_1, c_2)} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A}^{(c_d, c_e)} \end{bmatrix}. \quad (2.12)$$

The constraint matrices  $\mathbf{C}^{(c_d, c_e)}$  can be assembled analogously into a global  $\mathbf{C}$ . The matches are collected for all stereo pairs into the global set of matches

$$\mathcal{M} = \bigcup_{(c_d, c_e) \in \mathcal{S}} \mathcal{M}^{(c_d, c_e)}, \quad (2.13)$$

of size  $M = |\mathcal{M}|$ . Reindexing needs to be performed such that any match can be identified across all stereo pairs, e.g. using a unique re-ordering mapping  $\zeta : (\mathbb{N}, \mathbb{N}, \mathbb{N}, \mathbb{N}) \rightarrow \mathbb{N}$  so that  $m_n = m_{\zeta(i, j, d, e)} = m_{\iota(i, j)}^{(c_d, c_e)} \in \mathcal{M}$ . When we assemble the matches and matrices from the separate camera pairs like this, we still do not generate any interactions between those separate camera pairs. If we solve the assembled graph matching problem it would still be equivalent to solving the distinct stereo problems separately.

However, this global representation paves the way for introducing multiview conflicts, interactions between matches from different camera pairs. Consider a stereo match  $m_{\iota(i, j)}^{(c_1, c_2)} = (\mathbf{p}_i^{(c_1)}, \mathbf{p}_j^{(c_2)})$  going from camera  $c_1$  to  $c_2$ , and a match  $m_{\iota(i, k)}^{(c_1, c_3)} = (\mathbf{p}_i^{(c_1)}, \mathbf{p}_k^{(c_3)})$  involving the same dot  $\mathbf{p}_i^{(c_1)}$ , but going to another camera  $c_3$ . Both matches can be independently triangulated into two 3D points  $\mathbf{r}_{\iota(i, j)}^{(c_1, c_2)}$  and  $\mathbf{r}_{\iota(i, k)}^{(c_1, c_3)}$ . For a consistent 3D reconstruction result, those 3D points should coincide (up to some expected reconstruction error  $\epsilon$ ). Otherwise, the 3D points would be in *multiview conflict*, forming an inconsistent 3D reconstruction. An example is illustrated in Figure 2.8.

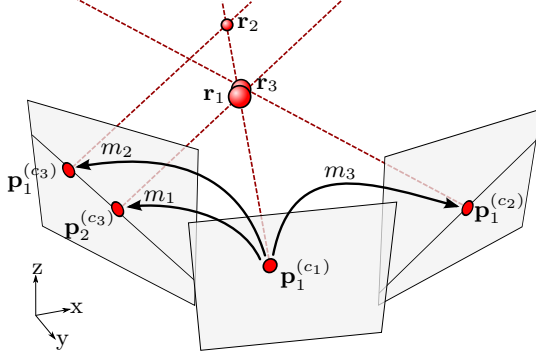


Figure 2.8.: Three camera views with extracted 2D markers. Stereo matches  $m_1 = m_{i(1,2)}^{(c_1,c_3)}$  and  $m_3 = m_{i(1,1)}^{(c_1,c_2)}$  correspond to almost overlapping reconstructed 3D points  $\mathbf{r}_1$  and  $\mathbf{r}_3$ . We thus enforce the simultaneous assignment of both matches by strengthening their pairwise potential during optimization. Additionally, while matches  $m_1$  and  $m_2$  are already in conflict from the one-to-one stereo matching constraint, our method now also sets  $m_2$  and  $m_3$  in conflict.

These multiview constraints are included into the optimization objective by incorporating additional constraints on the assignment vector  $\mathbf{x}$ . For each multiview conflict between two matches  $m_{i(j,d,e)}^{(c_d,c_e)}$  and  $m_{i(i,k)}^{(c_d,c_f)}$  we want to force their assignments  $\mathbf{x}_{\zeta(i,j,d,e)}$  and  $\mathbf{x}_{\zeta(i,k,d,f)}$  never to be active simultaneously (i.e., have value 1). From this idea, we define the set of multiview-consistent assignments as

$$\mathcal{V} = \left\{ \mathbf{x} \mid \mathbf{x}_{\zeta(i,j,d,e)} + \mathbf{x}_{\zeta(i,k,d,f)} \leq 1, \forall i, j, d, e : \left( \left\| \mathbf{r}_{i(j)}^{(c_d,c_e)} - \mathbf{r}_{i(i,k)}^{(c_d,c_f)} \right\|_2 > \epsilon \right) \right\}. \quad (2.14)$$

The multiview constraint set can equivalently be expressed with a constraint matrix  $\mathbf{V}$ . This constraint matrix is very sparse, containing one row with two non-zero entries per multiview constraint. It allows writing the set of multiview-consistent assignments in the form  $\mathcal{V} = \{\mathbf{x} \mid \mathbf{V}\mathbf{x} \leq \mathbf{1}\}$ . Then, the global matching problem becomes

$$\underset{\mathbf{x} \in \Psi}{\text{maximize}} \mathbf{x}^\top \mathbf{A} \mathbf{x} \quad (2.15)$$

with constraint set

$$\Psi = \{\mathbf{x} \in \{0, 1\}^M \mid \mathbf{C}\mathbf{x} = \mathbf{1}, \mathbf{V}\mathbf{x} \leq \mathbf{1}\} \quad (2.16)$$

We can additionally enforce consistent reconstructions by modifying the pairwise potentials in  $\mathbf{A}$ . If  $\mathbf{x}_{\zeta(i,j,d,e)}$  and  $\mathbf{x}_{\zeta(i,k,d,f)}$  correspond to multiview consistent matches, and are thus not in conflict, we set the corresponding entry in the affinity matrix  $\mathbf{A}_{\zeta(i,j,d,e), \zeta(i,k,d,f)} = 1$ . These values fill the empty blocks in the blocks of zeros in Eq. (2.12). This makes the graph matching objective Eq. (2.15) genuinely span across all camera pairs involved in the 3D reconstruction. However, this is no longer a classical graph matching problem, meaning we cannot directly use a graph matching solver.

### 2.3.3. Optimization with Multiview Constraints

Graph matching is a special case of the NP-hard quadratic assignment problem [BCPP98]. Numerous attempts exist to solve these kinds of problems approximately [CFSV04; FPV14]. To cope with the

highly combinatorial nature of the problem, most graph matching methods in computer vision solve the problem approximately by relaxing the constraints Eq. (2.9). The very successful spectral matching method [LH05] ignores the constraints altogether and replaces it with the norm constraint  $\|\mathbf{x}\|_2^2 \stackrel{!}{=} 1$ . Other methods respect the 1-to-1 constraint, but allow continuous solutions in the range  $\mathbf{x} \in (0, 1)$  instead of enforcing a binary solution. Then the constraint set Eq. (2.9) takes the form of the Birkhoff polytope [Zie95; BCPP98], a convex relaxation of the original constraint set, permitting much simpler and faster solution strategies. The Graduated Assignment (GA) method stays within the Birkhoff polytope by repeatedly projecting the current solution onto it using Sinkhorn’s balancing algorithm [GR96]. The fixed point methods in [LHS09; VCL+15] directly respect the binary constraint Eq. (2.9). They perform a combination of line search and projection onto this set using the Hungarian algorithm [LHS09]. The Sinkhorn and Hungarian projections are theoretically and practically connected by the theory of random walks [CLL10; LL11]. These projections originally work only for the two camera case. None of these methods can directly cope with constraints spanning multiple graphs, such as the proposed multiview constraints.

In this section, I present a generalized treatment of the Graduated Assignment (GA) method based on entropic regularization and alternating projections. My derivation paves the way for respecting the multiview constraints during optimization. First, the optimization problem is simplified by relaxing the binary constraint in  $\Psi$  to

$$\hat{\Psi} = \{\mathbf{x} \geq 0 \mid \mathbf{C}\mathbf{x} = \mathbf{1}, \mathbf{V}\mathbf{x} \leq \mathbf{1}\} \quad . \quad (2.17)$$

Optimization is then done with respect to the simpler constraint set,

$$\underset{\mathbf{x} \in \hat{\Psi}}{\text{maximize}} \quad E(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} \quad . \quad (2.18)$$

This turns the original optimization problem Eq. (2.15) into an inequality-constrained quadratic program (QP). But the affinity matrix  $\mathbf{A}$  is not negative semi-definite. Thus, this QP is not concave and cannot be uniquely solved in general; only local convergence can be achieved. While a strictly convex formulation of graph matching exists, see e.g. [ABK14] for a characterization, practical and theoretical results suggest that, surprisingly, the non-convex relaxation often yields superior results [LFF+14]. Novel convex reformulations of the original QP problem provide tight relaxations even of the original formulation [KKBL15], but don’t scale to big problems as in 3D reconstruction. This motivates finding a local optimum of the non-concave relaxation.

GA introduces a heuristic amplification term  $\beta$ . As we will see, this heuristic approach emerges naturally if an entropic regularizer is included into Eq. (2.18),

$$\underset{\mathbf{x} \in \Psi}{\text{maximize}} \quad \tilde{E}(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x} - \frac{1}{\beta} \mathbf{x}^\top (\ln \mathbf{x} - \mathbf{1}) \quad . \quad (2.19)$$

Here, I followed the convention that  $0 \ln(0) = 0$ . The entropy of the assignment,  $H(\mathbf{x}) = -\mathbf{x}^\top (\ln \mathbf{x} - \mathbf{1})$ , is used for regularization. This entropic regularizer is strictly concave. For small values of  $\beta$ , the contribution of this regularizer to  $\tilde{E}$  equalizes ("smoothes") the assignment solution, while for  $\beta \rightarrow \infty$  the original problem is recovered. This suggests successively solving a sequence of problems with more and more regularization (increasing  $\beta$ ). Such an approach has similarities with so-called barrier and smoothing methods [Ber99; CZ97; MN10]. Furthermore, entropic regularization was recently applied to the computation of optimal transport plans [Cut13] where it was shown to significantly simplify computation, a fact that we can also make use of.

Optimizing Eq. (2.19) for a specific  $\beta$  follows a scheme similar to the Frank-Wolfe algorithm. The objective function is first linearized around a previous solution  $\mathbf{y}$ ,

$$\underset{\mathbf{x} \in \Psi}{\text{maximize}} \quad \tilde{E}(\mathbf{x}) \approx \mathbf{y}^\top \mathbf{A} \mathbf{y} + (\mathbf{x} - \mathbf{y})^\top \mathbf{A} \mathbf{y} - \frac{1}{\beta} \mathbf{x}^\top (\ln \mathbf{x} - \mathbf{1}) \quad . \quad (2.20)$$

To simplify, we ignore terms constant with respect to  $\mathbf{x}$ , introduce an alias  $\mathbf{z} = \mathbf{A} \mathbf{y}$ , and write Eq. (2.20) component-wise

$$\underset{\mathbf{x} \in \Psi}{\text{maximize}} \quad \sum_{i=1}^M \mathbf{x}_i \mathbf{z}_i - \frac{1}{\beta} \mathbf{x}_i (\ln \mathbf{x}_i - 1) \quad . \quad (2.21)$$

As we will see shortly, this is a projection with respect to the Kullback-Leibler (KL) divergence, a distance measure between two vectors  $\mathbf{x} \in \mathbb{R}^M$  and  $\mathbf{m} \in \mathbb{R}^M$  defined as

$$\text{KL}(\mathbf{x}, \mathbf{m}) = \sum_{i=1}^M \mathbf{x}_i \ln \left( \frac{\mathbf{x}_i}{\mathbf{m}_i} \right) - \mathbf{x}_i \quad . \quad (2.22)$$

To see the relation to the KL projection, we rewrite Eq. (2.21) as

$$\underset{\mathbf{x} \in \Psi}{\text{maximize}} \quad -\frac{1}{\beta} \sum_{i=1}^M -\mathbf{x}_i \ln (\exp (\beta \mathbf{z}_i)) + \mathbf{x}_i \ln \mathbf{x}_i - \mathbf{x}_i \quad . \quad (2.23)$$

---

**Algorithm 1** Find a local minimum for the graph matching problem Eq. (2.18) by continuous relaxation and successively decreasing entropic regularization

---

```

1: Inputs:
    $\mathbf{A}$  ▷ Global affinity matrix from Eq. (2.12)
    $\mathbf{C}, \mathbf{V}$  ▷ Constraint matrices in  $\hat{\Psi}$ , Eq. (2.17)
    $\beta_0 = 1, \beta_{\text{step}} = 1.075, \beta_{\text{max}} = 30$  ▷ Default parameters for regularization schedule
    $t = 0.9$  ▷ Default binarization threshold
2:  $\beta \leftarrow \beta_0$ 
3:  $\mathbf{x} \leftarrow \mathbf{1}$ 
4: while  $\beta \leq \beta_{\text{max}}$  do
5:   repeat
6:      $\mathbf{z} \leftarrow \mathbf{A}\mathbf{x}$ 
7:      $\mathbf{x} \leftarrow \arg \min_{\mathbf{x} \in \hat{\Psi}} \text{KL}(\mathbf{x}, \exp(\beta\mathbf{z}))$  ▷ Solve Eq. (2.24) using Algorithm 2
8:   until converged
9:    $\beta \leftarrow \beta \cdot \beta_{\text{step}}$  ▷ Decrease influence of entropic regularizer in Eq. (2.19)
10: end while
11:  $\mathbf{x}_n \leftarrow \begin{cases} 1, & \text{if } \mathbf{x}_n \geq t \\ 0, & \text{otherwise} \end{cases}$  ▷ Binarize final solution
12: return  $\mathbf{x}$ 
    
```

---

Since  $\ln \mathbf{x} - \ln(\exp(\beta\mathbf{z})) = \ln \left( \frac{\mathbf{x}}{\exp(\beta\mathbf{z})} \right)$  we see that this can be written in the form

$$\underset{\mathbf{x} \in \hat{\Psi}}{\text{minimize}} \quad \text{KL}(\mathbf{x}, \exp(\beta\mathbf{z})) \quad . \quad (2.24)$$

Thus, the linearization in Eq. (2.20) is, in fact, identical to a KL projection onto the constraint set  $\hat{\Psi}$ .

The KL projection in Eq. (2.24) can be solved using the Alternating Generalized Projection algorithm, originally proposed by Bregman [Bre67] and generalized to inequality constraints in [CL81]. Censor and Zenios [CZ97] provide a self-contained derivation of this method based on dual coordinate ascent and give a convergence proof. Here, I just want to provide the necessary information to adopt this algorithm to our case. I also show how the specific structure of  $\hat{\Psi}$  leads to convenient simplifications.

The algorithm starts with the initial estimate, in our case  $\mathbf{x}^{(0)} = \exp(\beta\mathbf{z})$ . It then cyclically iterates through the *rows* of the constraints in  $\hat{\Psi}$ , with each iteration respecting a single row and each iteration improving upon the current iterate  $\mathbf{x}^{(k)}$ . This leads to a generalized Bregman projection which demands solving for  $\rho^{(k)}$  and  $\mathbf{x}^{(k+\frac{1}{2})}$  in

$$\nabla f \left( \mathbf{x}^{(k+\frac{1}{2})} \right) = \nabla f \left( \mathbf{x}^{(k)} \right) + \rho^{(k)} \mathbf{c} \quad , \quad (2.25)$$

$$\mathbf{c}^\top \mathbf{x}^{(k+\frac{1}{2})} = 1 \quad , \quad (2.26)$$

---

**Algorithm 2** Modified version of Bregman’s Alternating Generalized Projection algorithm for solving Eq. (2.24)

---

```

1: Inputs:
    $\mathbf{C} \in \{0, 1\}^{P \times M}$ ,  $\mathbf{V} \in \{0, 1\}^{|\mathcal{V}| \times M}$  ▷ Constraint matrices in  $\hat{\Psi}$ , Eq. (2.17)
    $\mathbf{x} \in \mathbb{R}_+^N$  ▷ Assignment vector to project from
2: repeat
3:   for  $r = 1$  to  $P$  do ▷ Loop through rows in 1-to-1 constraint matrix  $\mathbf{C}$ 
4:     Let  $\mathcal{I}(\mathbf{C}_r)$  be the set of non-zero indices in row  $r$  of matrix  $\mathbf{C}$ .
5:      $\rho \leftarrow \frac{1}{\sum_{i \in \mathcal{I}(\mathbf{C}_r)} \mathbf{x}_i}$ 
6:      $\mathbf{x}_i \leftarrow \mathbf{x}_i \cdot \rho, \forall i \in \mathcal{I}(\mathbf{C}_r)$  ▷ Solves Eqns. (2.25) and (2.26) by normalization
7:   end for
8:   for  $r = 1$  to  $|\mathcal{V}|$  do ▷ Loop through rows in multiview constraint matrix  $\mathbf{V}$ 
9:     Let  $\mathcal{I}(\mathbf{V}_r)$  be the set of non-zero indices in row  $r$  of matrix  $\mathbf{V}$ .
10:     $\rho \leftarrow \min \left( \mathbf{u}_r, \frac{1}{\sum_{i \in \mathcal{I}(\mathbf{C}_r^{\mathcal{M}})} \mathbf{x}_i} \right)$ 
11:     $\mathbf{x}_i \leftarrow \mathbf{x}_i \cdot \rho, \forall i \in \mathcal{I}(\mathbf{C}_r^{\mathcal{M}})$  ▷ Eq. (2.28)
12:     $\mathbf{u}_r \leftarrow \frac{\mathbf{u}_r}{\rho}$  ▷ Update dual variable
13:  end for
14: until converged
15: return  $\mathbf{x}$ 
    
```

---

where  $f(\mathbf{x}) = \text{KL}(\mathbf{x}, \exp(\beta \mathbf{z}))$ , cf. Eq. (2.24). If  $\mathbf{c}$  corresponds to an equality constraint (from matrix  $\mathbf{C}$ ), the algorithm simply picks  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k+\frac{1}{2})}$  as the next iterate. Multiview constraints represent inequality constraints, Eq. (2.14). Those require dual variables  $\mathbf{u} \in \mathbb{R}^{|\mathcal{V}|}$  initialized to  $\mathbf{u}^{(0)} = \mathbf{0}$ , and they require dual variable corrections. These corrections are quite essential as demonstrated by Kulis et al. [KSD09]. Based on the generalized projection parameter  $\rho^{(k)}$  obtained from solving Eqns. (2.25) and (2.26), these updates including the dual variable corrections take the form

$$\theta^{(k)} = \min(\mathbf{u}_r^{(k)}, \rho^{(k)}) \quad (2.27)$$

$$\nabla f(\mathbf{x}^{(k+1)}) = \nabla f(\mathbf{x}^{(k)}) + \theta^{(k)} \mathbf{c} \quad (2.28)$$

$$\mathbf{u}_n^{(k+1)} = \begin{cases} \mathbf{u}_n^{(k)} - \theta^{(k)}, & i = r \\ \mathbf{u}_n^{(k)}, & i \neq r \end{cases} \quad (2.29)$$

The structure of our problem permits very efficient closed-form solutions to these steps. The constraint matrices  $\mathbf{C}$  and  $\mathbf{V}$  contain only ones or zeros, and so do the rows Eq. (2.26). The update Eq. (2.26) can thus be written as  $\sum_{i \in \mathcal{I}(\mathbf{c})} \mathbf{x}_i = 1$ , where  $\mathcal{I}(\mathbf{c})$  is the set of non-zero indices in row  $\mathbf{c}$ . Inserting the gradients in Eq. (2.25) and exponentiating yields  $\mathbf{x}_n^{(k+\frac{1}{2})} = \mathbf{x}_n^{(k)} \exp(\rho^{(k)})$ ,  $\forall i \in \mathcal{I}(\mathbf{c})$ .

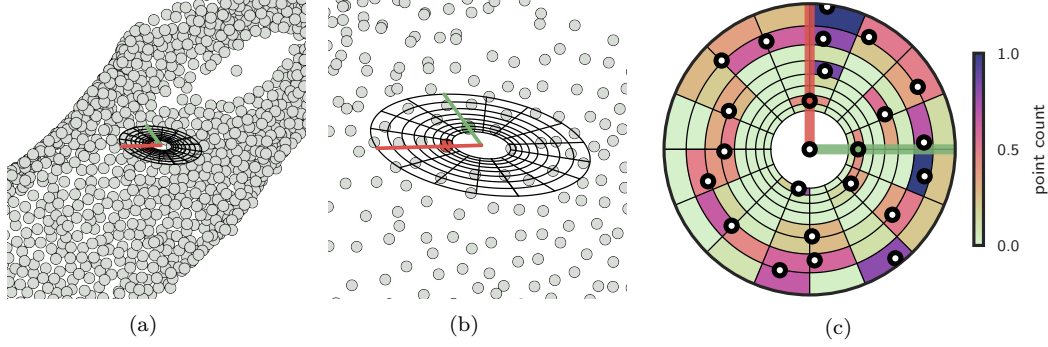


Figure 2.9.: (a) Point cloud with attached shape context coordinate frame and histogram bin edges. Each coordinate frame is given by a central point and two neighboring points that span the  $x$  and  $y$  coordinate axis. (b) Close-up view of the same situation. (c) Histogram bins of the shape context descriptor [BMP02] computed from the deformed spatial neighbors (white dots). The  $x$ - and  $y$ -axis are in green and red, respectively. Soft counting is used [LC04].

This equation can be inserted into Eq. (2.26), from which we obtain  $\exp(-\rho^{(k)}) = \sum_{j \in \mathcal{I}_r} \mathbf{x}_j^{(k)}$ , so that

$$\mathbf{x}_i^{(k+\frac{1}{2})} = \begin{cases} \frac{\mathbf{x}_i^{(k)}}{\sum_{j \in \mathcal{I}(\mathbf{c})} \mathbf{x}_j^{(k)}} & \text{if } i \in \mathcal{I}(\mathbf{c}) \\ \mathbf{x}_i^{(k)} & \text{if } i \notin \mathcal{I}(\mathbf{c}) \end{cases} \quad (2.30)$$

We can circumvent taking any exponents and logarithms by using exponentiated  $\mathbf{u}^{(k)}$ ,  $\lambda^{(k)}$  and  $\theta^{(k)}$ . These leads to further simplifications which are summarized in Algorithm 2. In the outer loop, the objective is linearized, performing a gradient ascent step at each iteration. Based on this linearized estimate, the KL projection is performed. This is summarized in Algorithm 1. The algorithm returns a set of active matches in  $\mathbf{x}$ , each match corresponding to a reconstructed 3D point. Therefore, the method returns an optimally consistent 3D reconstruction in form of a point cloud for each recorded frame of multiview data.

#### 2.3.4. Robust Registration across Large Motions

The deformation between two reconstructed 3D point clouds can only be revealed if correspondences between the 3D-points at different timepoints are available. This *tracking* is trivial with the quad pattern, Section 2.2, since each point has a unique identification, the  $3 \times 3$  color M-array code. In order to identify one of the randomly distributed dots, we look at its spatial relationships to its neighbors. These relationships can be turned into a discriminative representation, a feature descriptor that facilitates robust matching of points. The proposed method extends upon shape context [BMP02], a

feature descriptor that sets up histogram bins in uniform log-polar space, then counts the number of points that fall inside each bin.

However, shape context descriptors are not invariant under rotations and affine deformations. We thus enumerate every possible rotation and affine deformation by constructing possible local coordinate frames. Each coordinate frame is defined by two neighboring 3D points. The coordinate frame also provides a 3D normal so that a projective plane for the 2D shape context can be set up. This is visualized in Figure 2.9. The matching distance between two points is then taken as the minimum distance of all descriptors attached to these points.

An important benefit of this approach is that it implicitly takes care of any deformations by deforming the histogram axes accordingly. To further increase robustness, erroneous correspondences are detected by non-maximal suppression as in [Low04] and by checking geometric consistency in the local spatial neighborhoods of matches. Superficially, this approach bears resemblance to Spin Images [JH99], but the proposed descriptor explicitly constructs (affinely deformed) angular bins to increase distinguishability. From the works of Liu and Chen [LC04] and of Johnson and Hebert [JH99], I adopt the idea of soft counting: Instead of hard-assigning points to a single histogram bin, a point is weighted to the closest histogram bins according to the bilinear interpolation scheme. For example, a point lying exactly on the border of two bins is assigned to both bins with weight 0.5. In combination, this strategy yields a robust set of matches across even large deformations.

### 2.3.5. Results

The evaluation of the proposed method focuses on the integration of multiple cameras using the novel multiview constraints. To test this contribution in isolation, comparison to a baseline algorithm is performed. This method boils down to Graduated Assignment (GA) [GR96] on the distinct stereo pairs, using the same parameters and affinity score Eq. (2.6). Synthetic scenes with known 3D geometry were used for quantitative evaluation. Moreover, real-world scenes involving more than 5.000 random dots are used to assess the robustness and the runtime behavior of the algorithm.

#### Synthetic Scenes

Synthetic images of scenes with known ground truth geometry were generated. Three virtual cameras were placed around a given 3D mesh such that the viewing angle between each camera pair is of a predefined baseline angle  $b$ . The 3D mesh was equipped with artificial marker dots by randomly poisson-distributing points on the 3D mesh surface. The dots are then rendered into each camera using OpenGL. The resulting 2D points were discarded when they showed too much projective

distortion (more than  $60^\circ$  viewing angle towards the camera), simulating the real-world effect that marker detection often fails at too high perspective distortion. To simulate the effect of inexact marker extraction, normally distributed noise of scale  $\sigma^2$  was added to the 2D points. In each case, the multiview extension was compared to Graduated Assignment [GR96].

|                         | proposed <b>MV</b> | <b>GA</b>          | <b>GA-nodummy</b>  |
|-------------------------|--------------------|--------------------|--------------------|
| correct matches (%)     | $93.4\% \pm 2.0\%$ | $85.2\% \pm 3.2\%$ | $73.2\% \pm 4.1\%$ |
| number of false matches | $40.7 \pm 10.6$    | $92.5 \pm 15.5$    | $233.2 \pm 28.1$   |

Table 2.2.: Results of 3D reconstruction of a virtual sphere with known ground truth geometry.

**Sphere.** Table 2.2 gives results for a 3D sphere geometry with 1.000 random dots. The three virtual cameras had a  $40^\circ$  baseline between their viewing directions. The experiment was repeated 50 times with different random dots, each time adding normally distributed noise with variance  $\sigma^2 = 0.75$  pixels to their 2D projections. The dummy point weight was set to  $\omega_{\text{dummy}} = 0.6$ , the regularization strength  $\lambda$  to 1.0, the maximal reprojection error to reject non-epipolar matches to 4 pixels, and  $\epsilon$  in Eq. (2.14) was set to 3% of the diameter of the sphere. This experiment additionally tested the proposed explicit handling of dummy matches by comparing to Graduated Assignment without considering dummy matches and instead adding dual (slack) variables during constraint projection (denoted **GA-nodummy**). In effect, this comes down to using the constraint  $C^{(c_1, c_2)} x^{(c_1, c_2)} \leq 1$  in Eq. (2.9). Table 2.2 shows the average and the standard deviation of the percentage of correctly recovered matches in  $\mathbf{x}$  in relation to the ground truth assignment vector. The numbers are given in percent since the number of dots (thus number of ground truth matches) differs slightly for each experiment. The table also gives the number of false matches. Each false match would generate an outlier in the 3D reconstruction. Explicitly handling dummy points (**GA**) already greatly improves results (compared to **GA-nodummy**). The multiview constraints (**MV**) help to discard more than twice as many false matches compared to this baseline even for this extremely simple test case.

**Human.** A 3D model of a human was used, obtained as the average of a statistical body model based on laser scans [HSS+09]. Three virtual camera views were rendered at increasing baselines from  $20^\circ$  to  $80^\circ$ , shown for baseline  $b = 50^\circ$  in Fig. 2.10(a). 5 000 randomly distributed dots on the 3D geometry were projected into the virtual camera views, adding normally distributed noise of variance  $\sigma^2 = 0.75$  pixels to the 2D projections. To simulate detection errors, 10% of randomly selected 2D points were removed in each camera view. The dummy point weight was set to  $\omega_{\text{dummy}} = 1.0$ , the maximal reprojection error to reject non-epipolar matches to 4 pixels, and  $\epsilon$  in Eq. (2.14) that

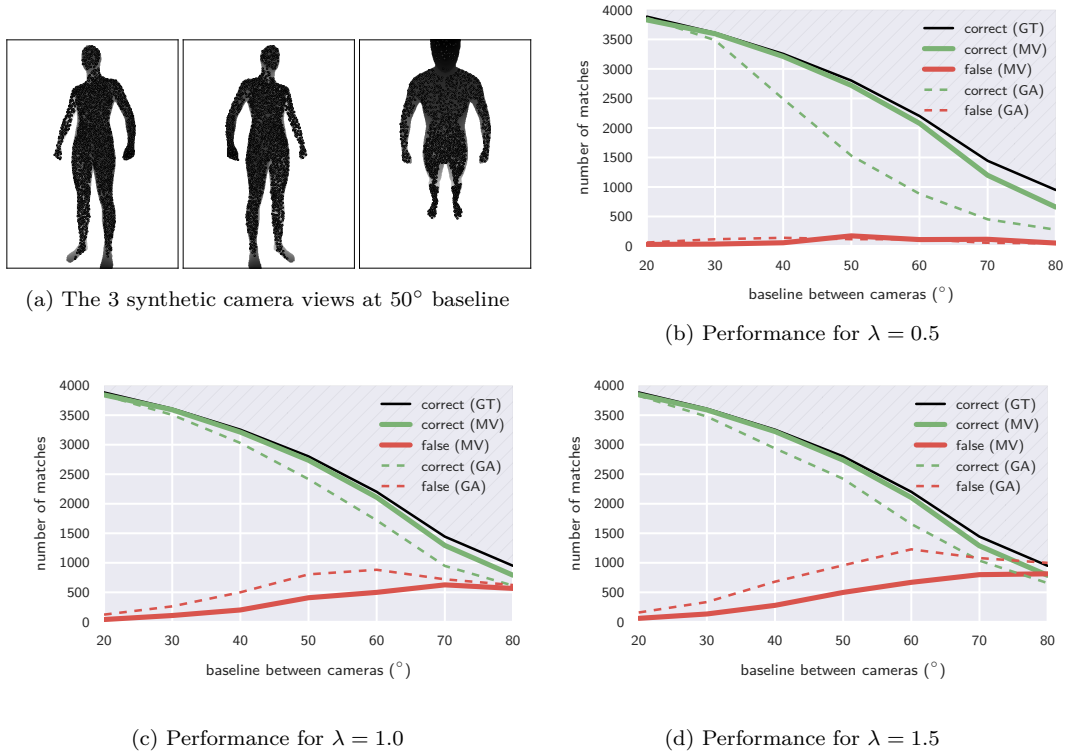


Figure 2.10.: Reconstruction results for synthetic scenes with known ground-truth geometry, depending on the distance (baseline) between three virtual cameras. The number of correct and false matches is compared between the algorithm with ("MV") and without ("GA") the proposed multiview extension. The maximal possible number of correct matches ("GT" for ground truth) is also shown.

controls the detection of multiview conflicts corresponding to 10 mm on the surface. The number of correctly and falsely recovered matches in  $\mathbf{x}$  is given in Figures 2.10(b) to 2.10(d). With increasing baseline, less and less matches are in the set of ground truth matches (black line in Figures 2.10(b) to 2.10(d)) The reason is that less and less points are simultaneously visible between pairs of cameras. Notice that the proposed multiview extension is closer to finding all correct matches. At moderate to large baselines, MV clearly improves upon GA (even though dummy points were added to the GA formulation; the original GA formulation performed even poorer). The results depend on the tuning parameter  $\lambda$  in Eq. (2.6). This parameter controls the strength of the spatial regularization in both algorithms. Increasing  $\lambda$  decreases regularization strength but does not change the differences between the methods significantly; both methods produce more false matches, Figure 2.10(d) shows

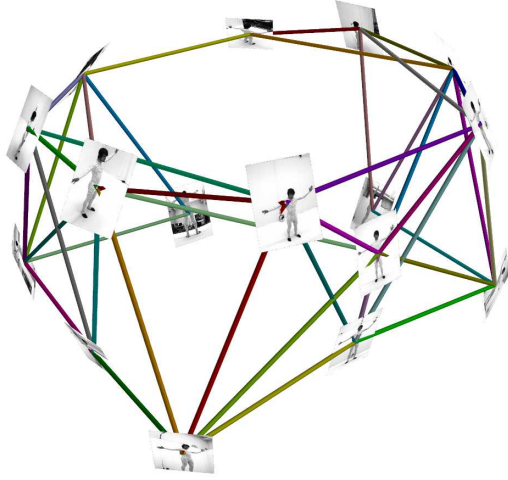


Figure 2.11.: Camera setup used for full-body 3D reconstruction. The visualization shows the captured images of the 16 cameras at their calibrated locations. The lines depict stereo camera pairs between which correspondences are estimated by the global correspondence estimation process.

$\lambda = 1.0$ , but even in this case GA is not able to obtain the same number of correct matches as the proposed MV algorithm.

### Real-World Scenes

A qualitative evaluation was performed for two application scenarios: Capturing full human body shape and capturing the deformation of a shoe. Both datasets were captured with 16 synchronized and calibrated video cameras running at 30 Hz and capturing images with  $1600 \times 1200$  pixels resolution. Here, only greyscale images were used. The marker dots were detected in the images as multi-scale extrema in the image pyramid. Markers were localized at sub-pixel accuracy via polynomial fitting in a  $5 \times 5$  pixel neighborhood.

**Full-Body 3D Reconstruction.** Figure 2.12 shows results for 3D reconstruction of a full-body performance. For capturing, a morph suit was printed with randomly distributed marker dots. The set of stereo cameras was constrained to pairs where the angle between the viewing directions of the cameras was less than  $70^\circ$ , since a larger baseline would result in stereo pairs with insufficient overlap. Still, this baseline can be considered extremely large for multiview stereo matching applications (usual baselines are around  $30^\circ$  [BPS+08; GS15]). The proposed method excels for such very sparse multi-camera setups. Figure 2.11 depicts the camera setup used here. The reconstructions are very robust, accurate, and temporally consistent even though no tracking was done at this stage. For this real-world dataset, I observed a big qualitative improvement compared to the GA method for regions on the body that are observed by three or even more cameras. I postulate that in such regions

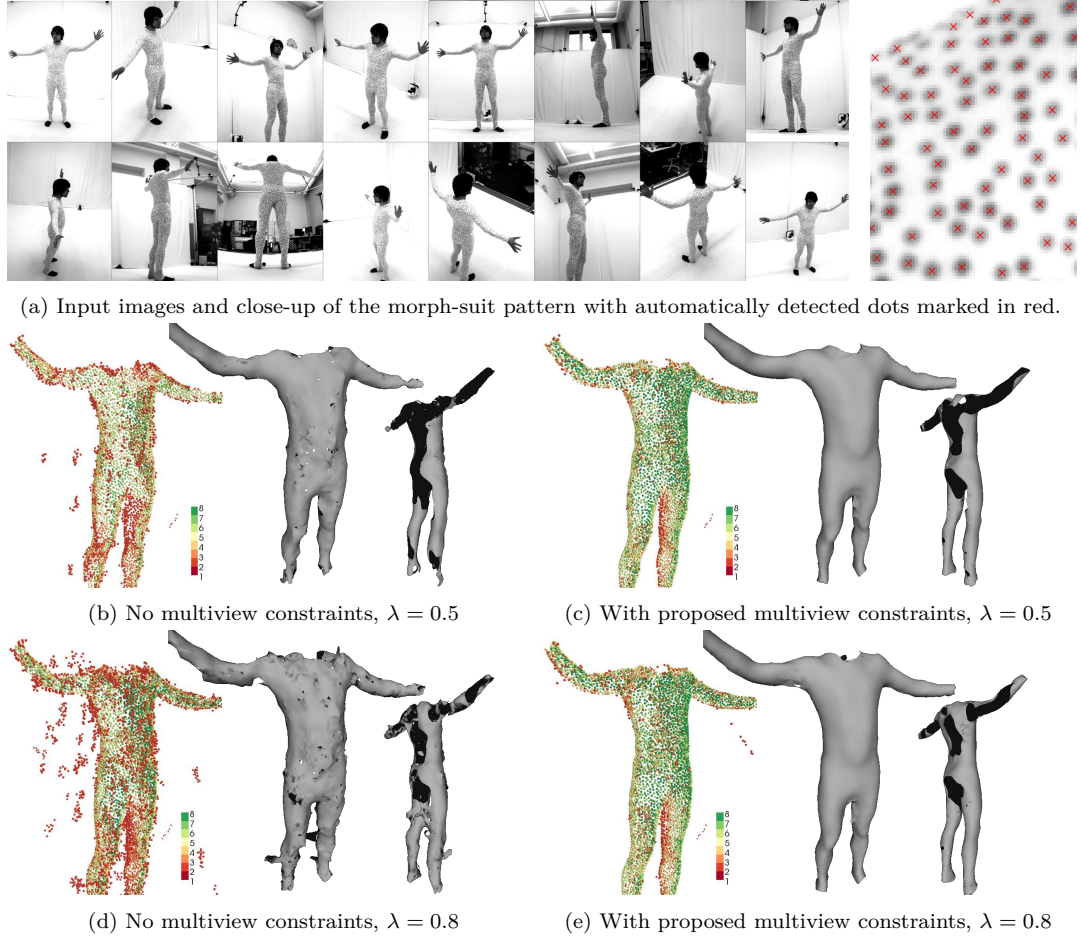


Figure 2.12.: Reconstruction results for capturing a full-body suit. The proposed multiview constraints clearly improve results over Graduated Assignment (GA). See text for details.

multiview constraints can be very effective because the additional constraint of multiview consistency provides strong hints at how to optimally break matching ambiguities.

The supplementary video shows the entire sequence. In the sequel, I focus on results for the first frame, Figure 2.12. The input images are shown in Figure 2.12(a). The results, for  $\lambda = 0.5$ , are shown in Figure 2.12(b) for Graduated Assignment without multiview constraints, and in Figure 2.12(c) *with* multiview constraints. The raw point cloud, Figure 2.12(b) left, reveals severe outliers far away from the true surface. There are also numerous outliers close to the true 3D surface which would be hard to filter.

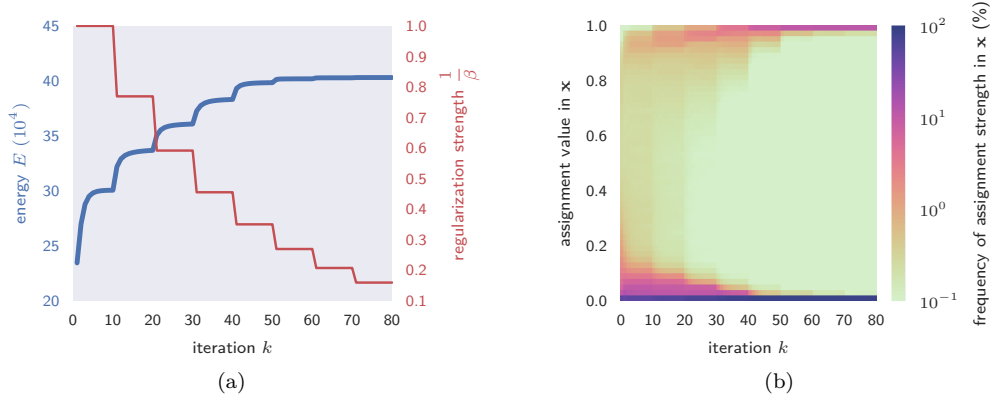


Figure 2.13.: Iterations of Algorithm 1 during optimization of a large graph matching problem involving multiview constraints. (a) Value of the optimization objective Eq. (2.18) (blue, without regularization) which increases and then converges as expected. The strength of the regularization  $\frac{1}{\beta}$  in Eq. (2.19) is shown in red. The proposed algorithm decreases the influence of this regularization during optimization to prevent early convergence to poor local minima. (b) Histogram of assignment values in the solution  $\mathbf{x}^{(k)}$  during iterative optimization. In later iterations, the solution is forced towards binary solutions (many values are either 0 or 1). Note: a logarithmic color scale was used to amplify small percentage values in the visualization.

It is interesting to estimate how many cameras are involved in the 3D reconstruction of each marker because this provides a hint at how consistent the reconstruction is. A post-processing step counts the number of cameras for each marker by finding marker tracks across multiple cameras and determining the length of these match tracks, a step which is especially needed in the GA result where stereo reconstructions happen separately. In Figure 2.12, the reconstructed points are colored according to the match track length. From this visualization, we can see that the multiview constraints enable a much more robust reconstruction, with each marker being reconstructed from as many camera matches as possible.

To obtain an approximate estimate of the 3D surface from the point clouds, Ball-Pivoting [BMR+99] with radius  $\rho = 30.0$  mm (approximate maximal distance between markers in the textile print) followed by one step of Taubin smoothing [Tau95] was performed in MeshLab [CCC+08]. The results using the proposed multiview constraints, Figure 2.12(c), are much cleaner compared to GA, Figure 2.12(b). But the multiview extension does more than filter outliers from the point cloud: the backside of the person, which is not well covered by many cameras in this setup, has fewer reconstruction holes when the multiview constraints are used. When we decrease the smoothness regularizer we obtain even better results for the back, Figure 2.12(e), while GA simply produces more noise, Figure 2.12(d). In

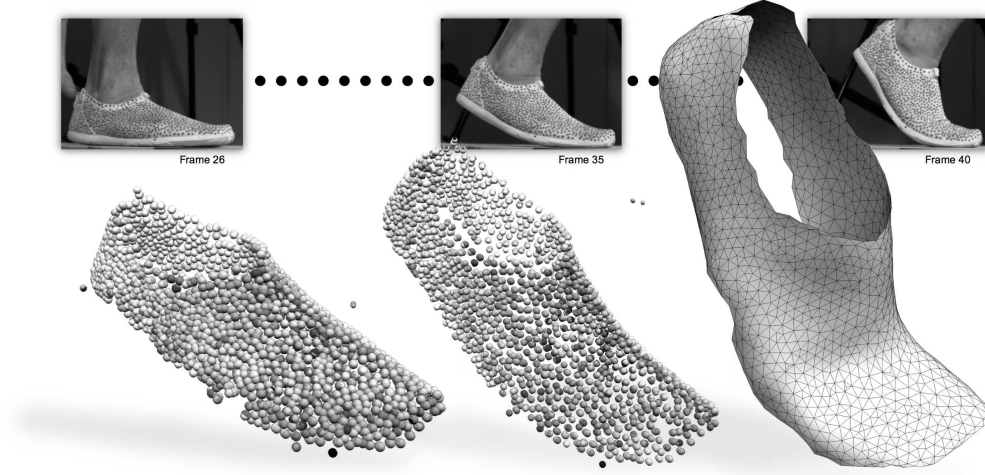


Figure 2.14.: 3D-reconstructed point clouds of a shoe while walking, 3 frames of the sequence shown here. Input images of one of the 16 cameras shown on top. The rightmost point cloud was meshed using Ball-Pivoting [BMR+99].

Chapter 3, we explore more elaborate techniques in order to obtain clean and hole-free 3D surfaces from such point clouds.

The human body dataset was used to analyze the convergence behavior of the algorithm. The solution  $\mathbf{x}$  at each (outer) iteration was observed while running Algorithm 1, with  $\lambda = 0.5$ . Figure 2.13(a) indicates that the optimization objective Eq. (2.18) (without regularization) indeed increases during the iterative optimization steps, and that it converges to a local optimum. A global optimum cannot be guaranteed due to the problem’s non-concavity. Recall that the proposed method decreases the strength of the regularization term in Eq. (2.19) during optimization, as visualized by the red line in Figure 2.13(a). I hypothesized earlier that this enforces binary solutions at later iterations. To check this, a histogram was computed at each iteration step  $k$ , showing the frequency of different values in the current assignment vector  $\mathbf{x}^{(k)}$ . In Figure 2.13(b), these histograms were stacked as columns to form an image. Later iterations, towards the right in Figure 2.13(b), indeed show a higher frequency of the values 0 and 1, confirming the hypothesis.

**Shoe in motion.** Figure 2.14 shows 3D reconstruction results obtained for a shoe in motion. The shoe is a prototype developed at adidas AG. Dots were quickly applied to this shoe using a permanent marker. The multiview-constrained graph matching algorithm outputs very clean and precise point clouds for each recorded frame. In particular, the meshed point cloud on the right of Figure 2.14

|                           | Dataset |         |           |                 | Runtime   |          |          |
|---------------------------|---------|---------|-----------|-----------------|-----------|----------|----------|
|                           | dots    | $M$     | ambiguity | $ \mathcal{V} $ | find dots | prepare  | optimize |
| Arm, Figure 3.4           | 1 002   | 180 415 | 15.1      | 1 814 615       | 44 s      | 21 s     | 51 s     |
| Leg, [PGF+16]             | 2 775   | 334 538 | 14.9      | 2 722 152       | 1 m 13 s  | 47 s     | 1 m 46 s |
| Shoe, Figure 2.14         | 621     | 255 686 | 42.7      | 7 624 050       | 29 s      | 51 s     | 2 m 37 s |
| Body suit, Figure 2.12(e) | 2 023   | 715 443 | 35.1      | 17 438 546      | 32 s      | 2 m 23 s | 7 m 16 s |

Table 2.3.: Runtimes for 3D reconstruction of a single frame using the proposed multiview graph matching method. From left to right, columns show data set name, average number of dots extracted from each image, number of potential matches in  $\mathbf{x}$ , average number of candidate matches in any other view per extracted dot, number of multiview conflicts in  $\mathcal{V}$ . Runtimes are given for the extraction of dots in all the images (column "find dots"), the preparation procedure (e.g. collection of potential matches and multiview conflicts), and the graph matching optimization step using Algorithm 1 (column "optimize").

shows the high fidelity of the reconstruction. By temporal tracking, we are able to very accurately measure the deformation of the shoe surface, Figure 2.17.

