Human Shape Estimation using Statistical Body Models

Dissertation

der Mathematisch-Naturwissenschaftlichen Fakultät

der Eberhard Karls Universität Tübingen

zur Erlangung des Grades eines

Doktors der Naturwissenschaften

(Dr. rer. nat.)

vorgelegt von

Matthew Loper

aus Long Branch, New Jersey, USA

Tübingen

2017

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation:	02.03.2017
Dekan:	Prof. Dr. Wolfgang Rosenstiel
1. Berichterstatter:	Prof. Dr. Hendrik Lensch
2. Berichterstatter:	Dr. Michael Black

Human Shape Estimation using Statistical Body Models

by

Matthew Maverick Loper

Submitted to the Wilhelm Schickard Institut for Computer Science
on January 26, 2017, in partial fulfillment of the
requirements for the degree of
Doctor rerum naturalium

Abstract

Human body estimation methods transform real-world observations into predictions about human body state. These estimation methods benefit a variety of health, entertainment, clothing, and ergonomics applications. State may include pose, overall body shape, and appearance.

Body state estimation is underconstrained by observations; ambiguity presents itself both in the form of missing data within observations, and also in the form of unknown correspondences between observations. We address this challenge with the use of a *statistical body model*: a data-driven virtual human. This helps resolve ambiguity in two ways. First, it fills in missing data, meaning that incomplete observations still result in complete shape estimates. Second, the model provides a statistically-motivated penalty for unlikely states, which enables more plausible body shape estimates.

Body state inference requires more than a body model; we therefore build observation models whose output is compared with real observations. In this thesis, body state is estimated from three types of observations: 3D motion capture markers, depth and color images, and high-resolution 3D scans. In each case, a forward process is proposed which simulates observations. By comparing observations to the results of the forward process, state can be adjusted to minimize the difference between simulated and observed data. We use gradient-based methods because they are critical to the precise estimation of state with a large number of parameters.

The contributions of this work include three parts. First, we propose a method for the estimation of body shape, nonrigid deformation, and pose from 3D markers. Second, we present a concise approach to differentiating through the rendering process, with application to body shape estimation. And finally, we present a statistical body model trained from human body scans, with state-of-the-art fidelity, good runtime performance, and compatibility with existing animation packages.

Thesis Supervisor: Michael J. Black
Title: Honorarprofessor

Acknowledgments

I would first like to thank my advisor, Michael Black. I have benefited greatly from his ability to communicate the science, utility, and fun of exploring opportunities in computer vision. He is a tireless advocate for his students and their work, including my own. His emphasis on making my approximations explicit will always be with me, and his taste for good research problems is impeccable. I owe a great deal to his patience and input.

Many have inspired me during the last few years, and I want to thank them for it. Thanks to Eric Rachlin for his humor and brainstorming sessions. Thanks to Gerard Pons-Moll for his contribution to the SMPL paper, and for all our amateur physics discussions. Thanks to Laura Seville, for the enthusiasm and fun she brings to every problem. Thanks to Javier Romero, for showing me what it means to be a great blend of scientist and engineer. And thanks to David Hirshberg, whose rigor with objective functions truly affected my view of problem solving.

I have received more than inspiration from people around me, and I want to acknowledge those who have shown me what it means to work hard and make things work. I would like to thank Alex Weiss for his practical sensibility and ability to summarize the bigger picture. Thanks to Naureen Mahmood, who has been a motivated collaborator and a kind person in all interactions. Thanks to Federica Bogo for her quiet intelligence, dry wit, and persistence with difficult problems. Thanks to Aggeliki Tsoli for letting me contribute to her work on measurement prediction.

I would like to thank Kathryn Rotondo, Max Loper, my parents, and those who stood by me during my years of education. I would also like to thank my brothers Eddie and Hoy; I look up them as computer scientists and outstanding people.

Finally, I want to thank my committee members: Hendrik Lensch, Christian Theobalt, Andreas Schilling, and Michael Black, for their input and assistance. Thanks especially to Hendrik for his patience throughout this process.