**Runtime.** Table 2.3 shows computation times of the unoptimized Python implementation running on a desktop computer with Intel Core i7-3770 processor and 16GB RAM, on a single core. The runtimes correspond to 3D reconstruction from a single frame of multiview video. The computational complexity does not primarily scale with the number of random dots but rather with the number of multiview conflicts, since the Bregman projection inner loops in Algorithm 2 take most of the computation time. The runtime is also related to the quality of calibration: Precise calibration allows to lower the value of the expected reconstruction error  $\epsilon$ , thereby reducing the number of match candidates  $M$  and the number of multiview constraints.

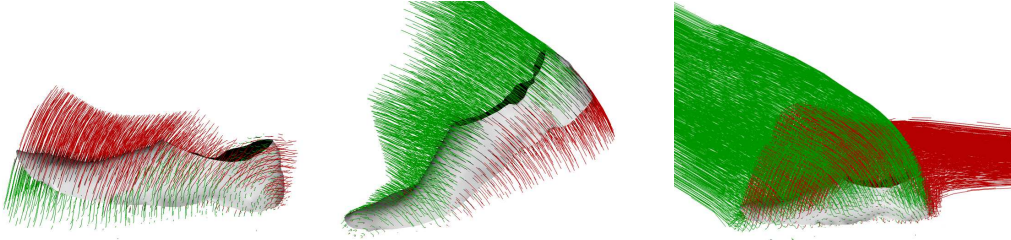


Figure 2.15.: Frame-to-Frame correspondences of the 3D reconstructions of the shoe, visualized as lines. Red lines are in the past and green lines in the future with respect to the current frame. The mesh is just for visualization and not used for tracking. *Left:* Correspondences around the touch-down phase of the walk cycle. *Middle:* Correspondences during plantar flexion. *Right:* All correspondences across the whole motion for reference. The proposed method is able to give robust tracking performance without producing any immediately visible outliers.

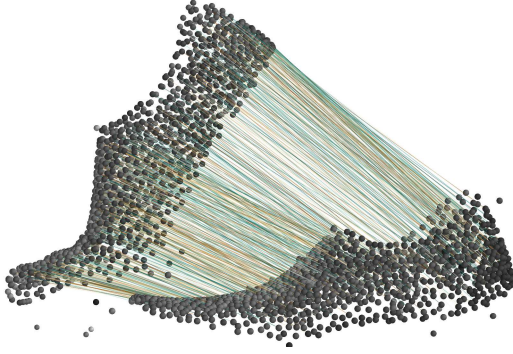


Figure 2.16.: Long-range correspondences computed between the rest-shape of a shoe and its deformed state. The shoe undergoes significant surface deformations: the shoe textile is stretched due to being put onto the foot and it is bend due to plantar flexion. The proposed feature descriptor nevertheless succeeds in obtaining robust correspondences.

**Tracking.** Figure 2.15 shows tracking results for the shoe dataset. Correspondences were computed between successive frames of the sequence. In each frame, local coordinate frames were generated from all combinations of 5 neighboring points. On average, this generated approximately 35 coordinate frames for each of the ca. 1200 points of each frame, each giving a shape context descriptor as described in Section 2.3.4. Correspondences were filtered using non-maximal suppression as in [Low04] (with threshold 0.8) and by discarding locally non-rigid correspondences. Despite such heavy filtering, a large quantity of frame-to-frame correspondences were successfully established.

Figure 2.16 shows correspondences obtained between the rest pose (shoe not put on) and a deformed state of the shoe during the complete walk cycle. The deformations between the two point clouds are quite large. Nevertheless, lots of correspondences are found without producing any severe outliers. This opens the possibility for performing deformation analysis on these kinds of heavily deforming 3D surfaces.

## 2.4. A first use case: Deformation and Strain Analysis

The methods presented in this chapter allow for calculating a full displacement field of a deforming 3D surface. For deformation analysis applications, engineers are usually interested in the strain at each surface point. I now show how to obtain such a full-field strain distribution from captured 3D data.

### 2.4.1. From 3D Reconstructions to Principal Deformations

Let  $\mathbf{r}^{(i)} \in \mathbb{R}^3$  and  $\mathbf{d}^{(i)} \in \mathbb{R}^3$  be 3D points in the reference and deformed configuration, respectively. To compute the strain of a point, we need to measure the change in distances and angles of neighboring points. We choose the  $K$  nearest points to  $\mathbf{r}^{(i)}$ . Let those neighbors be  $\mathbf{R}_{\mathcal{N}(i)} \in \mathbb{R}^{K \times 3}$  and  $\mathbf{D}_{\mathcal{N}(i)} \in \mathbb{R}^{K \times 3}$  (neighboring 3D points stacked on top of each other in a matrix). First, the rigid

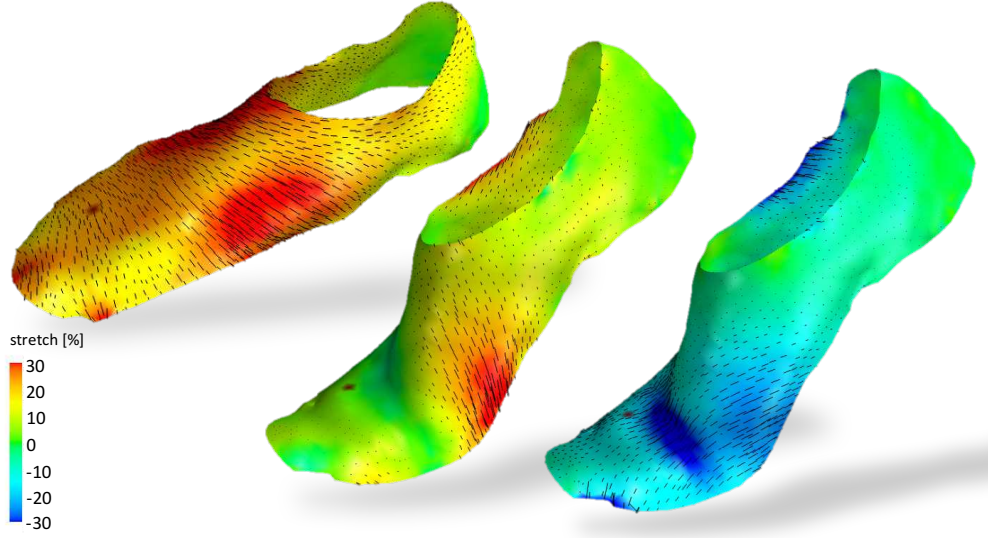


Figure 2.17.: Strain analysis of a shoe. *Left*: Major principal strain (in %) between the rest shape of the shoe (not shown) and the deformed state after putting the shoe on a foot. *Middle*: Major and *Right*: Minor principal strain between the leftmost shape and the shape during plantar flexion. Black lines visualize the principal strain direction.

motion must be subtracted. This can be done by translating each neighborhood  $\mathcal{N}(i)$  such that  $\mathbf{r}^{(i)}$  and  $\mathbf{d}^{(i)}$  are moved to the origin, then rotating the points such that their normal is  $[0, 0, 1]^\top$ . After this step, both point neighborhoods lie approximately in the 2D plane, so we can drop the  $z$  components to obtain the 2D points  $\hat{\mathbf{R}}_{\mathcal{N}(i)} \in \mathbb{R}^{K \times 2}$  and  $\hat{\mathbf{D}}_{\mathcal{N}(i)} \in \mathbb{R}^{K \times 2}$ , essentially performing orthogonal projection. For the sake of brevity, I now drop the subscript in  $\hat{\mathbf{R}}_{\mathcal{N}(i)}$  and  $\hat{\mathbf{D}}_{\mathcal{N}(i)}$  and use  $\hat{\mathbf{R}}$  and  $\hat{\mathbf{D}}$  instead.

The in-plane deformations can be quantified using the deformation gradient. We assume the deformation of the points in the neighborhood  $\mathcal{N}(i)$  can be sufficiently approximated by a continuous linear function, so that  $\mathbf{d} = \varphi(\mathbf{r}) \approx \mathbf{F}\mathbf{r}$ , which means that  $\nabla\varphi \approx \mathbf{F}$ . This assumption holds in most strain analysis scenarios as long as no cracks form on the surface, and as long as the neighborhood  $\mathcal{N}(i)$  stays small enough. The deformation gradient  $\mathbf{F}$  can be obtained from measured data by solving the least squares-problem

$$\underset{\mathbf{F}}{\text{minimize}} \quad \|\hat{\mathbf{R}}\mathbf{F}^\top - \hat{\mathbf{D}}\|_{\mathbf{F}}^2, \quad (2.31)$$

which is a similar approach as in [PAXG09]. The solution can be written in closed-form as

$$\mathbf{F}^\top = (\hat{\mathbf{R}}^\top \hat{\mathbf{R}})^{-1} \hat{\mathbf{R}}^\top \hat{\mathbf{D}} = \mathbf{A}^{-1} \mathbf{B} \quad (2.32)$$

with  $\mathbf{A} \in \mathbb{R}^{2 \times 2}$  and  $\mathbf{B} \in \mathbb{R}^{2 \times 2}$  given by

$$\mathbf{A}_{i,j} = \sum_{k=1}^K \hat{\mathbf{R}}_{k,i} \hat{\mathbf{R}}_{k,j} \quad , \quad \mathbf{B}_{i,j} = \sum_{k=1}^K \hat{\mathbf{R}}_{k,i} \hat{\mathbf{D}}_{k,j} \quad . \quad (2.33)$$

Thus, only a  $2 \times 2$  matrix inversion and a  $2 \times 2$  matrix-matrix product are needed to compute  $\mathbf{F}$ .

The deformation gradient  $\mathbf{F}$  maps directly to rotation-invariant strain measures like the Green - St-Venant strain tensor  $\mathbf{E} = \frac{1}{2}(\mathbf{F}^\top \mathbf{F} - \mathbf{I})$  and the Cauchy-Green deformation tensor  $\mathbf{C} = \mathbf{F} \mathbf{F}^\top$  [BW08; BK00; Neu09]. These quantities are very important for computer simulation [Hau04], but interpreting and visualizing them directly is not intuitive.

Engineers usually want to analyze how much and in which principal direction the material deforms. To this end, residual 2D rotation is factored from the deformation gradient  $\mathbf{F}$  using polar decomposition [SHD92]. This factors  $\mathbf{F}$  into a rotation part  $\mathbf{Q} \in \mathcal{SO}(3)$  and a symmetric in-plane deformation component  $\mathbf{U}$ , so that  $\mathbf{F} = \mathbf{Q} \mathbf{U}$ . Since  $\mathbf{U}$  is symmetric, it has two real eigenvalues which directly yield the principal strains

$$\lambda_{1,2} = \frac{1}{2} \text{Tr}(\mathbf{U}) \pm \sqrt{\frac{\text{Tr}(\mathbf{U})^2}{4} - \det(\mathbf{U})} \quad . \quad (2.34)$$

The corresponding eigenvalues represent the principal strain directions

$$\mathbf{e}_1 = \begin{bmatrix} \lambda_1 - \mathbf{U}_{1,1} \\ \mathbf{U}_{1,0} \end{bmatrix} \quad , \quad \mathbf{e}_2 = \begin{bmatrix} \lambda_2 - \mathbf{U}_{1,1} \\ \mathbf{U}_{1,0} \end{bmatrix} \quad . \quad (2.35)$$

The major strain of the material is then given by  $\lambda_{\text{major}} = \max(\lambda_1, \lambda_2)$ .

The deformation model may seem very crude. In place of orthogonal projection onto the 2D plane, we could use a procedure respecting the curvature of the surface. We could replace the simple deformation model in  $\varphi$  by a higher-order polynomial function. But this may introduce additional noise into the strain fields. There is a tradeoff between smoothness, precision, locality, and computation speed. This tradeoff is not only related to deformation model complexity; the size of the neighborhood  $\mathcal{N}(i)$  also plays an important role. Alternative approaches are discussed in [PAXG09; SOS09] and can be implemented alternatively.



Figure 2.18.: A gallery of visualizations of full 3D strain fields, showing the major deformation of loose-fit and tight-fit garments during various sports motions. Recorded with a 16-camera camera setup. The 3D strain fields are color-coded in percent (50 % means the textile stretched by a factor of 1.5, e.g. from 10mm to 15mm). Some of the 3D reconstructions are rendered from the point of view of one of the of the 16 cameras, some from free viewpoints (those with white background). Note that the system can, to a certain extent, handle wrinkles.

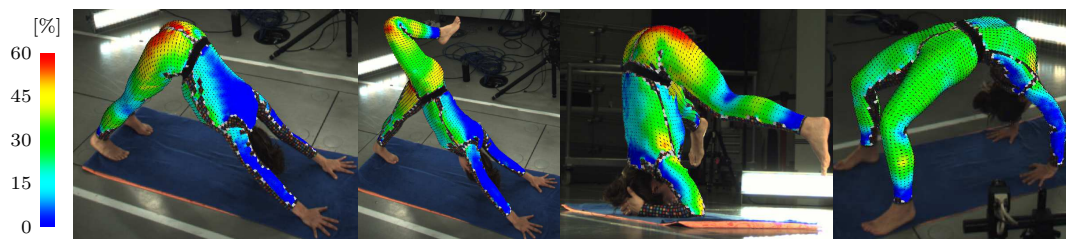


Figure 2.19.: Visualizations of the full 3D strain field, showing the major deformation of a tight-fit garment during various yoga motions. Experiments were performed at the adidas innovation team laboratory. The 3D strain fields are color-coded in percent. The 3D reconstruction is rendered from the point of view of one of the of the 12 cameras.

### 2.4.2. Results

Figure 2.17 shows the full-field strain distribution of the previously covered shoe dataset. The visualization shows the strain between rest-state and rest-state with foot. It also shows the deformation happening during plantar flexion by visualizing the strain field between rest-state with foot and the frame in motion showing maximal plantar flexion. These kinds of visualizations enable testing, validating, and improving (textile) material designs.

The quad pattern also allows 3D deformation analysis. Since the print size of the quads is known, absolute strains can be measured. The pattern was used in numerous setups, for example for measuring loose-fit as well as tight-fit garments as shown in Figure 2.18. The company adidas AG started using the quad pattern 3D reconstruction method along with the presented deformation analysis system. Some results measured by adidas are shown in Figure 2.19

### 3. Data-Driven Animation of Deforming Muscles

Realistic virtual humans are becoming commonplace in movies and interactive computer graphics applications. The lifelike appearance of virtual characters depends not only on accurate rendering and motion but also on faithful modeling of subtle pose-dependent effects, primarily deformations which are due to muscles and soft tissue. Currently, creating such subtle effects is a challenging process - usually requiring tedious manual work by experienced animators as well as physics-based simulation of deformation effects [LGK+12]. Despite producing believable results, these approaches have several drawbacks. Customization of muscle models to specific virtual subjects is not easy. Control and simulation of these models may be computationally expensive and require tuning of many parameters. Furthermore, since the underlying muscle model is still an approximation of anatomical reality, truly fine-scale muscle deformation may not be reproducible.

In this chapter, a data-driven approach for muscle modeling is presented. The approach employs 3D reconstruction methods from Chapter 2 to obtain real-world muscle deformation data. This data shows complex inter-dependencies between muscle activity of real subjects and their physical constitution, external forces, and motion. These phenomena are learned automatically by the proposed muscle model. While the muscle model focuses on the shoulder-arm-complex, the underlying methodology

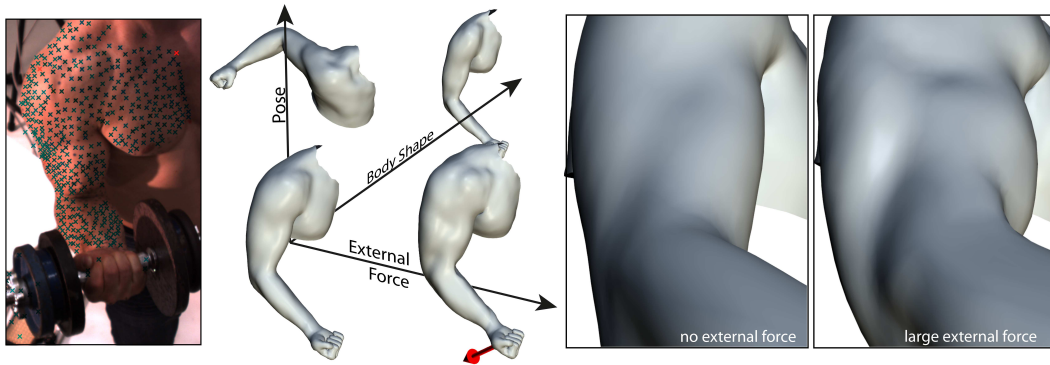


Figure 3.1.: Multiview recordings of human subjects performing various muscle exercises (left) are used to build a statistical model for synthesizing muscle deformations (middle) that can be altered according to pose, body shape, and external forces. The model can simulate fine-scale bulges and dimples on the skin caused by various muscle strands (right).

can work as a blueprint and as a proof of concept, enabling the application of similar concepts to other body parts, to the whole body, or across other people.

The muscle model takes as input standard skeletal motion parameters to specify the motion to be executed. It additionally incorporates the parameters that control the appearance of the subject in terms of physique and training level: the subject’s body mass index (BMI), muscularity, height. This allows one to smoothly transfer the muscle deformation between subjects of diverse physique, training level and muscularity. Finally, the model takes as input an external force vector acting on the hand. The proposed model reproduces realistic muscle shape as an interplay of the above factors and generalizes well to other movements, people, and mechanical scenarios beyond the training dataset.

Building such a data-driven model poses new challenges in acquisition, model learning, and parameterization. I extend the 3D acquisition methods from Chapter 2 so that one can obtain highly detailed and spatiotemporally coherent geometry of deforming skin, capturing individual muscle strands, fat tissue, as well as tangential stretching and shifting of skin. I propose a method for registering such data to enable scalable and robust non-rigid registration of thousands of 3D reconstructions with minimal user interaction. Finally, I introduce a new semi-parametric learning approach to build a data-driven model from the 3D reconstructed input data. Thanks to a sparse approximate formulation of non-linear kernel regression, the final muscle model requires little memory, can handle huge training datasets, and can be evaluated in real-time. At the same time, it can produce non-linear muscle deformation effects that cannot be achieved by previous data-driven methods. The muscle model provides new animations of the shoulder-arm complex by controlling a small set of intuitive parameters.

#### 3.1. Related Work

**Data-driven Muscle Modeling** was studied by Allen et al. [ACP02]. They acquire static range scans of the torso and the arm of a single person in different poses. Performing nearest-neighbor interpolation allows them to generate deformations for new poses. Park and Hodgins [PH06] used a VICON mocap system and a set of 350 reflective markers to capture skin deformation. In [PH08], they describe a non-linear kernel regression method for obtaining a surface model at this resolution (350 markers) from a smaller set of 40 – 50 skeletal markers. They later describe a method to generate a better skeletal model for the shoulder with additional virtual joints, using a capture set-up of 200 markers [HPH10]. Drawing inspiration from these works, I build a model with a wider scope and higher detail. Specifically, unlike these methods, I model statistical variations of muscle deformations across multiple people. This allows additionally incorporating external forces and in turn enabling

the modeling of isometric muscle contractions in static poses which is an effect largely overlooked previously.

**Data-driven Face Capture and Modeling.** Modeling of human facial expressions is of specific interest for character animation. Blanz and Vetter [BV99] build a linear deformation model (using PCA) for static faces using a set of registered 3D laser scans. Vlastic et al. [VBPP05] present a multi-linear face model that models personal identity, facial expression and visemes as discrete variables. Attempts have also been made to capture the elastic properties of facial muscles: Bickel et al. [BBO+09] propose a marker-based framework for capturing muscle strain in response to stress induced by a force probe. I argue that such force-dependent deformations do not have to be modeled by parameters of elasticity, but can be directly modeled using a purely data-driven model.

**Statistical Body Modeling.** Allen et al. [ACP03] analyze shape variations of 3D scans of people standing in the same pose. The SCAPE body model [ASK+05] generalizes this work by modeling surface deformation as a function of both body shape and pose, assuming that both effects are independent (using 70 scans from one subject in multiple poses and 37 scans of multiple subjects in one pose). Later, it was shown by Allen et al. [ACPH06] and Hasler et al. [HSS+09] that body pose and shape can also be modeled simultaneously, which allows scanning several subjects in different poses to analyze a broader range of variations. Statistical body models like these have applications in image and video recognition [HAR+10; BB08], animation and shape completion [ASK+05]. Unlike these methods, which are based on laser scans, the proposed method captures and models minute muscle deformation and tangential skin-shifting on multiple subjects. It constitutes a large training database of more than 32 000 meshes and contributes to this area by modeling the effect of external forces on muscle deformation and stretching of skin.

**Anatomical and Physics-based Modeling.** Anatomically accurate modeling of human muscle tissue and bone deformation has applications in sports medicine and kinesiology, alongside computer graphics. In the recent past, Teran et al. [TSB+05] have shown impressive results by simultaneously modeling muscles, bones, and tendon properties using the Visible Human dataset and simulating them using a Finite Element method. A similar approach is also used for building an anatomically based face muscle model [SNF05] and for realistic hand animation [SKP08]. Lee et al. [LST09] describe a comprehensive system for the biomechanical modeling of the upper human body. An extensive review of physiologically based modeling is given in the survey of Lee et al. [LGK+12]. Such physiologically based simulation methods are complementary to the proposed approach. Although they are more

accurate and yield elaborate shape and elastic deformation priors, tuning these parameters and applying them to different subjects is not straightforward. Simulating these complex mechanical systems is also computationally demanding. This chapter describes a data-driven approach that is easy-to-use, computationally efficient and that adapts quickly to additional people.

**Example-based Deformation.** Learning general surface deformations from examples [SRC01] given by artists has conceptual similarities to data-driven modeling. Lewis et al. [LCF00] have presented one of the first approaches to smoothly blend deformations defined at specific training poses. Wang et al. [WPP07] introduced a fast example-based skinning method based on predicting deformation gradients [SP04] from example meshes. Context-aware skeletal shape deformation [WSLG07] takes a similar path by learning polar-decomposed deformation gradients as residuals on top of a skeleton that is rigged by an artist. These methods provide deformation control using a limited number of training poses modeled by the artist, while my approach provides deformation control by intuitive parameters and learns from a large dataset of 3D reconstructions of real people.

**Shape Parameterization.** It is interesting to note how all these related approaches parametrize 3D shape. Park and Hodgins [PH08] model surface vertex displacements as offsets from the bone skeleton. Anguelov et al. [ASK+05] model surface triangle deformations using deformation gradients [SP04]. Hasler et al. [HSS+09] use the relative rotations between triangles as a rotation-invariant encoding, which is, however, hard to use alongside positional constraints on vertices. The presented work employs a deformation gradient encoding that is based on this prior work but combines it with a skinning prior as in [WSLG07] to make it usable alongside an artist-controllable skeleton. With ideas from [HAR+10], the encoding is compressed even further, providing a deformation gradient decomposition scheme that gives a minimal representation of the shape deformations. This compressed encoding reduces the burden on the learning method, especially for large training datasets.

## 3.2. Method

When a person moves, a complex biological control system of muscles, bones, and tendons is activated. In the shoulder-arm complex alone, more than 20 muscles contribute to the motion - not taking into account multiple pennate branches in muscles such as the *deltoid*, Figure 3.2. The largest arm muscles are the *biceps brachii* and the *triceps brachii* in the upper arm which show large pose- and force-dependent deformations. On the one hand, muscles are intentionally or subconsciously activated through neural signals. On the other hand, muscles react to external influences, i.e. forces or torques acting on joints. As a result, the body moves, the muscles bulge, and the soft tissue and skin deform,

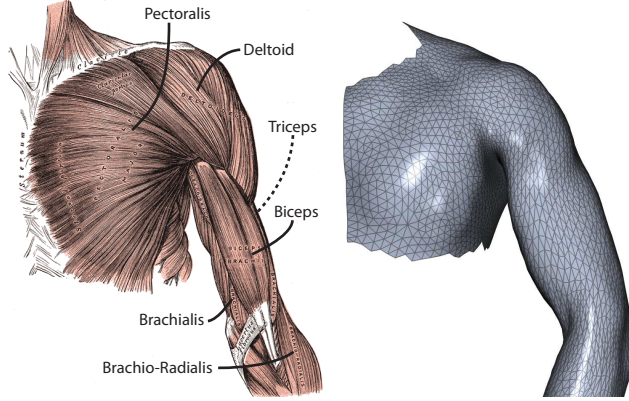


Figure 3.2.: Left: Anatomy of the shoulder-arm complex (*Gray's Anatomy* - Henry Gray, 1858) , depicting a few participating muscles. Right: Surface mesh of the same area as obtained by the muscle reconstruction method.

a process whose biological mechanisms are well-studied in anatomy and kinesiology [Eno08]. However, muscle shape and deformation also depend on a person's physical constitution, training level, tissue, and fibre composition, and many more factors. It is no wonder that the design of biologically accurate physics-based forward simulation models that account for all parameters influencing the muscle and skin deformation is challenging, if not impossible, even today [TSB+05; DAA+07].

It is a reasonable abstraction of the true biomechanical process to assume that the main external influence on muscle bulges are forces acting on the rigid skeletal parts that the muscles are connected to. Such forces also create secondary effects such as the intake of air into the lungs, the deformation of the spine to stabilize the body, and slight translations of bones. The proposed muscle model focuses on the shoulder-arm-complex, as by its motion range and anatomical complexity it can be considered a downscaled version of the full-body [ACP02]. Slow to medium-fast motions (which can be interpreted as *quasi-static* according to biomechanics literature [Eno08, “When is a movement fast ?”]) are considered. Non-static motion would require capturing dynamics of soft tissues, referred to as wobbling of body masses in physiology. However, the quasi-static assumption is fulfilled in the acquired training set of 3D skin surface measurements since the subjects perform the muscle exercises slowly and naturally.

The muscle model can be seen as a mapping  $\Psi$  from a few intuitive, yet biologically motivated, input parameters (body pose  $\theta$ , body shape  $\beta$ , external forces  $\gamma$ ) to a body surface  $\mathbb{M}$  that exhibits plausible muscle bulges and skin deformation, hence

$$\mathbb{M} = \Psi(\theta, \beta, \gamma) \quad . \quad (3.1)$$

**Body Pose.** The muscle model embeds a simplified bone skeleton with a shoulder and an elbow joint. Each joint is parametrized as a 3-degree of freedom ball and socket joint, i.e. in total six parameters  $\theta$  determine the skeleton pose.

**Body Shape** includes the body mass index (BMI), muscle proportion (as a percentage of body mass), and the height of the person as relevant physiological parameters  $\beta$  that influence muscle bulges.

**External Forces** are parametrized as force magnitudes and direction vectors. The external force acts on the hand, but to express this force locally to the body parts the force direction is rotated into the coordinate frame of the lower and upper arm (in total,  $2 \cdot 3 + 1$  scalars). The kind of deformations that are induced by external force are visually analyzed in Figure 3.8 on page 60, which shows two reconstructions acquired using the method presented in the next section.

#### 3.2.1. Experiment Setup

In order to model muscle deformations in a data-driven way, they must be captured from the real world from real subjects. This requires a strict experiment protocol. 10 male subjects were recruited (between 20 to 34 years of age, of different body shapes). Five of the subjects practice body building regularly, the remaining subjects exercise less and cover a wide variety of body shapes, from corpulent to thin. The following physiological parameters are recorded for each subject: weight, height, bone mass, body fat ratio, and muscle proportion (using a body fat scale).

Each subject performed the same 11 predefined arm motions. An exact description of the movements is provided in Appendix A. In some movements, the shoulder-arm joints were articulated separately. In others, articulations were combined. Five of these movements were repeated with a barbell in the hand, to simulate external forces. 8 barbells were used, weighing between 0.5 kg and 17.5 kg<sup>1</sup>. Additionally, some free motions like boxing, dancing, or flexing were performed by the subjects, which are used for cross validation. Per subject, around 30 – 40 motions were captured at 30 Hz, each consisting of approximately 100 frames. The range of exercises in the dataset is designed to sufficiently sample the pose, body shape and force parameter dimensions serving as input to our model.

I argue that the force parameter is sufficiently sampled even though only a barbell is used. The (gravitational) force from the barbell always points in the same direction: downwards; but if torso,

---

<sup>1</sup>Some subjects were not able, maybe also not confident enough to lift that much weight. The exercises were then performed up to the maximum weight they could lift.

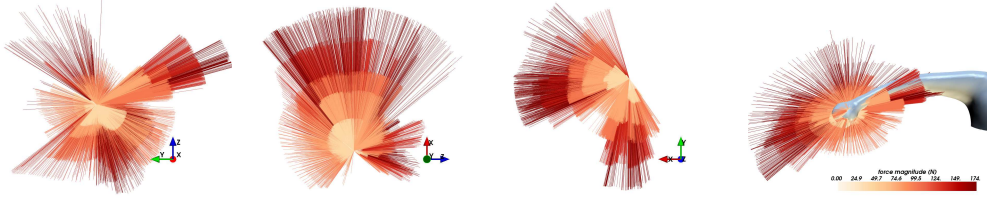


Figure 3.3.: Directions of captured external forces in the relative coordinate frame of the lower arm, rendered as lines in 3 orthogonal projections in the x-, y-, and z-plane; and on the right in context of the arm template mesh. The different viewpoints help assessing the density of the force direction vectors in 3D space. As one can see, the force directions are sampled densely in many practically important directions.

upper, and lower arm are rotated, the force direction changes *relative* to those body parts, thus it changes *relative* to the lines of action of the participating muscles. A body-builder uses this fact to train his or her muscles with barbells only, even if those muscles induce force in different, sometimes opposing directions (e.g. agonist and antagonist). Just like the body-builder, one can carefully select the body-building movements and body postures which effectively allows sampling many different *relative* force directions. Figure 3.3 validates this for the dataset at hand: the sampled force directions are visualized after being rotated into the local coordinate system of the lower arm in 3D. This sampling is very dense in many practically important directions. Some of those directions of external force were only captured with small magnitude. This is easily explained by the difference in strength of the different muscles in the arm. For example, shoulder abduction is much harder to perform under load compared to a flexion of the lower arm.

To capture the muscles in action, a multi-camera acquisition setup was used consisting of 16 synchronized and calibrated FireWire video cameras, each recording at a resolution of  $1600 \times 1200$  pixels and a frame rate of 30 Hz, Figure 3.4(a). The cameras were arranged in a convergent setup around the shoulder-arm region. To facilitate space-time reconstruction, a pattern of dense colored dots was applied to the arm, the shoulder and the torso (random peppering of black, dark green and blue dots) using film make-up, Figure 3.4(b). Depending on the shape of the subject, around 800 to 1200 markers were applied with a maximum inter-marker distance of about 5mm. Application of make-up took around 30 – 45 minutes per person. While most markers were randomly placed on the subject, 11 markers were placed at anatomically equivalent locations with a red pen, inspired by marker placements for motion capture, for example, at the elbow, extremities at ulna and radius near the hand, tip of scapula and humerus (red points are well visible in Figure 3.4(d) and Figure 3.5(a)). These anatomical markers are used in a template initialization step and enable template matching across captured subjects.

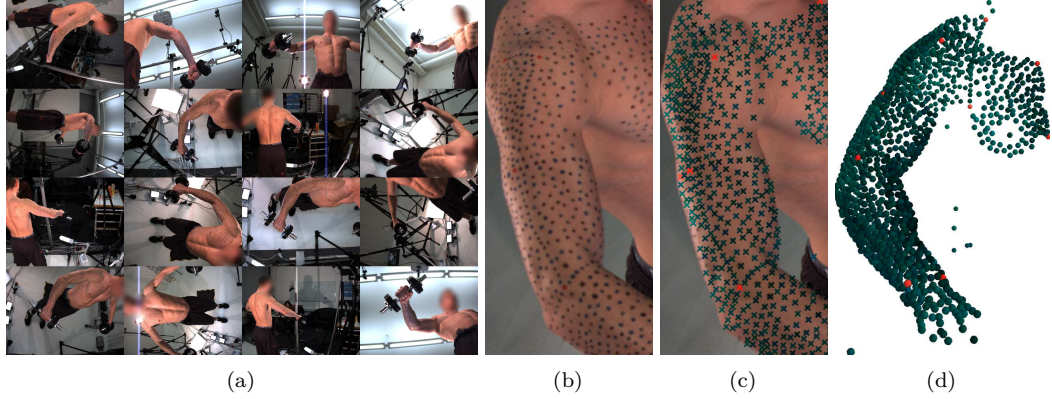


Figure 3.4.: Capturing muscle deformations: (a) Subjects are recorded by 16 synchronized video cameras. (b) The random-dot pattern applied to the skin. (c) Automatic detections of dots. (d) Reconstructed 3D point cloud.

### 3.2.2. Multiview 3D Reconstruction and Registration

The 3D reconstruction pipeline consists of four steps: 3D marker reconstruction, template initialization, template registration across frames, and articulated motion estimation. These steps are described in the following.

**3D Marker Reconstruction.** In each frame of the multi-camera video, Figure 3.4(a), markers are detected with sub-pixel precision using Gaussian blob templates, Figure 3.4(c). To match markers across views, the approach presented in Section 2.3 is used. The result is a dense point cloud of 3D markers for each time step, Figure 3.4(d). However, these point clouds are not yet in temporal correspondence across frames.

**Template Initialization.** All point clouds are brought into correspondence by registering them to a common template mesh. This approach is advantageous: it not only puts into correspondence all the movements from one subject, it also automatically brings different subjects into correspondence. The template mesh  $\mathbb{M}^{(0)}$  used here has  $N = 5\,152$  vertices and has been extracted from the arm section of the average mesh of human body scans provided by [HSS+09]. The template initialization step needs to be performed only once for each subject, in a body pose where the arm is straight and the shoulder abducted by  $90^\circ$ . This initialization pose features the least occlusions.

To fit the template mesh to the reconstructed point cloud of the above-mentioned initialization pose, the 11 anatomical markers are manually marked as shown in Figure 3.5(a). The rest of the fitting procedure is fully-automatic and consists of a rough pose alignment to these landmarks, followed by a

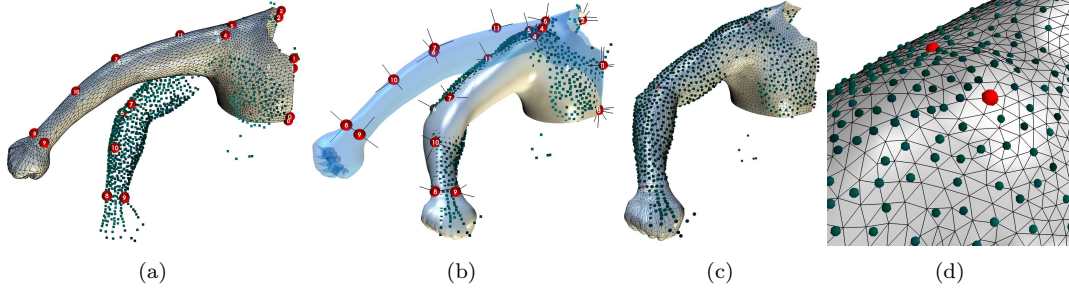


Figure 3.5.: Template initialization step, performed once per subject. (a) The landmarks (red) are selected in the 3D-reconstructed point cloud (green). Landmarks correspond to predefined landmark vertices in the template mesh (grey). (b) The template mesh (transparent blue) is warped to fit the landmarks (red), yielding an approximated surface estimate (grey). Notice that this works well even though the template mesh and initialization point cloud show slight differences in pose. (c) The estimate is refined using non-rigid registration to fit the template to the true geometry of the arm. (d) The zoomed-in view shows that the registered mesh closely fits the reconstructed 3D points.

non-rigid fine registration. The approximate alignment warps the template towards the 11 landmarks by means of Laplacian deformation with rotation correction to compensate for the rotation-variance of differential coordinates [SKR+06]. Such a rotation correction scheme allows the template mesh and the selected initialization pose to deviate, as is the case in Figure 3.5, which is beneficial since subjects rarely manage to exactly reproduce a requested initialization pose. The mesh-based deformation method also tends to work better than skeleton-based Inverse Kinematics (IK), since it can easily deal with differing bone lengths across subjects, gives a precise fit to the anatomical landmarks, and avoids skinning artifacts in the training data. To reproduce the fine-scale detail of the subjects’ arm geometry, a subsequent step performs non-rigid registration. Matches are established between 3D points of the point cloud and vertices on the template mesh, the latter given by the barycentric coordinates relative to the enclosing triangles. In this process, points are matched conservatively: matches further apart than a preset threshold are discarded, which effectively ignores outliers in the 3D reconstruction. The mesh is then displaced towards these closest points with as-rigid-as-possible surface deformation [SA07] utilized as regularization. This amounts to solving a sparse linear system containing the mesh Laplacian and the positional constraints, followed by re-computation of the Laplacian coordinates and search for new correspondences, iteratively. Usually, 10 iterations are sufficient until the mesh tightly fits the 3D-reconstructed point cloud, as confirmed by Figure 3.5(c) and the close-up view in Figure 3.5(d).



Figure 3.6.: Dense correspondences (lines) between marker point clouds automatically align the initialization template to all other ca. 3 000 poses of each subject. Correspondence lines are colored to make individual matches better distinguishable. Occluded areas (dark grey) are interpolated.

**Marker Matching and Template Registration Across Motions.** Once the template is initialized, it has to be aligned to all other captured poses of a subject. Instead of relying on template tracking and thereby suffering from drift, 3D markers are robustly matched across huge pose differences using the shape context matching method described in Section 2.3.4. This gives point-to-point correspondences between the template-initialized pose and any other pose of the same subject, as visualized by the lines in Figure 3.6. Similar to the template initialization step, the mesh is then deformed to globally fulfil the shape context correspondences while interpolating occluded areas. Interpolation is done using Laplacian deformation with rotation correction [SKR+06]. In fact, the template mesh offers a strong prior for extending marker correspondences. After the template is deformed using the set of shape context matches, new correspondences are found by searching within the local neighborhood of the deformed template and matching closest predicted marker locations for each unmatched marker. This process of marker matching, template deformation and correspondence update is iterated until no new correspondences can be added, which usually takes around 5 – 10 iterations. Note that since this template alignment procedure is applied to the data of each individual, alignment of the entire database comes for free. User intervention is needed only for the template initialization step, once for each subject in the database. Since all poses are independently matched to the initialization pose, this registration pipeline step can be parallelized trivially (parallelization was implemented using GNU Parallel [Tan11]). Figure 3.8 demonstrates that 3D reconstructions obtained using this approach are able to measure muscle deformations at high detail.

**Articulated Motion Estimation.** All reconstructed poses are now registered based on a template mesh, but training the muscle model additionally requires the joint angles for these poses. The template mesh was thus prepared with a skinned and rigged skeleton, defining the rotations of the 4 main parts of the shoulder-arm complex: Torso, upper arm, lower arm, and hand, as shown in Figure 3.7. A rigid body rotation  $\mathbf{R}^{(p,b)}$  and translation  $\mathbf{t}^{(p,b)}$  are found that best map the rest-pose vertices  $\mathbf{v}^{(0)}$  of each body part  $b \in \mathcal{B}$  to the vertices  $\mathbf{v}^{(p)}$  in the observed pose  $p$  by minimizing the distance

$$\mathbf{R}^{(p,b)}, \mathbf{t}^{(p,b)} = \arg \min_{\mathbf{R} \in \mathcal{SO}(3), \mathbf{t} \in \mathbb{R}^3} \sum \left\| \mathbf{w}^{(b)} (\mathbf{R} \mathbf{v}^{(0)} + \mathbf{t} - \mathbf{v}^{(p)}) \right\|_2^2 \quad (3.2)$$

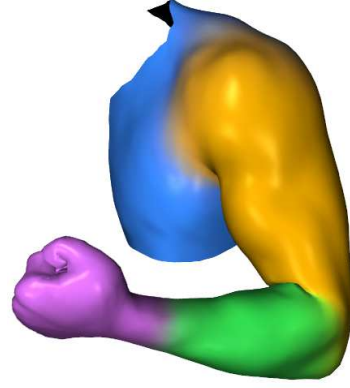


Figure 3.7.: Visualization of skinning weights  $\mathbf{w}$

for each body part separately. We define  $\mathbf{w}^{(b)} \in (0, 1)^N$  to correspond to the given skinning weights for each body part  $b \in \mathcal{B}$ . An initial rotation is found by solving the orthogonal Procrustes problem using SVD [Ume91]. From this starting point, a more accurate rigid transformation is estimated by iterative minimization using a Newton-Raphson method and the twist representation of the rotation, similar to [BM98] but without the need for a kinematic chain. These segment transformations are aligned between different subjects by expressing them relative to a skeleton rigged inside the template mesh. Similar to conventional motion-capture, rotation angles are extracted from  $\{\mathbf{R}^{(p,b)} \mid b \in \mathcal{B}\}$ , resulting in rotation angles  $\boldsymbol{\theta}^{(p)}$  for all body parts.

### 3.2.3. Shape Parameterization

Surface deformation is often generated from joint angles  $\boldsymbol{\theta}$  only, through skeleton skinning, a widely used practice in animation [JDKL14; KCZO08; MTLT88; VBG+13]. However, the proposed muscle model is supposed to generate muscle surface deformation as a function of body pose ( $\boldsymbol{\theta}$ ), shape ( $\boldsymbol{\beta}$ ) and external forces representing triggers for muscle activation ( $\boldsymbol{\gamma}$ ). To include the additional input dimensions, force  $\boldsymbol{\gamma}$  and body shape  $\boldsymbol{\beta}$ , the proposed model adopts a two layered representation. The first layer is provided by triangle-based quaternion blend skinning [WSLG07]. The second layer accommodates for residual deformations between the skinned mesh and the actual detailed skin and muscle deformation. These residuals are learned by the mathematical model  $\Psi$  from the training data.

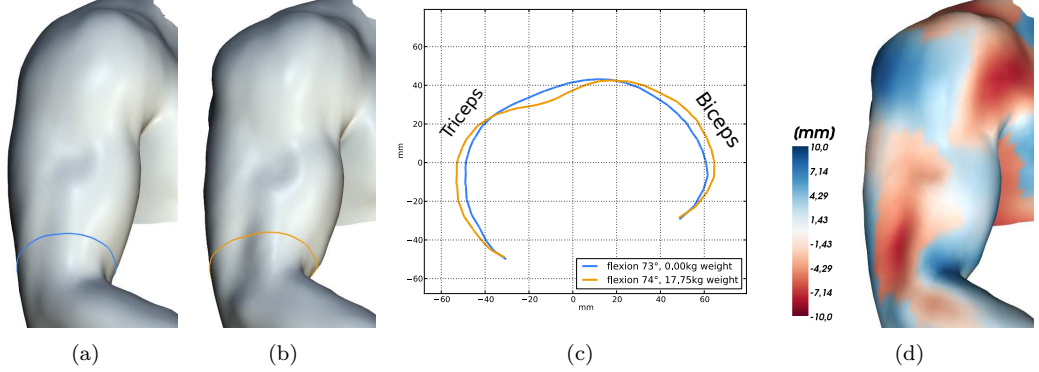


Figure 3.8.: Muscle deformations due to external load in elbow flexing motion, as captured for one subject in the dataset. (a) Flexing with no external load (b) Flexing with 17.75 kg of barbell weight in one hand (c) Cross-sectional cut along the horizontal plane at the bottom of the upper arm. Subtle bulges can be seen on triceps and biceps muscles, which work as an agonist-antagonist pair to keep the arm in equilibrium against the external load. (d) Vertex displacements from (a) to (b) visualized on the mesh. Bulges are shown in blue, and dimples are shown in red (an animated view is shown in the supplementary video).

To this end, it is necessary to encode the shape, converting the mere vertex coordinates  $\mathbf{V}^{(p)} \in \mathbb{R}^{N \times 3}$  of the mesh  $\mathbb{M}^{(p)}$  in pose  $p$  into a representation from which we can easily learn shape changes. If 3D positions  $\mathbf{V}^{(p)}$  are used directly, rotations cannot be modeled easily because rotations are non-linear within that representation. Even if deformations were encoded in pose space [LCF00; SRC01], this representation would still cause artifacts because arm lengths and shapes vary between subjects. I therefore build upon and extend ideas on deformation gradient encoding [SP04] to learn rotation and stretch transformation per template triangle from the training data.

In order to introduce this idea, let us consider a single triangle in pose  $p$  with vertices  $\mathbf{v}_1^{(p)}, \mathbf{v}_2^{(p)}, \mathbf{v}_3^{(p)} \in \mathbb{R}^3$ . Following [SP04], we represent its deformation gradient from a rest pose  $s$  as

$$\mathbf{D}^{(p)} = \mathbf{T}^{(p)} \cdot (\mathbf{T}^{(s)})^{-1} \quad (3.3)$$

where the local transformation of the triangle is given by stacking two edge vectors and the triangle normal as follows:

$$\mathbf{T}^{(p)} = \left[ \begin{array}{c} (\mathbf{v}_2^{(p)} - \mathbf{v}_1^{(p)}) \mid (\mathbf{v}_3^{(p)} - \mathbf{v}_1^{(p)}) \mid \frac{\mathbf{n}^{(p)}}{\|\mathbf{n}^{(p)}\|_2} \end{array} \right] \in \mathbb{R}^{3 \times 3} \quad , \quad (3.4)$$

and for  $\mathbf{T}^{(s)}$  correspondingly. Here,  $\mathbf{n}^{(p)} = (\mathbf{v}_2^{(p)} - \mathbf{v}_1^{(p)}) \times (\mathbf{v}_3^{(p)} - \mathbf{v}_1^{(p)})$  is the triangle face normal. Notice that the deformation gradient in Eq. (3.3) is an adaptation of the 2D deformation gradient in

Eq. (2.32) to 3D triangle meshes. The formulation can be simplified by setting  $(\mathbf{T}^{(s)})^{-1} = \mathbf{T}^{(s)} = \mathbf{I}$ , i.e. by constraining the rest-pose triangle to lie in the  $xy$ -plane with edge lengths 1, with the benefit of achieving a compact deformation representation.

Let us now adopt the above-described concept for the whole mesh  $\mathbb{M}$  (not necessarily in a pose  $p$  from the training set). For each of the  $M$  triangles, we have a deformation gradient  $\mathbf{D}^{(j)} \in \mathbb{R}^{3 \times 3}$ ,  $j \in 1, \dots, M$ . This deformation gradient encoding is translation invariant. To decode the deformation gradients and recover the absolute vertex positions  $\hat{\mathbf{V}} \in \mathbb{R}^{N \times 3}$ , we need at least one anchor point  $\{\mathbf{p}^{(a)} \in \mathbb{R}^3 \mid a \in \mathcal{A}\}$  that breaks the translational invariance, where  $\mathcal{A} \subseteq \{1, \dots, N\}$ ,  $\mathcal{A} \neq \emptyset$  is the set of vertex indices corresponding to the anchors. The anchor positions  $\mathbf{p}^{(a)}$  are predefined, for example located at a vertex on the torso that shall be fixed in space. Note that Eq. (3.3) is a linear operator, implicating that the decoding process is equivalent to a Poisson partial differential equation with Dirichlet boundary conditions at the anchors [SP04; BSPG06]. A common practice is to express both Eq. (3.3) and the boundary conditions in a least squares sense. Then, for the special case of  $\mathbf{T}^{(s)} = \mathbf{I}$  considered here, reconstruction takes a particularly simple form:

$$\begin{aligned} \hat{\mathbf{V}} = \arg \min_{\mathbf{V} \in \mathbb{R}^{N \times 3}} \sum_{j=1}^M \left\| \begin{bmatrix} -1 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \mathbf{V}_{\mathcal{V}(j)} - \begin{bmatrix} \mathbf{D}_{1,1}^{(j)} & \mathbf{D}_{2,1}^{(j)} & \mathbf{D}_{3,1}^{(j)} \\ \mathbf{D}_{1,2}^{(j)} & \mathbf{D}_{2,2}^{(j)} & \mathbf{D}_{3,2}^{(j)} \end{bmatrix} \right\|_2^2 \\ + \omega \sum_{a \in \mathcal{A}} \left\| \mathbf{p}^{(a)} - (\mathbf{V}_a)^\top \right\|_2^2, \end{aligned} \quad (3.5)$$

where  $\mathcal{V}(j)$  gives the vertex indices for the  $j$ th triangle and  $\omega$  is the predefined anchor strength, which is safely set to 100. Eq. (3.5) can be expressed compactly with a linear operator  $\mathbf{G} \in \mathbb{R}^{2M \times N}$  having only two non-zero entries ( $-1$  and  $1$ ) in each row:

$$\hat{\mathbf{V}} = \arg \min_{\mathbf{V}} \left\| \mathbf{G}\mathbf{V} - \hat{\mathbf{D}} \right\|_{\text{F}}^2 + \omega \sum_{a \in \mathcal{A}} \left\| \mathbf{p}_a - \mathbf{v}_a \right\|_2^2. \quad (3.6)$$

$\hat{\mathbf{D}} \in \mathbb{R}^{2M \times 3}$  is given by stacking the deformation gradients  $\mathbf{D}^{(j)}$  but ignoring the third columns from  $\mathbf{D}^{(j)}$ . This third column contains the triangle normal and is not needed for the solution of Eqns. (3.5) and (3.6), as proven by Botsch et al. [BSPG06]. However, the triangle normal is still necessary for computing  $\mathbf{D}^{(j)}$  from the training examples, Eq. (3.3). Furthermore, it is essential for the decomposition step described in the following.

**Decomposition into residual transformations.** Dropping the subscript  $j$  for now, we separate the rotation and stretch components of the deformation gradient  $\mathbf{D} = \mathbf{R}\mathbf{S}$  by polar decomposition [SHD92]. The rotation  $\mathbf{R} \in \mathcal{SO}(3)$  can be represented in axis-angle representation  $\mathbf{r} \in \mathfrak{so}(3)$  using the

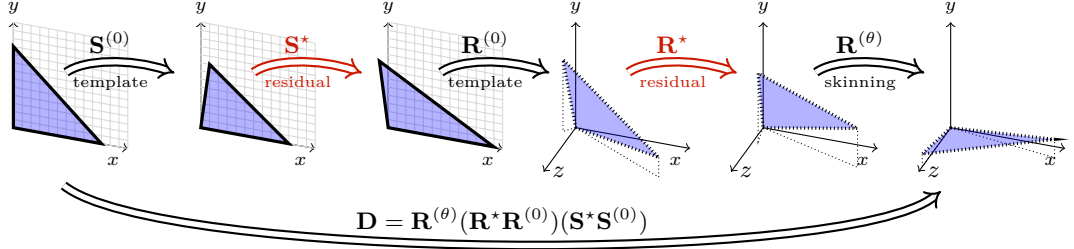


Figure 3.9.: Visualization of the decomposition scheme of Eq. (3.8). The deformation gradient  $\mathbf{D}$  describes a transformation between a canonical  $xy$ -plane triangle (leftmost) and a triangle as measured in the captured data.  $\mathbf{D}$  is decomposed into shear/stretch transformations in the 2D plane, followed by 3D rotations. While some deformations are given by the template or computed from skinning,  $\mathbf{S}^*$  and  $\mathbf{R}^*$  (red) describe shape changes and muscle bulges. These residuals are learned by the muscle model  $\Psi$ .

log and exponential maps to perform conversions to and from this representation:  $\mathbf{R} = \exp(\mathbf{r}) \leftrightarrow \mathbf{r} = \log(\mathbf{R})$ . Rodrigues' formula provides a closed-form solution for this step [MLS94]<sup>2</sup>. This turns the redundant representation in  $\mathcal{SO}(3)$  with 9 parameters per triangle into one that has only 3 parameters per triangle, with the added benefit of representing rotation around an axis linearly. Since it was assumed earlier that  $\mathbf{T}^{(s)} = \mathbf{I}$ , the remaining stretch deformation  $\mathbf{S}$  completely happens in the 2D plane, and thus is compressible as well:  $\mathbf{S}$  has only 3 non-zero components for representing  $xy$ -plane scaling,  $\mathbf{S}_{1,1}$   $\mathbf{S}_{2,2}$ , and shear  $\mathbf{S}_{1,2}$ . These can be assembled into a vector  $\mathbf{s} = [\mathbf{S}_{1,1} \mathbf{S}_{2,2} \mathbf{S}_{1,2}]^\top$ . In summary, there are 3 scalars in  $\mathbf{r}$  and 3 scalars in  $\mathbf{s}$  for each of the  $M$  triangles. Thus, the deformation gradient encoding maps from vertex coordinates in  $\mathbb{R}^{N \times 3}$  to  $\mathbb{R}^{M \times 6}$  (for the specific template mesh used here  $N = 5152$  and  $M = 10164$ ).

The deformation gradient  $\mathbf{D}$  of a triangle is decomposed further to yield a shape encoding better suited for learning. Let  $\mathbf{R}^{(\theta)}$  be the pose-dependent rotation for the triangle given by articulated segment rotations from body part rotation angles  $\theta$ . The deformation of the triangle with respect to this simple pose-dependent transformation is given as

$$\mathbf{D}^{(\theta)} = \mathbf{R}^{(\theta)} \mathbf{D}^{(0)} = \mathbf{R}^{(\theta)} (\mathbf{R}^{(0)} \mathbf{S}^{(0)}) \quad (3.7)$$

<sup>2</sup>The log map can be tricky to implement due to numerical instabilities, a fact that is not well covered in the literature. Instabilities can be fixed by converting first to quaternions, then to rotation vectors; or by following the tips in <https://rip94550.wordpress.com/2008/09/28/angle-and-axis-of-rotation-2-the-two-correct-answers/>

where  $\mathbf{D}^{(0)} = \mathbf{R}^{(0)}\mathbf{S}^{(0)}$  is the deformation gradient from the unit triangle in the  $xy$ -plane to the triangle in the template pose. However, the reconstructed shapes show residual deformations that appear both as the rotation ( $\mathbf{R}^*$ ) and stretch/shear ( $\mathbf{S}^*$ ) components

$$\mathbf{D} = \mathbf{R}^{(\theta)}(\mathbf{R}^*\mathbf{R}^{(0)})(\mathbf{S}^*\mathbf{S}^{(0)}) \quad . \quad (3.8)$$

Figure 3.9 visualizes the decomposition process.

As discussed before, residual deformations can be represented in a compact manner using only 6 scalars per triangle: 3 for representing the rotation in axis-angle representation, and 3 for the stretch matrix. Together, these parameters compose a residual deformation gradient  $\mathbf{D}^* = \mathbf{R}^*(\mathbf{R}^{(0)}\mathbf{S}^*(\mathbf{R}^{(0)})^\top)$  that maps the skinned mesh to the observed mesh, such that  $\mathbf{D} = \mathbf{R}^{(\theta)}\mathbf{D}^*\mathbf{D}^{(0)}$ . The final shape representation vector  $\mathbf{y} \in \mathbb{R}^{6M}$  that represents the residual transformations  $(\mathbf{R}_j^*, \mathbf{S}_j^*)$  for all the triangles is given by

$$\mathbf{y} = \text{vec} \left( \begin{bmatrix} (\mathbf{S}_1^*)^\top & (\mathbf{R}_1^*)^\top \\ \vdots & \vdots \\ (\mathbf{S}_M^*)^\top & (\mathbf{R}_M^*)^\top \end{bmatrix} \right) \quad . \quad (3.9)$$

**Details of the Skinning Approach.** The body part rotations  $\mathbf{R}^{(\theta)}$  are either given by the articulated motion estimation in the training phase, Eq. (3.2), or from input data such as user input or motion capture in the testing phase. To obtain the per-triangle rotation  $\mathbf{R}_j^{(\theta)}$ , rotations are combined according to the blending weights from two nearby body parts (e.g. upper and lower arm). Let the nearby body parts for triangle  $j$  be  $b_1$  and  $b_2$ , then the corresponding blending weights are  $\alpha_j^{(b_1)} = \left( \sum_{i \in \mathcal{V}(j)} \mathbf{w}_i^{b_1} \right) / 3$ ,  $\alpha_j^{(b_2)} = \left( \sum_{i \in \mathcal{V}(j)} \mathbf{w}_i^{b_2} \right) / 3$ , averaged from the weights  $\mathbf{w}$  of the vertices  $\mathcal{V}(j)$  of triangle  $j$ . The rigid rotations from the two body parts  $\mathbf{R}^{(b_1)}, \mathbf{R}^{(b_2)}$  are blended within their axis-angle representations using the exponential map to obtain faithful blending [WSLG07]:

$$\mathbf{R}_j^{(\theta)} = \mathbf{R}^{(b_1)} \exp \left( \alpha_j^{(b_2)} \log \left( (\mathbf{R}^{(b_1)})^\top \mathbf{R}^{(b_2)} \right) \right) \quad . \quad (3.10)$$

We can assume that  $\alpha_j^{b_1} + \alpha_j^{b_2} = 1$  by construction, since the template mesh has no areas where more than two body parts are close to each other. Most importantly, notice that the blend skinning completely acts in deformation gradient space, on the per-triangle rotations  $\mathbf{R}_j$ , instead of acting on the vertex positions directly. This skinning method prevents the infamous *candy wrapper* artifacts [JDKL14] and, since it works on a per-triangle basis, matches well with the proposed per-triangle shape encoding.

### 3.2.4. Deformation Learning

The aim is now to learn a model that can generate the arm-shoulder shape from given input parameters such as pose, body shape, and external load, in such a way that the generated shape realistically shows the deformations as seen in the recorded data. All  $P$  captured meshes are encoded via Eq. (3.9) to a collection of shape vectors  $\mathbf{y}^{(p)}$ ,  $p \in 1, \dots, P$ , that are in turn assembled in a training matrix  $\mathbf{Y} \in \mathbb{R}^{P \times 6M}$ ; and equivalently with the corresponding  $L$ -dimensional input parameters  $\mathbf{x}^{(p)} = [(\boldsymbol{\theta}^{(p)})^\top, (\boldsymbol{\beta}^{(p)})^\top, (\boldsymbol{\gamma}^{(p)})^\top, 1]^\top \in \mathbb{R}^L$  to form  $\mathbf{X} \in \mathbb{R}^{P \times L}$ . The constant 1 is added to include a bias term [Bis06]. The learning task is to estimate a function  $\Psi : \mathbb{R}^L \rightarrow \mathbb{R}^{6M}$  that is able to generate a shape vector from a novel, previously unseen  $\mathbf{x}$ .

An obvious choice for function  $\Psi$  is a linear regression model which can be trained by minimizing the sum of squared differences between the training examples and the model predictions, and therefore results in the following linear least squares problem

$$\underset{\mathbf{W} \in \mathbb{R}^{L \times 6M}}{\text{minimize}} \quad \|\mathbf{X}\mathbf{W} - \mathbf{Y}\|_{\text{F}}^2 \quad . \quad (3.11)$$

The evaluation phase is then a simple matrix multiplication,  $\Psi_{\text{linear}}(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ . This linear model usually is less prone to overfitting (due to the large size of the dataset) and generalizes well along dimensions where limited training data is available, e.g. the body shape of the person. However, for the pose-specific parameters (the  $L_\theta < L$  joint angles  $\boldsymbol{\theta} \in \mathbb{R}^{L_\theta}$ ), the linear model fails to produce details such as fine concavities for certain elbow poses.

Therefore, additional non-linearities have to be learned for a more accurate non-linear model. Since the captured dataset sufficiently covers the space of  $L_\theta$  pose parameters, it is possible to build such a model without overfitting. Non-parametric methods such as kernel ridge regression (KRR) are well suited in this situation [Bis06]. The main idea is to replace the  $L_\theta$ -dimensional input parameters by a much higher (potentially infinite) dimensional feature vector,  $\phi(\boldsymbol{\theta}) : \mathbb{R}^{L_\theta} \rightarrow \mathbb{R}^H$ , where  $H$  is the dimensionality of the higher-dimensional space. Gathering all feature vectors into  $\boldsymbol{\Phi} \in \mathbb{R}^{P \times H}$  gives the new high-dimensional representation of the whole dataset, with rows  $\boldsymbol{\Phi}_p = \phi(\boldsymbol{\theta}^{(p)})$ . To avoid overfitting with this huge number of features, regularization (with strength  $\lambda \in \mathbb{R}_+$ ) is needed. The coefficients of a linear model learned in higher-dimensional space are then given by

$$\hat{\mathbf{V}} = \underset{\mathbf{V} \in \mathbb{R}^{H \times 6M}}{\arg \min} \quad \|\boldsymbol{\Phi}\mathbf{V} - \mathbf{Y}\|_{\text{F}}^2 + \frac{\lambda}{2} \|\mathbf{V}\|_{\text{F}}^2 \quad . \quad (3.12)$$

One can apply the well-known Kernel Trick [Wel; Bis06] by using [PP12, p. 18, Eq. 155] to rewrite the solution of Eq. (3.12) as

$$\hat{\mathbf{V}} = (\mathbf{\Phi}^\top \mathbf{\Phi} + \lambda \mathbf{I}_H)^{-1} \mathbf{\Phi}^\top \mathbf{Y} = \mathbf{\Phi}^\top (\mathbf{\Phi} \mathbf{\Phi}^\top + \lambda \mathbf{I}_P)^{-1} \mathbf{Y} = \mathbf{\Phi}^\top (\mathbf{K} + \lambda \mathbf{I}_P)^{-1} \mathbf{Y} \quad . \quad (3.13)$$

The last step introduces the Gram matrix  $\mathbf{K} = \mathbf{\Phi} \mathbf{\Phi}^\top$ . The kernel trick circumvents the definition of the high-dimensional  $\mathbf{\Phi}$  altogether. Instead, the Gram matrix can be specified directly from a kernel function  $\mathbf{K}_{p,q} = \kappa(\boldsymbol{\theta}^{(p)}, \boldsymbol{\theta}^{(q)})$ , precomputed over all pairs of training samples  $p, q \in 1, \dots, P$ . Here, the Gaussian kernel was chosen, relating to the non-linear map  $\phi$  as follows

$$\phi(\boldsymbol{\theta}^{(p)})^\top \phi(\boldsymbol{\theta}^{(q)}) = \kappa(\boldsymbol{\theta}^{(p)}, \boldsymbol{\theta}^{(q)}) = \exp \left( -\frac{\|\boldsymbol{\theta}^{(p)} - \boldsymbol{\theta}^{(q)}\|_2^2}{\sigma} \right) , \quad (3.14)$$

where  $\sigma > 0$  is a hyper-parameter corresponding to the scale of the Gaussian kernel. During evaluation, the kernel ridge regression (KRR) synthesizes pose-dependent shapes by

$$\Psi_{\text{KRR}}(\boldsymbol{\theta}) = \hat{\mathbf{V}}^\top \phi(\boldsymbol{\theta}) = \mathbf{C}^\top \mathbf{\Phi} \phi(\boldsymbol{\theta}) = \mathbf{C}^\top \begin{bmatrix} \kappa(\boldsymbol{\theta}, \boldsymbol{\theta}^{(1)}) \\ \vdots \\ \kappa(\boldsymbol{\theta}, \boldsymbol{\theta}^{(P)}) \end{bmatrix}_{(P \times 1)} \quad (3.15)$$

where the auxiliary coefficients  $\mathbf{C} \in \mathbb{R}^{P \times 6M}$  were introduced by substituting  $\hat{\mathbf{V}} = \mathbf{\Phi}^\top \mathbf{C}$ . If the same substitution is done in the least squares problem Eq. (3.12), its dual formulation is recovered:

$$\underset{\mathbf{C} \in \mathbb{R}^{P \times 6M}}{\text{minimize}} \|\mathbf{K} \mathbf{C} - \mathbf{Y}\|_{\text{F}}^2 + \frac{\lambda}{2} \text{Tr}(\mathbf{C}^\top \mathbf{K} \mathbf{C}) \quad . \quad (3.16)$$

Since the Gram matrix  $\mathbf{K}$  is symmetric (even positive semi-definite), the gradient of this objective function takes the form  $\mathbf{K}(\mathbf{K} \mathbf{C} - \mathbf{Y}) + \lambda \mathbf{K} \mathbf{C}$ . Setting this gradient to zero gives the optimal weights

$$\mathbf{C} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{Y} \quad . \quad (3.17)$$

Therefore, it is not necessary to actually work with the high-dimensional feature space of  $\mathbf{\Phi}$ , which is an important result in machine learning [Bis06]. I shall note that this result can alternatively be derived using the reproducing kernel Hilbert space and the representer theorem [DD04; Wel] or from a full Bayesian treatment in the context of Gaussian Processes [KC05].

Training the kernel ridge regression, Eq. (3.17), requires solving a linear system of size  $P \times P$  for  $6M$  right hand sides - recall that  $\mathbf{Y}$  in Eq. (3.17) is a  $P \times 6M$  matrix. This is prohibitively expensive for the captured dataset of  $P \approx 32,000$  meshes. On top of this, such a model would need to compute

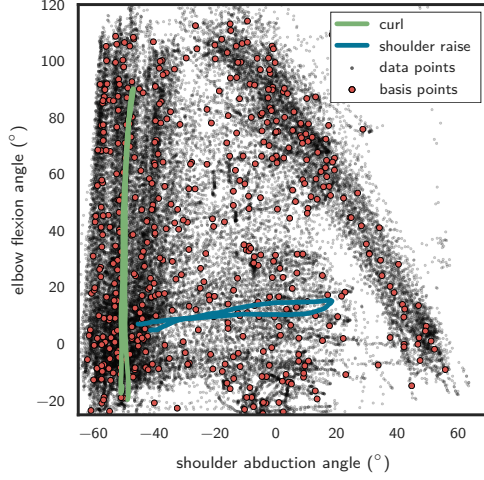


Figure 3.10.: Illustration of the basis approximation in pose space: Each small black dot is one sample in the dataset of  $F \approx 32\,000$  meshes, plotted according to elbow flexion and shoulder abduction angles. These are the most important pose parameters in the model, and the plot clearly shows how densely these parameters are sampled. The blue curve shows a typical shoulder abduction motion, the green curve shows a biceps curl. The muscle model finds  $B$  basis vectors  $\mathbf{B}$  in pose space, shown in red. These basis vectors approximate the full non-linear kernel regression in Eq. (3.20).

the kernel function  $P$  times at test time, which would prevent interactive use of the muscle model. Kim and Kwon [KK10; KK08] face a similar problem in the scope of image super-resolution. They provide a solution I take inspiration from: a sparse approximation of the kernel ridge regression. The approach is based on finding a set of  $B (\ll P)$  basis points  $\mathbf{B} \in \mathbb{R}^{B \times L_\theta}$  from the pose input parameters. Then, the solution is estimated only with respect to those few basis points,

$$\underset{\mathbf{C}^{(B)} \in \mathbb{R}^{B \times 6M}}{\text{minimize}} \left\| \mathbf{K}^{(B)} \mathbf{C}^{(B)} - \mathbf{Y} \right\|_F^2 + \frac{\lambda}{2} \text{Tr} \left( (\mathbf{C}^{(B)})^\top \mathbf{K}^0 \mathbf{C}^{(B)} \right), \quad (3.18)$$

where  $\mathbf{K}_{b,p}^{(B)} = \kappa((\mathbf{B}_b)^\top, \theta^{(p)})$ ,  $b \in 1, \dots, B$  contains the kernel distance from all data points to all basis points, and the basis Gram matrix  $\mathbf{K}_{b,c}^{(0)} = \Phi^{(0)}(\Phi^{(0)})^\top = \kappa((\mathbf{B}_b)^\top, (\mathbf{B}_c)^\top)$  has the distances of all basis points to each other, with  $b, c \in 1, \dots, B$ . The solution of the minimization is then  $\mathbf{C}^{(B)} = \left( (\mathbf{K}^{(B)})^\top \mathbf{K}^{(B)} + \lambda \mathbf{K}^{(0)} \right)^{-1} \mathbf{K}^{(B)} \mathbf{Y}$ . In this case it is impossible to further simplify to the form of Eq. (3.17), since the regularization in Eq. (3.18) measures distances over the reproducing kernel Hilbert space of the basis  $\mathbf{B}$  [DD04; Wel; KK10]. Nevertheless, this scheme is faster during training time. During test time, the kernel needs to be evaluated only with the few  $B$  basis vectors:

$$\Psi_{\text{SKRR}}(\theta) = (\mathbf{C}^{(B)})^\top \begin{bmatrix} \kappa(\theta, (\mathbf{B}_1)^\top) \\ \vdots \\ \kappa(\theta, (\mathbf{B}_B)^\top) \end{bmatrix}_{(B \times 1)}. \quad (3.19)$$

The basis  $\mathbf{B}$  is obtained by running K-means clustering on the pose parameters of the training set, utilizing the cluster center locations as the basis points. Essentially, this clusters the pose space, as demonstrated in Figure 3.10.

In the final model, linear regression as in Eq. (3.11) is combined with the sparse kernel regression from Eq. (3.18); to learn such a model one needs to minimize the following expression:

$$\underset{\substack{\mathbf{C}^{(B)} \in \mathbb{R}^{B \times 6M} \\ \mathbf{W} \in \mathbb{R}^{L \times 6M}}}{\text{minimize}} \left\| \begin{bmatrix} \mathbf{K}^{(B)} & \mathbf{X} \end{bmatrix} \begin{bmatrix} \mathbf{C}^{(B)} \\ \mathbf{W} \end{bmatrix} - \mathbf{Y} \right\|_{\text{F}}^2 + \frac{\lambda}{2} \text{Tr} \left( (\mathbf{C}^{(B)})^\top \mathbf{K}^0 \mathbf{C}^{(B)} \right) . \quad (3.20)$$

Training the muscle model is implemented by solving for the optimal coefficients:

$$\begin{bmatrix} \mathbf{C}^{(B)} \\ \mathbf{W} \end{bmatrix} = \left( \begin{bmatrix} \mathbf{K}^{(B)} \mathbf{K}^{(B)} & \mathbf{K}^{(B)} \mathbf{X} \\ \mathbf{X}^\top \mathbf{K}^{(B)} & \mathbf{X}^\top \mathbf{X} \end{bmatrix} + \begin{bmatrix} \lambda \mathbf{K}^{(0)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \right)^{-1} \begin{bmatrix} \mathbf{K}^{(B)} \\ \mathbf{X}^\top \end{bmatrix} \mathbf{Y} . \quad (3.21)$$

Regularization on the kernel-transformed pose parameters is induced by matrix  $\mathbf{K}^{(0)}$ , and not by the identity as in Eq. (3.17). This is absolutely essential as it controls model complexity inside the non-linearity of the kernel. In Eq. (3.21), we can see that training the model requires inverting (or prefactorizing) a matrix of dimension  $(B+L) \times (B+L)$ , solving for  $6M$  right-hand sides. The trained model can then synthesize new meshes by mapping new input parameters  $\mathbf{x}$  (with the subset  $\mathbf{x}_{\mathcal{I}(\theta)}$  denoting the pose parameters  $\theta$ ), to the corresponding deformation as follows:

$$\Psi(\mathbf{x}) = \begin{bmatrix} \mathbf{C}^{(B)} \\ \mathbf{W} \end{bmatrix}^\top \begin{bmatrix} \kappa(\mathbf{x}_{\mathcal{I}(\theta)}, (\mathbf{B}_1)^\top) \\ \vdots \\ \kappa(\mathbf{x}_{\mathcal{I}(\theta)}, (\mathbf{B}_B)^\top) \\ \mathbf{x} \end{bmatrix}_{(B+L \times 1)} . \quad (3.22)$$

The rationale behind combining linear and non-linear regression is that this combination better generalizes to novel poses and allows meaningful extrapolation. Compared to linear regression, non-linear regression often performs worse at extrapolation since it relies on distances between data points. A new data point might be in a previously unseen area, far away from the training data points. Most non-linear regression methods then tend to output zero. In this worst case, the muscle model would set the deformation gradients to zero and reproduce the template mesh as its best guess. This is not the case for linear regression, as it maps an entire input domain to the output domain. By combining linear and non-linear regression, we can capture subtle non-linearities in deformations given

|                  | ours  | linear | [HSS+09] |
|------------------|-------|--------|----------|
| curl low weight  | 5.57  | 9.29   | 24.23    |
| curl high weight | 5.94  | 8.86   | 24.87    |
| shoulder spin    | 14.66 | 14.19  | 28.04    |
| arm rotate       | 9.33  | 11.30  | 29.72    |
| boxing           | 14.28 | 15.58  | 49.52    |
| freestyle        | 9.00  | 10.20  | 30.80    |

Table 3.1.: Average errors in vertex displacement (in mm) for our method, a linear model trained with the proposed shape encoding, and a linear regression trained with the encoding from [HSS+09]. Evaluation was performed on movements not used for training.

that we have enough training data. The results indicate both visual and measurable improvement over pure linear modeling.

### 3.3. Results

As described in Section 3.2.1, 10 male subjects were captured, each performing 30 to 40 movements. Each motion sequence is approximately 80 to 100 frames long, resulting in a dataset of approximately 32 000 meshes. The muscle model was trained on a set of 16 input parameters: 3 angles of rotation each at the shoulder and elbow joints<sup>3</sup>, 3 body shape parameters given by height, BMI, and muscularity, and 7 force parameters given by the 3-dimensional force vector of the barbell weight in the local coordinate frames of the lower and upper arm and their magnitude.

**Quantitative Evaluation.** Table 3.1 reports the accuracy of the muscle model. The accuracy was evaluated quantitatively by cross validation, comparing the synthesized meshes with the ground-truth meshes on 6 movements, each of a different subject and each *not* part of the training set. The accuracy was measured in the  $\mathbb{R}^3$  Euclidean space of vertex coordinates. Since the shape encoding is translation-invariant, the meshes need to be aligned rigidly before comparison. Occluded and self-overlapping regions are ignored in all reported experiments since no reliable and noise-free ground truth mesh can be reconstructed in these areas. For comparison purposes, a purely linear model was trained given by Eq. (3.11). Table 3.1 reveals that the proposed non-linear model is superior to a purely linear model. To test the influence of the proposed shape encoding, the linear model was additionally trained on a state-of-the-art second-order deformation encoding [HSS+09]. The shape encoding of Hasler et al. [HSS+09] fails to provide comparative quantitative results since their encoding does not contain the base layer of a skinned skeleton which results in misaligned body parts.

<sup>3</sup>Modeling lower arm surface deformations requires at least two degrees of freedom at the elbow: While the anatomical elbow joint can be sufficiently approximated by one degree of freedom, rotation of radius and ulna in the axis of the lower arm using joints in the hand actually results in rotational deformations in the lower arm. Hence, ulna and radius are unified in the simple animation skeleton (this is usual practice in computer animation). Simply using the full three degrees of freedom at the elbow also simplifies computation, as no true inverse kinematics are required.

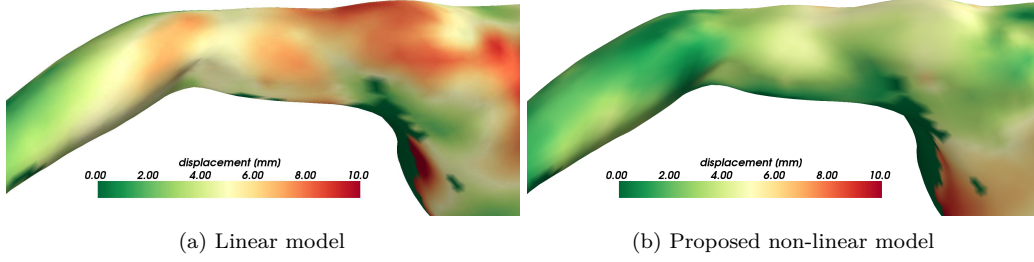


Figure 3.11.: Synthesized meshes for one pose of a *shoulder lift* motion, colored by difference to ground-truth. The measurement was not part of the training set. The proposed non-linear model shows significantly lower error.

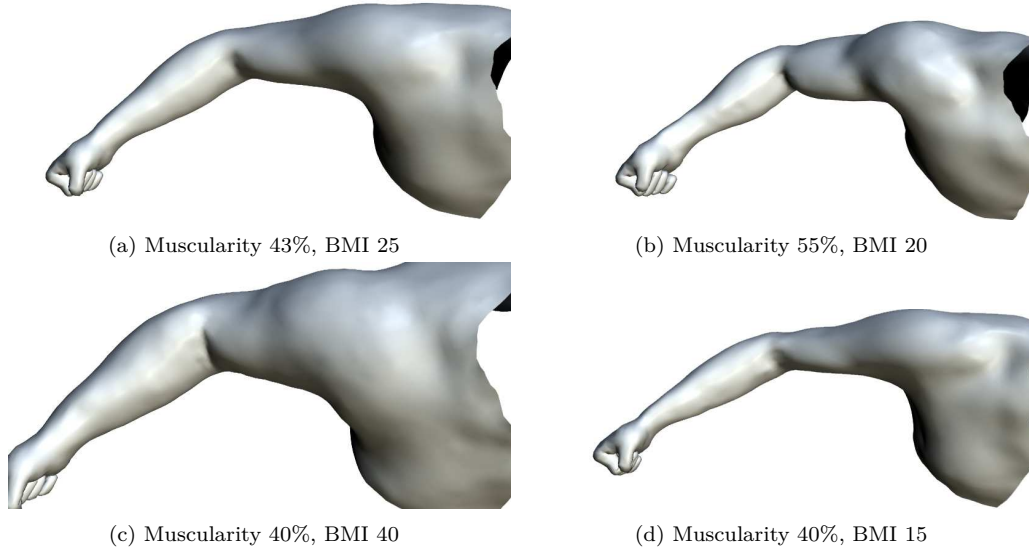


Figure 3.12.: BMI and muscularity are changed to produce new body shape variations with the statistical muscle model.

The proposed kernel-learning method has three hyper-parameters: kernel scale  $\sigma$  in Eq. (3.14), regularization strength  $\lambda$ , and number of basis points  $B$ . I observed that the learning method is robust to different hyper-parameter settings. The best results according to cross-validation were achieved for  $\sigma = 0.1$  and  $\lambda = 10.0$ . Different basis sizes  $B$  were tested: More basis points yield slightly better accuracy but take longer to evaluate. The value  $B = 512$  was found to be a good compromise. Figure 3.11 visualizes the error of the proposed non-linear muscle model against the linear model.

**Visual Results.** The muscle model can be interactively used by an artist to synthesize complex muscle effects by changing pose, shape, or external forces, and to smoothly vary parameter values.

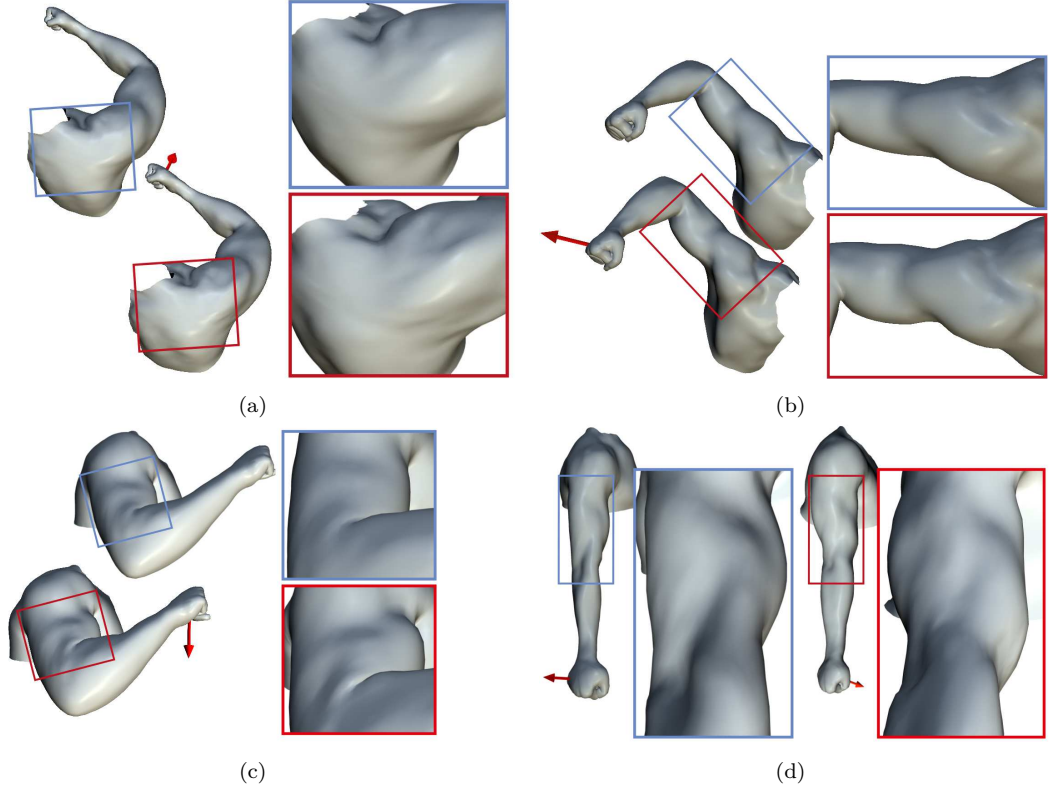


Figure 3.13.: Artist-driven muscle deformations to poses and large external forces not captured in our dataset. (a) Deformation around the *scapula* bone that counteracts some of the force; (b) the *deltoid* and *biceps* muscles get sharply defined by force; (c) *triceps lateral head* and *biceps long head* get pronounced with force; (d) opposing activations of *biceps* and *triceps* as the force vector reverses direction.

Figure 3.12 shows plausible variations in body shape while changing BMI and muscularity parameters. Figure 3.13 demonstrates the muscle models’ major feature: to simultaneously model external forces along with the other parameters. The pose and external force parameters specified to produce these effects are beyond the capture range in the training dataset. One can see physiological effects such as the sharp pronouncement of muscle strands in reaction to external forces (the *deltoid* and *biceps* muscles in Figure 3.13(b), the *biceps* and *triceps* muscles in Figure 3.13(c)), complex skin deformation around joint capsules (*scapula joint* at the back of the shoulder in Figure 3.13(a)), and the marked difference in skin deformation with respect to the direction of the force vector (loose skin around the elbow joint in Figure 3.13(d)). These effects are achieved on the fly using our prototype tool through editing the skeletal pose, body shape and the force vector.

|            | KRR   | decode | fps | size   |
|------------|-------|--------|-----|--------|
| $B = 256$  | 5 ms  | 22 ms  | 37  | 54 MB  |
| $B = 512$  | 10 ms | 22 ms  | 31  | 105 MB |
| $B = 1024$ | 18 ms | 22 ms  | 25  | 207 MB |

Table 3.2.: Performance metrics for varying number of basis vectors  $B$  (first column) for the muscle model. Column "KRR" corresponds to the model prediction runtime, Eq. (3.22). The decoding runtimes (column "decode") consist of several steps including skinning with Eq. (3.10), reversing the decomposition in Eq. (3.8), and solving for the vertex coordinates from the deformation gradients, Eq. (3.6). All models achieve realtime performance while keeping a low memory footprint.

**Runtime Performance.** At runtime, the trained muscle model is extremely fast, requiring only simple matrix operations. Interactive feedback is already possible with an unoptimized Python implementation of the muscle model. Furthermore, the model has a low memory footprint owing to the strategy of sparse-kernel learning and the requirement to store just a few basis vectors: for 512 basis points 105 MB are needed, making the muscle model suitable for deployment in resource constrained applications such as character animation in game consoles. Evaluation times and model sizes for other configurations are given in Table 3.2. The runtimes were measured on a Linux system with Intel i7 3.4 GHz CPU and 16 GB RAM. To obtain the training dataset, 3D reconstruction of 32 000 meshes has to be performed, which takes several hours even on multiple machines. However, the training phase takes no longer than 25 minutes to perform shape encoding, K-means and kernel regression over the full dataset of approximately 32 000 meshes.

### 3.4. Discussion

The proposed muscle model generalizes well beyond the range of inputs captured in the training set but starts to fail when these parameters are far off (e.g, external force that is twice the maximum of the captured barbell weight fails in many poses). A model with fewer parameters, such as the linear regression model, might generalize slightly better than the more accurate non-linear model for such extreme input parameter extrapolation - a tradeoff between model accuracy and model simplicity well known in machine learning [Bis06]. In this sense, the shoulder spin motion in Table 3.1 can be seen as a concrete failure case where the linear model slightly outperforms the non-linear model. Capturing more people and training on a larger dataset would ameliorate the results. Capturing the effects of time-dependent dynamics, such as the wobbling of body fat and soft tissue, is another important area that can benefit from the contributions in 3D data acquisition.

The muscle model embeds a simplistic skeleton structure, akin to those used in animations for computer games. While I view this as the most important limitation of the model, the simplistic skeleton nevertheless demonstrates natural and detailed animation results. In future work, physiologically correct skeletons and physics-based muscle models can further improve the accuracy of the model. A complex skeleton will also enable modeling of subtle secondary movements. This includes motion of the clavicle and scapula which cannot be explained by the current control skeleton. Clavicle deformations are therefore only moderately visible when animating arm pose and external force, even though they are present in the captured data.

It is obvious that capture resolution of the dataset is superior to previous methods, but using the same method with even higher-resolution cameras is necessary for explaining subtler deformations in the future (e.g. vertebral notches, more detail on the clavicle bone). In the future, data-driven models like the proposed one may help to better individualize physiological muscle models to match different people. Similarly, captured surface deformation can be combined with physiological information, recorded from EMG sensors or gathered from biophysical simulation. This potential for cross-fertilization needs to be explored in future work.

## 4. Sparse Localized Deformation Components

Nowadays, time-varying dynamic geometry with very fine shape detail can be generated and rendered at very high visual fidelity. When creating such content, artists usually rely on a low-dimensional control parameterization, for example a kinematic skeleton rig to control surface motion via joint angles, a muscle system simulating facial motion and tissue deformation, or a physics-based simulation model generating realistically deforming cloth or other materials. Despite increasing expressive power of such parameterizations and simulations, producing such realistic animations from scratch is a labor-intensive process, in particular since it is highly non-trivial to design or customize a specific parameterization to a new object to be animated. Performance capture techniques were thus developed that measure detailed, time-varying surface models of the real world [TAS+10]. Two such methods were presented in Chapter 2 for capturing skin and cloth as highly detailed and temporally coherent surface mesh sequences. However, the applicability of performance capture in animation production has been limited so far because a low-dimensional control parameterization for the captured data was missing. Convenient re-use, editing, or analysis of the captured input data was therefore not possible.

To overcome this problem, some recent work suggested to fit specific parametric models to such input data: simple linear regression models [ASK+05; HSS+09], skeletons or bone transformations [ATTS08; KSO10; LD12], or even pre-modeled facial blendshapes [LWP10]. These methods require additional prior knowledge about the underlying physical process of the animation (e.g. a template shape or a parametric physics model), or additional data needs to be recorded alongside the capture process. For example, the muscle model from Chapter 3 requires a rigged skeleton which in turn provides kinematic input parameters (bone angles estimated via inverse kinematics), as well as measured body shape parameters like the BMI of the person that was captured. The muscle model is then built around these specific input parameters. Such supervised approaches are inherently object-type specific and not adoptable to general data.

Dimensionality reduction techniques for animation parameterization rely less on specific prior information and are thus applicable to a broader range of captured data. Many of them are, in essence, matrix decomposition techniques that explain observed deformations as a linear combination of certain factors, or deformation components. Previously used dimensionality reduction techniques,

like Principal Component Analysis (PCA), faithfully reproduce input data and maintain certain compression guarantees, but the individual computed dimensions usually lack interpretable meaning and are of global support, i.e. their modification acts on the whole mesh. In contrast, in order to generate professional animation, artists require crisp controls on plausible local effects which can be edited in an intuitive manner [Hav06].

In this chapter, I introduce a new efficient, easy-to-implement, and versatile data-driven approach to decompose a mesh sequence into intuitively meaningful sparse deformation components, without the need to model the underlying physical process that generated the input data. The method takes inspiration from matrix decomposition methods such as Non-Negative Matrix Factorization (NMF), Robust PCA, and Sparse PCA that have recently become very popular in machine learning, image processing and computer vision. At its core, the new method relies on a sparsity-inducing regularizer that is designed for the input modality of (captured) mesh sequences. It further includes a mechanism to automatically find components that are not only *sparse* but also *localized* on the mesh. I coin these *sparse localized deformation components*, in short SPLOCS. The approach allows incorporating user-provided input in order to guide the sparse decomposition. Additionally, I contribute a novel, efficient optimization and initialization scheme to compute the decomposition.

In addition to the publication in [NVW+13], this chapter provides a more detailed derivation of the optimization method. The chapter also discusses limitations and convergence behavior of SPLOCS more critically and more comprehensively, with supplementary results demonstrating those limitations. I open-sourced a reference implementation of the algorithm<sup>1</sup>.