Contents

1 Introduction **19**

1.1 Thesis Statement 19

1.2 Introduction 19

1.3 Problem Statement 20

1.4 Motivation 21

1.4.1 Games and Animation 22

1.4.2 Ergonomics 22

1.4.3 Health 23

1.4.4 Clothing retail 23

1.5 Challenges and Solutions 23

1.6 Contributions 24

1.6.1 Motion and Shape from Sparse Markers 24

1.6.2 Differentiable Rendering for Body Shape Estimation 26

1.6.3 A Skinned Multi-Person Body Model 27

1.7 Thesis Outline 27

1.8 Published Papers 28

1.9 Related Papers 29

1.10 Related Patents 30

2 Related Work **31**

2.1 Human Body Modeling 31

2.1.1 Models Using Geometric Primitives 31

2.1.2 Artist-Driven Skinned Models 32

2.1.3 Data-Driven Models 33
2.2 Shape Estimation with Body Models 35

3 Shape Estimation from Sparse Markers 37
3.1 Introduction 37
3.2 Prior work 40
3.3 Body Model 43
3.4 Markers on the Body and in the World 44
3.4.1 Defining a Marker Set 45
3.4.2 Parameterizing Markers 46
3.5 Objectives 48
3.6 Optimization 49
3.7 Marker Selection 52
3.8 Results 55
3.8.1 Quantitative Shape Analysis 55
3.8.2 Archival Mocap (CMU) 57
3.8.3 Gender Estimation 58
3.8.4 Pose Estimation Results 58
3.9 Soft Tissue Deformation Results 59
3.9.1 Exaggerated Soft-Tissue Deformation 60
3.9.2 Soft-Tissue Retargeting 60
3.10 Conclusion and Discussion 61

4 Shape Estimation from RGB and Range Images 69
4.1 Introduction 69
4.2 Related Work 71
4.3 Defining our Forward Process 75
4.4 Differentiating our Forward Process 76
4.4.1 Differentiating Appearance 77
4.4.2 Differentiating Projection 77
4.4.3 Differentiating Intensity with Respect to 2D Image Coordinates 78

4.4.4 Software Foundation 79
4.5 Programming in OpenDR: Hello World 80
4.6 Experiments 82
4.6.1 Body Shape from Kinect 83
4.7 Conclusions 86

5 A Skinned Multi-Person Linear Model **89**
5.1 Introduction 89
5.2 Related Work 92
5.3 Model Formulation 96
5.4 Training 104
5.4.1 Pose Parameter Training 105
5.4.2 Shape Parameter Training 108
5.4.3 Optimization summary 109
5.5 SMPL Evaluation 110
5.5.1 Quantitative Evaluation 110
5.5.2 Sparse SMPL 112
5.5.3 Visual Evaluation 112
5.5.4 Run-time Performance 113
5.5.5 Compatibility with Rendering Engines 113
5.6 Discussion 114
5.7 Conclusions 117
5.8 Appendix 118
5.8.1 Mathematical Notation 118

6 Conclusion **129**
6.1 Future Directions 130

List of Figures

1-1 Body models use two types of parameters to create human body shapes. Training parameters are learned during a preprocess, and define the space of body shapes; run-time parameters select a body within that space. 20

1-2 Various kinds of observations are used in this work: markers in Chapter 3, color and depth in Chapter 4, and registrations in chapter 5. These images were generated from data used in this thesis. 21

1-3 Data-driven body models are used to fit models to markers (Chapter 3), images (Chapter 4), and registrations (Chapter 5). While Chapter 5 focuses on learning a body model, the other two chapters focus on shape and pose inference. 25

1-4 In Chapter 3, the body surface is predicted from marker locations. It works because we use a statistically trained body model to estimate body shape, pose, soft tissue deformation, and marker placement, all from the marker data. 26

1-5 Differentiable rendering produces not only the rendering in (a), but also per-pixel derivatives such as those shown in (b), (c), and (d). Differentiable rendering helps estimate parameters with gradient-based methods. 27

1-6 The SMPL body model produces meshes from pose and shape parameters, and can represent a wide variety of human bodies. Here, meshes produced by SMPL (in brown) are superimposed with ground-truth body registrations (in gray). 28

2-1 Basic geometric primitives have been used for modeling bodies and recovering pose. 32

2-2 Various works have tried to address the artifacts of LBS. 33

2-3 In SCAPE, vertices are unstitched into edges; these edges are first deformed according to shape and pose parameters, and finally restitched into a connected mesh. (Reprinted from [42]) 34