### 4.1. Related Work

**Skeletal Rigging.** Artists typically resort to parametric models for synthesizing mesh animations. The de-facto standard parameterization for producing articulated motion is linear blend skinning (and its extensions), often combined with hierarchical kinematic skeletons [JDKL14]. The difficulty of fitting such a parametrized skeletal model to example meshes given by an artist is evaluated in [MG03; WSLG07]. State-of-the-art methods can fit simple hierarchical skeletons to articulated mesh sequences [ATTS08] and can automatically construct rigs that explain body shape variations [HTRS10]. However, such skinned skeletons often miss fine-scale surface deformations. The missing details can be re-introduced in pose space using the Eigenskin method [KJP00], which essentially comes down to clustered PCA (with overlapping regions) and linear regression in pose space, and only works given sufficient training data. More general skinning decomposition methods fit unordered

---

<sup>1</sup><http://github.com/tneumann/splocs>

non-hierarchical collections of arbitrary [KSO10] or rigid [LD12] bone transformations and blend weights to mesh sequences. While being useful for quick rendering purposes using GPUs, these are not so useful for editing: The bones offer many degrees of freedom that quickly produce deformations far beyond the input range, leading to unintuitive editing, implausible deformations, or even artifacts. Above cited approaches differ from SPLOCS in the deformation representation (linear blend skinning vs. vertex displacements). They prescribe a fixed sparsity (e.g. exactly 4 joint influences per vertex) while the proposed method finds the suitable sparsity for every single vertex. However, the sparse decomposition method can benefit from skinning decomposition methods as a preprocessing step in order to subtract approximate articulations. More advanced deformations, such as detailed folds or smaller pose changes, can then be explained and modeled better using the new sparse decomposition method.

**Facial Animation and Editing.** Blendshapes are widely used for facial animation and are another highly important animation parameterization. Customizing blendshapes to a target face is a labor-intensive process. Adopting a given facial blendshape template model to a set of examples was demonstrated by Li et al. [LWP10]. To enable more local modifications, Tena et al. [TDM11] cluster the human face into regions and learn a clustered PCA model from marker-based motion capture data. Such direct and local manipulation is also possible with SPLOCS. Additionally, direct manipulation [LA10; SILN11] is possible without the explicit blending step required by Tena et al. [TDM11]. An elaborate facial editing system with many interesting user interaction concepts (that are compatible to sparse localized deformation components) was published by Lau et al. [LCXS09]. It relies on a facial prior based on a huge expression database. In contrast, SPLOCS automatically correspond to an intuitive-to-control parameterization of arbitrary mesh sequences, without being limited to faces.

**Simulating Deformations.** Physical simulation is often used to parametrize complex deformations based on material parameters and collisions. Simulation can also be constrained to lie in an artist-designed subspace [STC+12]. Fitting physical simulation models to captured mesh data is a very challenging task [SNF05; SGA+10; MBT+12]. Estimating the model parameters is often complex, requiring a suitable material model. Learned models are entirely object-type specific and do not expose the learned lower-dimensional space, which SPLOCS is all about. Pure data-driven simulation methods such as [GRH+12; KGBS11; ASTH09] circumvent the need for physical simulation (almost) completely by learning from real-world data. As recently shown by Kim et al. [KKN+13], successfully learning complex deformations from data is possible but requires a dauntingly huge amount of

well-structured training data, which is particularly rare for captured data. A decomposition method that is able to build a latent deformation space that generalizes beyond the training data and allows for user input, as offered by sparse localized deformation components, can help to overcome this limitation.

**Mesh Deformation.** Apart from parameterization based deformation techniques, direct mesh animation methods have emerged with ideas similar to static mesh deformation and with the goal to overcome some of the limitations of parametric models [SSP07; KG08; FB11]. These are powerful methods for editing but solve a different problem (direct manipulation of animations). In contrast, the present chapter studies the exploration of an underlying lower-dimensional deformation space (that can then be used for manipulation but also for other tasks).

**Low Dimensional Deformation Spaces.** A more general problem consists of the discovery and navigation of a latent space that is able to explain deformations given by (captured) data. The motivation comes from recent success in capturing performances directly from the real world, performances of human body movements, [TAS+10] and Chapter 2, or of detailed facial expressions [BHB+11]. Captured performances show great surface detail such as dimples, skin folds and cloth wrinkles. Editing captured geometry at each surface point or at each frame in a sequence is highly impractical. Graphics researchers proposed editing methods that transfer ideas from machine learning to identify a lower-dimensional space of deformations based on linear (PCA) or non-linear dimensionality reduction techniques [LWH+12; CH12; TR12]. Feng et al. [FKY08] deform a mesh and reproduce fine-scale details from given input data and control points by using kernel canonical correlational analysis [MRB03] on top of low-dimensional skeletal deformation. Some methods resort to Independent Component Analysis (ICA) [HKO01], for example for motion editing [CSFP07]. Shape and pose variations of human bodies can also be learned and modified using linear regression models [ASK+05; HSS+09], e.g. by modifying height, weight, gender; a principle that I extended with non-linear modeling in Chapter 3. Multi-linear models can be used to represent variation in human facial expressions and identity [VBPP05]. A key problem with these approaches is that the extracted components show global correlation of deformation, whereas artists often require specific deformation control, e.g. of specific muscle groups which are directly discovered by the sparse localized deformation decomposition. Meyer and Anderson [MA07] use the Varimax rotation [Har76, Chapter 13] of the PCA components as a basis that is localized, not directly for editing but for boosting computation speed. Experimentally, I found that Varimax yields unsatisfactory components that show global artifacts in the presence of noise and limited (captured) data, Section 4.3.3.

**Sparse Decompositions.** A major shortcoming of PCA is its tendency to produce components which involve all original variables - in the context of mesh deformations, this means every vertex deforms in each component. This is not suitable for animation decomposition. Several alternatives have been explored over the recent decades, such as non-negative matrix factorization [LS99] which finds positive components and Sparse PCA [CJ01; ZHT06; JIU03], which introduces a sparsity-inducing norm such as  $\ell_1$ , often dropping the orthogonality constraint. The fact that such decomposition methods happen to retrieve a localized set of variables, such as face or brain regions in (fMRI) images [VGP+11], made them popular in computer vision, signal processing and medical imaging. But they have not yet been explored in an animation processing context. Jenatton [Jen11] mentions two different formulations of sparse PCA: deflation methods [Mac09] consider a single component at each iteration step that estimates a subspace one dimension at a time, a strategy I use for *initialization*. The second formulation, matrix factorization methods [MBPS09], consist of a non-convex formulation to simultaneously optimize for all the principal components, an idea I adopt for *global optimization* of deformation components. Sparse decomposition methods are also related to independent component analysis (ICA) [HKO01]; it is known that both methods solve the same problem in a limited scenario - when there is no noise and the input and output dimensionality are the same - see Olshausen and Field [OF97]. In the real world, these conditions are not satisfied. The new sparse decomposition method outperforms the previously used ICA decomposition in terms of localized control.

Thanks to the success of compressed sensing and sparse linear models, minimization of sparsity-inducing norms can be achieved efficiently [BPC+11; BT09]. An excellent introduction and a comprehensive overview are given by Bach et al. [BJMO11] and Parikh and Boyd [PB14]. In computer graphics, such methods can help to find correspondences between meshes [PBB+12], can improve the robustness against outliers within the classical Iterative Closest Point (ICP) algorithm that is used in rigid registration [BTP13], or even allow the modeling of 3D astronomical nebulae from a single image [WLM13]. Deng et al. [DBD+13] showed that sparsity-inducing norms are very suitable to obtain *local* modifications of 3D surfaces, but they only showed this capability on constrained static meshes which are popular in architecture. While Deng et al. [DBD+13] achieved local editing of static meshes, the approach presented here provides local editing based on a latent deformation space that is automatically extracted from any captured mesh sequence.

## 4.2. Method

My method aims to decompose captured or animated mesh sequences into sparse, localized, and intuitive-to-control deformation components. The input data to the algorithm consists of a mesh

sequence with  $F$  frames. Let  $f \in 1, \dots, F$  denote the index of the current frame. Each frame consists of  $N$  vertex positions  $\mathbf{v}_i^{(f)}$ ,  $i \in 1, \dots, N$ . The mesh topology is equal for all frames, and the vertices are in temporal correspondence. All vertices are assembled in a single animation matrix  $\mathbf{X} \in \mathbb{R}^{F \times 3N}$ . Each row of this matrix contains all vertex positions in that  $f$ :

$$\mathbf{X} = \begin{bmatrix} (\mathbf{v}_1^{(1)})^\top & (\mathbf{v}_2^{(1)})^\top & \dots & (\mathbf{v}_N^{(1)})^\top \\ (\mathbf{v}_1^{(2)})^\top & (\mathbf{v}_2^{(2)})^\top & \dots & (\mathbf{v}_N^{(2)})^\top \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{v}_1^{(F)})^\top & (\mathbf{v}_2^{(F)})^\top & \dots & (\mathbf{v}_N^{(F)})^\top \end{bmatrix} \quad (4.1)$$

For convenience, let us assume the vertex coordinates in the animation matrix are expressed as residual displacements to a “mean shape”  $\hat{\mathbf{x}}$  of the mesh (e.g. the first frame or the average of all frames).<sup>2</sup> Before assembling the animation matrix, rigid alignment must be performed in order to subtract global translation and rotation.

#### 4.2.1. Sparse PCA for mesh sequence decomposition

We are looking for an appropriate matrix factorization of  $\mathbf{X}$  into  $K \in \mathbb{Z}_+$  deformation components in matrix  $\mathbf{C} \in \mathbb{R}^{K \times 3N}$  with weights  $\mathbf{W} \in \mathbb{R}^{F \times K}$ , hence

$$\mathbf{X} = \mathbf{W}\mathbf{C} \quad (4.2)$$

The above model is comparable to the widely used blendshape parameterization technique [Osi03], where the weights  $\mathbf{W}$  are usually key-framed by an animator and the components  $\mathbf{C}$  (blendshapes minus mean shape) are prepared by a modeler.

The space of solutions to Eq. (4.2) has to be regularized by additional constraints on  $\mathbf{W}$  and  $\mathbf{C}$ . For example, PCA (Principal Component Analysis) constrains the components to be orthogonal, forcing  $\mathbf{C}^\top \mathbf{C} \stackrel{!}{=} \mathbf{I}$ . However, the principal components exert global influence on the whole mesh and are therefore unintuitive for artistically meaningful localized editing [LA10], Figure 4.4(a).

Sparse components can be discovered by imposing sparsity on the solution of Eq. (4.2). To this end, a sparsity-inducing norm such as the  $\ell_1$  norm is introduced as a regularizer  $\Omega(\mathbf{C})$  acting on

---

<sup>2</sup>Therefore, the algorithm expects that this rest shape was subtracted from the vertex coordinates in a pre-processing step, in the form  $\mathbf{X}_f \leftarrow \mathbf{X}_f - \hat{\mathbf{x}}$  and accordingly  $\mathbf{v}_i^{(f)} \leftarrow \mathbf{v}_i^{(f)} - \hat{\mathbf{v}}_i$ . It is also necessary to normalize  $\mathbf{X}$  by its standard variation (across frames and vertices) after subtracting the mean to make the algorithm invariant to the amount of deformation of different datasets. This arbitrary scaling can be reversed for editing or display.

the components. This yields *Sparse PCA* [ZHT06; Jen11]. Following their framework, the matrix factorization can be formulated as a joint regularized minimization problem,

$$\begin{aligned} & \underset{\mathbf{W}, \mathbf{C}}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{X} - \mathbf{WC}\|_{\text{F}}^2 + \Omega(\mathbf{C}) \quad , \\ & \text{subject to} \quad \mathbf{W}_{*,k} \in \mathcal{V}_q(\mathbf{W}_{*,k}) \text{ , } \forall k \in 1, \dots, K \text{ , } q \in 1, 2 \quad . \end{aligned} \quad (4.3)$$

The constraint set  $\mathcal{V}_q$  constrains the control weights of the decomposition, which is not only important for the optimization but also allows for specific animation-based priors. The first choice,

$$\mathcal{V}_1(\mathbf{w}) = \left\{ \mathbf{w} \in \mathbb{R}^F \mid \max(|\mathbf{w}|) = 1 \right\} \quad , \quad (4.4)$$

forces the maximum weight of component  $k$  across the whole animation (across the corresponding column  $\mathbf{W}_{*,k}$ ) to be either 1 or  $-1$ . This allows negative weights which work well for modeling body shape variations. For facial animations, where the weights should be constrained to match the well-established blendshape model, we can use the second constraint set

$$\mathcal{V}_2(\mathbf{w}) = \left\{ \mathbf{w} \in \mathbb{R}^F \mid \max(\mathbf{w}) = 1 \wedge \mathbf{w} \geq 0 \right\} \quad . \quad (4.5)$$

Furthermore, the constraints on the weights  $\mathbf{W}$  are essential for optimizing the objective in Eq. (4.3). They prevent the weights from getting too large. Optimization without weight constraints would drive the components completely to zero and the weights would increase towards  $\infty$ . This will become apparent when discussing the nature of the regularizer  $\Omega$ .

To find an appropriate regularizer for (captured) mesh sequences or animations, let us observe that the rows of  $\mathbf{C}$  are aligned akin to the 3D vertex position layout in Eq. (4.1). Triplets in the rows of  $\mathbf{C}$  thus form three-dimensional vectors, formally  $\mathbf{C}_{k,\mathcal{I}(i)} = [x, y, z]_k^{(i)}$  with  $\mathcal{I}(i) = \{3i, 3i+1, 3i+2\}$  (remember that  $i \in 1, \dots, N$  denote vertex indices). Every triplet  $\mathbf{C}_{k,\mathcal{I}(i)}$  thus corresponds to the  $x$ ,  $y$ , and  $z$  displacements of vertex  $i$  in component  $k$ . While regularizing  $\mathbf{C}$  with the element-wise  $\ell_1$  norm would induce sparsity, it would ignore this inherent group structure: it would treat all entries of the displacement vectors separately. The regularizer should only take into account whether a vertex  $i$  is displaced by component  $k$  at all, irrespective of the displacement direction. A good choice towards this goal is to regularize the (unsquared) *lengths* of the displacement vectors,

$$\Omega(\mathbf{C}) = \sum_{k=1}^K \sum_{i=1}^N \lambda_i^{(k)} \left\| \mathbf{C}_{k,\mathcal{I}(i)} \right\|_2 \quad , \quad (4.6)$$

with  $\lambda^{(k)} \in \mathbb{R}_+^N$ . This norm, called the  $\ell_1/\ell_2$  norm, induces a form of group sparsity [WNF09; BJMO11]. The spatially-varying regularization parameters  $\lambda^{(k)}$  (for each component  $\mathbf{C}_k$ ) are called support maps - an important innovation that is exploited to provide local support in the deformation components.

#### 4.2.2. Local Support

To derive a deformation basis where the displacements are spatially confined, I stipulate that each deformation component  $\mathbf{C}_k$ ,  $k \in 1, \dots, K$  should be centered around the set of vertices  $\mathbf{v}_j \in \mathcal{J}(k) \subset \{\mathbf{v}_1, \dots, \mathbf{v}_N\}$  that show the largest displacements for that component. Centered around these vertices, a fuzzy support region is defined as follows. For every component  $k$ , let  $\mathbf{d}^{(k)} \in \mathbb{R}^N$  be the geodesic distance of each vertex in the rest-pose mesh to the vertices of  $\mathcal{J}(k)$ . The range  $[d_{\min}, d_{\max}]$  is defined to allow the control of the size of the support regions. Geodesic distances are mapped from this range to  $[0, 1]$  (with clipping when out of this range), relating geodesic distances to the regularization strength as follows

$$\lambda_i^{(k)} = \lambda \cdot \begin{cases} 0 & \text{if } \mathbf{d}_i^{(k)} < d_{\min} \quad , \\ 1 & \text{if } \mathbf{d}_i^{(k)} > d_{\max} \quad , \\ \left( \frac{\mathbf{d}_i^{(k)} - d_{\min}}{d_{\max} - d_{\min}} \right) & \text{otherwise} \quad , \end{cases} \quad (4.7)$$

with  $\lambda \in \mathbb{R}_+$  being a user-defined tuning parameter. This simply maps the geodesic distance linearly to regularization strength. It therefore changes the regularization strength of each component locally for each vertex.

The support regions  $\lambda^{(k)}$ , and thus the regularization in Eq. (4.6), are iteratively updated during the optimization of Eq. (4.3) by re-computing distances  $\mathbf{d}^{(k)}$  at every iteration and for each component. The heat method presented by Crane et al. [CWW13] performs the geodesic distance computation very quickly on the (constant rest shape) mesh by simply solving two pre-factored sparse linear systems per component.

In practice, the set  $\mathcal{J}(k)$  is constructed greedily by picking the vertex showing maximal activations for each component  $k$ ,  $\mathcal{J}(k) = \{\mathbf{v}_j\}$ , with  $j = \arg \max_i \|\mathbf{C}_{k, \mathcal{I}(i)}\|_2$  and only a single point in  $\mathcal{J}(k)$ . Figure 4.7 demonstrates the effect of automatic local support. Alternatively, the set of vertices in the center of each support region  $\mathcal{J}(k)$  can also be obtained through user-input.

**User Constraints.** Experiments show that, in most cases, the automatically found components already correspond to intuitively meaningful dimensions. Sometimes, however, artists prefer a different deformation space to the one found automatically, or they choose to have stricter control on the

deformation region. SPLOCS therefore provides a simple user interface, that allows an artist to paint the center of a region for which a deformation component shall be found. This feature fits perfectly into the support region concept explained above, where the mask of vertices stroked by the artist can be conveniently fed into the optimization as the center vertices  $\mathcal{J}(k)$ . Due to diffusion by geodesic distances, this stroke mask does not need to be very precise.

### 4.2.3. Optimization Algorithm

Due to the non-convexity of Eq. (4.3) and the added objective of local support, the problem is difficult to optimize directly. However, when keeping either  $\mathbf{W}$  or  $\mathbf{C}$  constant the problem can be solved for the other variable. The whole optimization is thus performed with an iterative block coordinate descent method that alternates between the two optimization tasks [MBPS09]. This process requires initial values for the components and weights. An established practice is to initialize with PCA components, which works well for general Sparse PCA, but that strategy conflicts with the extension for local support and does not reach a good solution in practice. A novel initialization strategy is required.

**Initialization.** To iteratively find the initial components  $\mathbf{C}_k^{(0)}$  and corresponding weights  $\mathbf{W}_{*,k}^{(0)}$ , a greedy deflation algorithm is used. Beginning without any components, each step of the algorithm adds one component, iterating from  $k = 1$  to  $k = K$ . Before computing the first component, the residual  $\mathbf{R}^{(0)}$  is initialized to the animation matrix  $\mathbf{X}$ . The  $k$ th component is approximately optimized such that it maximally explains the variance in the residual from the previous step,  $\mathbf{R}^{(k-1)}$ . Then, this new component is subtracted from the data, leaving a residual  $\mathbf{R}^{(k)} \in \mathbb{R}^{F \times 3N}$ . Let us now formalize how the component  $\mathbf{C}_k^{(0)}$  at step  $k \in 1, \dots, N$  is estimated from the previous residual  $\mathbf{R}^{(k-1)}$ . First, a local support region has to be constructed. The algorithm selects the single vertex  $\mathcal{J}(k) = \{\mathbf{v}_j\}$  with the largest displacement in the previous residual  $\mathbf{R}^{(k-1)}$ .<sup>3</sup> Around this vertex, a support region  $\mathbf{\Lambda}_k^{(0)}$  is built according to Eq. (4.7). The next step is to approximate an initial component  $\mathbf{C}_k^{(0)}$  and the corresponding initial weights  $\mathbf{W}_{*,k}^{(0)}$ . Note that the vertices in the support region  $\mathbf{\Lambda}_k^{(0)}$  undergo motion across all frames of the animation, motion that is recorded in the residual  $\mathbf{R}^{(k-1)}$  and can be factorized into the product of our requested component  $\mathbf{C}_k^{(0)}$  and weight vector  $\mathbf{W}_{*,k}^{(0)}$ . In order to force the initial component to have only local influence, we want to perform such a factorization only with respect to the motion inside the support region (according to the values in  $\mathbf{\Lambda}_k^{(0)}$ ). To formalize this, let  $\hat{\mathbf{A}}^{(k)} \in \mathbb{R}^{3N \times 3N}$  be a diagonal matrix version of the initial support map, broadcasted for

<sup>3</sup>Note that even though only single vertices are collected in  $\mathcal{J}(k)$  here, the set notation needs to be kept in order to conform to any arbitrary user input masks. In this case  $\mathcal{J}(k)$  may contain multiple vertices. The initialization procedure also works in this case.

every  $x, y, z$  tuple of each vertex, such that  $\hat{\Lambda}_{a,b}^{(k)} = \lambda_{(a \setminus 3)}^{(k)} \delta_{a,b}$  for  $a, b \in 1, \dots, 3N$  (with  $\setminus$  denoting integer division and  $\delta_{a,b}$  denoting the Kronecker delta). For initialization, the factorization objective for estimating the  $k$ th initial component and its weights are

$$(\mathbf{C}_k^{(0)})^\top, \mathbf{W}_{*,k}^{(0)} = \arg \min_{\mathbf{c} \in \mathbb{R}^{3N}, \mathbf{w} \in \mathbb{R}^F} \frac{1}{2} \left\| \mathbf{R}^{(k-1)} (1 - \hat{\Lambda}^{(k)}) - \mathbf{w} \mathbf{c}^\top \right\|_F^2. \quad (4.8)$$

In essence, this heuristic looks for a rank-1 approximation of the previous residual  $\mathbf{R}^{(k-1)}$ , weighted by the current support map  $\lambda^{(k)}$ . The closed-form solution goes back to the celebrated theorem by Eckart and Young [EY36], but it requires an SVD of an  $F \times 3N$  matrix, which is still quite expensive to compute for large meshes. Therefore, we solve it only for the vertices in  $\mathcal{J}(k)$  to obtain the initial weights  $\mathbf{W}_{*,k}^{(0)}$  across the whole animation (recall that the weights on the vertices  $\mathcal{J}(k)$  are highest in the support map, making this a suitable approximation). Solving Eq. (4.8) only with respect to vertices  $\mathcal{J}(k)$  gives us the component at those vertices only. Yet, the factorization provides initial weights  $\mathbf{W}_{*,k}^{(0)}$ , which allow solving Eq. (4.8) with the weight variable  $\mathbf{w}$  fixed to these initial weights  $\mathbf{W}_{*,k}^{(0)}$ . This leads to a closed-form solution of Eq. (4.8):

$$\mathbf{C}_k^{(0)} = \left( \frac{(\mathbf{W}_{*,k}^{(0)})^\top \mathbf{R}^{(k-1)} (1 - \hat{\Lambda}^{(k)})}{\left\| \mathbf{W}_{*,k}^{(0)} \right\|_2^2} \right)^\top. \quad (4.9)$$

The initialization process adds one component at a time until it reaches the required number of  $K$  components (that is preset by the user). Clearly, such a heuristic solution is not optimal with respect to Eq. (4.3), but it gives a good initialization that is already spatially localized and was found to work well in practice.

**Optimization of weights.** Given the components  $\mathbf{C}$ , optimization of Eq. (4.3) with respect to the weights  $\mathbf{W}$  is a constrained linear least squares optimization problem. It is separable, as the constraints act on the weight vector  $\mathbf{W}_{*,k}$  of each component separately. One can thus use the block-coordinate descent algorithm, [Ber99, Section 2.7], which optimizes weights for each component  $k$  successively, similar to [MBPS09]. In practice, the weights from the previous iteration  $\mathbf{W}^{(t-1)}$  can be re-used (“warm restart”). The objective Eq. (4.3) can be written with respect to one component  $k$ , allowing minimization just for the weights of that  $k$ th component and leaving all other weights constant:

$$\mathbf{W}_{*,k}^{(t)} \leftarrow \arg \min_{\mathbf{w} \in \mathcal{V}_q} \frac{1}{2} \left\| \mathbf{X} - \left( \sum_{l=1, l \neq k}^K \mathbf{W}_{*,l}^{(t-1)} \mathbf{C}_l \right) - \mathbf{w} \mathbf{C}_k \right\|_F^2. \quad (4.10)$$

The closed-form solution takes the form

$$\mathbf{W}_{*,k}^{(t)} \leftarrow \text{Proj}_{\mathcal{V}_q} \left( \frac{\left( \mathbf{X} - \mathbf{W}^{(t-1)} \mathbf{C} + \mathbf{W}_{*,k}^{(t-1)} \mathbf{C}_k \right) (\mathbf{C}_k)^\top}{\|\mathbf{C}_k\|_2^2} \right) . \quad (4.11)$$

The operation  $\text{Proj}_{\mathcal{V}_q}(\mathbf{w}) = \arg \min_{\mathbf{u} \in \mathcal{V}_q} \|\mathbf{w} - \mathbf{u}\|_2^2$  performs a projection of the weights onto the constraint set  $\mathcal{V}_q$ , depending on the chosen  $q$ . Projection onto the constraints  $\mathcal{V}_1$  and  $\mathcal{V}_2$  has an analytic solution. The constraint set  $\mathcal{V}_1$ , Eq. (4.4), corresponds to the  $\ell_\infty$  norm unit sphere<sup>4</sup>; the projection is evaluated via

$$\left( \text{Proj}_{\mathcal{V}_1}(\mathbf{w}) \right)_f = \begin{cases} \text{sgn}(\mathbf{w}_f) & \text{if } |\mathbf{w}_f| > 1 \vee f = \arg \max_{g \in 1, \dots, F} |\mathbf{w}_g| \\ \mathbf{w}_f & \text{otherwise} \end{cases} , \quad (4.12)$$

The blendshape-like constraint set  $\mathcal{V}_2$ , Eq. (4.5), corresponds to the intersection of the positive orthant and the  $\ell_\infty$  norm unit sphere, so the projection operator becomes

$$\text{Proj}_{\mathcal{V}_2}(\mathbf{w}) = \text{Proj}_{\mathcal{V}_1}(\max(0, \mathbf{w})) . \quad (4.13)$$

It is sufficient to perform a single iteration (for each  $k$ , so  $t \in 1, \dots, K$ ) of weight optimization before optimizing the sparse components in the next step.

Some mesh datasets are densely sampled and feature tens of thousands of vertices, resulting in a considerable amount of computation time to compute Eq. (4.11). It is possible to optimize this step by updating a running residual  $\mathbf{R}^{(t)} = \mathbf{X} - \mathbf{W}^{(t)} \mathbf{C}$  at each weight update iteration. Nevertheless, the involved matrix multiplications still involve notably big matrices. To accelerate weight optimization, weight updates can be performed on a random subset of vertices only, in a similar fashion as stochastic gradient descent. Since the components likely involve many vertices for big meshes, the weights tend to be over-determined - even by a subset of vertices.

**Optimization of sparse components.** The optimization of  $\mathbf{C}$  given fixed  $\mathbf{W}$  can be tackled using convex optimization. Many algorithms exist that are able to optimize objectives regularized by the  $\ell_1/\ell_2$  norm. In the setting considered here, the Alternating Direction Method of Multipliers (ADMM) [BPC+11] was observed to quickly converge<sup>5</sup>. The basics of ADMM are covered extensively in [BPC+11]. In the following, I will provide the details necessary to apply ADMM to the optimization

<sup>4</sup>Notice that the sparse optimization literature, e.g. [PB14, Section 6.5], almost exclusively mentions projections onto  $\ell_p$ -balls and not  $\ell_p$ -unit spheres as required here. That is why the projection operators given here are slightly different.

<sup>5</sup>Results of experiments with the alternative FISTA method [BT09] are reported in Section 4.3

of sparse components. First, Eq. (4.3) needs to be modified by introducing an auxiliary variable  $\mathbf{Z} \in \mathbb{R}^{K \times 3N}$ . With its help, the optimization objective, Eq. (4.3) with fixed  $\mathbf{W}$  is rewritten in a form that decouples the regularizer from the data term:

$$\begin{aligned} & \underset{\mathbf{C}, \mathbf{Z}}{\text{minimize}} \quad \frac{1}{2} \|\mathbf{X} - \mathbf{WC}\|_{\text{F}}^2 + \Omega(\mathbf{Z}) \\ & \text{subject to } \mathbf{C} - \mathbf{Z} = \mathbf{0} \quad . \end{aligned} \quad (4.14)$$

To solve this substitute minimization problem, ADMM initializes a Lagrange multiplier  $\mathbf{U} \in \mathbb{R}^{K \times 3N}$  to zero<sup>6</sup> and then iterates the following steps

$$\mathbf{C} \leftarrow \arg \min_{\mathbf{C}} \frac{1}{2} \|\mathbf{X} - \mathbf{WC}\|_{\text{F}}^2 + \frac{\rho}{2} \|\mathbf{C} - \mathbf{Z} + \mathbf{U}\|_{\text{F}}^2 \quad , \quad (4.15)$$

$$\mathbf{Z} \leftarrow \arg \min_{\mathbf{Z}} \left( \Omega(\mathbf{Z}) + \frac{\rho}{2} \|\mathbf{C} - \mathbf{Z} + \mathbf{U}\|_{\text{F}}^2 \right) \quad , \quad (4.16)$$

$$\mathbf{U} \leftarrow \mathbf{U} + \mathbf{C} - \mathbf{Z} \quad . \quad (4.17)$$

The first step, updating  $\mathbf{C}$  in Eq. (4.15), is a linear least squares problem that can be solved analytically via

$$\mathbf{C} \leftarrow (\mathbf{W}^{\top} \mathbf{W} + \rho \mathbf{I})^{-1} (\mathbf{W}^{\top} \mathbf{X} + \rho (\mathbf{Z} - \mathbf{U})) \quad . \quad (4.18)$$

As long as the penalty parameter  $\rho$  is constant,  $(\mathbf{W}^{\top} \mathbf{W} + \rho \mathbf{I})$  can be prefactorized using Cholesky factorization to quickly re-solve Eq. (4.18) at each iteration step of ADMM.

The second step, updating the dual variable  $\mathbf{Z}$  in Eq. (4.16), corresponds to the  $\ell_1/\ell_2$ -norm's proximal operator. A closed-form solution is [BJMO11, Chapter 3.3]

$$\mathbf{Z}_{k, \mathcal{I}(i)} \leftarrow \text{Prox}_{\frac{1}{\rho} \Omega}(\mathbf{p}^{(k, i)}) = \max \left( 0, 1 - \frac{\lambda_i^{(k)}}{\rho \|\mathbf{p}^{(k, i)}\|_2} \right) \mathbf{p}^{(k, i)} \quad , \quad (4.19)$$

with  $(\mathbf{p}^{(k, i)} = \mathbf{Z}_{k, \mathcal{I}(i)} + \mathbf{U}_{k, \mathcal{I}(i)}) \in \mathbb{R}^3$ . A fixed penalty parameter of  $\rho = 10$  works well for all datasets considered<sup>7</sup>. Running 10 iterations of ADMM suffices to optimize the components for each iteration step of the decomposition algorithm. The Lagrange multipliers  $\mathbf{U}$  are stored during the whole procedure.

---

<sup>6</sup>The rationale behind this is well explained in [BPC+11]. Essentially, ADMM solves the decoupled substitute minimization problem using dual ascent on the augmented Lagrangian. This represents an immense benefit: Since the two terms in the optimization objective completely decouple (here, data term and regularizer), each iteration involves optimization of just one of the terms (plus a quadratic penalty), often solvable in closed-form.

<sup>7</sup>Strategies to adjust  $\rho$  are discussed in [BPC+11] and are, for example, used in Chapter 5

In summary, the whole optimization procedure interleaves the following steps: re-computation of support maps based on geodesic distances, optimization of weights using block-coordinate descent and optimization of components using ADMM.

**Convergence.** Finding a stopping criterion to detect convergence of the objective function Eq. (4.3) is tricky due to its non-convexity and the added spatially varying regularization. A commonly used stopping criterion is when the change of the objective function falls below a threshold. But the SPLOCS objective function cannot be guaranteed to decrease for each iteration step because the support maps  $\lambda^{(k)}$  are changing during optimization. Therefore, the proposed algorithm monitors the average change of the objective function across the last 5 iterations. If this averaged change falls below a threshold  $\epsilon$ , the algorithm stops.

**Tunable Parameters.** In summary, the sparse localized deformation decomposition method takes the following user-specified parameters: the number  $K$  of desired deformation components, the minimal and maximal geodesic distance for the support maps  $d_{\min}$  and  $d_{\max}$ , as well as  $\lambda$  in Eq. (4.7) that sets the importance of the local support term. Moreover, users have to choose whether the weights are allowed to be negative (use  $\mathcal{V}_1$ , Eq. (4.4)) or not ( $\mathcal{V}_2$ , Eq. (4.5)) and if they want to use a specific frame (e.g. the first one) or the average of all frames as the rest shape.

### 4.3. Results

The versatility of the method is demonstrated using a variety of captured data sets, with multiple potential applications being explored: intuitive editing and shape analysis of faces (Section 4.3.3), full-body models (Section 4.3.6), muscle motion (Section 4.3.4) and cloth motion (Section 4.3.5). Table 4.1 lists the processed datasets, method parameters, computation times and reconstruction accuracy. The supplementary demonstration video illustrates many of these results in motion. Extensive cross validation experiments were performed in order to assess how well the decomposition method generalizes for data that was not used during training. In addition to those results, which were published in [NVW+13], this section investigates the convergence behavior of the method and yields insights into the choice of the  $\ell_1/\ell_2$  optimizer.

#### 4.3.1. Evaluation of Optimization Algorithm

Recall that updating the sparse components  $\mathbf{C}$  involves optimizing a sum of a smooth, continuously differentiable data term plus the non-smooth convex regularizer  $\Omega$  based on the  $\ell_1/\ell_2$  norm. ADMM is

#### 4. Sparse Localized Deformation Components

| Dataset              |        |     | Algorithm Parameters |                    |                        |           |            |            |       | Results |                  |  |
|----------------------|--------|-----|----------------------|--------------------|------------------------|-----------|------------|------------|-------|---------|------------------|--|
| Source               | $N$    | $F$ | $K$                  | $\hat{\mathbf{x}}$ | $q$ of $\mathcal{V}_q$ | $\lambda$ | $d_{\min}$ | $d_{\max}$ | iters | time    | $E_{\text{RMS}}$ |  |
| Face [BHB+11]        | 40 000 | 322 | 80                   | first              | 2                      | 2         | 0.1        | 0.6        | 49    | 6m57s   | 0.39             |  |
| Face [ZSCS04]        | 23 725 | 384 | 50                   | first              | 2                      | 1         | 0.1        | 0.7        | 82    | 3m58s   | 0.76             |  |
| Face [VWB+12]        | 44 153 | 566 | 50                   | first              | 2                      | 2         | 0.1        | 0.7        | 113   | 14m09s  | 0.67             |  |
| Body-scans [HSS+09]  | 6 449  | 111 | 100                  | avg.               | 1                      | 1         | 0.1        | 1.0        | 34    | 33s     | 0.76             |  |
| Cloth folds [WVL+11] | 10 684 | 459 | 200                  | avg.               | 1                      | 25        | 0.01       | 0.3        | 18    | 4m39s   | 0.29             |  |
| Muscles, Chapter 3   | 3 467  | 242 | 80                   | avg.               | 1                      | 5         | 0.1        | 0.4        | 77    | 1m36s   | 1.08             |  |

Table 4.1.: Overview of processed datasets. From left to right, columns show source of data set, number of vertices  $N$ , number of frames  $F$ . The columns grouped under “Algorithm Parameters” show: number of components  $K$ , reference frame used as rest shape (first frame or average), type of constraint for weights (allow non-negativity with  $q = 1$  or not with  $q = 2$ ),  $\lambda$  to control sparsity regularization,  $d_{\min}$  and  $d_{\max}$  to control size of support regions. Data below “Results” are the number of outer iterations for global optimization until convergence, computation time in minutes, and reconstruction error  $E_{\text{RMS}}$  using the measure from [KSO10].

just one suitable method to tackle this problem. There are other algorithms for optimization involving the  $\ell_1/\ell_2$  term, including the popular FISTA (Fast Iterative Shrinkage-Thresholding Algorithm). Using FISTA for computing SPLOCS leads to different runtime and convergence characteristics that are worth further investigation. Per iteration, both FISTA and ADMM solve the proximal mapping of the  $\ell_1/\ell_2$  norm, Eq. (4.19); FISTA additionally computes the gradient at each iteration (a simple matrix multiplication in our case) while ADMM inverts a linear system, Eq. (4.18). As preprocessing step, FISTA requires estimating the Lipschitz constant of the gradient of the data term and ADMM needs a pre-factorization of the linear system to be solved at each iteration step using Cholesky decomposition. For this reason, ADMM iterations are about 33 % slower than one FISTA iteration step (within the SPLOCS framework). However, the convergence behavior of ADMM is significantly better: Figure 4.1 compares the convergence behavior of FISTA [BT09] and ADMM, using the same number of 10 interleaved component optimization steps between updating support maps and weights. The same figure additionally plots convergence behavior when using the proposed accelerated (stochastic) weight updates, where only 1% of vertices were randomly chosen at each iteration step for the weight update in Eq. (4.11). The stochastic weight update strategy only works well for large meshes, Figure 4.1(b), where the accelerated updates (green line) achieve almost the same convergence as the full update (blue line).

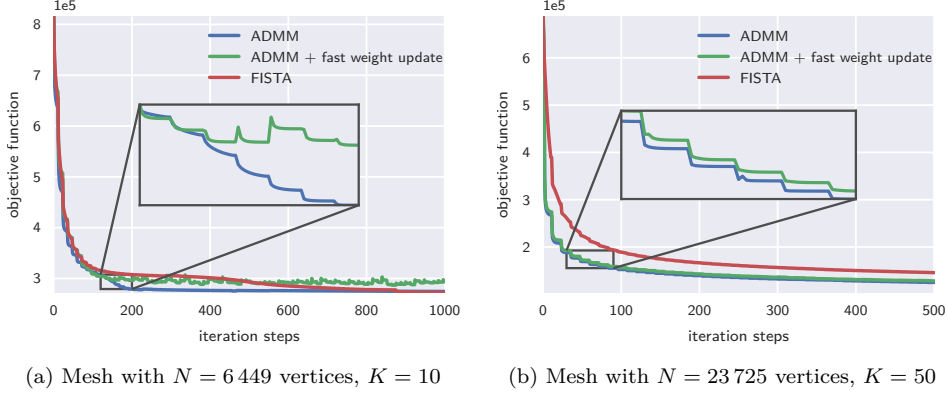


Figure 4.1.: Convergence: value of objective function, Eq. (4.3), vs. iteration steps. Different  $\ell_1/\ell_2$  minimizers (ADMM and FISTA) for optimizing the components are compared. The minimization of the components is interleaved with the optimization of the weights using block-coordinate descent every 10 iterations, which results in the step-like convergence curve - well visible in the zoomed-in insets. The plots also compare with *fast weight update* option of the proposed method (which uses only a random selection of 1% of the vertices for block-coordinate descent). (a) shows the convergence curve for a dataset with a low number of vertices (from [HSS+09]) and low number of SPLOCS to compute. Fast weight update can show stochastic behavior in this case, rendering it unsuitable for this dataset. The graph clearly demonstrates that FISTA converges to the same local minimum as ADMM, only at a much slower convergence rate. (b) shows the convergence of SPLOCS on a high-resolution mesh from [ZSCS04]. Using fast weight update provides a boost in computation speed for such datasets, while the convergence is almost the same. Notice that the graph shows both the inner iterations of ADMM as well as the outer iterations, thus the iteration axis reaches beyond the outer iteration values given in Table 4.1.

**Runtime.** Computation times of the algorithm are given in Table 4.1 and were measured on a desktop machine with Intel i7 CPU running at 3.40GHz and 16GB RAM. Runtime grows with increasing number of vertices  $N$ , number of frames in the sequence  $F$ , and number of components  $K$ . Notice that the Python implementation of SPLOCS is neither optimized nor explicitly parallelized.

#### 4.3.2. Quantitative Evaluation

Since sparse localized deformation decomposition can be viewed as an unsupervised dimensionality reduction technique, it is interesting to assess the generalizability of the extracted dimensions to unseen data. To this end, an evaluation was performed on a dataset with 111 full-body scans of people approximately standing in the same pose, obtained from Hasler et al. [HSS+09]. The scans were randomly split into training (55 scans) and test set (56 scans). Clustered PCA [TDM11], PCA, and SPLOCS (without and with automatic locality) were used to decompose the training datasets,

#### 4. Sparse Localized Deformation Components

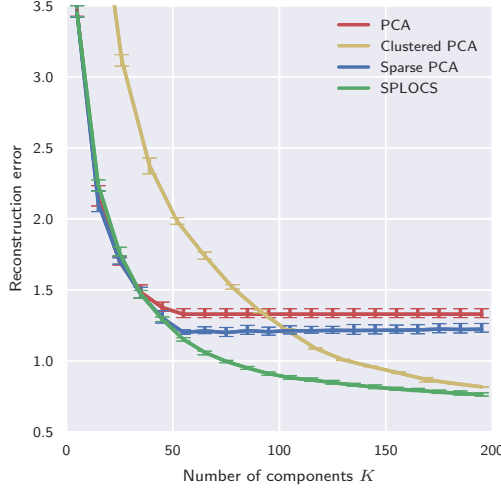


Figure 4.2.: Generalization to unseen data: reconstruction error (y-axis) with respect to the number of components used (x-axis) for a set of body scans from [HSS+09] that were not part of the training data. SPLOCS outperforms other related approaches in generalizing to unseen data from small training sets. The error bars show the maximal/minimal reconstruction error across 3 random splits of the data set.

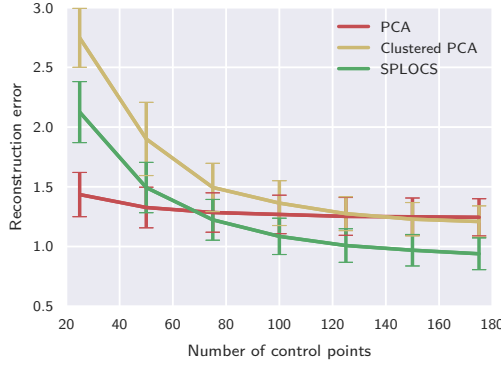


Figure 4.3.: Reconstruction error with respect to sparse control points: plot of the reconstruction error (y-axis) of different approaches when used for regularizing a sparse number of control points (x-axis) for test poses which are not part of a small training dataset [ZSCS04]. Error bars show the standard deviation across 50 trials.

with varying number of components  $K$ . To reconstruct the test set Eq. (4.2) is inverted which allows measuring the difference in vertex coordinates with the error metric introduced in [KSO10, Section 5]. Figure 4.2 reports the reconstruction error over 3 different random splits of the data set. For Clustered PCA, the number of regions was set to 13, and  $K$  was the total number of all components and all regions, not the number of components per region as in [TDM11]. SPLOCS was also compared to non-localized classical Sparse PCA in this experiment by performing SPLOCS without the proposed locality term (constant  $\lambda_i^{(k)} = 0.5$ ). The results indicate that the proposed method *with* localization generalizes better to new body shapes outside the training set than PCA or even plain Sparse PCA.

In order to assess SPLOCS’ capacity to provide editing beyond limited input data, an experiment was performed on the facial performance dataset provided by Zhang et al. [ZSCS04]. A fixed number of  $K = 50$  components was extracted using PCA, Clustered PCA, and SPLOCS. As input training

data, 19 randomly selected frames from the 365 frame dataset of Zhang et al. [ZSCS04] were used. For each of the remaining frames, sparse control points were randomly picked on the mesh. Then, weights  $\mathbf{W}$  were computed such that the reconstructed vertices best fit only those selected control points. The resulting complete mesh was compared to the ground truth, again using the error metric from [KSO10]. The experiment was repeated in 5 trials with a different training set. Each trial was further repeated 10 times using different random control points. Figure 4.3 gives the average reconstruction error in relation to number of control points. Due to the locality of SPLOCS components, the proposed method requires a certain number of control points until it outperforms PCA. For this dataset, this happens at around 70 control points. This corresponds approximately to the number of 50 components used here, so each component needs approximately one control point. With increasing number of control points, the superior accuracy of SPLOCS compared to PCA becomes even more apparent. SPLOCS also consistently outperforms Clustered PCA.

To understand the reason why SPLOCS generalizes so well to data it was not trained on, consider a facial geometry sequence in which both eyebrows are always raised together. PCA cannot reproduce a mesh showing only one raised eyebrow. In contrast, SPLOCS finds localized components for every eyebrow separately, and can thus reproduce their motion separately, even though such deformations were not seen in the training data. In essence, sparse localized deformation components provide a richer latent deformation space.

#### 4.3.3. Facial performances

The proposed method can be used to learn deformation components of captured facial animations that mimic the control properties of blendshapes, a widely used representation that animation artists use to craft facial animations and which are usually created in a tedious manual process. The automatically detected SPLOCS components are spatially localized and thus limit the influence of modifications to confined regions that correspond to individual, intuitively meaningful effects (e.g. twitching of an eyebrow or a nostril).

**Comparison to Important Related Work.** The above-mentioned spatially confined and intuitive edits are hard to perform with components from global decomposition methods such as PCA or ICA. Figure 4.4 illustrates this fact on facial performance data [ZSCS04]. Even without any facial prior or user input, sparse localized deformation components are spatially confined to local regions and already resemble artistically modeled blendshapes for controlling a facial animation [Osi03].

Figure 4.4(c) visualizes the PCA components after Varimax rotation, a technique used by Meyer and Anderson [MA07] for extraction of key points. Applied to captured data, the components

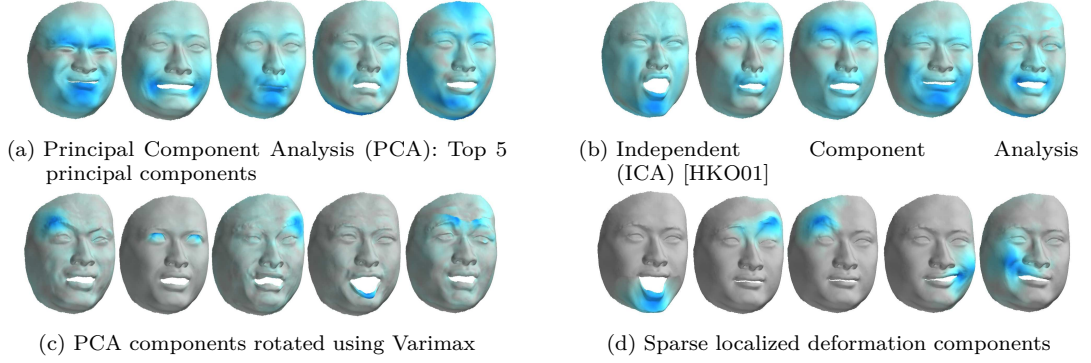


Figure 4.4.: Comparison of general decomposition methods on the captured face dataset of [ZSCS04]. Color coding shows the magnitude of vertex displacements inside the components, from grey (zero) to dark blue (maximum). (a)-(b) The deformation components of PCA and ICA act globally on each vertex in the mesh, prohibiting local modification. (c) Varimax shows certain locality, but this cannot be controlled and shows artifacts for captured data. (d) Sparse localized deformation components show sparse and local deformations on confined regions, which is important i.e. for artistic editing.

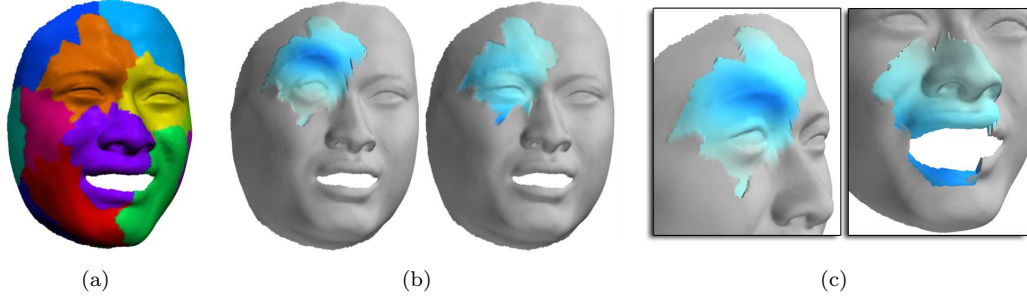


Figure 4.5.: Skinning Decomposition [KSO10] (a) The underlying segmentation. (b) First and second principal component in one of the regions. (c) Close up views clearly show artifacts arising when activating a single component. Those artifacts have to be removed with an explicit blending step that SPLOCS does not require.

show severe artifacts, rendering Varimax unusable for direct editing. Notice that Varimax and PCA components only differ in a rotation (in  $\mathbb{R}^K$ ), so Varimax performs exactly the same as PCA with respect to measured reconstruction error and generalization.

Figure 4.5 shows the drawbacks of Clustered PCA, where spectral clustering [TDM11] is used to provide a region segmentation. PCA components are confined to these fixed regions. Within these regions, components exhibit the same limitations as PCA components: they fail to produce confined and independent effects - all PCA components have to be activated to achieve a specific deformation effect. Additionally, blending at region boundaries is needed to remove artifacts [TDM11].

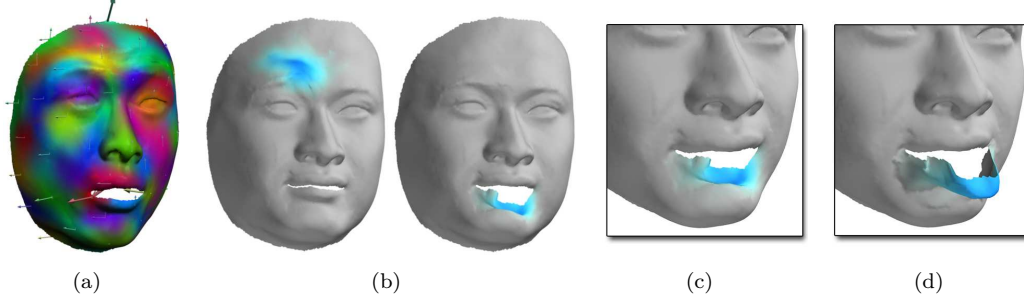


Figure 4.6.: Skinning decomposition [KSO10] (a) Vertex influences (blending weights) (b) Influence and motion of 2 bones in isolation (c) Allowing only one bone parameter to be active (setting all other bone transformations to zero) produces artifacts. (d) Arbitrary translation of one of the bones found on the lips causes artifacts since bones offer too many degrees of freedom.

SPLOCS was also compared to skinning decomposition methods that learn explicit bone transformations, the method of Kavan et al. [KSO10] being a representative example. Individual bone transformations can be interpreted as weighted transformation nodes (sometimes called cluster deformer), and allow modification of surface mesh data by translating and rotating a 3D handle. Two key problems arise which interfere with the target applications that SPLOCS is designed for, problems which are visualized in Figure 4.6. Firstly, the bone transformations only achieve reasonable deformation when modifying several bones in concert, preventing bone transformation handles from providing individual control. For example, eyebrow-wrinkles are produced by setting transformations of at least 2 specific bones in a peculiar, hardly intuitive way. Secondly, bone transformations themselves do not provide intuitive control parameters at all. For example, editing the translation of a bone quickly results in deformation artifacts when parts of the mesh are “pulled” outside of the mesh, Figure 4.6(d). Notice that translation and rotation invariance is inherent to the underlying skinning representation. The supplementary video specifically and clearly visualizes the deficits of these related decomposition methods.

**Automatic Local Support.** Figure 4.7 shows the benefit of the spatial locality term, Eq. (4.7), when decomposing the dataset provided by Valgaerts et al. [VWB+12]. This example illustrates how imposing local support regions helps SPLOCS to separate motion of distant regions which are co-activated in the original actor’s performance. One could say SPLOCS helps to model regions influenced by individual muscle groups separately, such as the left eyebrow motion which is distinct from the right eyebrow’s motion.

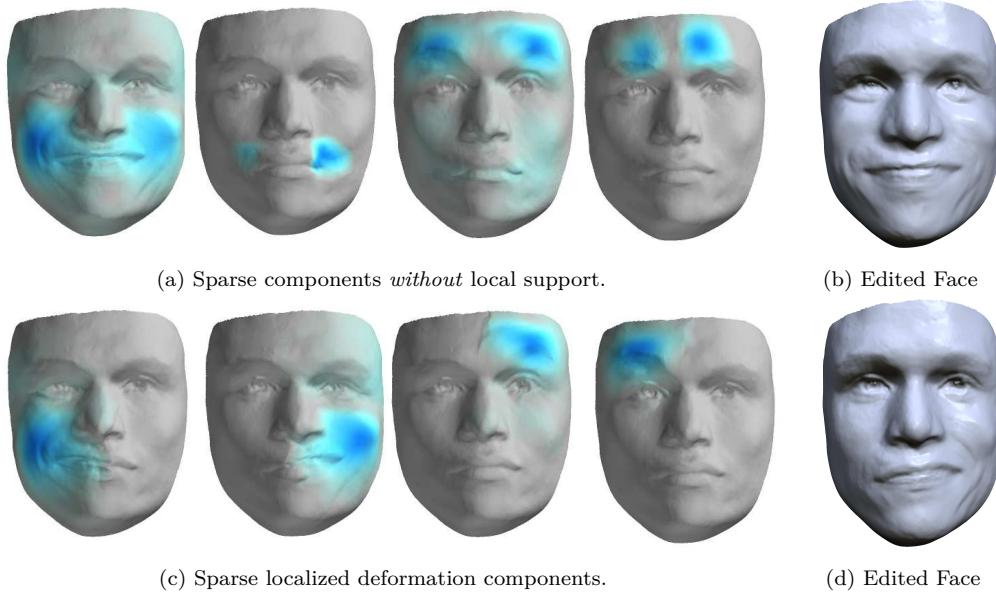


Figure 4.7.: Effect of automatic local support demonstrated on a face dataset by Valgaerts et al. [VWB+12]. Without local support in (a), spatially distant regions are co-activated. In (c), with local support, left and right side brows and cheek are separated, even though these separate deformations were never captured in the input. This allows generating novel facial expressions, for example raising only the right cheek as in (d), whereas non-local components only allow global edits such as in (b).



Figure 4.8.: The proposed method allows completely new, exaggerated facial expressions far beyond the input range, here based on the dataset of Beeler et al. [BHB+11]. Control points used to obtain the edit are shown as small blue dots.

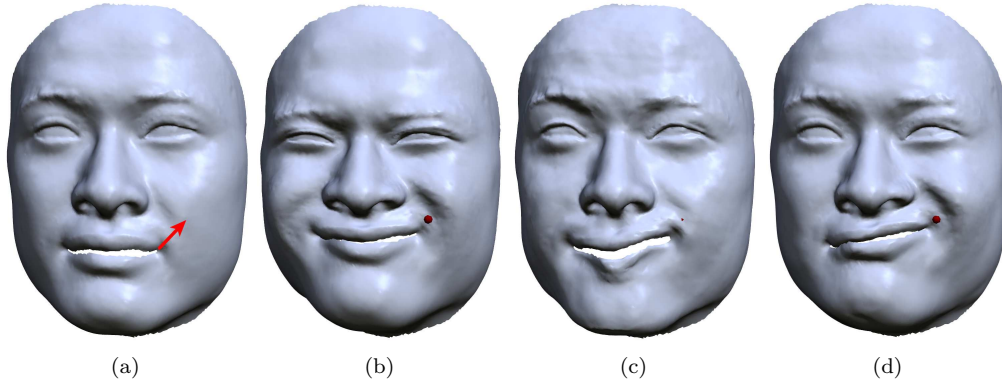


Figure 4.9.: Results of direct manipulation of the underlying basis: (a) rest pose mesh with a given user constraint (red arrow). (b) Result based on PCA basis, showing unintuitive global deformation - even though the constraint is on the right side of the mouth, the left side of the mouth as well as the eyebrows change. (c) result in Varimax basis shows artifacts. (d) Sparse localized deformation components provide artistically meaningful local edits.

**Artistic Control.** Figure 4.8 shows artistic edits using sparse localized components on a face animation captured by Beeler et al. [BHB+11] (that has been downsampled to 40k vertices). With the simple, blendshape-like decomposition scheme of sparse localized deformation components, it is straightforward to apply the direct manipulation method of Lewis and Anjyo [LA10]: positional edits on mesh vertices act in the linear deformation space of the sparse localized components. Therefore, edits of control points are confined within a plausible deformation space, whereas bone transformations would provide unspecific degrees of freedom as discussed above. Based on the direct manipulation scheme, captured facial animations can be conveniently edited beyond the captured motion using only a few input constraints for crisp direct manipulation. Moreover, effects can be reproduced plausibly even if they are missing from the input data, for example independent articulation of separate eye-brow movement. This editing paradigm has also been tested on the decomposed dataset of Zhang et al. [ZSCS04] for comparing to related decomposition approaches. Figure 4.9 shows the edits based on just a single control point, comparing PCA, Varimax, and SPLOCS components. It can be seen that the sparse components allow for localized edits constrained in a plausible space of face motion, a characteristic that artists need [Hav06]. In contrast, in case of PCA, single vertex edits have unintuitive global effects on the entire face.

**User interaction.** The option of user constraints allows for the introduction of semantic information into the very process of the decomposition. This is provided by a simple interface that allows an artist to draw a few strokes on the mesh surface, Figure 4.10(a) and Figure 4.10(c). The strokes

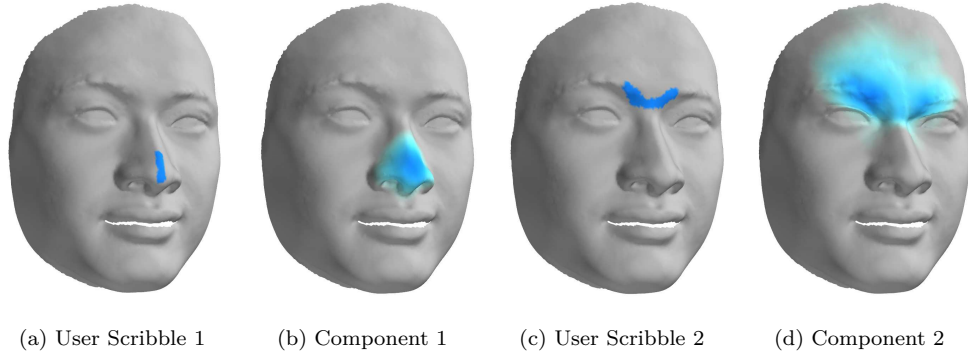


Figure 4.10.: Incorporating user constraints: A rough scribble in form of a binary mask shown in (a) and (c), is turned into a deformation component around the selected region, shown in (b) and (d).

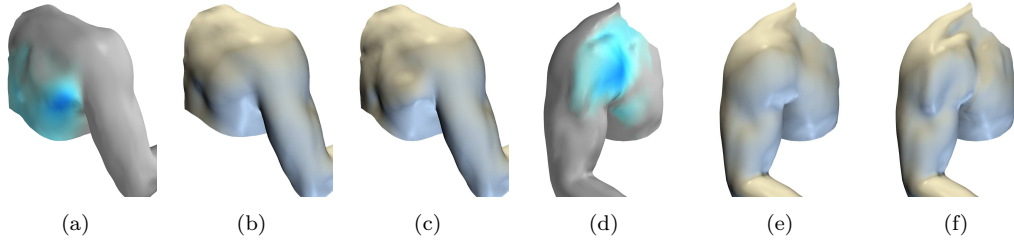


Figure 4.11.: Sparse localized deformation components automatically extracted from muscle deformation, corresponding to the clavicle and one strand of the deltoid muscle. (a) component area for the clavicle component, (b) mesh in rest pose and (c) exaggerated effect of the clavicle deformation component. (d) Component area discovered on the deltoid muscle, (e) mesh in rest pose and (f) effect when exaggerating the deltoid component, resulting in noticeable bulges of the muscle strand. Such confined muscle effects do not actually happen in reality, but are useful for precise editing and may be used for muscle activation analysis.

define a binary mask which marks the region of interest for one of the SPLOCS components. Not all components need to be user-constrained - if only some components have user-supplied scribbles, the support regions of the remaining components can be inferred automatically using the proposed optimization approach (while the user-constrained ones stay fixed).

#### 4.3.4. Muscle deformations

To showcase its use in biomechanics, the SPLOCS algorithm was applied to the captured moving arm geometry from Chapter 3 exhibiting muscle-induced surface deformation. A subset was selected from the full captured dataset, showing deformation of the arm and shoulder in a curl motion done once

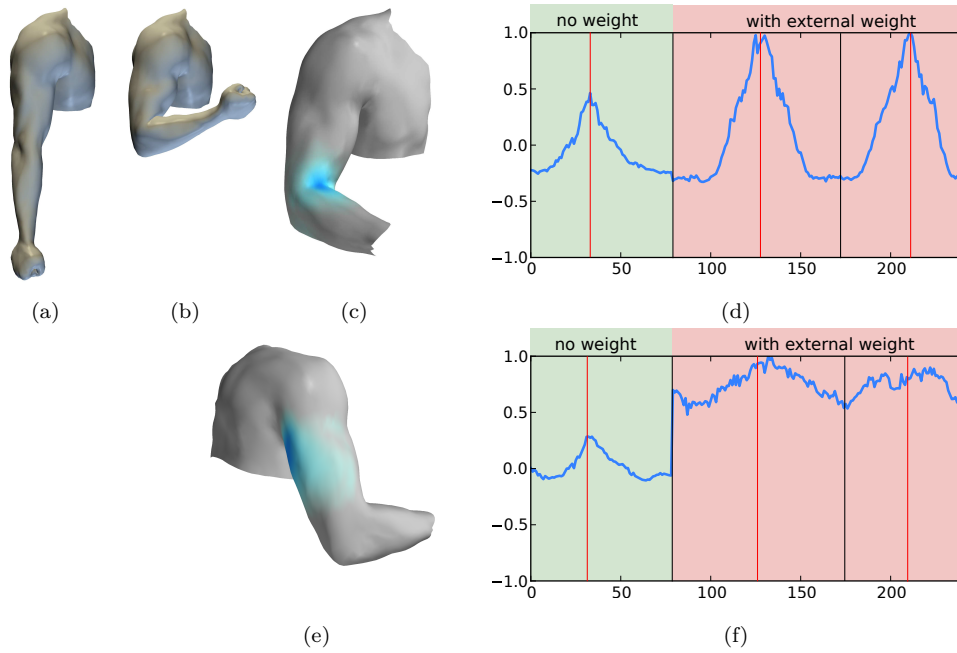


Figure 4.12.: Automatically found activations of components corresponding to specific muscles. The subject flexes the elbow from pose (a) to pose (b). SPLOCS happens to identify biomechanically meaningful deformation components: (c) Component corresponding to twitching of the tendon at the elbow joint. (d) Activation graph (weights across the sequence for the tendon component). (e) Triceps muscle component. (f) Activation graph: the black vertical lines correspond to pose (a), red vertical lines to (b). The subject performs one motion cycle without any weight in hand, and then two cycles holding an external load in the hand. This is clearly reflected in the higher activation of both components.

without a weight and twice with a 14 kg barbell in hand. These meshes have already been temporally aligned and registered to a simple 3-piece bone skeleton, Chapter 3. Prior to sparse component extraction, the skeleton was used to subtract the articulated upper- and lower-arm rotation by aligning all meshes to the template skeleton pose, leaving only the residual muscle deformations. Figure 4.11 visualizes the sparse localized deformation components, extracted without user constraints. The components visually correspond to local bulges that correspond to actual muscle groups like the deltoid, biceps, and triceps muscles. Some components show residual skeletal deformations that are not explained by the simple 3-piece skeleton, e.g. the complex motion of the clavicle underneath the skin.

Figure 4.12 shows activations of some muscle components extracted from the input sequence. One component represents the tendon deformation at the elbow joint that is twitching during this



Figure 4.13.: Left: 3 sparse localized deformation components on a performance capture sequence showing cloth folds, Right: effect of exaggerating one component.

sequence, Figure 4.12(c). The activation weights  $\mathbf{W}_{*,k}$  plotted in Figure 4.12(d) clearly reflect elbow twitches due to change in the pose of the arm. The triceps muscle, Figure 4.12(e), is summoned to counteract the external force on the hand, which is clearly depicted by the activation graph of the triceps component in Figure 4.12(f). In other words, SPLOCS is able to provide biomechanically informative muscle activation graphs without requiring the user to explicitly fit a physiological muscle template to the data. In the same way, surface noise in the data due to limitations of the acquisition method can be isolated: systematic noise will be decomposed into separate components. The user can look through all  $K$  components and select the artifact-inducing components for deletion, keeping a set of  $\mathbf{C}_{\mathcal{K}}$  artifact-free components and corresponding weights  $\mathbf{W}_{*,\mathcal{K}}$  (where  $\mathcal{K} \subset \{1, \dots, K\}$  are the indices of the components not marked for deletion). Projecting the data into the remaining subspace removes the artifacts (by computing  $\mathbf{W}_{*,\mathcal{K}}\mathbf{C}_{\mathcal{K}}$ ). Furthermore, animators can selectively emphasize the effect of individual muscle groups in an animation by key-framing the weights  $\mathbf{W}$  or, for example, by doubling the activation of certain muscle components. In this way, an animator can create a biologically inaccurate but artistically desirable appearance of an animation which is often needed in VFX productions, an effect that is demonstrated in the supplementary video.

#### 4.3.5. Cloth deformations

Sparse localized deformation components also enable visualizing cloth deformation patterns in captured mesh sequences, such as the detailed full-body motion data including cloth wrinkles captured by Wu et al. [WVL+11] from multiview video. It is possible to derive a set of features that describe high-frequency detail of finer-scale folds. We just need to separate the lower-frequency motion coming from the limbs and register all meshes onto a single template pose. The moving cloth folds then appear as floating residuals on the template mesh. Sparse decomposition identifies localized fold patterns on the cloth surface that can be visualized and controlled individually, Figure 4.13. This hints at

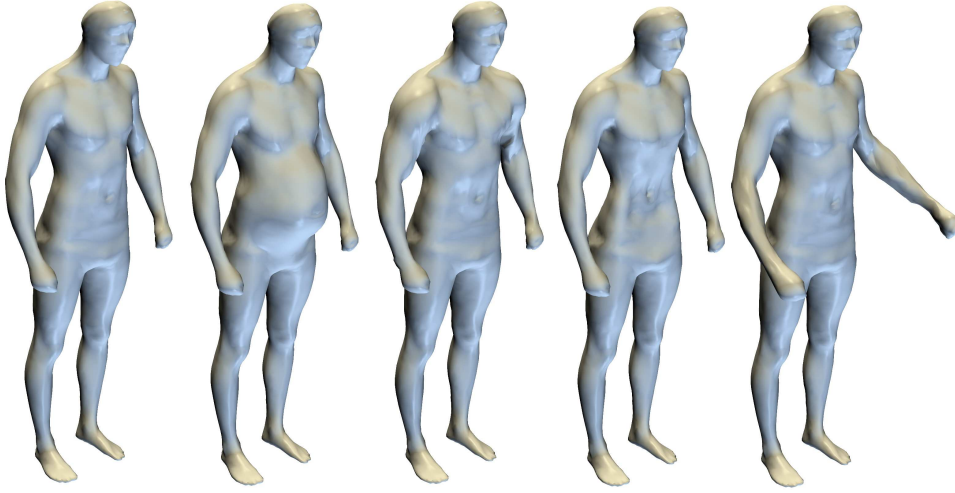


Figure 4.14.: Automatically extracted deformation components on a database of human body scans from [HSS+09], shown by fully activating a single component. From left to right: rest shape, belly size, shoulder muscles, waist girth, pose of one arm. Notice how the activation of a single component can already produce spatially localized and semantically meaningful effects. SPLOCS even isolates pose variations into separate components.

potential applications of sparse localized deformation components in data-driven cloth simulation and upsampling.

#### 4.3.6. Statistical shape modeling, processing, and visualisation

In statistical shape processing, researchers often face the problem of aligning large data sets of shape models or scans that show variants of a class of shapes. These shapes may exhibit certain local shape variations, but may also differ in proportions or pose. Certain statistical properties or correlations of the shapes shall be learned, while certain other effects shall be factored out. The proposed decomposition method can be of great help in such a setting. Let me exemplify this for the problem of building a static statistical shape model of human shape variations from a large corpus of laser scans of real human subjects, accomplished with PCA and bilinear models in previous work [ASK+05; HSS+09]. For instance, Hasler et al. [HSS+09] provide a data set of 111 humans of different shape and gender that were asked to stand in the same pose during a scan. From this set one may want to capture shape variations and neglect variations due to pose. Hasler and other researchers resorted to deformation-based scan alignment relative to a template to get dense correspondences of the surfaces. However, pose variations due to different joint angles/hand poses still persist after

correspondence estimation as people can never stand in exactly the same pose. Applying PCA to that set thus leads to components which intertwine pose and shape variations, dimensions that are supposed to be separated. Somewhat surprisingly, SPLOCS is able to automatically separate effects of pose and shape of specific body parts, Figure 4.14, because deformation related to pose and shape variations get grouped into different components. The deformation components may be used to meaningfully pose-normalize all scans by manually removing the pose-specific deformation components and projecting the scans into the remaining subspace. Such an operation would be difficult by any other means. At the same time, the automatically extracted components reveal intuitive localized dimensions of shape variation, even without the need to pose-normalize the scans *before* decomposition. These dimensions correspond to more meaningful deformations than those found in previous approaches, for instance, waist girth, belly size, or hip size. The dimensions can be individually explored and changed independently of global body pose. I foresee that this general idea could be used on scan sets with much stronger pose variation, enabling more robust shape-retrieval under pose variation and allowing for co-segmentation learning from shape sets.

#### 4.4. Discussion

Sparse Localized Deformation Components present a versatile decomposition method for space-time mesh sequence data that is applicable to many settings: mesh editing, control, scan alignment, construction of static and parametric shape models. The major advantage of this approach is that it is very general yet simple to implement. Quantitative experiments show that SPLOCS provide excellent generalizability compared to other methods like PCA, Clustered PCA, Varimax, and non-localized Sparse PCA.

However, deformation components are learned based on the vertex displacements from a rest shape; for decomposition of articulated motions showing rotations, previous methods might be preferable[KSO10; ATTS08]. I would like to give an illustrating example of this limitation. Figure 4.15 visualizes the sparse deformation components automatically extracted from a galloping horse sequence provided by Sumner and Popović [SP04]. It is apparent that even with simple vertex displacement encoding, the sparse localized deformation decomposition can extract and segment meaningful parts from this animation corresponding to the limbs of the horse. However, linear combination of these components is not suited for modeling curvilinear paths of the limbs due to articulated rotation of joints. Since the arced motion trail of the leg cannot be described by a linear path, a multitude of components are required to fit it, some of them showing a shrinking effect of the leg.

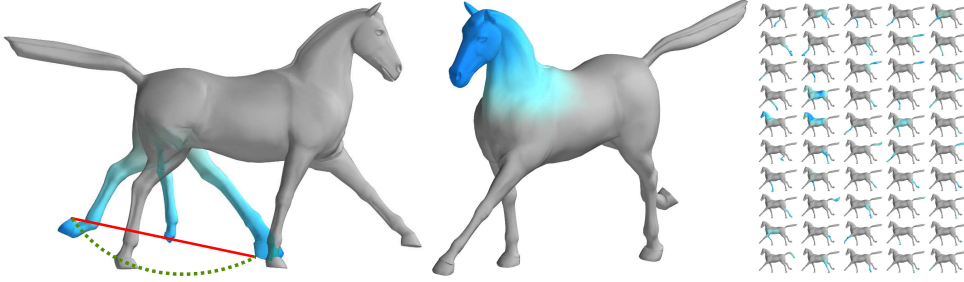


Figure 4.15.: Limitations: SPLOCS extracts linear components representing vertex displacements. This is not suitable for animations involving rotations (left). In this case, the linear path (red) is extracted for the leg motion instead of the green path. Despite this limitation, the motion of the teetering head (right) is captured well. The segmentation into body parts is already very convincing (right), but is expected to further improve when alternative deformation representations are used.

SPLOCS excels at modeling local high-detail deformations such as muscle bulges, cloth folds and slight pose changes after a pre-processing step for approximate pose alignment. This requires pose alignment of the input data. In the future, the pose alignment step may be achieved with an automatic skeletal decomposition methods such as [KSO10; ATTS08; LD12]. Another possibility is to investigate rotation-invariant deformation encodings or hierarchical schemes in Eq. (4.3), suggested already in the publication [NVW+13]. Based on this idea, a first step in this direction has indeed recently been done by Huang et al. [HYZ+14].

Similar to other data-driven deformation methods, SPLOCS can only learn deformation effects visible in the input animation. The option for user constraints allows for the introduction of semantic information into the very process of decomposition. In the future, additional priors such as symmetry or elasticity/material parameters are worth investigating. Right now, SPLOCS automatically extracts components which are symmetric (e.g. left and right side of the face), but this can only be guaranteed if such a symmetric behavior is visible in the input data. For example, SPLOCS cannot provide meaningful deformation components on the right eyebrow if only the left eyebrow moves in the input data. Asymmetry is a very important property of individual faces, but in the future it would be interesting to extend the SPLOCS data-term to incorporate user-given knowledge about symmetry.

An important limitation of the current approach is that the locality-inducing support maps are moving around during optimization. Their behavior is not analytically expressed in the objective function. This causes problems when monitoring convergence. It would be more elegant to reformulate the objective function to directly incorporate the support maps as a variable to minimize over. This approach might introduce discrete variables or high non-linearity. Nevertheless, it can assist in

selecting even better algorithms for optimizing the components. Such a formulation would also help in better understanding the theoretical reasons for SPLOCS performs so well at extracting local and meaningful deformation components. Future work should additionally explore alternative locality-inducing regularizers, e.g. structured sparsity-inducing norms [MJOB11].

In this chapter, I evaluated ADMM against FISTA for computation of sparse components. It is worth hypothesizing about the reason why ADMM converges quicker. In contrast to ADMM, FISTA additionally incorporates the concept of momentum: it keeps a velocity during optimization to extrapolate on the optimal point of the objective function. After FISTA runs for many iterations, it has built up lots of momentum information towards the optimal point and consequently is able to converge very quickly. However, within the SPLOCS framework, there are additional terms to optimize for (the weights  $\mathbf{W}$ ), and we have to switch between optimizing for components and optimizing for weights. For this reason, the  $\ell_1$ -based optimizer for the components is not required to iterate until perfect convergence is attained - here only 10 iterations were used. For only few iteration steps, FISTA cannot build enough momentum to outperform ADMM. Furthermore, ADMM does an implicit gradient update in Eq. (4.18), while FISTA performs an explicit gradient step. In general, implicit updates tend to converge much faster, a fact that is also known from physics simulation where implicit time-stepping usually requires fewer steps than explicit time-stepping. This partly explains the improved convergences of ADMM in contrast to FISTA.

Controlling and exploring the latent deformation space, as offered by SPLOCS, has wider implications in the domain of statistical shape processing. Therefore, another promising research direction is to extend the described theory of sparse deformation components to account for dynamics or physical material properties which can potentially improve the realism of interactive applications such as computer games. I hope that my publication of a reference implementation of the SPLOCS algorithm<sup>8</sup> will stimulate further research in all above-mentioned areas.

---

<sup>8</sup><http://github.com/tneumann/splocs>

## 5. Compressed Manifold Modes for Mesh Processing

Nowadays, 3D reconstruction systems produce a vast amount of 3D data, mostly in the form of triangular mesh surfaces, surfaces that need to be processed for tasks like geometry registration, dense spatiotemporal tracking, filtering, or segmentation. A suitable representation of such surface data can greatly help in these tasks. As an analogy, consider how typical signal processing tasks can be elegantly formulated within a signals' spectral representation, a data representation obtained through the Fourier transform. This general idea has been extended to the domain of 3D geometry processing by Taubin [Tau95] who first generalized the classical Fourier basis to 3D meshes using the eigenfunctions of the Laplace-Beltrami operator.

The eigenfunctions of the Laplace-Beltrami operator of a 3D surface define a basis, known as the manifold harmonic basis (MHB). The Laplace-Beltrami operator is known to capture intrinsic properties of the surface's shape and is invariant under isometric deformation. As such, its eigenfunctions constitute a compact and elegant basis for spectral shape processing that is independent of the actual shape representation and parameterization. In fact, the MHB is unique and characteristic of the geometric and topological properties of the shape. Several successful applications have been proposed that take advantage of these desirable properties, for spectral geometry filtering, compression, and surface editing [VL08; BKP+10; LZ10]. The MHB provides an efficient and robust formulation for estimating shape correspondences, e.g. within the functional maps framework [OBCS+12].

Similar to the Fourier spectrum, the manifold harmonics (MHs) have global spatial support; i.e. each function acts on all the vertices of the mesh. This means that the functions are not easily interpretable, they act on all parts of the mesh. For example, we humans understand the shape of a hand as consisting of the palm and five fingers, but this arrangement is not directly represented within the MHB basis (e.g. Figure 5.12 on page 118). The global support of MHs also renders them very sensitive to topological noise due to holes and occlusions that often appear in scanned meshes. This reduces the utilizeability of the MHB for practical shape matching problems in the real world [BRLB14].

These arguments resemble the observations brought forth in Chapter 4 about PCA components: they have global support, are not easily interpretable, act on every part of the mesh - just like the

MH basis. In fact, by transferring the general idea from Chapter 4 it is possible to build an intrinsic basis for a static shape (mesh) that tackles these disadvantages, a basis that inherits the beneficial property of orthogonality and shape-awareness from the MHB, yet features easily controlled local support. I coin this basis the *compressed manifold basis (CMB)* and the individual basis functions *compressed manifold modes (CMM)*. To this end, I extend the theory of compressed eigenfunctions of differential operators that was recently presented in the context of computational physics [OLCO13a] to the setting of general manifold surfaces in computer graphics. Based on this theory, I provide a complete mathematical derivation and numerical method for extracting the compressed manifold basis for an input triangle mesh, given a discretization of the Laplace-Beltrami operator.

It turns out that compressed manifold modes automatically identify key shape features of the underlying mesh. They automatically group confined local regions like protrusions and ridges into separate basis functions, Figure 5.1. Due to their unique spatial locality, they are robust to significant geometric and topological noise, typical for partial scans from 3D reconstruction systems such as those described in Chapter 2. Thus, the CMB can be considered a tool for robust shape analysis and matching. At the same time, CMB is an orthogonal basis and can reconstruct any function defined on the shape, up to an arbitrary degree of precision. I evaluate the CMB towards developing potential applications in shape matching, shape approximation and feature detection. I open-sourced a reference implementation of the algorithm as well as all scripts to reproduce the results in this chapter<sup>1</sup>.

---

<sup>1</sup><https://github.com/tneumann/cmm>

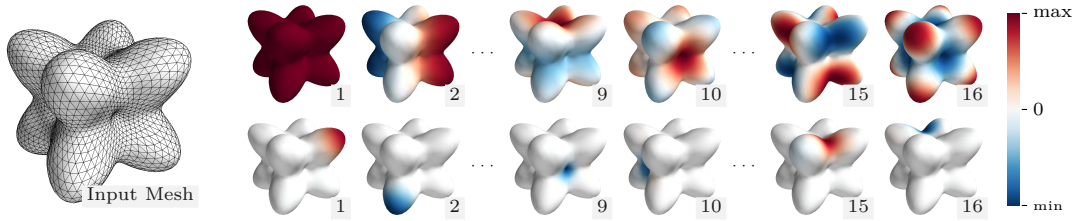


Figure 5.1.: Comparison of different types of eigenfunctions of the Laplace-Beltrami operator of the input mesh (leftmost). *Top Row*: manifold harmonics have global support. *Bottom Row*: the proposed compressed manifold modes (CMMs) have local support and are confined to specific local features, like protrusions and ridges. Here, 8 of the CMMs are found for the 8 protrusions at the corners (1 to 8, only 2 shown here), 6 concentrate at each of the dents (2 shown here), and 12 CMMs automatically form at the valleys between the protrusions.

## 5.1. Related Work

Before introducing the new compressed manifold basis (CMB), let me first give some background on the manifold harmonic basis (MHB) and on its applications in various areas in computer graphics. I will also review recent results on obtaining compactly supported eigenfunctions.

**Manifold Harmonic Basis.** The Laplace-Beltrami operator  $\Delta$  on a 2D manifold surface embedded in 3D space induces a set of eigenfunctions  $\phi_k$  which satisfy the classical equations

$$-\Delta\phi_k = \lambda_k\phi_k, \quad k \in \mathbb{N}, \quad \lambda_k \in \mathbb{R}, \quad (5.1)$$

where  $\lambda_k$  are the eigenvalues of the operator. The eigenfunctions form a basis that is called the manifold harmonic basis (MHB) [VL08]. In most practical applications, only a finite subset of  $K$  eigenfunctions corresponding to the smallest eigenvalues is required.

**Applications.** Due to their compactness, encoding efficiency, isometric invariance, and computational efficiency, manifold harmonics can be found in countless applications in geometry processing; a comprehensive introduction is given in the course notes of Levy and Zhang [LZ10]. Vallet and Lévy [VL08] present several mesh filtering applications and introduce an algorithm that is able to compute the MHB on very large meshes. The recently proposed functional map framework [OBCS+12] uses maps between functions on shapes, for example to transfer segmentations or to estimate correspondences between deformed shapes. The functional map framework is built upon the manifold harmonic basis since this basis is invariant to isometric deformations of the shapes. Pokrass et al. [PBB+12] show how to extend this idea and robustly estimate correspondences between surfaces by inducing sparsity on the functional map. In contrast, the compressed manifold modes described here impose sparsity not on the map between two bases, but on the support of the eigenfunctions themselves.

It is worth mentioning additional applications of the MH basis since the proposed CMM basis is evaluated for some of these application contexts. Various point signatures based on Laplace-Beltrami eigenfunctions are successfully used in shape matching [Rus07; SOG09] where points are described based on the characteristic of heat diffusion on the surface around this point. The connection of the Laplace-Beltrami operator to heat flow equations allows expressing this point description task with the help of the MHB. Other applications include mesh parameterization [MTAD08], where a generalized eigendecomposition of the Laplace-Beltrami operator enables a conformal mapping of the surface to the 2D domain. Goes et al. [GGV08] use the connection between the Laplace-Beltrami

operator and the heat diffusion process for segmenting shapes. The MHs can also help in compressing meshes [KG00] and are used to constrain deformation energies for accurate facial tracking [BWP13]. Recently, the concept of Laplacian eigenfunctions has been brought to image processing: [EKB14] explores applications like color conversion. While I will discuss compressed manifold modes on meshes, the general idea presented here can be easily adapted to images, point clouds, and volume data (like CT scans).

**Sparsity-inducing norms in graphics.** The main mathematical tool used in this chapter is sparsity-inducing regularization. Section 4.1 (paragraph "Sparse Decompositions") covers the most important related work in this domain. One paper deserves to be mentioned here in addition: Rustamov [Rus11] proposes a multiresolution kernel that is centered locally around a specific point on the mesh, using also the sparsity-inducing  $\ell_1$  norm. Due to a similar sparsity-inducing objective, these multi resolution kernels look similar to some of the first compressed modes presented here. However, they look very different for higher  $K$  of compressed basis functions. Another important difference is that the multi-resolution kernels of [Rus11] are defined with respect to some given central point, which requires additional user input that is not necessary for the CMB; furthermore, these kernels do not form a basis, whereas the CMB are explicitly constructed that way.

**Compressed Modes.** Ozoliņš et al. [OLCO13a] propose to find *compressed* eigenfunctions of a general differential operator. To this end, they add a sparsity-inducing  $\ell_1$ -norm into a variational formulation of a problem of type Eq. (5.1). Given a parameter  $\mu \in \mathbb{R}_+$  that controls the sparsity, they arrive at

$$\underset{\phi_k}{\text{minimize}} \sum_{k=1}^K \langle \phi_k, \Delta \phi_k \rangle + \mu |\phi_k|_1, \quad \text{s.t.} \quad \langle \phi_k, \phi_j \rangle = \delta_{kj}, \quad (5.2)$$

where  $\delta_{kj}$  is the Kronecker delta, used here to enforce orthogonality of the eigenfunctions;  $\langle \cdot, \cdot \rangle$  denotes the inner product between functions in a Hilbert space. Interestingly, those compressed eigenfunctions were proven to have *compact support* [BCO14]: They are non-zero only in a confined region of the domain. The size of the compact support can be controlled by  $\mu$ : a large value for  $\mu$  will result in smaller support regions. Until now, compressed eigenfunctions have been predominantly employed on 1D and 2D domains for applications in physics and partial differential equations [OLCO13a]. My approach extends this work and provides compressed Laplace-Beltrami eigenfunctions on three-dimensional mesh surfaces and for 3D shape processing.

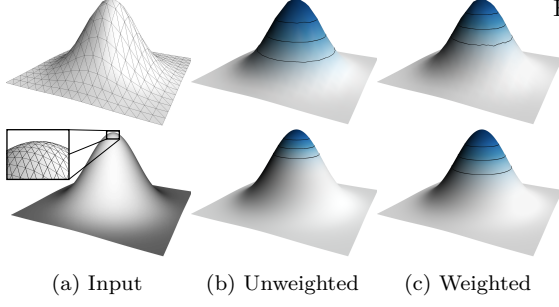


Figure 5.2.: The method correctly handles the area-weighted Laplacian for computing compressed eigenfunctions. (a) This is shown for a low-resolution (top row, 512 triangles) and a high-resolution (bottom row, 32000 triangles) mesh. (b) Without area-weighted Laplacian, eigenfunctions are resolution-dependent. (c) Correct weighting results in resolution-independent eigenfunctions.

## 5.2. Method

As a preliminary, it is first necessary to discretize both the Laplace-Beltrami operator  $\Delta$  and the eigenvalue equation Eq. (5.1) as well as the minimization problem Eq. (5.2). With those building blocks in place, I will then derive an algorithm that is able to compute compressed manifold modes by solving Eq. (5.2).

### 5.2.1. Discretization

This work concentrates on triangle meshes. A popular discretization of the Laplacian  $\Delta$  for a triangle mesh with  $N$  vertices may be realized as a sparse matrix  $\mathbf{L} \in \mathbb{R}^{N \times N}$  with the cotangent weights [MDSB02]. Along with these weights, which only respect angles between edges, proper resolution independence can be achieved with the lumped mass matrix  $\mathbf{D}$  such that  $\mathbf{D}_{i,i}$  contains the surface area around vertex  $i$ . This discretization corresponds to the finite-element formulation of the Laplace-Beltrami operator as detailed in [LZ10], which leads to a generalized eigenvalue problem as explained in the following.

With the Laplace-Beltrami operator defined, the eigenfunctions can now be discretized to eigenvectors. Given a mesh with  $N$  vertices, the first  $K$  eigenvectors are assembled into a matrix  $\Phi \in \mathbb{R}^{N \times K}$ , where each column  $\Phi_{*,k}$  is one eigenvector. The eigenvectors of the manifold harmonic basis (MHB) are governed by [VL08]

$$-\mathbf{L}\Phi_{*,k} = \lambda_k \mathbf{D}\Phi_{*,k}, \quad k \in 1, \dots, K, \quad (5.3)$$

where  $\lambda_k$  are the eigenvalues. To solve for the eigenvectors in  $\Phi$ , one can use an off-the-shelf sparse iterative eigensolver or the efficient band-by-band computation method presented by Vallet and Lévy [VL08].

For computing the proposed compressed manifold basis, the variational formulation Eq. (5.2) is extended from [OLCO13a], in order to make it applicable to triangle meshes. In particular,

the area-matrix  $\mathbf{D}$  must be respected. Without  $\mathbf{D}$ , the eigenbasis is not independent of the mesh resolution, as demonstrated in Figure 5.2. A proper discretization of Eq. (5.2) thus takes the form

$$\underset{\Phi}{\text{minimize}} \quad \text{Tr}(\Phi^\top \mathbf{L} \Phi) + \mu \|\Phi\|_1, \quad \text{subject to } \Phi^\top \mathbf{D} \Phi = \mathbf{I} \quad . \quad (5.4)$$

The next section explains an efficient algorithm that solves this minimization problem.

### 5.2.2. Reformulation using ADMM

The optimization problem Eq. (5.4) can be reformulated in such a way that it can be solved efficiently using the alternating direction method of multipliers (ADMM). ADMM was already used in Section 4.2.3 to optimize for the sparse localized components. Again, I will provide the necessary details how the general ADMM framework can be applied to the problem at hand. I refer interested readers to the comprehensive introduction to ADMM by Boyd et al. [BPC+11]. In its most general form, ADMM can solve (convex) problems that involve two functions and constraints,

$$\underset{\mathbf{x}, \mathbf{z}}{\text{minimize}} \quad f(\mathbf{x}) + g(\mathbf{z}), \quad \text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \quad , \quad (5.5)$$

where in this general formulation the objective terms consist two functions  $f : \mathbb{R}^M \rightarrow \mathbb{R}$ ,  $g : \mathbb{R}^P \rightarrow \mathbb{R}$ , with  $\mathbf{x} \in \mathbb{R}^M$ ,  $\mathbf{z} \in \mathbb{R}^P$ ,  $\mathbf{c} \in \mathbb{R}^C$ , and relation matrices  $\mathbf{A} \in \mathbb{R}^{C \times M}$ ,  $\mathbf{B} \in \mathbb{R}^{C \times P}$ . The optimization problem Eq. (5.4) can be reformulated in the above form of Eq. (5.5). To this end, the orthogonality constraint is replaced with an indicator function

$$\iota(\Phi) = \begin{cases} 0 & \text{if } \Phi^\top \mathbf{D} \Phi = \mathbf{I} \\ \infty & \text{otherwise} \end{cases} \quad . \quad (5.6)$$

This lets us transform the optimization objective Eq. (5.4) into a sum of three functions:

$$\underset{\Phi}{\text{minimize}} \quad \text{Tr}(\Phi^\top \mathbf{L} \Phi) + \mu \|\Phi\|_1 + \iota(\Phi) \quad . \quad (5.7)$$

ADMM can be extended for optimizing functions containing three blocks, but these extensions cannot be guaranteed to converge in each case [CHYY16]. However, general three-block ADMM is actually not necessary in our case<sup>2</sup>, we just need an appropriate splitting strategy. Such a strategy was recently introduced by [WHML13] and can be adapted to the optimization problem at hand. The

---

<sup>2</sup>The generalization of ADMM would also introduce a third relation matrix along  $\mathbf{A}$  and  $\mathbf{B}$  into Eq. (5.4). But the relation matrices are not a necessity for modeling the CMM objective Eq. (5.4). They will just be exploited for splitting the energy function, allowing easier solution of the splitted sub-problems. Thus, a third relation matrix is not needed, we just need a clever way to split the sum of three functions in Eq. (5.7).

idea is to use one of the three functions as the “main function”  $f$  in Eq. (5.5) and group the remaining functions into  $g$ . For computing CMMs, experiments confirmed that choosing  $\iota$  as the main function works best. Applying this splitting strategy allows the following reformulation of Eq. (5.4):

$$\begin{aligned} & \underset{\Phi, \mathbf{S}, \mathbf{E}}{\text{minimize}} \quad \iota(\Phi) + \text{Tr}(\mathbf{E}^\top \mathbf{L} \mathbf{E}) + \mu \|\mathbf{S}\|_1, \\ & \text{subject to} \quad \Phi - \mathbf{S} = \mathbf{0}, \quad \Phi - \mathbf{E} = \mathbf{0}. \end{aligned} \quad (5.8)$$

Notice how the two separate coupling constraints force the variable  $\Phi$  to be equal to  $\mathbf{S}$  and  $\mathbf{E}$ . If those constraints are fulfilled exactly, Eq. (5.4) is recovered. Ultimately, all three parameters  $\Phi$ ,  $\mathbf{S}$ , and  $\mathbf{E}$  correspond to the same values (the compressed manifold modes) at the optimal solution, but the three variables can and will be allowed to be different during optimization.