3-1 **Optimizing shape and markers.** Left: initial guess of markers, v_i, on the template shape in the canonical pose (blue). Right: Shape and marker locations after optimization. Optimized marker locations, $\tilde{m}_i$, are shown in red. Note that they have moved (see inset). 45

3-2 **Marker transformations.** In the latent coordinate space (left) we project a marker, $\tilde{m}_i$ into a basis defined by the nearest vertex: specifically by its normal, an arbitrary normalized edge, and the cross product between them. This provides a pose invariant representation for the marker. When the body pose changes (right), we then compute the location of the marker, $\hat{m}(\tilde{m}_i, \beta, \theta_t, \gamma_t)$, in the observed frame. 47

3-3 **Marker sets.** The union of all markers illustrates the 114 possible markers we considered. Yellow markers correspond to a standard 47-marker Vicon set. The 20 orange markers were found to improve shape estimation the most. The union of yellow and orange markers corresponds to our 67-marker set used for capturing shape and soft-tissue motion. White markers were deemed redundant and were not used. . 53

3-4 **Marker selection residuals.** The plot shows the mesh shape reconstruction error as a function of marker count. 55

3-5 **Effects of marker number on reconstruction error.** The mean and standard deviations of distance residuals indicate how the marker number affects reconstruction. *Top:* Shape reconstruction error. This is computed as the mean absolute distance between the true body shape (as represented by the alignment of the template to a scan) and the body shape estimated by MoSh reposed to match the registered mesh. *Bottom:* Held-out marker error across all sequences. This measures errors in both shape and pose but is inflated by marker placement error and marker movement. In both plots, 68.2% ($\pm 1\sigma$) of the residuals are contained between the error bars. 63

3-6 **Shape reconstruction.** First row: raw 3D scans from a high-resolution scanner. Second row: registered meshes obtained by precisely aligning a template mesh, with the same topology as our model, to the scans. These registered meshes faithfully capture the body shape and are used for our quantitative analysis. Third row: our model with shape, β, estimated from only 67 markers. Here we estimate the pose, θ, of our model to match the registered meshes to facilitate comparison. Bottom row: Distance between second and third rows. The heat map shows Euclidean distance from the registered mesh to the nearest point on the surface of the body estimated by MoSh; blue means zero and red means ≥ 4 cm. 64

3-7 **Shape from markers.** We show the effect of the number of markers (5, 10, 25, 47, 67) on the registration error (in m) of the estimated shape. Far right: reference image of the subject. 65

3-8 **CMU bodies.** Extracted shapes (bottom) and reference images (top) for several CMU subjects. Shape and pose is computed with MoSh using 47 Vicon markers only. 65

3-9 **CMU mocap.** Example meshes extracted from the CMU mocap dataset and representative frames from the animation. All shapes and poses are estimated automatically using only 47 markers. See **accompanying video** to see these and other results for CMU. 66

3-10 **Motion of soft tissue.** Some representative samples are shown. In each pair, the left image is without modeling dynamics (body shape fixed) and the right with with dynamics (body shape varying). Each image shows the full body and a detail region. Green balls correspond to the mocap markers. Red balls correspond to the simulated marker locations. Allowing body shape to change over time better captures soft tissue deformations. Note that, with dynamics, the predicted markers much more closely match the observed markers. 67

3-11 **Retargeting soft-tissue motions.** Top row: Body part segmentation for human and stylized characters. Middle row: retargeting pose and soft-tissue motion of an actor (left) to a stylized female character (middle), with heat maps (right) illustrating the percentage of soft-tissue deformation; blue means zero and red means ≥ 20 percent deformation. Bottom row: retargeting to another stylized character. See **accompanying video** to visualize the soft-tissue motions. . . . 68

4-1 Partial derivative structure of the renderer. 77

4-2 Constructing a renderer in OpenDR. 80

4-3 Minimizing an objective function given image evidence. The derivatives from the renderer are used by the minimize method. Including a translation-only stage typically speeds convergence. 81

4-4 Illustration of optimization in Figure 4-3. In order: observed image of earth, initial absolute difference between the rendered and observed image intensities, final difference, final result. 81

4-5 Optimizing a function of the rendered image to match a function of image evidence. Here the function is an edge filter. 81

4-6 **Rendering performance versus resolution.** For reference, 640x480 is 0.3 million pixels. Left: with rendering only. Right: with rendering and derivatives. 82