The equivalence to the standard formulation of ADMM from Eq. (5.5) can be seen by first substituting  $\mathbf{x} = \Phi$  and  $f = \iota$ . Variable  $\mathbf{z} = [\mathbf{E}^\top, \mathbf{S}^\top]^\top$  and function  $g$  are then block-separable with

$$g\left(\begin{bmatrix} \mathbf{E} \\ \mathbf{S} \end{bmatrix}\right) = \begin{bmatrix} \text{Tr}(\mathbf{E}^\top \mathbf{L} \mathbf{E}) \\ \mu \|\mathbf{S}\|_1 \end{bmatrix}. \quad (5.9)$$

Finally, the relation matrices from Eq. (5.5) may be written as

$$\mathbf{A} = \begin{bmatrix} \mathbf{I} \\ \mathbf{I} \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} -\mathbf{I} & 0 \\ 0 & -\mathbf{I} \end{bmatrix}, \quad \mathbf{c} = \mathbf{0}. \quad (5.10)$$

### 5.2.3. Numerical Algorithm

Rephrasing the original minimization problem Eq. (5.4) in the form of Eq. (5.8) makes it possible to apply the ADMM method [BPC+11, Section 3.1.1] to optimize for the CMB. To this end, the Lagrange multiplier  $\mathbf{U} \in \mathbb{R}^{2N \times K}$  is introduced, consisting of two blocks  $\mathbf{U} = [\mathbf{U}^{(E)}; \mathbf{U}^{(S)}]$  that correspond to the two auxiliary variables  $\mathbf{E}$  and  $\mathbf{S}$ . Given an initial value for  $\Phi$ , those variables are initialized to  $\mathbf{E} \leftarrow \Phi$ ,  $\mathbf{S} \leftarrow \Phi$ , and  $\mathbf{U} \leftarrow \mathbf{0}$  at the start of optimization. The algorithm then comes down to iterating between the following steps

$$\Phi \leftarrow \arg \min_{\Phi} \iota(\Phi) + \frac{\rho}{2} \left\| \begin{bmatrix} \Phi \\ \Phi \end{bmatrix} - \begin{bmatrix} \mathbf{E} \\ \mathbf{S} \end{bmatrix} + \begin{bmatrix} \mathbf{U}^{(E)} \\ \mathbf{U}^{(S)} \end{bmatrix} \right\|_{\text{F}}^2, \quad (5.11)$$

$$\mathbf{E} \leftarrow \arg \min_{\mathbf{E}} \text{Tr}(\mathbf{E}^\top \mathbf{L} \mathbf{E}) + \frac{\rho}{2} \left\| \Phi - \mathbf{E} + \mathbf{U}^{(E)} \right\|_{\text{F}}^2, \quad (5.12)$$

$$\mathbf{S} \leftarrow \arg \min_{\mathbf{S}} \mu \|\mathbf{S}\|_1 + \frac{\rho}{2} \left\| \Phi - \mathbf{S} + \mathbf{U}^{(S)} \right\|_{\text{F}}^2, \quad (5.13)$$

$$\mathbf{U} \leftarrow \mathbf{U} + \begin{bmatrix} \Phi \\ \Phi \end{bmatrix} - \begin{bmatrix} \mathbf{E} \\ \mathbf{S} \end{bmatrix} . \quad (5.14)$$

Notice that the original ADMM formulation [BPC+11, Section 3.1.1] updates Eq. (5.12) and Eq. (5.13) in a single step, involving  $g(\mathbf{z})$  in Eq. (5.5), whereas here they can be separated into independent blocks in Eq. (5.9).

The individual update steps are small optimization problems on their own, but they can be performed efficiently. The objective of the minimization Eq. (5.11) can be transformed into

$$\iota(\Phi) + \frac{\rho}{2} \left( 2 \left( \|\Phi - \mathbf{Y}\|_{\mathbf{F}}^2 - \|\mathbf{Y}\|_{\mathbf{F}}^2 \right) + \|\mathbf{E} - \mathbf{U}^{(E)}\|_{\mathbf{F}}^2 + \|\mathbf{S} - \mathbf{U}^{(S)}\|_{\mathbf{F}}^2 \right) , \quad (5.15)$$

where  $\mathbf{Y} = \frac{1}{2} (\mathbf{S} - \mathbf{U}^{(S)} + \mathbf{E} - \mathbf{U}^{(E)})$ . To see this, note that the underlined term in Eq. (5.15) expands to  $2 \|\Phi\|_{\mathbf{F}}^2 - 2 \text{Tr}(\Phi^\top (\mathbf{E} - \mathbf{U}^{(E)})) - 2 \text{Tr}(\Phi^\top (\mathbf{S} - \mathbf{U}^{(S)}))$ . We can recombine this with the other terms in Eq. (5.15) to arrive at  $\iota(\Phi) + \frac{\rho}{2} \left( \|\Phi - \mathbf{E} + \mathbf{U}^{(E)}\|_{\mathbf{F}}^2 + \|\Phi - \mathbf{S} + \mathbf{U}^{(S)}\|_{\mathbf{F}}^2 \right)$  which is equal to the objective in Eq. (5.11). Ignoring terms in Eq. (5.15) that don't change the minimum allows reducing Eq. (5.11) to

$$\Phi \leftarrow \arg \min_{\Phi} \|\Phi - \mathbf{Y}\|_{\mathbf{F}}^2 \quad \text{subject to} \quad \Phi^\top \mathbf{D} \Phi = \mathbf{I} . \quad (5.16)$$

Since  $\mathbf{D}$  contains the vertex area weights, it is a positive diagonal matrix and trivial to invert. Substituting  $\Phi = \mathbf{D}^{-\frac{1}{2}} \Psi$  yields

$$\Phi \leftarrow \mathbf{D}^{-\frac{1}{2}} \left( \arg \min_{\Psi^\top \Psi = \mathbf{I}} \left\| \mathbf{D}^{-\frac{1}{2}} \Psi - \mathbf{Y} \right\|_{\mathbf{F}}^2 \right) . \quad (5.17)$$

By SVD factorization we obtain the orthogonal matrix  $\mathbf{V} \in \mathbb{R}^{K \times K}$  and the diagonal matrix of singular values  $\mathbf{W} \in \mathbb{R}^{K \times K}$  such that  $(\mathbf{D}^{\frac{1}{2}} \mathbf{Y})^\top (\mathbf{D}^{\frac{1}{2}} \mathbf{Y}) = \mathbf{V} \mathbf{W} \mathbf{V}^\top$ . This leads to the closed-form solution of Eq. (5.17):

$$\Phi \leftarrow \mathbf{D}^{-\frac{1}{2}} \left( \mathbf{Y} \mathbf{V} \mathbf{W}^{-\frac{1}{2}} \mathbf{V}^\top \right) . \quad (5.18)$$

A proof of the last step appears in various sources, e.g. in [LO14, Theorem 1]. Thus, step Eq. (5.11) requires an SVD of a  $K \times K$  matrix - the number of vertices  $N$  is irrelevant. Luckily, we almost always have  $K \ll N$ .

To solve the minimization problem Eq. (5.12), we set its derivative to zero. The solution is then

$$\mathbf{E} \leftarrow \rho \left( \rho \mathbf{I} - \mathbf{L} - \mathbf{L}^\top \right)^{-1} \left( \Phi + \mathbf{U}^{(E)} \right) , \quad (5.19)$$

so this update step solves  $K$  very sparse linear systems in each iteration step. By prefactorizing  $(\rho \mathbf{I} - \mathbf{L} - \mathbf{L}^\top)$ , e.g. using Cholesky factorization, this step can be significantly accelerated. Updating the factorization is only necessary when  $\rho$  changes.

Finally, the third step of updating  $\mathbf{S}$  in Eq. (5.13) is separable for each entry  $\mathbf{S}_{i,k}$ . It has a simple closed-form solution that may be expressed concisely using the proximal operator of the  $\ell_1$  norm [BJMO11],

$$\mathbf{S}_{ij} \leftarrow \text{Prox}_{\frac{\mu}{\rho} \|\cdot\|_1}(\mathbf{V}_{i,k}) = \text{sgn}(\mathbf{V}_{i,k}) \max\left(|\mathbf{V}_{i,k}| - \frac{\mu}{\rho}, 0\right), \quad (5.20)$$

where  $\mathbf{V}_{i,k} = \Phi_{i,k} + (\mathbf{U}^{(S)})_{i,k}$  is introduced for brevity. With these building blocks at hand, the steps in Eqns. (5.11)–(5.14) can be implemented. However, we still need to clarify how to monitor convergence of the algorithm and how to set the penalty parameter  $\rho$ .

#### 5.2.4. Convergence

ADMM is guaranteed to converge only for closed, convex (but not necessarily smooth)  $f$  and  $g$ . However, the minimization task in Eq. (5.4) contains the *non-convex* constraint of orthogonality of the eigenvectors, so convergence to a global minimum cannot be guaranteed. Nevertheless, the proposed method finds local minima which are suitable for practical applications.

The convergence can be monitored with the help of the dual and primal residual. The dual residual  $r_{\text{dual}} = \rho \|\mathbf{S} - \mathbf{S}^{(\text{old})} + \mathbf{E} - \mathbf{E}^{(\text{old})}\|_{\text{F}}$  is defined as the difference of the optimization variables before (superscript "old") and after the corresponding updates of Eq. (5.12) and Eq. (5.13). The primal residual for the optimization problem at hand is  $r_{\text{pri}} = \|2\Phi - \mathbf{E} - \mathbf{S}\|_{\text{F}}$ . When both  $r_{\text{pri}}$  and  $r_{\text{dual}}$  fall below some numerical threshold, the optimization stops. It is beneficial to set the parameter  $\rho$  automatically for faster convergence (in contrast to Chapter 4, where this improvement was not observed). The automatic selection of  $\rho$  is implemented by adopting the adjustment strategy discussed in [BPC+11, Section 3.4.1].<sup>3</sup> This means that the only parameters that the user has to specify for computing the CMB are  $\mu$ , which sets the sparsity and controls the locality, and  $K$ , which is the number of CMMs to compute.

### 5.3. Results

The compressed manifold modes feature some intriguing properties: they are geometry-aware, have local support and form an orthonormal basis. Those properties will be explored in this

<sup>3</sup>This automatic selection of  $\rho$  is based on the balance between the primal residual  $r_{\text{pri}}$  versus the dual residual  $r_{\text{dual}}$ ,  $\rho$  is increased when the primal residual is significantly bigger than the dual residual and vice versa. For the threshold parameters in the  $\rho$  adjustment formula [BPC+11, Eq. 3.13], the values recommended in [BPC+11] are used.

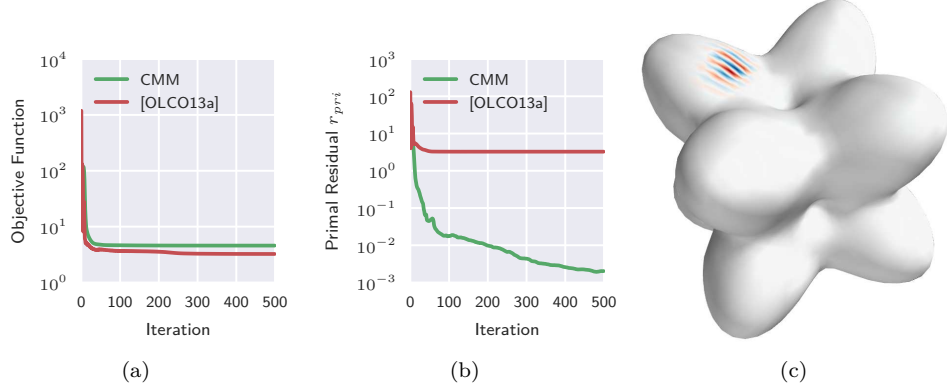


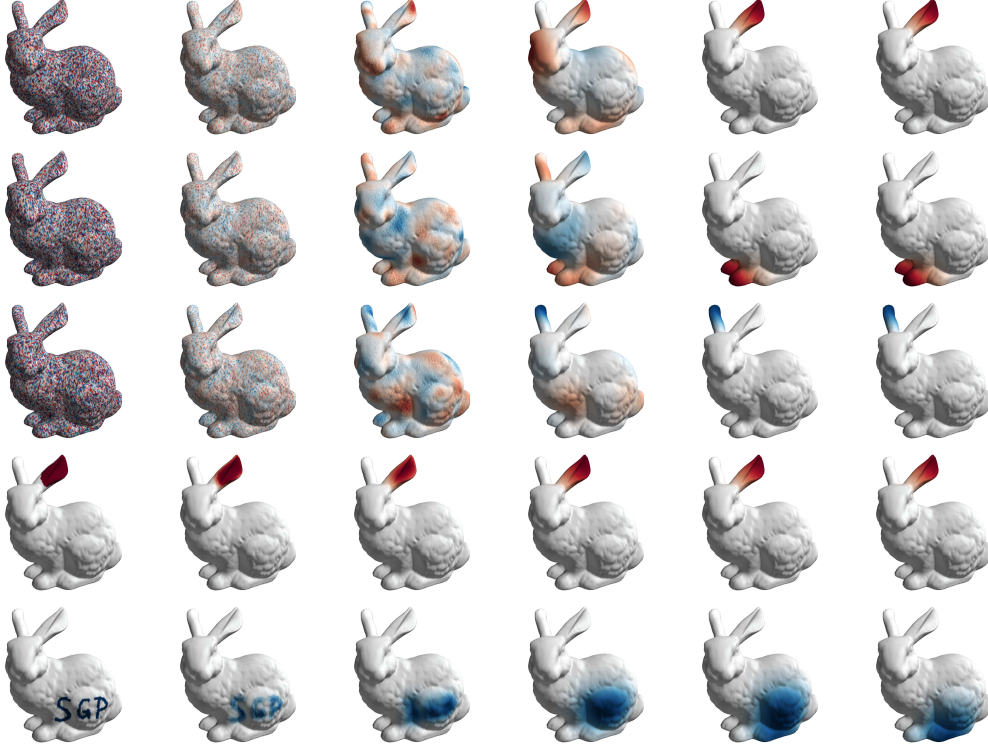
Figure 5.3.: Comparison of convergence of the proposed numerical algorithm (green) vs the direct extension of [OLCO13a] to 3D meshes (red). (a) The objective function value across iterations seems to indicate better convergence of the algorithm by Ozoliņš et al. [OLCO13a], but (b) plotting the primal residual demonstrates that the solutions produced by [OLCO13a] can be infeasible (with respect to the coupling constraints in Eq. (5.8)) because it is stuck at high primal residual. In contrast, the proposed optimization scheme converges. (c) Some of the eigenfunctions computed by [OLCO13a] show oscillations, which do not appear with the new method (compare with the result in Figure 5.1).

section, alongside potential applications in shape matching, segmentation, and localized editing. The proposed optimization has also been evaluated thoroughly. Notice that all the results given here are completely reproducible, especially the tables and figures, using the source code available from <https://github.com/tneumann/cmm>.

### 5.3.1. Evaluation of the Optimization Algorithm

**Comparison to [OLCO13a].** The convergence of the optimization algorithm, Section 5.2.3, was compared to a reimplementaion of the algorithm given in [OLCO13a]. Since [OLCO13a] cannot handle arbitrary  $\mathbf{D}$ , the mass matrix had to be set to  $\mathbf{D} = \mathbf{I}$  for this comparison. While this does not compare directly to the proposed algorithm (that may include  $\mathbf{D} \neq \mathbf{I}$ ), it nevertheless compares to the splitting and penalty adjustment strategy employed here. The mesh from Figure 5.1 was used for comparison. In this example, only the proposed algorithm is able to find a feasible solution, as can be seen in Figure 5.3. I believe that this behavior is caused mainly by the splitting strategy that groups the two convex functions together and selects the non-convex  $\iota$  as “main function”  $f$ .

**Initialization.** The method can be initialized with some random  $\Phi$ . Typically, the algorithm converges to the same set of basis functions, although the ordering is sometimes different. For example, the proposed method always finds the same  $K = 6$  CMMs on the hand mesh for the five fingers and



(a) initialization (b) iteration 2 (c) iteration 10 (d) iteration 50 (e) iteration 200 (f) iteration 1000

Figure 5.4.: Sensitivity to initialization. Here, a single CMM ( $K = 1$ ) was computed for different initializations. Each row shows this single CMM for different iteration steps of the optimization algorithm. (a) shows the initialization; the first three rows are random, the last two rows depict user-drawn scribbles (b) - (f) show the manifold harmonic after a different number of iteration steps. Notice that for different initializations, the method may converge to different local minima.

the palm, Figure 5.12(b). Similarly, the set of eigenfunctions of the mesh shown in Figure 5.1 are always the same, but their ordering can be different because some CMMs have eigenvalues that are close to each other. Figure 5.4 shows the CMB basis for different initializations when computing only one mode,  $K = 1$ . It can be observed that different initializations lead to different local minima. Instead of starting from some random initialization, the user can give a rough scribble as initialization, a feature visualized in the last two rows in Figure 5.4. The ability to incorporate user input might prove useful for certain applications. With suitable user-given initializations, the method converges quicker; for example, the compressed mode on the ear of the bunny is found quickly when the user gives a segmentation of the ear as initialization (already at around 10-50 iterations in the fourth

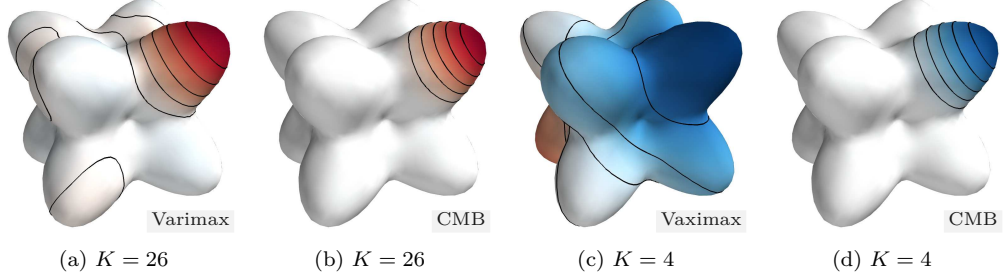


Figure 5.5.: (a) Varimax manifold harmonics show residual oscillations on the whole shape, as indicated by the black isolines. (b) In contrast, the proposed CMMs are truly sparse and zero almost on the whole shape. This effect is amplified when only  $K = 4$  instead of  $K = 26$  basis functions are computed: in this case, (c) the Varimax harmonics show global support while in (d) the proposed CMMs are still sparse.

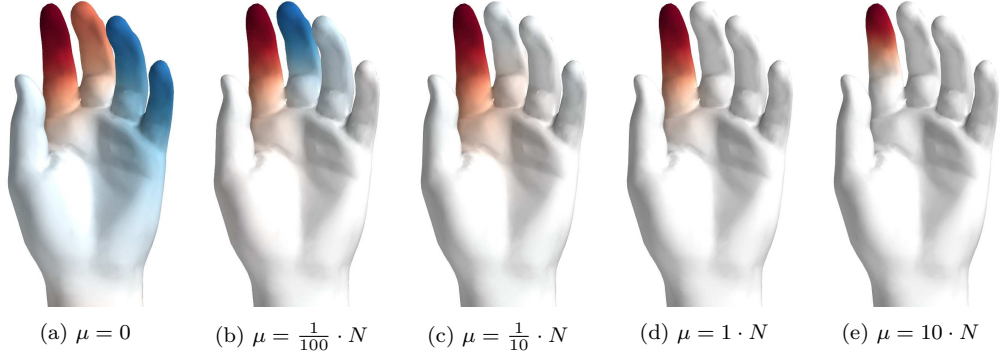


Figure 5.6.: The parameter  $\mu$  controls the locality of the CMMs. A large  $\mu$  will give small local support.  $N$  is the number of vertices in the mesh.

row of Figure 5.4). In contrast, random initializations require more iterations (at least 200 for the bunny mesh, first to third row in Figure 5.4). It is possible to initialize CMMs from the MHB rotated by Varimax [Har76, Chapter 13], as it was done in Figure 5.8. Since the Varimax initialization is consistent and not random, this strategy usually yields reproducible results - even when  $K$  is quite low. This hints at the importance of a proper, localized initialization, but further investigation is needed.

### 5.3.2. Properties of Compressed Manifold Modes

**Comparison to Varimax.** Varimax finds a unitary transformation of the eigenspace that aims to localize the basis by maximizing the second-order moments [Har76, Chapter 13]. Example applications in computer graphics can be found in [SBCBG11; MA07]. Applying the Varimax method on the

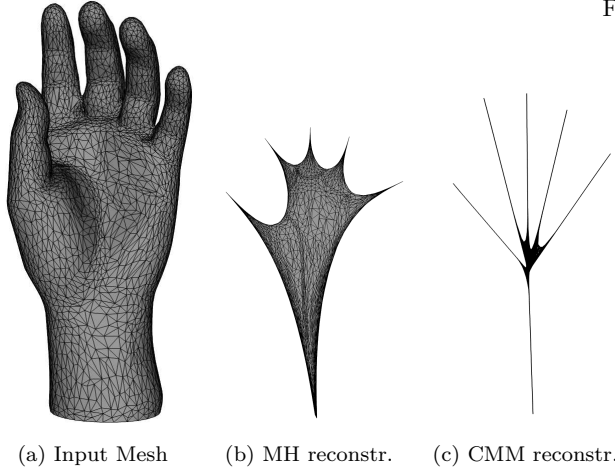


Figure 5.7.: Reconstruction of vertex positions from a small number of  $K = 6$  (b) manifold harmonics and (c) compressed manifold modes. While the MHB basis better retains the original shape, the CMB reconstruction is abstract since every mode only affects a limited number of vertices.

MHB indeed localizes the functions, but global oscillations remain on the mesh: The eigenfunctions are often not exactly zero and are thus not sparse (compare Figure 5.5(a) to Figure 5.5(b)). Also, for small  $K$  the locality of Varimax diminishes completely, Figure 5.5(c), while the CMMs are always local, Figure 5.5(d).

**Influence of Varying Sparsity.** The locality of the CMMs can be controlled by varying the parameter  $\mu$ , which is not possible with Varimax. This is demonstrated in Figure 5.6: large  $\mu$  results in smaller local support. To additionally make the parameter  $\mu$  invariant to the number of vertices  $N$ , one can multiply it by  $N$ .

**Shape Approximation.** Since both MHs and CMMs form a basis, it is possible to use them for encoding actual mesh coordinates. This process was described for MHs by Vallet and Lévy [VL08]. Specifically, we have the transformation to "frequency space" given by  $\hat{\mathbf{x}} = \mathbf{D}\Phi^T \mathbf{x}$ , and its inverse transformation by  $\mathbf{x} = \Phi \hat{\mathbf{x}}$ . Here,  $\mathbf{x}$  are the  $x$ ,  $y$  and  $z$  coordinates of the vertices. For  $K \ll N$ , applying the transform followed by its inverse gives an approximation of the mesh. Comparing those approximations can help gaining a better understanding of the properties of CMB compared to MHB. MHs show better compression guarantees and retain the approximate shape of the whole mesh, Figure 5.7(b). When  $K$  is increased, the MH approximation will move all the vertices towards the input mesh. In contrast, the CMM approximation is more abstract and almost looks like a skeleton of the mesh, Figure 5.7(c). When we look at reconstructions with increasing  $K$ , each additional CMM will introduce a localized change to the approximated shape.

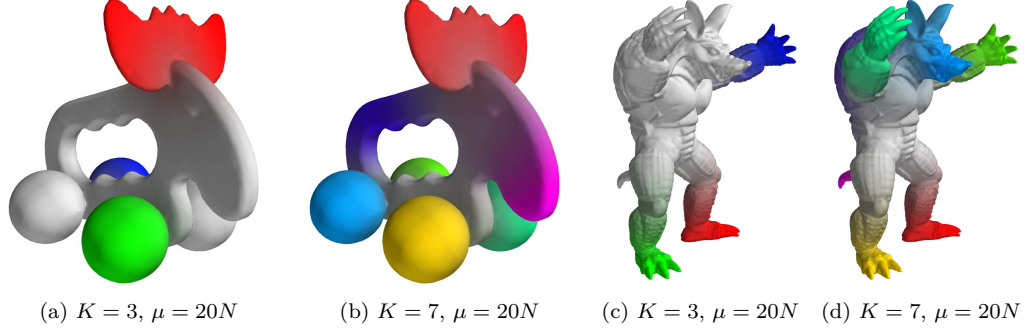


Figure 5.8.: Soft segmentation using CMMs for two meshes and a different number of modes  $K$ . Protrusions and geodesic extremities are segmented by the first CMMs and colorized here with a unique color for each CMM. Grey areas are not covered by any of the first CMMs.

| mesh                | basis | $K$ | $\mu$ | error  | size    | time   |
|---------------------|-------|-----|-------|--------|---------|--------|
| hand<br>(868)       | MHB   | 10  | 5     | 9.34   | 68 KB   | 0.03 s |
|                     | CMB   | 10  | -     | 14.39  | 43 KB   | 3 s    |
|                     | MHB   | 30  | 5     | 4.00   | 203 KB  | 0.08 s |
|                     | CMB   | 30  | -     | 3.97   | 156 KB  | 29 s   |
| fertility<br>(4994) | MHB   | 10  | 5     | 45.02  | 390 KB  | 0.18 s |
|                     | CMB   | 10  | -     | 17.54  | 216 KB  | 44 s   |
|                     | MHB   | 40  | 5     | 7.92   | 1.5 MB  | 0.39 s |
|                     | CMB   | 40  | -     | 6.93   | 933 KB  | 560 s  |
| bunny<br>(34 834)   | MHB   | 10  | 1.25  | 36.89  | 2.7 MB  | 1.4 s  |
|                     | CMB   | 10  | -     | 121.28 | 1.7 MB  | 419 s  |
|                     | MHB   | 40  | 1.25  | 20.49  | 10.6 MB | 3.5 s  |
|                     | CMB   | 40  | -     | 20.68  | 8.2 MB  | 1357 s |

Table 5.1.: Quantitative comparison of the proposed compressed manifold basis (CMB) to the manifold harmonic basis (MHB). The number of vertices  $N$  is given in brackets below the mesh name. See text for further details.

**Segmentation.** Figure 5.8 offers another insight into the behavior of CMMs; here, each CMM was given a unique color, and the mesh was colored according to where the CMMs are non-zero. This shows how the CMM approximation covers the shape by adding important features to the mesh one by one as  $K$  grows larger. The CMMs automatically form at high-curvature regions and topological protrusions of the mesh. To comply with sparsity regularization, the first  $K$  CMMs ignore flat regions or regions connecting geodesic extremities. Thus, CMMs can be understood to perform an importance ranking, and areas of lesser information density may be covered by no single CMM (grey areas in Figure 5.8). Figure 5.9 further demonstrates this in the context of mesh decimation. While I do not claim that CMMs outperform state-of-the-art mesh simplification or segmentation algorithms, I argue that these visualizations hint at the intriguing properties of compressed modes. Potential applications in mesh decimation and segmentation are worth exploring in the future.

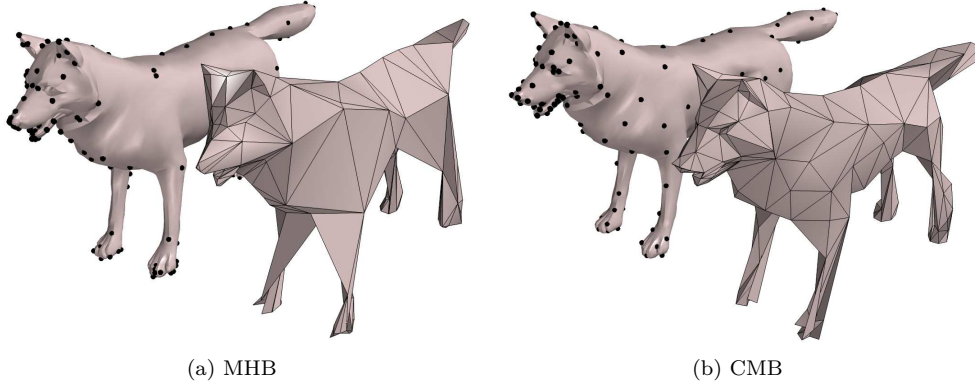


Figure 5.9.: Compressed Manifold Modes cluster at regions with locally high Gaussian curvature. The maxima within these regions turn out to be important shape-aware points (black points). To demonstrate this, these maxima were extracted across all manifold harmonic functions for the manifold harmonic basis in (a) and the compressed manifold basis in (b). These are then used as control points for mesh decimation, here with  $K = 200$  points. The extrema points of the compressed manifold harmonics (b) work much better as local shape-adaptive detectors, which is also reflected in the recovered abstracted mesh.

**Time and Space Complexity for Reconstruction.** Table 5.1 lists results of a quantitative evaluation of the CMB in comparison to the MHB for different number of components  $K$  and for different meshes. The reconstruction error of the vertex coordinates ( $\ell_2$  norm of the difference between reconstruction and ground truth) approaches that of the MHB when the number of basis functions  $K$  goes up. It is also interesting to compare the required memory size for the MHB, stored as a dense array (64bit double), to the CMB stored as a sparse matrix in CSR format. The storage cost of CMB is smaller since the matrix of basis functions  $\Phi$  contains many zeros. The runtimes in Table 5.1 were measured on a Linux system with Intel i7 3.4GHz CPU and 16GB RAM. Notice that this comparison is between the unoptimized Python/NumPy implementation for computing the CMB and the highly optimized ARPACK eigensolver to compute the MHB.

**Insensitivity to noise.** Since the MH basis is based on the intrinsic Laplace operator it is robust to various kinds of noise. It can be empirically shown that the CMM basis functions share this property with the MH basis. In Figure 5.10, Gaussian noise was added to the vertex positions of the meshes (with  $\sigma = 50\%$  of the average edge length of the mesh). The CMMs of the noisy meshes are usually the same, but the ordering or the sign of the CMMs may flip. In order to observe how the CMB reacts to topological noise, small holes were cut in some example meshes. In Figure 5.11 one can see that, up to ordering and sign flip, the CMMs align well even in this case. In both experiments, it was observed that the ordering aligns much more robustly when a good initialization is used.

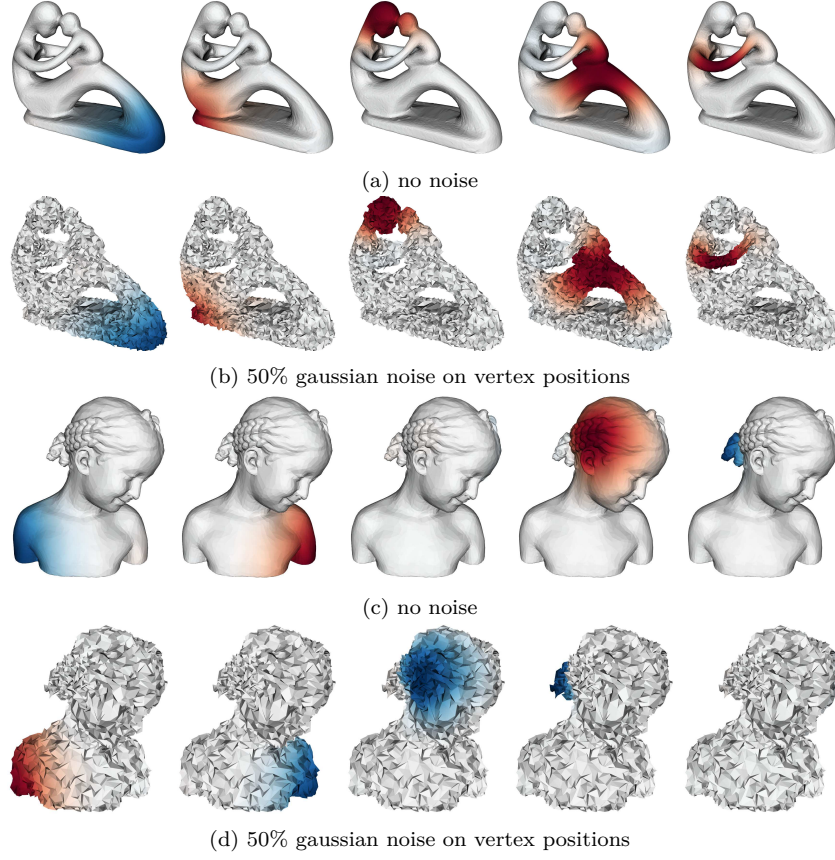


Figure 5.10.: Insensitivity to noise. (a) and (c) show the  $K = 5$  CMMs on two different meshes without noise. (b) and (d) show CMMs after adding Gaussian noise to the vertex positions of the mesh. The CMM basis functions align well between the original and the noisy mesh, but the ordering and the sign of the CMMs may flip. Note that on the second mesh, one basis function is located on the backside of the mesh and thus is not visible (3rd column in (c) and 5th column in (d)).

Therefore, Varimax was used for initialization. In the following section, I discuss insensitivity to larger holes and to isometric deformations.

### 5.3.3. Applications in Shape Matching

Typically, shape matching involves first robust feature detection and matching, and then geometry-aware regularization that extends this correspondence to the entire shape. For practical shape correspondence, both these steps have to be robust to geometric and topological surface noise that inevitably occurs in real world scanning systems. Occlusions and partial scans make this problem

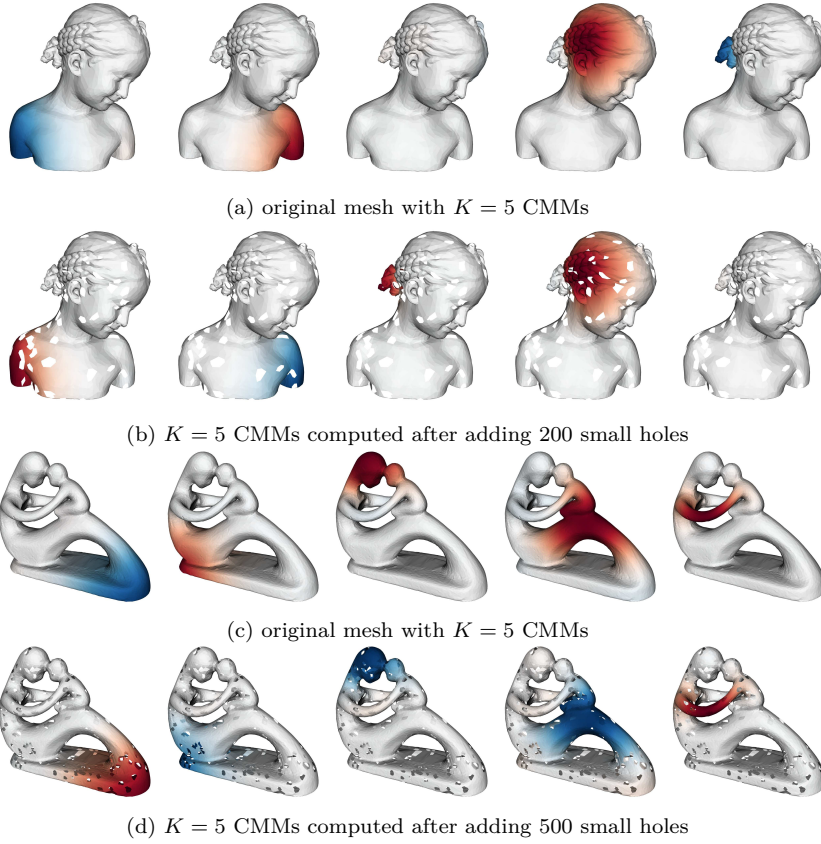


Figure 5.11.: Up to ordering and sign flip, the CMMs align even after adding topological noise in form of small holes.

even more challenging. As an example, the shape descriptor presented in Section 2.3.4 gave us robust feature matches, and the as-rigid-as-possible template registration procedure presented in Section 3.2.2 gave us a full correspondence field even in case of occlusion.

Ovsjanikov et al. [OBCS+12] use the manifold harmonic basis to propose the elegant framework of functional maps, a framework that can solve this registration problem very effectively. Often, the manifold harmonic bases between two shapes do not exactly correspond to each other. The functional map provides full correspondence of the two shapes even in this case, but only if the sparse input correspondences (point or region features) that are required to construct the map are correct. On the other hand, if the shapes are related by an isometry, the manifold harmonics align well. Then, the functional map yields a diagonal correspondence matrix. Pokrass et al. [PBB+12] use this idea for regularizing the functional map to be sparse and close to a diagonal. However, this is limited to

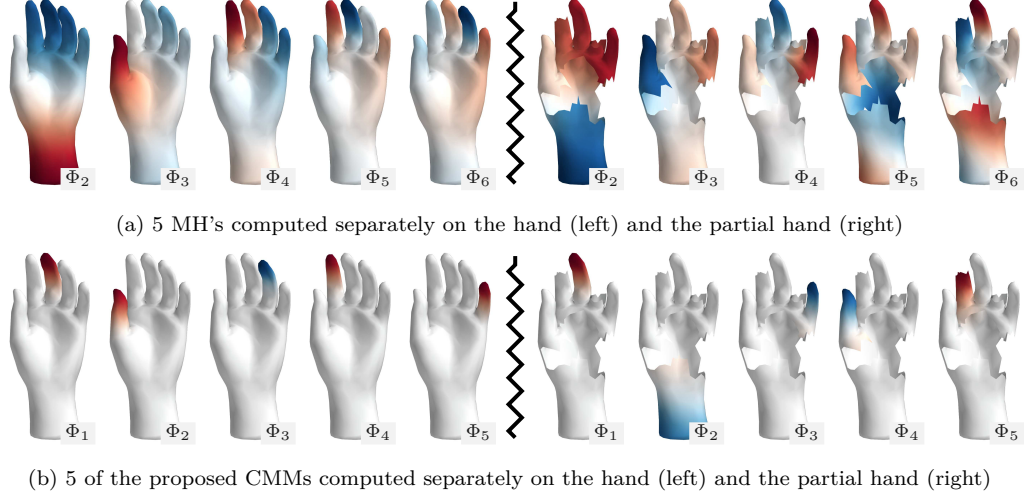


Figure 5.12.: Basis functions of a hand mesh (left side) and a partial shape of the hand emulating typical scanning artifacts (right side). (a) The MHB basis functions do not correspond well between partial meshes. (b) For every CMM on the full hand, left, we can find a CMM on the partial hand, right. One exception is the ring finger which is completely missing on the right side. Thus, up to sign flip and ordering, the CMMs align very well.

situations where the manifold harmonics can be aligned between the shapes. One important case where this assumption fails is if the mesh has large holes or features severe topological noise due to partial scans. Figure 5.12 shows an example where the mesh of a hand is badly corrupted by manually simulated holes. “Dense” harmonics of the MHB are severely corrupted by these holes and no longer align with the original mesh. However, up to a sign flip and ordering, CMMs still align robustly with those of the original mesh. At the same time, the CMMs align very well in the case of purely isometric deformations, as demonstrated in Figure 5.13. Notice that in both examples the same set of CMMs is recovered, even though completely different random initializations were used for the different meshes.

Because of their spatial locality and robustness, compressed manifold modes can be used as shape features in the first stage of shape matching. But since the CMB provides an orthogonal basis that is invariant to isometric deformations, they can simultaneously be used in the second stage of shape-aware regularization. This second aspect was tested by using the CMB to replace the MHB in the functional map framework as follows. In the discrete setting, the functional correspondence between two shapes,  $S_1$ , resp.  $S_2$ , can be given as a simple  $K \times K$  matrix  $\mathbf{C}$ . Both shapes are equipped a  $K$ -dimensional basis, discretized in the basis matrices  $\Phi^{(S_1)}$  and  $\Phi^{(S_2)}$ , respectively. The rows in  $\Phi^{(S_1)}$  and  $\Phi^{(S_2)}$  can be seen as  $K$ -dimensional point clouds (in eigenspace), and the matrix  $\mathbf{C}$

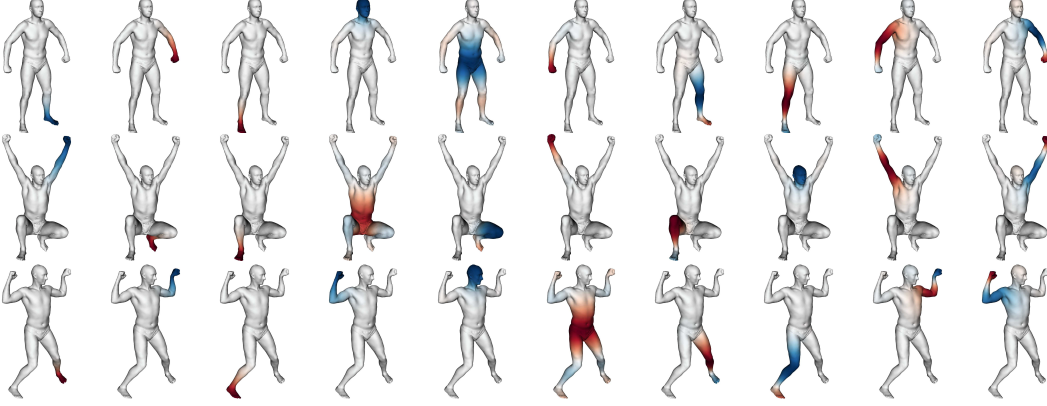


Figure 5.13.: Compressed manifold modes are insensitive to isometric deformations, up to ordering and sign flip. Each row shows  $K = 10$  CMMs on a different mesh from the SCAPE database [ASK+05].

aligns those point clouds by  $\Phi^{(S_2)} = \Phi^{(S_1)} \mathbf{C}^\top$ .<sup>4</sup> Thus,  $\Phi^{(S_1)} \mathbf{C}^\top$  is a rotated basis such that the basis vectors (columns) overlap with those in  $\Phi^{(S_2)}$ . A necessary condition for  $\mathbf{C}$  to be an area-preserving vertex-to-vertex map is that it must be orthogonal [OBCS+12, Theorem 5.1].

To illustrate CMMs in a shape matching scenario, let us take two shapes with ground-truth correspondence differing by isometric deformations. Those given correspondences are then used to find the optimal orthogonal  $\mathbf{C}$  for different numbers of basis functions  $K$ . One can then measure how much error (in geodesic distance) the resulting map between the basis functions introduces, depending on the number of eigenfunctions  $K$ . This is exactly the same experiment as performed in [OBCS+12, p. 3], but here applied using the new CMB. As in [OBCS+12], the unweighted Laplacian is used for these experiments. Despite the CMB not being as information-dense as the MHB, it achieves similar geodesic error with a fairly low number of basis functions  $K$ . It can be observed that around 35 compressed modes are good enough to meet the accuracy of dense harmonics, Figure 5.14. Even more interesting is the phenomenon that the correspondence matrix  $\mathbf{C}$  is much sparser for the CMB when compared to the MHB (Figure 5.14(b) versus Figure 5.14(c)). Note that sparsity of  $\mathbf{C}$  was never enforced, at no point during the estimation of  $\mathbf{C}$ . A direct implication of this is that the matrix  $\mathbf{C}$  is closer to a permutation when computed based on the CMB. For matching partial shapes, this effect is even more apparent, as seen in the middle and bottom row in Figure 5.14. The respective correspondence matrix  $\mathbf{C}$  with CMB is again much closer to a permutation matrix, whereas that of the MHB shows more off-diagonal entries and is much denser. I argue that this is a strong indicator

<sup>4</sup>This is the usual basis transformation. It may also be written as  $(\Phi^{(S_2)})^\top = \mathbf{C} (\Phi^{(S_1)})^\top$  but we take the same notation as Pokrass et al. [PBB+12] since it involves less confusing brackets.

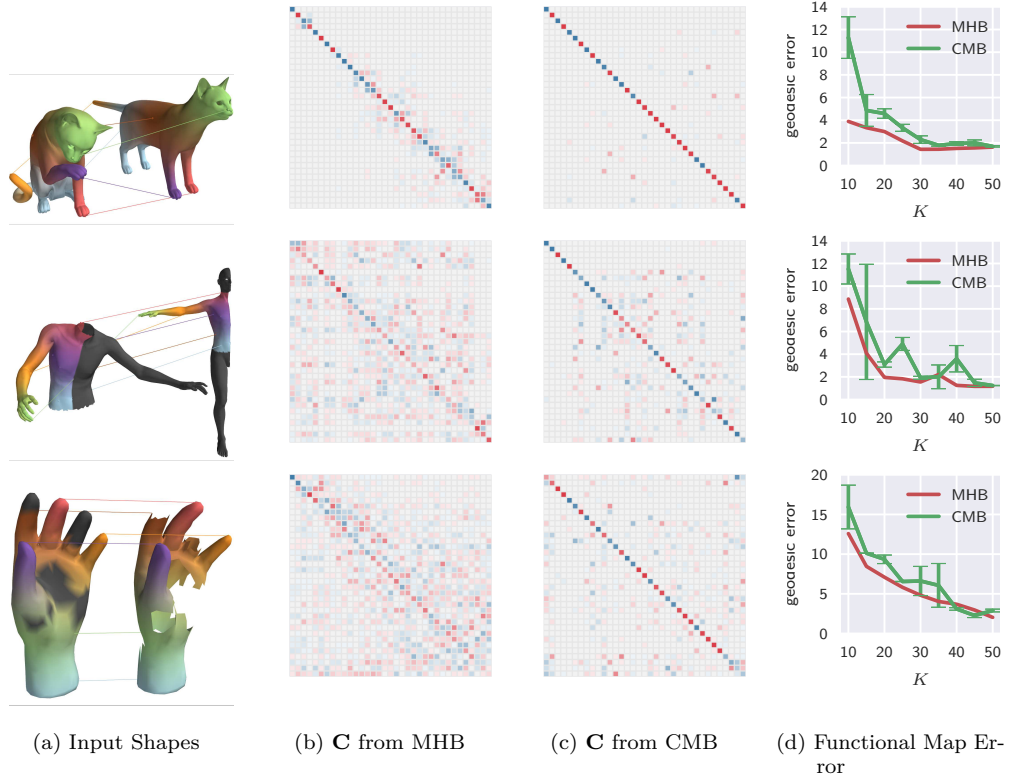


Figure 5.14.: Visualizing the functional map correspondence in three different scenarios. (a) Two meshes are given with known ground-truth vertex-to-vertex correspondence. (b) shows the functional representation of the correspondence between the MHB of both shapes (for  $K = 35$ ). (c) The functional map between the CMB clearly shows much higher sparsity (for  $K = 35$ ). Notice how the effect of sparsity in the correspondence matrix is retained even in cases of partial correspondences between meshes with huge holes (middle and bottom row). (d) visualizes the mapping error of the maps between MHB and CMB. With increasing number of basis functions  $K$ , the error of the CMB approaches that of the MHB. Error bars show the standard deviation of the geodesic error across 5 different random initializations of the CMB.

that CMMs can be extremely useful for shape matching or even partial shape matching that is robust against large holes.

#### 5.3.4. Localized Editing

The local nature of CMMs can be used for mesh manipulation by local edits that preserve fine surface detail and overall rigidity of the shape. Successful mesh editing paradigms such as the classical Laplacian surface editing framework [SCOL+04] or extensions using non-linear deformation energies

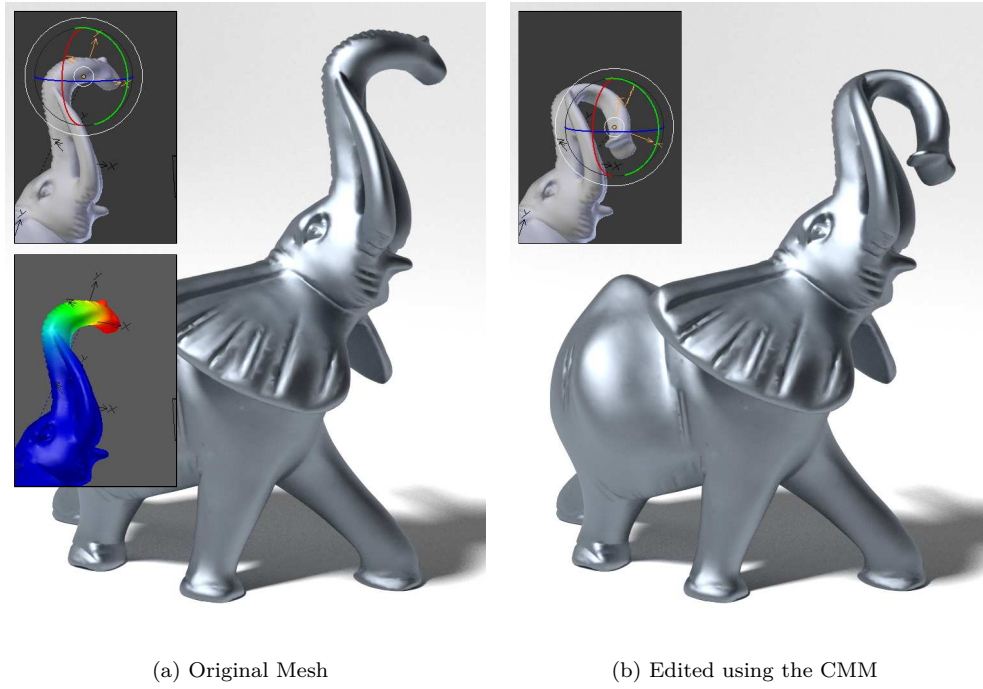


Figure 5.15.: The automatically found CMMs can be used as skinning weights that automatically concentrate on important parts of the mesh, for example on the trunk of the elephant. CMM skinning weights can be used as a 3D modeling tool to produce local edits. (a) The original elephant, where the insets show the manipulator attached to a CMM found automatically on the trunk. The second inset visualizes the CMM. (b) This handle can be used to move and rotate the trunk, achieving a local and smooth edit.

[TSSH13; JBK+12] usually requires the user to supply handles and constraints on the surface. For example, to move the arm of a virtual human character, a handle is placed at the hand; then, other parts of the body need to be explicitly “pinned” such that they don’t move. Essentially, this circumscribes a soft deformation region. Compressed manifold modes can help to automatically find such local regions. The first few CMMs happen to latch onto protrusions such as legs and arms of a character, or the nose and the chin of a face. At the same time, due to  $\ell_1$ -induced sparsity, they are completely zero for the rest of the mesh. In other words, CMMs output automatically-detected shape-aware regions for editing the mesh.

Figure 5.15 demonstrates that CMMs can be used in the context of linear blend skinning. In this case, the individual basis functions  $\Phi_{*,k}$  must be scaled so that their values are between 0 and 1 after optimization, so they can be provided as per-vertex skinning weights. A transformation handle may then be placed automatically at the center of the support area where the CMM is non-zero. A

smooth and local surface deformation can be achieved by translating and rotating the handle. The automatically generated handles are very similar to cumbersome rig handles that an animator might set by hand.

#### 5.4. Discussion

Section 5.3.4 demonstrated the possibility to use CMMs as skinning weights for mesh editing applications. The compressed modes can act as meaningful weights for such editing, but the CMMs lie in the range  $[-1, 1]$ , hence can be negative. It would be desirable to produce skinning weights that are positive everywhere and that sum up to one for all the vertices of the shape, similar to the bounded biharmonic weights of Jacobson et al. [JBK+12]. CMMs cannot meet these requirements and do not fulfil exact interpolation constraints. Adding these kinds of constraints, e.g. constraining the weights to be between 0 and 1 and to sum up to one, unfortunately renders the  $\ell_1$  regularization term useless: when we force  $\sum_{k=1}^K \Phi_{i,k} \stackrel{!}{=} 1, \forall i \in 1, \dots, N$  this would mean that  $\|\Phi\|_1 = N$  for any  $\Phi$  fulfilling these constraints, making optimization with this  $\ell_1$  norm meaningless. But while bounded biharmonic weights also achieve some level of sparsity, they require the user to provide a set of point constraints or a skeleton rig. This is not needed for CMMs. The CMB automatically provides meaningful “areas” that can be used to restrain certain edits. Future research in this direction should focus on how to combine such constraints elegantly with a sparsity-inducing norm, for example using the non-convex  $\ell_p$ -norm with  $0 < p < 1$ , or with the help of norms that induce structured sparsity. CMMs may also regularize sparse input positional constraints for applications such as posing a shape, key-framing an animation, or tracking a deformable object in videos - applications that are yet to be explored. Additionally, the CMB can be used for projecting certain deformation energies, thereby reducing their dimensionality and thus computationally simplifying and regularizing the deformation process.



Figure 5.16.: Some of the  $K = 300$  higher CMMs computed on the hand mesh show higher frequencies and small vibrations in a specific area in the mesh, similar to wavelets. However, computing that many modes with the proposed algorithm takes computation time on the order of minutes.

The Laplace-Beltrami eigenfunctions are related to the heat kernel operator for heat diffusion on the mesh surface. This heat kernel has several applications, particularly in shape matching [SOG09]. Investigating the effect of sparsity and  $\ell_1$ -minimization in the heat diffusion framework was partly done by Rustamov [Rus11], but only in the context of a single-point heat kernel. CMMs can help in further investigations. Local support was proven for compressed modes in the two-dimensional plane [BCO14]; I showed empirically that this also seems to be the case on manifolds. Proving theoretically that the obtained functions are connected on the manifold is a related question open for future research.

Because of their spatial locality, higher-order compressed modes disintegrate into tiny vibrations on local surface patches, Figure 5.16. If we can extend the CMB in a multi-resolution framework such that it yields a compact basis, it will have many applications akin to the wavelet basis. With similar inspiration, Ozoliņš et al. [OLCO13b] recently introduced compressed plane waves which are shift-invariant basis functions of the Laplace operator and resemble wavelets of increasing frequency. But this work requires the shift operator to be well-defined, which is possible for regular domains, e.g. on the 2D plane, where a natural grid structure can be superimposed on the domain. Building a shift-invariant multi-resolution basis for general manifolds is an open and challenging problem, with implications in many fields.

One important point I would like to discuss is the non-convexity of the objective function in Eq. (5.2) which leads to a certain dependence on the initialization, a dependence that was rigorously evaluated in this chapter. As mentioned, initializing compressed modes with Varimax can yield a deterministic result in some cases, but that doesn't solve the underlying problem. Lai et al. [LLO14] recently proposed a convex relaxation of Eq. (5.2). However, their approach involves optimizing over an  $N \times N$  matrix, which means that it is infeasible for meshes with a large number of vertices  $N$ . In my opinion, this is still an avenue for future work.

Apart from the dependence on initialization, the current optimization algorithm is comparatively slow in converging to a precise solution, especially for computing a large number  $K$  of compressed modes. This is a common problem with ADMM: While the algorithm is known for its quick convergence to approximate solutions, it usually takes a lot of iterations to converge to a precise solution. While I observed more robust convergence compared to [OLCO13a], Figure 5.3, the proposed method seems to run into numerical problems for very large  $K$  (depending on the size of the mesh at around  $K > 100$ ) and very irregularly tessellated meshes, producing a few compressed modes with artifacts. In the future, different acceleration and supervision strategies may help to ameliorate these issues. Similarly, novel optimization methods might improve upon the numerical problems: The constraint set  $\Phi^\top \mathbf{D} \Phi = \mathbf{I}$  enforced in Eq. (5.4) has a manifold structure called the Stiefel manifold

[AMS08, Chapter 3]. Optimization algorithms exist that can optimize directly on this manifold [BMAS14]. It is not easy to directly apply such methods since they usually expect a smooth and convex objective function, while the  $\ell_1$ -norm is only convex but not smooth. Future work should thus concentrate on combining proximal methods and non-linear optimization on manifolds.

The general framework of obtaining localized functions of a differential operator on mesh manifolds may be extended in other important ways. For example, computing the CMM basis on the sphere would lead to compressed spherical harmonics and certainly would lead to interesting applications in rendering. Discretizing the Laplacian for color images instead of meshes and then computing the compressed eigenfunctions may lead to new applications in image processing. Similarly, it would be interesting to see how the compressed basis functions of other operators or generators defined on the mesh look like: The proposed approach is not restricted only to the Laplace-Beltrami operator. Different discretizations of the Laplace-Beltrami operator are mentioned by Wardetzky et al. [WMKG07]. Other schemes like the Killing vector field [BCBSG10] also come to mind. Using the concept on elastic surface energies [TSSH13] may provide localized and sparse vibration modes. All these options are worth exploring in the future.

Finally, I would like to mention that specific applications in geometry processing need to use the CMB in a judicious way that makes sense to the application in question. In this chapter, I have shown implications of the CMB to various applications such as shape matching, shape approximation and shape editing. However, I have not developed complete applications that compete with the state-of-the-art algorithms specifically designed for these scenarios. For example, it would certainly be interesting to see whether the CMB approach can help obtaining even better registration results in the reconstruction framework described in Chapter 2. I will leave this to future work and hope that the public release of the source-code<sup>5</sup> will help in this investigation.

---

<sup>5</sup><https://github.com/tneumann/cmm>

## 6. Summary and Future Work

In this work I have presented novel approaches all along the 3D geometry processing pipeline, ranging from surface acquisition and registration to data-driven modeling and editing. Chapter 2 introduces two new methods to reconstruct dense and spatiotemporally consistent geometry of deforming 3D surfaces. These approaches rely on patterns applied onto the surface, allowing for robust acquisition of moving and deforming surfaces, with examples of garments, walking shoes, and bulging human muscles. The captured 3D surface data lends itself to full-field strain analysis, which opens up new application areas in textile engineering or material sciences. Most importantly, the acquisition methods measure the surfaces' true tangential deformation with high accuracy. Thus, they are able to capture important effects such as the sliding and shifting of the skin on top of soft tissue and muscle.

In Chapter 3, I exploit these new acquisition capabilities to build the first comprehensive data-driven model for muscle deformation of the shoulder-arm region, spanning the parameter spaces of body pose, body shape and external forces. To this end, the acquisition pipeline is extended by a scalable registration method that jointly aligns thousands of dynamic 3D reconstructions from multiple people. I also contribute a novel shape parameterization and an effective semi-parametric non-linear learning approach. This combination provides the main ingredients for an expressive data-driven model, a model that can be evaluated in real-time and has low memory requirements. The result is a trained model that reproduces fine-scale, anatomically realistic muscle deformations at high quality not attained by previous data-driven methods. Such a model offers realistic character animation capabilities and can provide a basis for detailed biomechanical analysis of human muscles.

Chapter 4 widens the scope of data-driven modeling to the unsupervised setting and introduces sparse localized deformation components (SPLOCS). By embracing the concepts of sparsity and locality, SPLOCS decompose captured surface mesh data into components that are spatially localized and identify local features of shape deformation. Unlike previous data-driven approaches, it allows controlling the dimensions of the underlying space of deformations. This provides a powerful tool for artistic editing, data exploration, and statistical shape processing. I demonstrate novel capabilities for intuitive and localized artistic editing for a wide variety of datasets, ranging from facial performance captures and full-body motion to muscle and cloth deformations. Experiments showed that SPLOCS

achieve improved generalizability compared to the state-of-the-art, which means that the method can better predict data that it was not trained on.

In Chapter 5, I transfer the concepts of sparsity and locality from decomposition of 3D deformation to the analysis of 3D shapes. I propose a novel intrinsic basis for 3D surfaces, a basis consisting of compressed manifold modes (CMM). CMMs have local support around key shape features that are automatically detected. I present a new numerical algorithm that extracts such a local shape basis for a given 3D mesh. My experiments reveal the unique capabilities and properties of CMMs, such as invariance to noise and to partial scans. I show potential applications in shape matching, suggesting a huge potential for future application for 3D reconstruction and registration systems, such as the ones presented in Chapters 2 to 3 of this thesis.

The 3D reconstruction methods that I proposed in this thesis provide a solid foundation for future improvements and extensions. The quad pattern-based reconstruction method from Section 2.2 is designed for parallelized GPU implementation, and the current implementation is close to achieving real time capture performance. So we should concentrate on optimizing the algorithm implementation for current GPU architectures. At the time of writing this thesis, the quad pattern method has already been extensively used by adidas AG. Their captured sequences provide a huge and very diverse benchmark dataset that can help to further increase the practicality and robustness of the system.

The multiview conflicts proposed in Section 2.3 may help in the general (markerless) 3D reconstruction settings. Jensen et al. [JDV+14] suggest that matching ambiguity in areas of missing or repeatable texture is the major reason for holes in multiview-stereo reconstructions. I believe that the proposed multiview conflicts tackle precisely this problem, so if we incorporate this idea into a state-of-the-art multiview-stereo method like [GS15] we could further the state-of-the-art of multiview-stereo estimation. The proposed global graph matching framework further offers opportunities to incorporate additional structure constraints or priors, depending on the application in question. One important example is loop-consistency [SRM12] which can be incorporated in the graph matching framework to improve optical flow estimation required in free-viewpoint video applications, especially in cases of unstructured and uncalibrated multiview data [LKM14]. Apart from transferring these ideas to the mentioned domains of multiview-stereo estimation, optical flow, and free-viewpoint video, further development of the random-dot based 3D reconstruction method should concentrate on improving runtime and scalability of the approach. This would facilitate even denser reconstructions from even higher-resolution cameras, in a shorter amount of time. To achieve this goal, I suggest exploring novel ways to adapt the elegant framework of sparsity-inducing norms

---

and the convex optimization method of ADMM employed in Chapters 4 to 5 to the multiview graph matching scenario.

The data-based animation method presented in Chapter 3 was evaluated on the shoulder-arm region, but it can work as a framework for modeling arbitrary body parts and should easily scale to even larger datasets or the full-body. However, the deformations that the approach can model are limited to the animation rig provided. For example, the simple animation rig used in this work cannot explain motion of the scapula and clavicle bones. Consequently, clavicle bone deformations are averaged between subjects and cannot be controlled by the artist. An obvious way to tackle this limitation is to embed physiologically correct skeleton rigs into the capture data. A less obvious, but all the more appealing idea is to use SPLOCS to discover the additional articulated complexity that the low-fidelity skeleton cannot explain. However, for such an approach to succeed we must combine unsupervised learning methods like SPLOCS with the supervised learning methods from Chapter 3. This is a promising approach that should be explored in future research.

I already discussed limitations of SPLOCS and offer diverse ideas how to fix them in Section 4.4. But there is one limitation that I am not happy with in particular, and so I would like to repeat the following request to future researchers: The locality-inducing support maps are not analytically expressed in the objective function of SPLOCS. Apart from not being mathematically elegant, this results in problems with monitoring convergence and requires a very specific, greedy initialization method. Future research should formulate the geodesic distance regularization term (that is currently simulated with the support maps) in a mathematically precise way. This may lead to discrete variables or strong non-convexity, maybe it even recovers exactly the proposed optimization framework. But it would certainly help in better understanding the theoretical underpinnings of sparse localized components, would fix the convergence problem, and it would probably convince even more researchers to use SPLOCS in novel application use cases.

The framework of compressed manifold modes is a very interesting one to explore in the future. While I present a sparse localized basis for 3D geometry based on the Laplace-Beltrami operator, the framework may be able to construct sparse and localized bases for other operators and generators as well, e.g. on arbitrary manifolds such as spheres (*compressed* spherical harmonics), images, or on elastic deformation energies of physically simulated surfaces. These options are worth exploring in future work.

All projects presented in this thesis also bear potential for future research in various areas outside computer graphics and computer vision. For example, machine learning relies heavily on decomposition and dimensionality reduction techniques such as PCA, autoencoders, multidimensional scaling, and linear discriminant analysis - to name just a few. These techniques allow for efficient data analysis as

well as feature selection and feature extraction. In the future, the locality-preserving techniques for data decomposition and basis construction proposed in Chapter 4 and 5 shall help to improve such machine learning tasks for datasets that exhibit a geodesic distance metric - e.g. image sequences or network datasets.

In a similar way, the dynamic 3D reconstruction techniques proposed in this thesis can be helpful in a number of application scenarios. For example, they provide textile engineers with precise deformation measurements that in turn assist in virtual prototyping and design processes. In fact, Colaianne et al. [CSS+16] just recently used the system presented in Section 2.2 for evaluating a new method that constructs patterns for made-to-scan compression garments. In the future, researchers in orthopaedics and biomechanics shall use the proposed 3D reconstruction systems to gain insights into the temporal and spatial strategies of human muscle control. A first step in this direction was recently accomplished in our own group [PGF+16]: for the first time, we simultaneously recorded surface deformations and Electromyography (EMG) data of muscles in the lower body along with foot pressure distribution during balance tasks. Surface deformation measurements promise precise insights into the temporal and spatial strategies for muscle control during balance tests and may lead to improved assessment and therapy planning. However, they require precise biomechanical modeling of the underlying process. This may be achieved through a combination of data-driven methods from this thesis and biomechanical simulation models like OpenSIM [DAA+07]. Biomedical applications also require new anatomically plausible decomposition techniques that subtract the motion of the limbs (the kinematics) from the bulging of the muscles, so that muscle bulging can be precisely quantified. The SPLOCS technique provided first results in this direction (Figure 4.12), but more research is needed to connect sparse decomposition techniques with anatomically realistic priors.

By the time this thesis is written, further research inspired by the presented approaches has been published. With a similar goal as this work, Bogo et al. [BRLB14] set out to capture detailed tangential surface deformations for later use in data-driven modeling. They stamp patterns onto the skin to help in parameterization registration. Bogo et al. use 22 structured-light scanning units to acquire very fine geometry whose parameterization can be corrected by estimating the tangential shift of the stamp pattern. They provide stunning resolution and high detail of dynamic body shape variation, albeit at much higher hardware cost compared to the systems presented in this thesis. Data captured with this system is used by Pons-Moll et al. [PMRMB15] to extend data-based human body modeling by learning time-dependent soft-tissue deformations like wobbling of body masses. Tsoli et al. [TMB14] provide a non-linear data-driven model for breathing motion. Inspired by SPLOCS, Yang et al. [YYZ+14] learn a semantic full-body model that explicitly models variations in pre-segmented body parts separately. These methods provide another layer towards the realism of data-driven

---

animation since they capture additional phenomena of the dynamic behavior of human body shapes in motion. However, such methods require very costly capture setups. In contrast, I helped in building a system that extends an existing data-based model to new subjects by capturing them with a Kinect sensor [RNVT13]. Independently of my work on shape parameterization for muscle modeling in Section 3.2.3, Freifeld and Black [FB12] developed a very similar shape parameterization that also reduces triangle deformations to the minimal 6 degrees of freedom per triangle. They theoretically analyze this parameterization and find that this parameterization induces a manifold structure; fittingly, Freifeld and Black name this parameterization *Lie Bodies*. However, they apply their parameterization only to linear modeling from static laser scan data. Data-based modeling as explored in Chapter 3 maps parameters like external force to a predicted 3D surface deformation. The inverse problem amounts to predicting external force from 3D surface deformations, a process that is studied by Sagawa et al. [SYA+15]. They show promising first results and also compare with EMG measurements, thus providing a first proof-of-concept that muscular effort may be predictable from 3D surface measurements. This is a very interesting direction that begs for further investigation.

My publication of sparse localized deformation components in [NVW+13] directly influenced publications in computer vision and computer graphics, presumably thanks to the reference implementation I open-sourced. Huang et al. [HYZ+14] find sparse localized deformation components within a deformation gradient encoding, similar to the one proposed in Section 3.2.3. This allows modeling curvilinear paths and thus directly tackles the limitation shown in Figure 4.15. However, the hierarchical nature of articulated deformations cannot be correctly modeled with deformation gradients: the component on the upper leg will still keep the lower leg straight because deformation gradients do not respect the inherent kinematic chain of articulated motion sequences. Therefore, it is still an avenue for future work to automatically discover the kinematic chain of a mesh sequence using sparse localized components. My algorithms have also been used for image classification in computer vision by Lee and Chellappa [LC14], who adopt SPLOCS to the setting of facial expression recognition from video data. This shows the wide applicability of the SPLOCS algorithm. Independently, Brunton et al. [BBW14] have developed a multilinear wavelet basis that is also localized, but that locality is fixed to the mesh topology and not adapted like in the SPLOCS algorithm. Bernard et al. [BGH+15] provide a new local deformation model that is similar to SPLOCS. It also models sparse components but enforces smoothness of the components instead of explicitly dictating locality of the deformation components using geodesic support maps (like in SPLOCS). This fixes limitations regarding convergence monitoring. The approach also allows for deformation components of differing size and does not require localized initialization. If smoothness is enforced on the deformation components as in [BGH+15], this can be useful for modeling human body or brain shapes. However,

I argue that this does not apply for modeling complex wrinkles since such wrinkle deformations are not smooth by nature. But the correct modeling of such wrinkles is crucial for high-resolution facial animations. Also, the smooth components proposed by Bernard et al. are not necessarily local, necessitating a post-processing splitting step [BGH+15]. In their paper, Bernard et al. [BGH+15] also provide further evaluation of SPLOCS on different datasets.

One of the main limitations of compressed manifold modes (Chapter 5) is the relatively high computation time compared to the classical manifold harmonic basis. This has been tackled in follow-up publications. Kovnatsky et al. [KGB15] combine non-linear smooth optimization on manifolds with ADMM, which provides a new framework for quicker solution of the CMM optimization problem. Houston [Hou15] shows that momentum ADMM [GOSB14] can be adopted for quicker computation of CMMs. Additionally, Houston introduces compressed *eigenvalues* that induce an ordering on the compressed modes. Recent shape correspondence estimation techniques combine the concept of locality with the power of trainable convolutional neural networks. The framework of Boscaini et al. [BMM+15] allows incorporating any shape basis, including the compressed manifold modes - an idea that needs to be explored in future work.

I hope that my contributions to the domain of reconstruction, analysis, and editing of dynamically deforming 3D surfaces spawn future research in the domain of computer graphics and computer vision. I am curious to see my methods being adopted and extended in exciting new application areas.

## A. Motion Protocol for Capturing Arm Muscle Deformations

370 motions were recorded at 30 Hz from 10 different subjects, each having a length of around 100 frames. After taking out 7 motions for cross validation experiments, approximately 32 000 meshes were left for training. A strict recording protocol of motion was maintained for capturing the dataset. The 10 subjects had to perform the same predefined arm articulations, first without external load, then with external load. Finally, some free motions differing between subjects are recorded.

### Articulations with external load

The subjects were asked to repeat the following motions, first without weight, then with a barbell in hand. The barbell weight was raised in the following way: 0.5 kg, 1.0 kg, 2.0 kg, 4.0 kg, 7.5 kg, 10.0 kg, 14.0 kg, 17.75 kg. Recording stopped when subjects could not lift the weight anymore. I now give a specific description of the motions.

**Arm Flexion in Supination.** This is the classical biceps curl move, the palm of the hand facing upwards. Practices mainly biceps brachii. Figure A.1(a).

**Arm Flexion in Pronation.** The same biceps curl, but with the palm of the hand facing towards the ground. Mainly practices biceps brachii and brachialis muscle. Figure A.1(b).

**Arm Extension.** This motion is targeted at the triceps muscle and starts with the upper body position horizontal to the ground. For this, a bench was put inside the capture volume so that subjects could rest with the left arm on the bench while recording the right arm. The arm is then moved from a flexed position and extended upwards. Figure A.1(c).

**Shoulder Abduction and Adduction.** The extended arm is lifted up within the coronal plane. Practices mainly deltoid and supraspinatus muscle. Figure A.1(d).

**Shoulder Upward Rotation.** While shoulder abduction up to  $90^\circ$  is performed by rotation in the glenohumeral joint, further rotation upwards is performed by a rotation of the scapula and even the spine. This motion was captured separately with the elbow flexed upwards. The exercise used to capture this is called the military press in body building. It exercises deltoid, trapezius, pectoralis major and serratus anterior muscles around the shoulder and in the back. The triceps also requires activation to extend the arm. Figure A.1(e).

### Articulations without external load

The following movements were recorded just once without any external load.

**Shoulder Flex.** The shoulder is lifted upwards within the sagittal plane. Figure A.1(f).

**Shoulder Spin 1.** Rotation of the shoulder is performed while the elbow is flexed. Figure A.1(g).

**Shoulder Spin 2.** Rotation of the shoulder is performed while the elbow is extended. We specifically ask the subjects to exercise the full rotational motion. This is an interesting motion since the observed deformations are hard to generate with classical skinning methods from computer graphics. Figure A.1(h).

**Transverse Extension of the Shoulder.** The shoulder is rotated in the transverse plane while the elbow is flexed. This exposes the scapula and (depending on the body shape) other parts of the glenohumeral joint at the extreme angles. Figure A.1(i).

**Lifting 1 and 2.** This motion performs flexing of the elbow joint and abduction of the shoulder joint at the same time. A similar motion is repeated after rotation of the shoulder, Figure A.1(j) and Figure A.1(k).

### Additional movements

Each subject was asked to perform some free articulated movements. These exercises show extensive flexing of the shoulder, dancing, freestyle articulations of the arm, boxing, and drinking. Usually, these captured sequences are a bit longer (sometimes up to 500 frames). Some of the additional sequences were left out of training and were used for cross validation. Four of the movements are shown in Figure A.2.

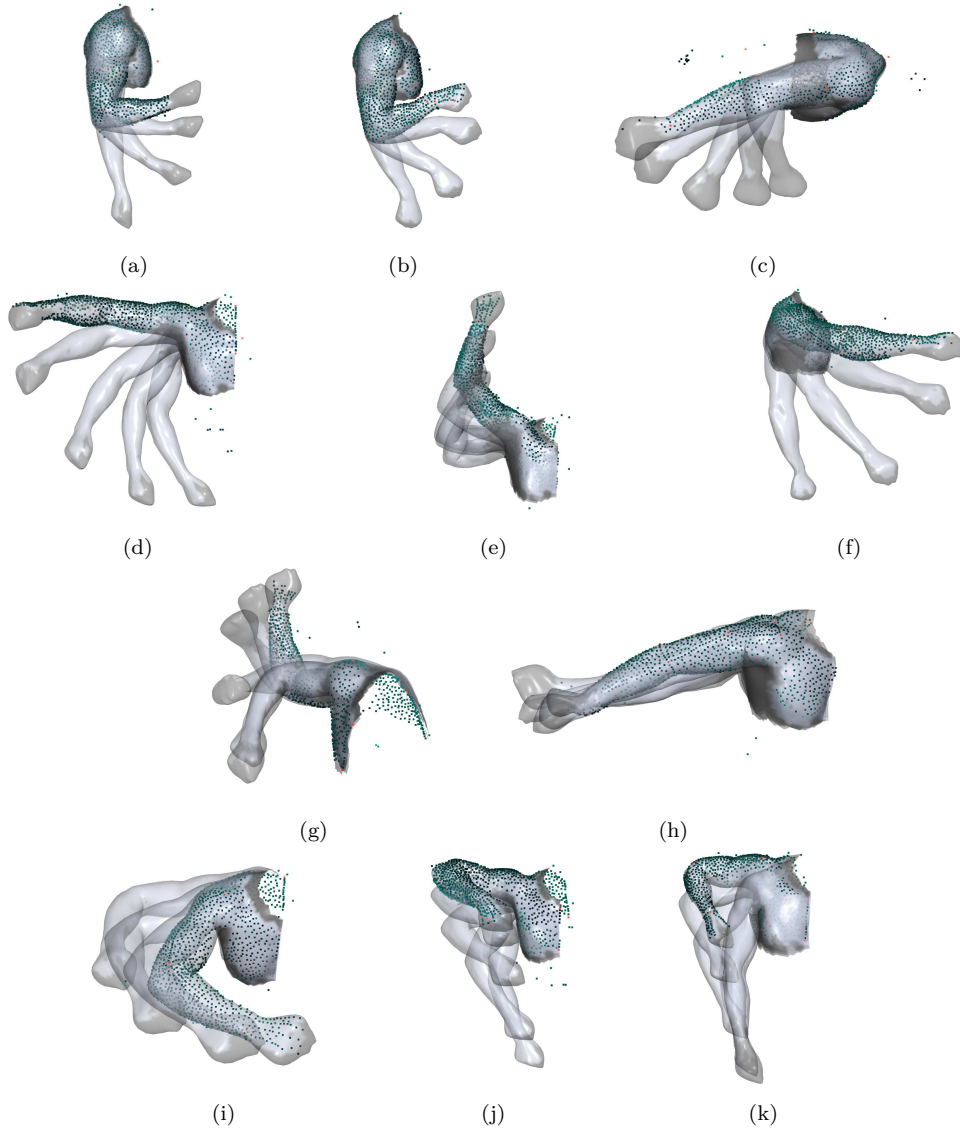


Figure A.1.: Motions recorded in the arm-muscle deformation dataset. This only shows the set of basic motions recorded every subject, additional articulated motions were recording for each subject, some of them shown in Figure A.2.

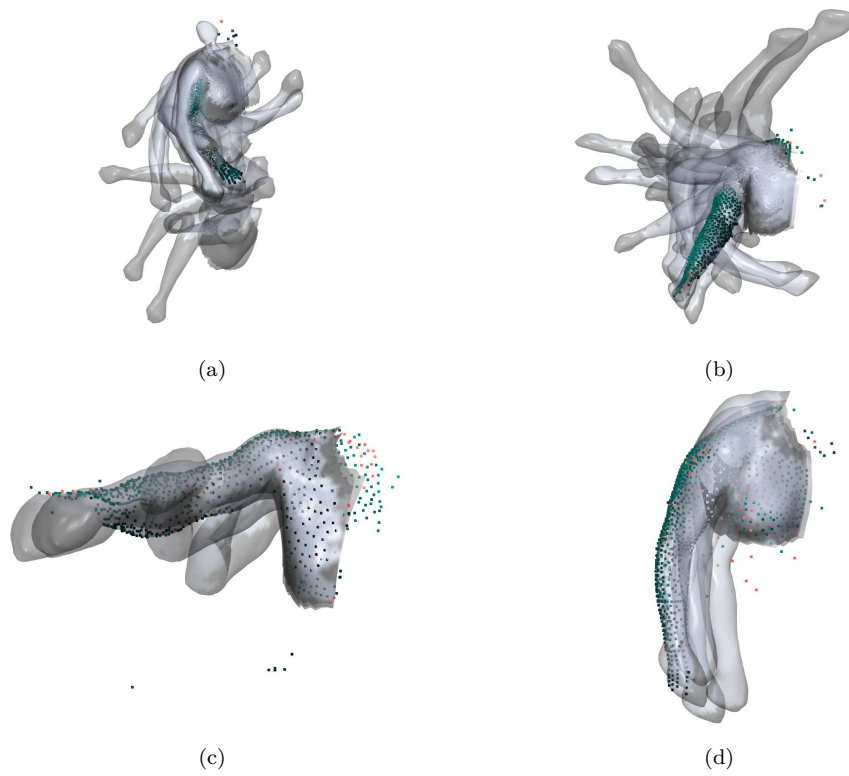


Figure A.2.: Subset of additionally recorded motions, different for each subject. (a) Dance motion, part of training set; (b) Freestyle motion, not part of training set, used for cross validation; (c) Boxing, not part of training set, used for cross validation; (d) shoulder spin, not part of training, used for cross validation

## Bibliography

- [ABK14] Y. Aflalo, A. Bronstein, and R. Kimmel. *Graph matching: relax or not?* 2014. arXiv: 1401.7623. URL: <http://arxiv.org/abs/1401.7623>.
- [ACP02] B. Allen, B. Curless, and Z. Popović. „Articulated body deformation from range scan data“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 21.3 (2002), pp. 612–619.
- [ACP03] B. Allen, B. Curless, and Z. Popović. „The space of human body shapes: reconstruction and parameterization from range scans“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22.3 (2003), pp. 587–594.
- [ACPH06] B. Allen, B. Curless, Z. Popović, and A. Hertzmann. „Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis“. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. 2006, pp. 147–156.
- [Agi] Agisoft. *Photoscan*. URL: <http://www.agisoft.com/> (visited on 04/01/2016).
- [AMS08] P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.
- [ASK+05] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. „SCAPE: shape completion and animation of people“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 24.3 (2005), pp. 408–416.
- [AST+08] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. „Performance capture from sparse multi-view video“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27.3 (2008), 98:1–98:10.
- [ASTH09] E. de Aguiar, L. Sigal, A. Treuille, and J. K. Hodgins. „Stable Spaces for Real-time Clothing“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 29.4 (2009), 106:1–106:9.

- [ATTS08] E de Aguiar, C Theobalt, S Thrun, and H.-P. Seidel. „Automatic Conversion of Mesh Animations into Skeleton-based Animations“. In: *Computer Graphics Forum (Proc. Eurographics)*. Vol. 27. 2. 2008, pp. 389–397.
- [Aut] Autodesk. *123D Catch*. URL: <http://www.123dapp.com/catch> (visited on 04/01/2016).
- [BB08] A. O. Bălan and M. J. Black. „The Naked Truth: Estimating Body Shape Under Clothing“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2008, pp. 15–29.
- [BBH08] D. Bradley, T. Boubekeur, and W. Heidrich. „Accurate multi-view reconstruction using robust binocular stereo and surface meshing“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2008.
- [BBO+09] B. Bickel, M. Bächer, M. A. Otaduy, W. Matusik, H. Pfister, and M. Gross. „Capture and modeling of non-linear heterogeneous soft tissue“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 28.3 (2009).
- [BBW14] A. Brunton, T. Bolkart, and S. Wuhler. „Multilinear wavelets: A statistical shape space for human faces“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2014, pp. 297–312.
- [BCBSG10] M. Ben-Chen, A. Butscher, J. Solomon, and L. J. Guibas. „On Discrete Killing Vector Fields and Patterns on Surfaces“. In: *Computer Graphics Forum* 29.5 (2010), pp. 1701–1711.
- [BCO14] F. Barekat, R. Caflisch, and S. Osher. „On the Support of Compressed Modes“. In: *UCLA CAM Report 14-14* (2014).
- [BCPP98] R. E. Burkard, E. Cela, P. M. Pardalos, and L. S. Pitsoulis. „The Quadratic Assignment Problem“. In: *Handbook of Combinatorial Optimization*. Ed. by D.-Z. Du and P. M. Pardalos. Kluwer, 1998, pp. 241–337.
- [Ber99] D. P. Bertsekas. *Nonlinear Programming*. 2nd ed. Athena Scientific, 1999.
- [BGH+15] F. Bernard, P. Gemmar, F. Hertel, J. Goncalves, and J. Thunberg. *Linear Shape Deformation Models with Local Support Using Graph-based Structured Matrix Factorisation*. 2015. arXiv: 1510.08291. URL: <http://arxiv.org/abs/1510.08291>.
- [BHB+11] T. Beeler, F. Hahn, D. Bradley, B. Bickel, P. Beardsley, C. Gotsman, R. W. Sumner, and M. Gross. „High-quality passive facial performance capture using anchor frames“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 30 (2011).

- 
- [BHV+11] G. J. Brostow, C. Hernández, G. Vogiatzis, B. Stenger, and R. Cipolla. „Video Normals from Colored Lights“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.10 (2011), pp. 2104–2114.
- [Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [BJMO11] F. R. Bach, R. Jenatton, J. Mairal, and G. Obozinski. „Optimization with Sparsity-Inducing Penalties“. In: *Foundations and Trends in Machine Learning* 4.1 (2011), pp. 1–106.
- [BK00] Y. Basar and W. B. Krätzig. *Theory of shell structures*. 2nd ed. VDI, 2000.
- [BKP+10] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, and B. Levy. „Smoothing“. In: *Polygon Mesh Processing*. AK Peters, 2010.
- [BM98] C. Bregler and J. Malik. „Tracking People with Twists and Exponential Maps“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1998, pp. 8–15.
- [BMAS14] N. Boumal, B. Mishra, P.-A. Absil, and R. Sepulchre. „Manopt, a Matlab Toolbox for Optimization on Manifolds“. In: *Journal of Machine Learning Research* 15 (2014), pp. 1455–1459. URL: <http://www.manopt.org>.
- [BMM+15] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. „Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks“. In: *Computer Graphics Forum (Proc. SGP)* 34.5 (2015), pp. 13–23.
- [BMP02] S. Belongie, J. Malik, and J. Puzicha. „Shape Matching and Object Recognition Using Shape Contexts“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.4 (2002), pp. 509–522.
- [BMR+99] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. „The ball-pivoting algorithm for surface reconstruction“. In: *IEEE Transactions on Visualization and Computer Graphics* 5.4 (1999), pp. 349–359.
- [BPC+11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. „Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers“. In: *Foundations and Trends in Machine Learning* 3.1 (2011), pp. 1–122.
- [BPS+08] D. Bradley, T. Popa, A. Sheffer, W. Heidrich, and T. Boubekeur. „Markerless garment capture“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27.3 (2008).

- [Bre67] L. Bregman. „The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming“. In: *USSR Computational Mathematics and Mathematical Physics* 7.3 (1967), pp. 200–217.
- [BRLB14] F. Bogo, J. Romero, M. Loper, and M. J. Black. „FAUST: Dataset and evaluation for 3D mesh registration“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 3794–3801.
- [BRR11] M. Bleyer, C. Rhemann, and C. Rother. „PatchMatch Stereo-Stereo Matching with Slanted Support Windows.“ In: *Proc. British Machine Vision Conference (BMVC)* i.1 (2011), pp. 14.1–14.11.
- [BSL+11] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. „A database and evaluation methodology for optical flow“. In: *International Journal of Computer Vision* 92.1 (2011), pp. 1–31.
- [BSPG06] M. Botsch, R. W. Sumner, M. Pauly, and M. Gross. „Deformation Transfer for Detail-Preserving Surface Editing“. In: *Proc. Vision, Modeling and Visualization (VMV)*. 2006, pp. 357–364.
- [BT09] A. Beck and M. Teboulle. „A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems“. In: *SIAM Journal on Imaging Sciences* 2.1 (2009), pp. 183–202.
- [BTP13] S. Bouaziz, A. Tagliasacchi, and M. Pauly. „Sparse Iterative Closest Point“. In: *Computer Graphics Forum (Proc. SGP)* 32.5 (2013).
- [BTV06] H. Bay, T. Tuytelaars, and L. Van Gool. „SURF: Speeded up robust features“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2006, pp. 404–417.
- [BTZ96] P. Beardsley, P. Torr, and A. Zisserman. „3D model acquisition from extended image sequences“. In: *Proc. European Conference on Computer Vision (ECCV)*. 1996, pp. 683–695.
- [BV99] V. Blanz and T. Vetter. „A morphable model for the synthesis of 3D faces“. In: *Proc. SIGGRAPH*. 1999, pp. 187–194.
- [BW08] J. Bonet and R. D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. 2nd ed. Cambridge University Press, 2008.
- [BWP13] S. Bouaziz, Y. Wang, and M. Pauly. „Online Modeling for Realtime Facial Animation“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 32.4 (2013), 40:1–40:10.

- [CCC+08] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. „MeshLab: an Open-Source Mesh Processing Tool“. In: *Sixth Eurographics Italian Chapter Conference*. 2008, pp. 129–136.
- [CFSV04] D. Conte, P. Foggia, C. Sansone, and M. Vento. „Thirty Years of Graph Matching“. In: *International Journal of Pattern Recognition and Artificial Intelligence* 18.3 (2004), pp. 265–298.
- [CGH14] Y. Chen, L. Guibas, and Q. Huang. „Near-optimal joint object matching via convex relaxation“. In: *Proc. International Conference on Machine Learning (ICML)*. 2014, pp. 100–108. arXiv: 1402.1473.
- [CH12] T. J. Cashman and K. Hormann. „A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape“. In: *Computer Graphics Forum* 31.2 (2012), pp. 735–744.
- [CHYY16] C. Chen, B. He, Y. Ye, and X. Yuan. „The Direct Extension of ADMM for Multi-block Convex Minimization Problems is Not Necessarily Convergent“. In: *Mathematical Programming* 155.1 (2016), pp. 57–79.
- [CJ01] C. Chennubhotla and A. Jepson. „Sparse PCA. Extracting multi-scale structure from data“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)* (2001), pp. 641–647.
- [CL81] Y. Censor and A. Lent. „An iterative row-action method for interval convex programming“. In: *Journal of Optimization Theory and Applications* 34.3 (1981), pp. 321–353.
- [CLL10] M. Cho, J. Lee, and K. M. Lee. „Reweighted Random Walks for Graph Matching“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2010, pp. 492–505.
- [CSFP07] Y. Cao, A. Shapiro, P. Faloutsos, and F. Pighin. „Motion editing with independent component analysis“. In: *Visual Computer* (2007).
- [CSS+16] M. Colaianni, C. Siegl, J. Süßmuth, F. Rott, and G. Greiner. „Shape Adaptive Cut Lines“. In: *Eurographics Workshop on Graphics for Digital Fabrication*. 2016.
- [Cut13] M. Cuturi. „Sinkhorn distances: Lightspeed computation of optimal transport“. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013. arXiv: arXiv:1306.0895v1.