4-7 **Differentiable rendering versus finite differencing.** Left: a rotating quadrilateral. Middle: OpenDR's predicted change in pixel values with respect to in-plane rotation. Right: finite differences recorded with a change to in-plane rotation. 83

4-8 Accuracy of measurement prediction for Kinect-based fitting compared to measurements from CAESAR scans or guessing the mean (uninformed). Left: root mean squared error (RMSE) in cm. Right: percentage of explained variance. 86

4-9 Reconstruction of subjects (each subject is a row). **First column:** original captured images, with faces blurred for anonymity. **Second column:** simulated images after convergence. **Third column:** captured point cloud together with estimated body model. **Fourth column:** estimated body shown on background point cloud. 88

5-1 SMPL is a realistic learned model of human body shape and pose that is compatible with existing rendering engines, allows animator control, and is available for research purposes. (left) SMPL model (orange) fit to ground truth 3D meshes (gray). (right) Unity 5.0 game engine screenshot showing bodies from the CAESAR dataset animated in real time. 90

5-2 **Models compared with ground truth.** This figure defines the color coding used in this chapter. The far right (light gray) mesh is a 3D scan. Next to it (dark gray) is a registered mesh with the same topology as our model. We ask how well different models can approximate this registration. From left to right: (light green) Linear blend skinning (LBS), (dark green) Dual-quaternion blend skinning (DQBS), (blue) BlendSCAPE, (red) SMPL-LBS, (orange) SMPL-DQBS. The zoomed regions highlight differences between the models at the subject's right elbow and hip. LBS and DQBS produce serious artifacts at the knees, elbows, shoulders and hips. BlendSCAPE and both SMPL models do similarly well at fitting the data. 91

5-3 **SMPL model.** (a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blendshape contribution only; vertex and joint locations are linear in shape vector $\vec{\beta}$. (c) With the addition of of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by dual quaternion skinning for the split pose. 97

5-4 **Rig-driven deformation.** Joint angles control the relative orientation of bone segments; the bone segments, in turn, affect the computed surface. 99

5-5 Sample registrations from the multipose dataset. 104

5-6 Sample registrations from the multishape dataset. 105

5-7 **Initialization of joints and blend weights.** Discrete part segmentation in (a) is diffused to obtain initial blend weights, $\mathcal{W}_I$, in (b). Initial joint centers are shown as white dots. 119

5-8 **Joint regression.** (left) Initialization. Joint locations can be influenced by locations on the surface, indicated by the colored lines. We assume that these influences are somewhat local. (right) Optimized. After optimization we find a sparse set of vertices and associated weights influencing each joint. 120

5-9 **Shape blend shapes.** The first three shape principal components of body shape are shown. PC1 and PC2 vary from -2 to +2 standard deviations from left to right, while PC3 varies from -5 to +5 standard deviations to make the shape variation more visible. Joint locations (red dots) vary as a function of body shape and are predicted using the learned regressor, $\mathcal{J}$. 120

5-10 Cumulative relative variance of the CAESAR dataset explained as a function of the number of shape coefficients. For SMPL the variance is in vertex locations, while for BlendSCAPE it is in triangle deformations. 121

5-11 Model fitting with intermediate stages. We fit both BlendSCAPE (blue) and SMPL-LBS, $M(\vec{\beta}, \vec{\theta})$, (red) to registered meshes by optimizing pose and shape. $\bar{\mathbf{T}} + B_S(\vec{\beta})$ shows the estimated body shape and $T_P(\vec{\beta}, \vec{\theta})$ shows the effects of pose-dependent blend shapes. Here we show SMPL-LBS, because T_P shows more variation due to pose than SMPL-DQBS. 122

5-12 Model generalization indicates how well we can fit an independent registration. Mean absolute vertex error versus the number of shape coefficients used. 123

5-13 Pose generalization error indicates how well a fitted shape generalizes to new poses. 124

5-14 **Animating SMPL.** Decomposition of SMPL parameters into pose and shape: Shape parameters, $\vec{\beta}$, vary across different subjects from left to right, while pose parameters, $\vec{\theta}$, vary from top to bottom for each subject. 125

5-15 Performance of SMPL and BlendSCAPE vary with the number of body shape coefficients used. Performance shown here is from a 2014 Macbook Pro. 126

5-16 **Parameterizing pose blend shapes.** (a) Pose blend shapes parameterized by Euler angles cause significant problems. (b) our proposed parameterization allows the head to rotate in either direction with natural deformations. 126

Chapter 1

Introduction

1.1 Thesis Statement

The precision of human body state estimation can be improved with the formulation of differentiable, statistically motivated forward models.