- [CVHC08] N. D. F. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. „Using multiple hypotheses to improve depth-maps for multi-view stereo“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2008, pp. 766–779.
- [CWW13] K. Crane, C. Weischedel, and M. Wardetzky. „Geodesics in Heat: A New Approach to Computing Distance Based on Heat Flow“. In: *ACM Transactions on Graphics* 32.5 (2013), 152:1–152:11.
- [CZ97] Y. A. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms and Applications*. Oxford University Press, 1997.
- [DAA+07] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen. „OpenSim: Open-Source Software to Create and Analyze Dynamic Simulations of Movement“. In: *Biomedical Engineering, IEEE Transactions on* 54.11 (2007), pp. 1940–1950.
- [DBD+13] B. Deng, S. Bouaziz, M. Deuss, J. Zhang, Y. Schwartzburg, and M. Pauly. „Exploring Local Modifications for Constrained Meshes“. In: *Computer Graphics Forum (Proc. Eurographics)* 32.2.1 (2013), pp. 11–20.
- [DD04] H. Daumé III and H. Daum. *From Zero to Reproducing Kernel Hilbert Spaces in Twelve Pages or Less*. 2004. URL: <http://pub.hal3.name/#daume04rkhs>.
- [EAJ+15] A. Elhayek, E. Aguiar, A. Jain, J. Tompson, L. Pishchulin, M. Andriluka, C. Bregler, B. Schiele, and C. Theobalt. „Efficient ConvNet-based Marker-less Motion Capture in General Scenes with a Low Number of Cameras“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3810–3818.
- [EKB14] D. Eynard, A. Kovnatsky, and M. M. Bronstein. „Laplacian colormaps: a framework for structure-preserving color transformations“. In: *Computer Graphics Forum* 33.2 (2014), pp. 215–224.
- [Eno08] R. M. Enoka. *Neuromechanics of Human Movement*. 4th ed. 2008. Chap. 3, p. 122.
- [EY36] C. Eckart and G. Young. „The approximation of one matrix by another of lower rank“. In: *Psychometrika* 1.3 (1936), pp. 211–218.
- [FB11] S. Fröhlich and M. Botsch. „Example-Driven Deformations Based on Discrete Shells.“ In: *Computer Graphics Forum (Proc. SGP)* 30.8 (2011), pp. 2246–2257.
- [FB12] O. Freifeld and M. J. Black. „Lie Bodies: A Manifold Representation of 3D Human Shape“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2012.

- [FH04] P. Felzenszwalb and D. Huttenlocher. „Efficient Belief Propagation for Early Vision“. In: *International Journal of Computer Vision* 70.1 (2004), pp. 41–54.
- [FK98] O. Faugeras and R. Keriven. „Variational Principles , Surface Evolution , PDE ’ s , Level Set Methods , and the Stereo Problem“. In: *Transactions on Image Processing* 7.3 (1998), pp. 336–344.
- [FKY08] W.-W. Feng, B.-U. Kim, and Y. Yu. „Real-time data driven deformation using kernel canonical correlation analysis“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27.3 (2008), 91:1–91:9.
- [FP08] Y. Furukawa and J. Ponce. „Accurate , Dense , and Robust Multi-View Stereopsis“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.8 (2008), pp. 1362–1376.
- [FPV14] P. Foggia, G. Percannella, and M. Vento. „Graph Matching and Learning in Pattern Recognition in the Last 10 Years“. In: *International Journal of Pattern Recognition and Artificial Intelligence* 28.1 (2014).
- [FTG03] V. Ferrari, T. Tuytelaars, and L. V. Gool. „Wide-baseline Multiple-view Correspondences“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2003, pp. 718–725.
- [FTV00] A. Fusiello, E. Trucco, and A. Verri. „Compact algorithm for rectification of stereo pairs“. In: *Machine Vision and Applications* 12.1 (2000), pp. 16–22.
- [Gal86] Z. Galil. „Efficient Algorithms for Finding Maximum Matching in Graphs“. In: *ACM Computing Surveys (CSUR)* 18.1 (1986), pp. 23–38.
- [GGV08] F. de Goes, S. Goldenstein, and L. Velho. „A Hierarchical Segmentation of Articulated Bodies“. In: *Proc. Eurographics Symposium on Geometry Processing SGP*. 2008, pp. 1349–1356.
- [GGWG12] T. Gevers, A. Gijzenij, J. van de Weijer, and J.-M. Geusebroek. „Pixel-Based Photometric Invariance“. In: *Color in Computer Vision*. 2012, pp. 47–68.
- [GKB03] I. Guskov, S. Klivanov, and B. Benjamin. „Trackable Surfaces“. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2003), pp. 251–257.
- [GM04] B. Goldluecke and M. Magnor. „Space-time isosurface evolution for temporally coherent 3D reconstruction“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2004, pp. 350–355.

- [GOSB14] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. „Fast Alternating Direction Optimization Methods“. In: *SIAM Journal on Imaging Sciences* 7.3 (2014), pp. 1588–1623.
- [GR96] S. Gold and A. Rangarajan. „A Graduated Assignment Algorithm for Graph Matching“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18.4 (1996), pp. 377–388.
- [GRH+12] P. Guan, L. Reiss, D. Hirshberg, A. Weiss, and M. J. Black. „DRAPE: DRessing Any PErson“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31.4 (2012), 35:1–35:10.
- [GS15] S. Galliani and K. Schindler. „Massively Parallel Multiview Stereopsis by Surface Normal Diffusion“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2015.
- [GSD+09] J. Gall, C. Stoll, E. De Aguiar, C. Theobalt, B. Rosenhahn, and H. P. Seidel. „Motion capture using joint skeleton tracking and surface estimation“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 1746–1753.
- [GTKK13] M. Grzegorzec, C. Theobalt, R. Koch, and A. Kolb. *Time-of-Flight and Depth Imaging. Sensors, Algorithms and Applications: Dagstuhl Seminar 2012 and GCPR Workshop on Imaging New Modalities*. Vol. 8200. Springer, 2013.
- [HAR+06] N. Hasler, M. Asbach, B. Rosenhahn, J. Ohm, and H. Seidel. „Physically based tracking of cloth“. In: *Proc. Vision, Modeling and Visualization (VMV)*. 2006, pp. 49–56.
- [HAR+10] N. Hasler, H. Ackermann, B. Rosenhahn, T. Thormahlen, and H.-P. Seidel. „Multilinear pose and body shape estimation of dressed subjects from image sets“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 1823–1830.
- [Har76] H. H. Harman. *Modern Factor Analysis*. 3rd ed. University Of Chicago Press, 1976.
- [Has14] N. Hasler. „Motion Capture“. In: *Computer Vision: A Reference Guide*. Ed. by K. Ikeuchi. Springer US, 2014, pp. 495–498.
- [Hau04] M. Hauth. „Visual simulation of deformable models“. PhD thesis. Eberhard Karls University of Tübingen, 2004.

- [Hav06] P. Havaldar. „Performance driven facial animation“. In: *ACM SIGGRAPH Course Notes*. 2006.
- [HG13] Q. X. Huang and L. Guibas. „Consistent shape maps via semidefinite programming“. In: *Computer Graphics Forum (Proc. SGP)* 32.5 (2013), pp. 177–186.
- [Hir05] H. H. Hirschmüller. „Accurate and efficient stereo processing by semi-global matching and mutual information“. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* 2 (2005), pp. 807–814.
- [HKO01] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. 2001.
- [HLP10] K. Hawick, A. Leist, and D. Playne. „Parallel graph component labelling with GPUs and CUDA“. In: *Parallel Computing* 36.12 (2010), pp. 655–678.
- [Hou15] K. Houston. *Compressed Manifold Modes: Fast Calculation and Natural Ordering*. 2015. arXiv: 1507.00644. URL: <http://arxiv.org/abs/1507.00644>.
- [HPH10] Q. Y. Hong, S. I. Park, and J. K. Hodgins. „A Data-driven Segmentation for the Shoulder Complex“. In: *Computer Graphics Forum (Proc. Eurographics)* 29.2 (2010), pp. 537–544.
- [HRT+09] N. Hasler, B. Rosenhahn, T. Thormählen, M. Wand, J. Gall, and H. P. Seidel. „Marker-less motion capture with unsynchronized moving cameras“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009), pp. 224–231.
- [HSS+09] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel. „A Statistical Model of Human Pose and Body Shape“. In: *Computer Graphics Forum (Proc. Eurographics)*. Vol. 28. 2. 2009, pp. 337–346.
- [HTRS10] N. Hasler, T. Thormählen, B. Rosenhahn, and H.-P. Seidel. „Learning skeletons for shape and pose“. In: *Proc. ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*. 2010, pp. 23–30.
- [HYZ+14] Z. Huang, J. Yao, Z. Zhong, Y. Liu, and X. Guo. „Sparse Localized Decomposition of Deformation Gradients“. In: *Computer Graphics Forum (Proc. Pacific Graphics)* 33.7 (2014), pp. 239–248.
- [HZ04] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. Cambridge University Press, 2004.

- [JBK+12] A. Jacobson, I. Baran, L. Kavan, J. Popović, and O. Sorkine. „Fast Automatic Skinning Transformations“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31.4 (2012), 77:1–77:10.
- [JDKL14] A. Jacobson, Z. Deng, L. Kavan, and J. P. Lewis. „Skinning: Real-time Shape Deformation“. In: *ACM SIGGRAPH 2014 Courses*. 2014.
- [JDV+14] R. Jensen, A. Dahl, G. Vogiatzis, E. Tola, and H. Aanaes. „Large scale multi-view stereopsis evaluation“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2014), pp. 406–413.
- [Jen11] R. Jenatton. „Structured Sparsity-Inducing Norms: Statistical and Algorithmic Properties with Applications to Neuroimaging“. PhD thesis. École Normale Supérieure Cachan, 2011.
- [JH99] A. E. Johnson and M. Hebert. „Using spin images for efficient object recognition in cluttered 3D scenes“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.5 (1999), pp. 433–449.
- [JTU03] I. T. Jolliffe, N. T. Trendafilov, and M. Uddin. „A Modified Principal Component Technique Based on the LASSO“. In: *Journal of Computational and Graphical Statistics* 12.3 (2003), pp. 531–547.
- [JWZ13] P. Jia, Y. Wang, and Y. Zhang. „Improvement in the independence of relaxation method-based particle tracking velocimetry“. In: *Measurement Science and Technology* 24.5 (2013), p. 055301. URL: <http://stacks.iop.org/0957-0233/24/i=5/a=055301?key=crossref.9781d971312ab8c1c9fc018578ddf64e>.
- [KC05] S. Keerthi and W. Chu. „A matching pursuit approach to sparse Gaussian process regression“. In: *Advances in Neural Information Processing Systems (NIPS)* (2005), pp. 643–650.
- [KCZO08] L. Kavan, S. Collins, J. Zara, and C. O’Sullivan. „Geometric Skinning with Approximate Dual Quaternion Blending“. In: *ACM Transactions on Graphics* 27.4 (2008), 105:1–105:23.
- [KG00] Z. Karni and C. Gotsman. „Spectral Compression of Mesh Geometry“. In: *Proc. SIGGRAPH*. 2000, pp. 279–286.
- [KG08] S. Kircher and M. Garland. „Free-form motion processing“. In: *ACM Transactions on Graphics* 27.2 (2008), 12:1–12:13.

- 
- [KGB15] A. Kovnatsky, K. Glashoff, and M. M. Bronstein. *MADMM: a generic algorithm for non-smooth optimization on manifolds*. 2015. arXiv: 1505.07676. URL: <http://arxiv.org/abs/1505.07676>.
- [KGBS11] L. Kavan, D. Gerszewski, A. Bargteil, and P.-P. Sloan. „Physics-inspired Upsampling for Cloth Simulation in Games“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 30.4 (2011), 93:1–93:9.
- [KH13] M. Kazhdan and H. Hoppe. „Screened poisson surface reconstruction“. In: *ACM Transactions on Graphics* 32.3 (2013), 29:1–29:13.
- [KJP00] P. G. Kry, D. L. James, and D. K. Pai. „EigenSkin : Real Time Large Deformation Character Skinning in Hardware“. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. 2000, pp. 153–160.
- [KK08] K. I. Kim and Y. Kwon. „Example-Based Learning for Single-Image Super-Resolution“. In: *Proc. DAGM Symposium on Pattern Recognition*. 2008, pp. 456–465.
- [KK10] K. I. Kim and Y. Kwon. „Single-Image Super-Resolution Using Sparse Regression and Natural Image Prior“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.6 (2010), pp. 1127–1133.
- [KKBC09] K. Kolev, M. Klodt, T. Brox, and D. Cremers. „Continuous global optimization in multiview 3D reconstruction“. In: *International Journal of Computer Vision* 84.1 (2009), pp. 80–96.
- [KKBL15] I. Kezurer, S. Z. Kovalsky, R. Basri, and Y. Lipman. „Tight Relaxation of Quadratic Matching“. In: *Computer Graphics Forum* 34.5 (2015), pp. 115–128.
- [KKN+13] D. Kim, W. Koh, R. Narain, K. Fatahalian, A. Treuille, and J. F. O’Brien. „Near-exhaustive Precomputation of Secondary Cloth Effects“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 32.4 (2013), 87:1–87:8.
- [KKSS00] K. N. Kutulakos, K. N. Kutulakos, S. M. Seitz, and S. M. Seitz. „A theory of shape by space-carving“. In: *International Journal of Computer Vision* 38.3 (2000), pp. 197–216.
- [KLM10] F. Klose, C. Lipski, and M. Magnor. „Reconstructing shape and motion from asynchronous cameras“. In: *Proc. Vision, Modeling and Visualization (VMV)* (2010), pp. 171–177.
- [Kra07] K. Kraus. *Photogrammetry. Geometry from images and laser scans*. 2nd ed. de Gruyter, 2007.

- [KSD09] B. Kulis, M. a. Sustik, and I. S. Dhillon. „Low-Rank Kernel Learning with Bregman Matrix Divergences“. In: *Journal of Machine Learning Research* 10 (2009), pp. 341–376.
- [KSO10] L. Kavan, P.-P. Sloan, and C. O’Sullivan. „Fast and Efficient Skinning of Animated Meshes“. In: *Computer Graphics Forum* 29.2 (2010), pp. 327–336.
- [KZ01] V. Kolmogorov and R. Zabih. „Computing Visual Correspondence with Occlusions via Graph Cuts“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2001, pp. 508–515.
- [LA10] J. Lewis and K. Anjyo. „Direct-manipulation blendshapes“. In: *IEEE Computer Graphics and Applications* 30.4 (2010), pp. 42–50.
- [LC04] D. Liu and T. Chen. „Soft shape context for iterative closest point registration“. In: *International Conference on Image Processing (ICIP)*. Vol. 2. 2004, pp. 1081–1084.
- [LC14] C. S. Lee and R. Chellappa. „Sparse localized facial motion dictionary learning for facial expression recognition“. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 3548–3552.
- [LCF00] J. P. Lewis, M. Cordner, and N. Fong. „Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation“. In: *ACM Transactions on Graphics*. 2000, pp. 165–172.
- [LCXS09] M. Lau, J. Chai, Y.-Q. Xu, and H.-Y. Shum. „Face poser: Interactive modeling of 3D facial expressions using facial priors“. In: *ACM Transactions on Graphics* 29.1 (2009), 3:1–3:17.
- [LD12] B. Le and Z. Deng. „Smooth Skinning Decomposition with Rigid Bones“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 31.6 (2012), 199:1–199:10.
- [LFF+14] V. Lyzinski, D. Fishkind, M. Fiori, J. T. Vogelstein, C. E. Priebe, and G. Sapiro. *Graph matching: Relax at your own risk*. 2014. arXiv: 1405.3133. URL: <http://arxiv.org/abs/1405.3133>.
- [LGK+12] D. Lee, M. Glueck, A. Khan, E. Fiume, and K. Jackson. „Modeling and Simulation of Skeletal Muscle for Computer Graphics: A Survey“. In: *Foundations and Trends in Computer Graphics and Vision* 7.4 (2012), pp. 229–276.

- 
- [LH05] M. Leordeanu and M. Hebert. „A Spectral Technique for Correspondence Problems Using Pairwise Constraints“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2005, 1482–1489 Vol. 2.
- [LHS09] M. Leordeanu, M. Hebert, and R. Sukthankar. „An Integer Projected Fixed Point Method for Graph Matching and MAP Inference.“ In: *Advances in Neural Information Processing Systems (NIPS)*. 2009, pp. 1114–1122.
- [Lin07] T. Lindeberg. „Scale-Space“. In: *Wiley Encyclopedia of Computer Science and Engineering*. 2007.
- [Lin14] T. Lindeberg. „Scale Selection“. English. In: *Computer Vision: A Reference Guide*. Ed. by K. Ikeuchi. Springer US, 2014, pp. 701–713.
- [Lin94] T. Lindeberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.
- [LKM14] C. Lipski, F. Klose, and M. Magnor. „Correspondence and Depth-Image Based Rendering: a Hybrid Approach for Free-Viewpoint Video“. In: *IEEE Trans. Circuits and Systems for Video Technology (T-CSVT)* 24.6 (2014), pp. 942–951.
- [LL11] J. Lee and K. M. Lee. „Hyper-graph Matching via Reweighted Random Walks“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 1633–1640.
- [LLN+10] C. Lipski, C. Linz, T. Neumann, M. Wacker, and M. Magnor. „High resolution image correspondences for video post-production“. In: *Proc. Conference on Visual Media Production (CVMP)* (2010), pp. 33–39.
- [LLN+12] C. Lipski, C. Linz, T. Neumann, M. Wacker, and M. Magnor. „High Resolution Image Correspondences for Video Post-Production“. In: *Journal of Virtual Reality and Broadcasting (JVRB)* 9(2012).8 (2012).
- [LLO14] R. Lai, J. Lu, and S. Osher. *Density matrix minimization with  $\ell_1$  regularization*. 2014. arXiv: 1403.1525. URL: <http://arxiv.org/abs/1403.1525>.
- [LO14] R. Lai and S. Osher. „A Splitting Method for Orthogonality Constrained Problems“. In: *Journal of Scientific Computing* 58.2 (2014), pp. 431–449.
- [Low04] D. G. Lowe. „Distinctive image features from scale invariant keypoints“. In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

- [LS99] D. D. Lee and H. S. Seung. „Learning the parts of objects by non-negative matrix factorization“. In: *Nature* 401 (1999), pp. 788–791.
- [LST09] S.-H. Lee, E. Sifakis, and D. Terzopoulos. „Comprehensive biomechanical modeling and simulation of the upper body“. In: *ACM Transactions on Graphics* 28.4 (2009), 99:1–99:17.
- [LWH+12] S. Levine, J. M. Wang, A. Haraux, Z. Popović, and V. Koltun. „Continuous Character Control with Low-Dimensional Embeddings“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31.4 (2012), 28:1–28:10.
- [LWP10] H. Li, T. Weise, and M. Pauly. „Example-based facial rigging“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 29.3 (2010), 32:1–32:6.
- [LZ10] B. Levy and R. H. Zhang. „Spectral Geometry Processing“. In: *ACM SIGGRAPH Course Notes*. 2010.
- [MA07] M. Meyer and J. Anderson. „Key Point Subspace Acceleration and soft caching“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 26.3 (2007).
- [Maa92] H.-G. Maas. „Complexity analysis for the establishment of image correspondences of dense spatial target fields“. In: *International Archives of Photogrammetry and Remote Sensing* 29 (1992), pp. 102–107.
- [Maa97] H.-G. Maas. „Mehrbildtechniken in der digitalen Photogrammetrie“. Habilitation Thesis. ETH Zurich, 1997.
- [Mac09] L. Mackey. „Deflation methods for sparse PCA“. In: *Advances in Neural Information Processing Systems (NIPS)*. 2009, pp. 1017–1024.
- [MAW+07] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J. M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. „Real-time visibility-based fusion of depth maps“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)* (2007).
- [MBPS09] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. „Online dictionary learning for sparse coding“. In: *Proc. International Conference on Machine Learning (ICML)*. ACM, 2009, pp. 689–696.
- [MBT+12] E. Miguel, D. Bradley, B. Thomaszewski, B. Bickel, W. Matusik, M. A. Otaduy, and S. Marschner. „Data-Driven Estimation of Cloth Simulation Models“. In: *Computer Graphics Forum (Proc. of Eurographics)* 31.2 (2012), pp. 519–528.

- [MDSB02] M. Meyer, M. Desbrun, P. Schröder, and A. H. Barr. „Discrete Differential-Geometry Operators for Triangulated 2-Manifolds“. In: *Visualization and Mathematics III*. 2002, pp. 35–57.
- [MG03] A. Mohr and M. Gleicher. „Building efficient, accurate character skins from examples“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22.2 (2003), pp. 562–568.
- [MGSHT15] M. Magnor, O. Grau, O. Sorkine-Hornung, and C. Theobalt, eds. *Digital Representations of the Real World: How to Capture, Model, and Render Visual Reality*. A K Peters/CRC Press, 2015.
- [MJOB11] J. Mairal, R. Jenatton, G. Obozinski, and F. Bach. „Convex and Network Flow Optimization for Structured Sparsity“. In: *Journal of Machine Learning Research* 12 (2011), pp. 2681–2720.
- [MLS94] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. 1st ed. CRC Press, 1994.
- [MN10] W. Murray and K.-M. Ng. „An algorithm for nonlinear optimization problems with binary variables“. In: *Computational Optimization and Applications* 47.2 (2010), pp. 257–288.
- [MOC+98] R. A. Morano, C. Ozturk, R. Conn, S. Dubin, S. Zietz, and J. Nissanov. „Structured Light Using Pseudorandom Codes“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.3 (1998), pp. 322–327.
- [MRB03] T. Melzer, M. Reiter, and H. Bischof. „Appearance models based on kernel canonical correlation analysis“. In: *Pattern Recognition* 36.9 (2003), pp. 1961–1971.
- [MS02] K. Mikolajczyk and C. Schmid. „An Affine Invariant Interest Point Detector“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2002, pp. 128–142.
- [MS04] K. Mikolajczyk and C. Schmid. „Scale & affine invariant interest point detectors“. In: *International Journal of Computer Vision* 60.1 (2004), pp. 63–86.
- [MTAD08] P. Mullen, Y. Tong, P. Alliez, and M. Desbrun. „Spectral Conformal Parameterization“. In: *Computer Graphics Forum* 27.5 (2008), pp. 1487–1494.
- [MTLT88] N. Magnenat-Thalmann, R. Laperrire, and D. Thalmann. „Joint-dependent local deformations for hand animation and object grasping“. In: *Proceedings on Graphics Interface* (1988), pp. 26–33.

- [Neu09] T. Neumann. „Deformation Analysis of Tight-Fit Clothing from Textured Body Scans“. Diploma Thesis. HTW Dresden, 2009.
- [NVH+13] T. Neumann, K. Varanasi, N. Hasler, M. Wacker, M. Magnor, and C. Theobalt. „Capture and statistical modeling of arm-muscle deformations“. In: *Computer Graphics Forum (Proc. of Eurographics)* 32.2 (2013), pp. 285–294.
- [NVT+14] T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker. „Compressed Manifold Modes for Mesh Processing“. In: *Computer Graphics Forum (Proc. of Symposium on Geometry Processing SGP)* 33.5 (2014), pp. 35–44.
- [NVW+13] T. Neumann, K. Varanasi, S. Wenger, M. Wacker, M. Magnor, and C. Theobalt. „Sparse Localized Deformation Components“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 32.6 (2013), 179:1–179:10.
- [NWV+12] T. Neumann, M. Wacker, K. Varanasi, C. Theobalt, and M. Magnor. „High Detail Marker based 3D Reconstruction by Enforcing Multiview Constraints“. In: *ACM SIGGRAPH 2012 Posters*. 2012, p. 59.
- [OBCS+12] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. „Functional Maps: A Flexible Representation of Maps Between Shapes“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 31.4 (2012), 30:1–30:11.
- [OF97] B. Olshausen and D. Field. „Sparse coding with an overcomplete basis set: a strategy employed by V1?“ In: *Vision Research* 37.23 (1997), pp. 3311–3325.
- [OLCO13a] V Ozoliņš, R Lai, R Ciflis, and S Osher. „Compressed Modes for Variational Problems in Mathematics and Physics“. In: *Proc. National Academy of Sciences* 110.46 (2013), pp. 18368–18373. arXiv: 1308.1758.
- [OLCO13b] V Ozoliņš, R Lai, R Ciflis, and S Osher. „Compressed plane waves yield a compactly supported multiresolution basis for the Laplace operator.“ In: *Proc. National Academy of Sciences* 111 (2013), pp. 1691–1696.
- [Osi03] J. Osipa. *Stop Staring: Facial modeling and animation done right*. Sybex, 2003.
- [PAXG09] B. Pan, A. Asundi, H. Xie, and J. Gao. „Digital image correlation using iterative least squares and pointwise least squares for displacement field and strain field measurements“. In: *Optics and Lasers in Engineering* 47.7-8 (2009), pp. 865–874.
- [PB14] N. Parikh and S. Boyd. „Proximal Algorithms“. In: *Foundations and Trends in Optimization* 1.3 (2014), pp. 123–231.

- [PBB+12] J. Pokrass, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro. „Sparse Modeling of Intrinsic Correspondences“. In: *Computer Graphics Forum (Proc. Eurographics)* 32.2 (2012), pp. 459–468. arXiv: 1209.6560.
- [PGF+16] L. Pogrzeba, S. Gassel, J. Friedrich, M. Klingner, F. Lischke, S. Hellbach, T. Neumann, H.-J. Böhme, and M. Wacker. „Towards estimation of muscle activity patterns for balance assessment from surface deformations in motion“. In: *Poster at International Symposium on the Neuromechanics of Human Movement*. 2016, to appear.
- [PH03] D. Pritchard and W. Heidrich. „Cloth motion capture“. In: *Computer Graphics Forum (Proc. Eurographics)* 22.3 (2003), pp. 263–271.
- [PH06] S. I. Park and J. K. Hodgins. „Capturing and animating skin deformation in human motion“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 25.3 (2006), pp. 881–889.
- [PH08] S. I. Park and J. K. Hodgins. „Data-driven modeling of skin and muscle deformation“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27.3 (2008), 96:1–96:6.
- [PKF07] J. P. Pons, R. Keriven, and O. Faugeras. „Multi-view stereo reconstruction and scene flow estimation with a global image-based matching score“. In: *International Journal of Computer Vision* 72.2 (2007), pp. 179–193.
- [PKS13] D. Pachauri, R. Kondor, and V. Singh. „Solving the multi-way matching problem by permutation synchronization“. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 1860–1868.
- [PMRMB15] G. Pons-Moll, J. Romero, N. Mahmood, and M. J. Black. „Dyna: A Model of Dynamic Human Shape in Motion“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 34.4 (2015), 120:1–120:14.
- [PP12] K. B. Petersen and M. S. Pedersen. *The Matrix Cookbook*. 2012. URL: <http://www2.imm.dtu.dk/pubdb/p.php?3274>.
- [PZB+09] T. Popa, Q. Zhou, D. Bradley, V. Kraevoy, H. Fu, A. Sheffer, and W. Heidrich. „Wrinkling Captured Garments Using Space-Time Data-Driven Deformation“. In: *Computer Graphics Forum (Proc. Eurographics)* 28.2 (2009), pp. 427–435.
- [RDHT14] N. Robertini, E. De Aguiar, T. Helten, and C. Theobalt. „Efficient Multi-view Performance Capture of Fine-Scale Surface Detail“. In: *International Conference on 3D Vision (3DV)*. 2014. arXiv: 1602.02023.

- [REH+15] K. Ruhl, M. Eisemann, A. Hilsmann, P. Eisert, and M. Magnor. „Interactive Scene Flow Editing for Improved Image-based Rendering and Virtual Spacetime Navigation“. In: *Proc. ACM Multimedia*. 2015, pp. 631–640.
- [RGPS05] P. Ruhnau, C. Guetter, T. Putze, and C. Schnörr. „A variational approach for particle tracking velocimetry“. In: *Measurement Science and Technology* 16.7 (2005), pp. 1449–1458. URL: <http://iopscience.iop.org/article/10.1088/0957-0233/16/7/007>.
- [RHB+11] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. „Fast Cost-Volume Filtering for Visual Correspondence and Beyond“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011, pp. 3017–3024.
- [RNVT13] N. Robertini, T. Neumann, K. Varanasi, and C. Theobalt. „Capture of Arm-Muscle Deformations using a Depth-Camera“. In: *Proc. European Conference on Visual Media Production (CVMP)*. 2013, 12:1–12:10.
- [RNWM11] L. Rogge, T. Neumann, M. Wacker, and M. Magnor. „Monocular Pose Reconstruction for an Augmented Reality Clothing System“. In: *Proc. Vision, Modeling and Visualization (VMV)*. 2011, pp. 339–346.
- [RRR+16] H. Rhodin, N. Robertini, C. Richardt, H.-P. Seidel, and C. Theobalt. „A Versatile Scene Model with Differentiable Visibility Applied to Generative Pose Estimation“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2016. arXiv: 1602.03725.
- [Rus07] R. M. Rustamov. „Laplace-Beltrami Eigenfunctions for Deformation Invariant Shape Representation“. In: *Proc. Eurographics Symposium on Geometry Processing SGP*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2007, pp. 225–233.
- [Rus11] R. M. Rustamov. „Multiscale Biharmonic Kernels“. In: *Computer Graphics Forum (Proc. Symposium on Geometry Processing SGP)* 30.5 (2011), pp. 1521–1531.
- [SA07] O. Sorkine and M. Alexa. „As-rigid-as-possible surface modeling“. In: *Proc. Eurographics Symposium on Geometry Processing SGP*. 2007, pp. 109–116.
- [SBB10] L. Sigal, A. Balan, and M. J. Black. „HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion“. In: *International Journal of Computer Vision* 87.1 (2010), pp. 4–27.
- [SBCBG11] J. Solomon, M. Ben-Chen, A. Butscher, and L. Guibas. „Discovery of Intrinsic Primitives on Triangle Meshes“. In: *Computer Graphics Forum (Proc. Eurographics)* 30.2 (2011), pp. 365–374.

- 
- [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. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2006), pp. 519–528. URL: <http://vision.middlebury.edu/mview/>.
- [SCOL+04] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. „Laplacian Surface Editing“. In: *Proc. Eurographics / ACM SIGGRAPH Symposium on Geometry Processing SGP*. 2004, pp. 179–188.
- [SGA+10] C. Stoll, J. Gall, E. Aguiar, S. Thrun, and C. Theobalt. „Video-based reconstruction of animatable human characters“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 29.6 (2010), 139:1–139:10.
- [SH07] J. Starck and A. Hilton. „Surface capture for performance-based animation“. In: *IEEE Computer Graphics and Applications* 27.3 (2007), pp. 21–31.
- [SHD92] K. Shoemake, M. Hill, and T. Duff. „Matrix Animation and Polar Decomposition“. In: *Proceedings of Graphics Interface*. 1992, pp. 258–264.
- [SHG+11] C. Stoll, N. Hasler, J. Gall, H.-P. Seidel, and C. Theobalt. „Fast articulated motion tracking using a sums of Gaussians body model“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2011, pp. 951–958.
- [SILN11] J. Seo, G. Irving, J. P. Lewis, and J. Noh. „Compression and direct manipulation of complex blendshape models“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 30.6 (2011), 164:1–164:10.
- [Sin14] S. N. Sinha. „Multiview Stereo“. In: *Computer Vision: A Reference Guide*. Ed. by K. Ikeuchi. Springer US, 2014, pp. 516–522.
- [SKP08] S. Sueda, A. Kaufman, and D. K. Pai. „Musculotendon Simulation for Hand Animation“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27.3 (2008), 83:1–83:8.
- [SKR+06] C. Stoll, Z. Karni, C. Rössl, H. Yamauchi, and H.-P. Seidel. „Template Deformation for Point Cloud Fitting“. In: *Proc. Point Based Graphics*. 2006, pp. 27–35.
- [SM04] V. Scholz and M. Magnor. „Cloth motion from optical flow“. In: *Proc. Vision, Modeling and Visualization (VMV)* (2004), pp. 117–124.
- [SMP03] P. Sand, L. McMillan, and J. Popović. „Continuous capture of skin deformation“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 22.3 (2003), pp. 578–586.

- [SNF05] E. Sifakis, I. Neverov, and R. Fedkiw. „Automatic determination of facial muscle activations from sparse motion capture marker data“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 24(3) (2005), pp. 417–425.
- [SOG09] J. Sun, M. Ovsjanikov, and L. Guibas. „A Concise and Provably Informative Multi-scale Signature Based on Heat Diffusion“. In: *Proc. Eurographics Symposium on Geometry Processing SGP* 28.5 (2009), pp. 1383–1392.
- [SOS09] H. Schreier, J. J. Orteu, and M. A. Sutton. *Image correlation for shape, motion and deformation measurements: Basic concepts, theory and applications*. Springer US, 2009.
- [SP04] R. W. Sumner and J. Popović. „Deformation transfer for triangle meshes“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 23.3 (2004), pp. 399–405.
- [SRC01] P.-P. J. Sloan, C. F. Rose III, and M. F. Cohen. „Shape by example“. In: *Proceedings of the 2001 symposium on Interactive 3D graphics (I3D)*. 2001, pp. 135–143.
- [SRM12] A. Sellent, K. Ruhl, and M. Magnor. „A loop-consistency measure for dense correspondences in multi-view video“. In: *Image and Vision Computing* 30.9 (2012), pp. 641–654.
- [SS02] D. Scharstein and R. Szeliski. „A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms“. In: *International Journal of Computer Vision* 47.1-3 (2002), pp. 7–42.
- [SSK+05] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor. „Garment Motion Capture Using Color-Coded Patterns“. In: *Computer Graphics Forum (Proc. Eurographics)* 24.3 (2005), pp. 439–447.
- [SSP07] R. Sumner, J. Schmid, and M. Pauly. „Embedded deformation for shape manipulation“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 26.3 (2007).
- [SSS08] N. Snavely, S. M. Seitz, and R. Szeliski. „Modeling the world from Internet photo collections“. In: *International Journal of Computer Vision* 80.2 (2008), pp. 189–210.
- [STC+12] C. Schumacher, B. Thomaszewski, S. Coros, S. Martin, R. Sumner, and M. Gross. „Efficient simulation of example-based materials“. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. 2012.
- [STG03] C. Strecha, T. Tuytelaars, and L. V. Gool. „Dense matching of multiple wide-baseline views“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 2003, pp. 1194–1201.

- 
- [SYA+15] R. Sagawa, Y. Yoshiyasu, A. Alspach, K. Ayusawa, K. Yamane, and A. Hilton. „Analyzing Muscle Activity and Force with Skin Shape Captured by Non-contact Visual Sensor“. In: *Proc. 7th Pacific Rim Symposium on Image and Video Technology (PSIVT2015)*. 2015, pp. 488–501.
- [SZB+07] M. Sormann, C. Zach, J. Bauer, K. Karner, and H. Bishof. „Watertight multi-view reconstruction based on volumetric graph-cuts“. In: *Scandinavian Conference on Image Analysis* (2007), pp. 393–402.
- [Sze10] R. Szeliski. *Computer Vision: Algorithms and Applications*. 1st ed. Springer, 2010.
- [Tan11] O Tange. „GNU Parallel - The Command-Line Power Tool“. In: *login: The USENIX Magazine* 36.1 (2011), pp. 42–47. URL: <http://www.gnu.org/s/parallel>.
- [TAS+10] C. Theobalt, E. de Aguiar, C. Stoll, H.-P. Seidel, and S. Thrun. „Performance Capture from Multi-View Video“. In: *Image and geometry processing for 3D Cinematography*. Ed. by R Ronfard and G Taubin. Springer, 2010, pp. 127–150.
- [Tau95] G. Taubin. „A signal processing approach to fair surface design“. In: *Proc. SIGGRAPH '95* (1995), pp. 351–358.
- [TDM11] J. R. Tena, F. De la Torre, and I. Matthews. „Interactive region-based linear 3D face models“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 30.4 (2011), 76:1–76:10.
- [TKR13] L. Torresani, V. Kolmogorov, and C. Rother. „A Dual Decomposition Approach to Feature Correspondence“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.2 (2013), pp. 259–271.
- [TMB14] A. Tsoli, N. Mahmood, and M. J. Black. „Breathing Life into Shape: Capturing, Modeling and Animating 3D Human Breathing“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 33.4 (2014), 52:1–52:11.
- [TR12] M. Tournier and L. Reveret. „Principal Geodesic Dynamics“. In: *ACM SIGGRAPH / Eurographics Symposium on Computer Animation*. 2012, pp. 235–244.
- [Tre13] M. A. Treiber. *Optimization for Computer Vision*. Springer, 2013.
- [TSB+05] J. Teran, E. Sifakis, S. S. Blemker, V. Ng-Thow-Hing, C. Lau, and R. Fedkiw. „Creating and Simulating Skeletal Muscle from the Visible Human Data Set“. In: *IEEE Transactions on Visualization and Computer Graphics* 11.3 (2005), pp. 317–328.

- [TSSH13] C. von Tycowicz, C. Schulz, H.-P. Seidel, and K. Hildebrandt. „An Efficient Construction of Reduced Deformable Objects“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 32.6 (2013), 213:1–213:10.
- [Ume91] S. Umeyama. „Least-Squares Estimation of Transformation Parameters Between Two Point Patterns“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13.4 (1991), pp. 376–380.
- [VBC+99] S. Vedula, S. Baker, R. Collins, T. Kanade, and P. Rander. „Three-Dimensional Scene Flow“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)*. 1999, 722–729 Vol. 2.
- [VBG+13] R. Vaillant, L. Barthe, G. Guennebaud, M.-P. Cani, D. Rohmer, B. Wyvill, O. Gourmel, and M. Paulin. „Implicit skinning“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 32.4 (2013), 125:1–125:12.
- [VBMP08] D. Vlastic, I. Baran, W. Matusik, and J. Popovic. „Articulated mesh animation from multi-view silhouettes“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27.3 (2008), 97:1–97:9.
- [VBPP05] D. Vlastic, M. Brand, H. Pfister, and J. Popović. „Face transfer with multilinear models“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 24.3 (2005), pp. 426–433.
- [VBZ+10] L. Valgaerts, A. Bruhn, H. Zimmer, J. Weickert, C. Stoll, and C. Theobalt. „Joint Estimation of Motion, Structure and Geometry from Stereo Sequences“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2010, pp. 568–581.
- [VCL+15] J. T. Vogelstein, J. M. Conroy, V. Lyzinski, L. J. Podrazik, S. G. Kratzer, E. T. Harley, D. E. Fishkind, R. J. Vogelstein, and C. E. Priebe. „Fast Approximate Quadratic programming for graph matching“. In: *PLoS ONE* 10.4 (2015). arXiv: [arXiv:1112.5507v3](https://arxiv.org/abs/1112.5507v3).
- [VGP+11] G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, and B. Thirion. „Multi-subject Dictionary Learning to Segment an Atlas of Brain Spontaneous Activity“. In: *Information Processing in Medical Imaging*. Vol. 6801. 2011, pp. 562–573.
- [VL08] B. Vallet and B. Lévy. „Spectral Geometry Processing with Manifold Harmonics“. In: *Computer Graphics Forum (Proc. Eurographics)* 27.2 (2008), pp. 251–260.
- [VRS14] C. Vogel, S. Roth, and K. Schindler. „View-Consistent 3D Scene Flow Estimation over Multiple Frames“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2014, pp. 263–278.

- 
- [VWB+12] L. Valgaerts, C. Wu, A. Bruhn, H.-P. Seidel, and C. Theobalt. „Lightweight binocular facial performance capture under uncontrolled lighting“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 31.6 (2012), 187:1–187:11.
- [VZBH08] K. Varanasi, A. Zaharescu, E. Boyer, and R. Horaud. „Temporal surface tracking using mesh evolution“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2008, pp. 30–43.
- [WCF07] R. White, K. Crane, and D. Forsyth. „Capturing and animating occluded cloth“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 26.3 (2007).
- [Wel] M. Welling. *Kernel Ridge Regression*. URL: <http://www.ics.uci.edu/~simswelling/teaching/KernelsICS273B/Kernel-Ridge.pdf>.
- [WHML13] X. Wang, M. Hong, S. Ma, and Z.-Q. Luo. „Solving Multiple-Block Separable Convex Minimization Problems Using Two-Block Alternating Direction Method of Multipliers“. In: *Pacific Journal of Optimization* 11.4 (2013). arXiv: 1308.5294.
- [WLM13] S. Wenger, D. Lorenz, and M. Magnor. „Fast Image-Based Modeling of Astronomical Nebulae“. In: *Computer Graphics Forum (Proc. of Pacific Graphics)* 32.7 (2013), pp. 93–100.
- [WMKG07] M. Wardetzky, S. Mathur, F. Kälberer, and E. Grinspun. „Discrete Laplace Operators: No Free Lunch“. In: *Proc. Eurographics Symposium on Geometry Processing SGP*. 2007, pp. 33–37.
- [WNEG14] M. Wacker, T. Neumann, J. Engel, and G. Gräfe. „Nutzung von Verfahren der Bildanalyse zur Baugrundbeurteilung“. In: *Tagungsband des Ohde-Kolloquiums*. Ed. by I. Herle. 2014.
- [WNF09] S. J. Wright, R. D. Nowak, and M. A. T. Figueiredo. „Sparse Reconstruction by Separable Approximation“. In: *IEEE Transactions on Signal Processing* 57.7 (2009), pp. 2479–2493.
- [WPP07] R. Y. Wang, K. Pulli, and J. Popović. „Real-time enveloping with rotational regression“. In: *ACM Transactions on Graphics* 26.3 (2007).
- [WSLG07] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. „Context-Aware Skeletal Shape Deformation“. In: *Computer Graphics Forum (Proc. EUROGRAPHICS)* 26.3 (2007), pp. 265–273.

- [WSVT13] C. Wu, C. Stoll, L. Valgaerts, and C. Theobalt. „On-set Performance Capture of Multiple Actors With A Stereo Camera“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*. Vol. 32. 6. 2013, 161:1–161:11.
- [Wu11] C. Wu. *VisualSFM: A Visual Structure from Motion System*. 2011. URL: <http://ccwu.me/vsfm/> (visited on 01/05/2016).
- [WVL+11] C. Wu, K. Varanasi, Y. Liu, H.-P. Seidel, and C. Theobalt. „Shading-based dynamic shape refinement from multi-view video under general illumination“. In: *International Conference on Computer Vision (ICCV)*. 2011, pp. 1108–1115.
- [WW04] B. J. van Wyk and M. a. van Wyk. „A POCS-based graph matching algorithm“. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.11 (2004), pp. 1526–1530.
- [YC07] J. Yao and W. K. Cham. „Robust multi-view feature matching from multiple unordered views“. In: *Pattern Recognition* 40.11 (2007), pp. 3081–3099.
- [YLL+14] J. Yan, Y. Li, W. Liu, H. Zha, X. Yang, and S. M. Chu. „Graduated consistency-regularized optimization for multi-graph matching“. In: *Proc. European Conference on Computer Vision (ECCV)*. 2014, pp. 407–422.
- [YWZ+15] J. Yan, J. Wang, H. Zha, X. Yang, and S. M. Chu. „Multi-View Point Registration via Alternating Optimization“. In: *AAAI Conference on Artificial Intelligence* (2015), pp. 3834–3840.
- [YYZ+14] Y. Yang, Y. Yu, Y. Zhou, S. Du, J. Davis, and R. Yang. „Semantic Parametric Reshaping of Human Body Models“. In: *3DV Workshop on Dynamic Shape Measurement and Analysis*. 2014.
- [ZHT06] H. Zou, T. Hastie, and R. Tibshirani. „Sparse Principal Component Analysis“. In: *Journal of Computational and Graphical Statistics* 15.2 (2006), pp. 265–286.
- [Zie95] G. M. Ziegler. *Lectures on Polytopes*. Springer, 1995.
- [ZKP10] C. Zach, M. Klopschitz, and M. Pollefeys. „Disambiguating visual relations using loop constraints“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010), pp. 1426–1433.
- [ZS08] R. Zass and A. Shashua. „Probabilistic graph and hypergraph matching“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008).

- [ZSCS04] L Zhang, N. Snavely, B. Curless, and S Seitz. „Spacetime Faces: High-Resolution Capture for Modeling and Animation“. In: *ACM Transactions on Graphics (Proc. SIGGRAPH)* 23.3 (2004), pp. 548–558.
- [ZZD15] X. Zhou, M. Zhu, and K. Daniilidis. „Multi-Image Matching via Fast Alternating Minimization“. In: *Proc. IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 4032–4040. arXiv: 1505.04845.
- [ŽL15] J. Žbontar and Y. LeCun. „Computing the Stereo Matching Cost with a Convolutional Neural Network“. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1592–1599. arXiv: 1409.4326.