1.2 Introduction

The human body is essential to every physical interaction we have with the world, and plays an important role in the communication of ideas and stories. We could not live without it. It should come as no surprise that there is much interest in the synthesis and analysis of the body: for animation, entertainment, communication, computer vision, and biomedical research.

The capture, analysis, and reproduction of body variation are greatly enabled by a *human body model*: a parameter-driven virtual human. As shown in Figure 1-1, a body model is a process that takes parameters as input and produces body shape as output. Such a model can help to recover the tremendous variety of shapes, poses, and configurations that bodies exhibit. The goal of this thesis is to improve the state of the art in precise model-driven human shape estimation: by improvements both to inference and to modeling.

While a body model is useful, it is not sufficient for inference. Observations

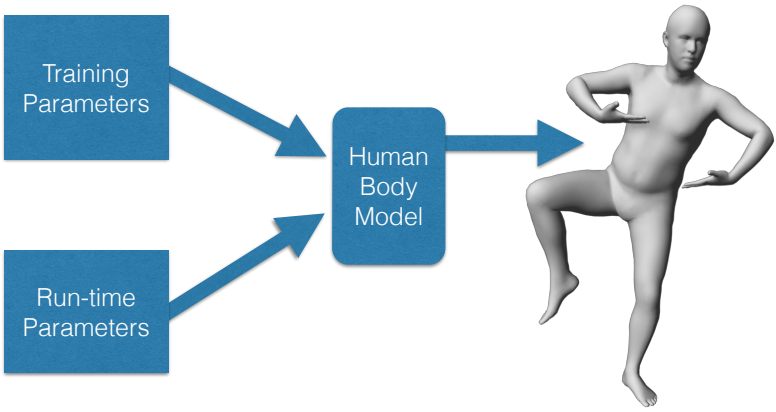


Figure 1-1: Body models use two types of parameters to create human body shapes. Training parameters are learned during a preprocess, and define the space of body shapes; run-time parameters select a body within that space.

typically take the form of images or marker data, which are not directly produced by a body model. In this work, we model three differentiable forward processes for the purpose of human state estimation, each of which produces a type of observable: as shown in Figure 1-2, one produces color and range images, one produces 3D marker locations, and one produces human body shapes. A body model helps us to predict observations; comparing these predictions with real observations enables parameter estimation.

1.3 Problem Statement

Our primary problem is the estimation of human pose and intrinsic shape from observations. Whereas we define *pose* by limb rotations and overall body translation, we define *intrinsic shape* to broadly mean "pose-independent shape characteristics." For example: whereas dancing implies pose variation, body weight change and interpersonal variation imply intrinsic shape variation. While this overall problem has

(a) Color and depth (b) Marker positions (c) Registrations

Figure 1-2: Various kinds of observations are used in this work: markers in Chapter 3, color and depth in Chapter 4, and registrations in chapter 5. These images were generated from data used in this thesis.

been investigated by others (see Chapter 2), our focus is on improving the precision of existing methods via inversion of well-conceived forward processes.

We specifically wish to demonstrate inference given three types of observations: markers, images, and registrations. While pose and intrinsic shape are estimated in all three cases, we also estimate other variables (such as marker placement, albedo, or soft tissue motion) that are specific to the form of observation.

1.4 Motivation

We distinguish between the scientific and the practical motivation for this work.

The *scientific* motivation of this work is to push the boundaries of generative inference in computer vision applications. We define *generative inference* as the process of adjusting model parameters such that the model output is similar to real-world observations. Unlike discriminative methods, which estimate parameters directly from observations, generative models simulate observations from parameters. Generative inference has the advantage that it can often produce lower error with less training

data than discriminative methods [1].

The *practical* motivations include applications for work, play, and health. We next review the potential practical applications of body-model driven inference.

1.4.1 Games and Animation

Motion capture is critical to the animation industry, but is nontrivial to automate. For example, *The Curious Case of Benjamin Button* required artists to add subtlety to the expressions captured with automated methods [2]; and *The Polar Express* required three capture stages with a total of 242 cameras and 151 face markers [3]. One reason for a good statistical model of the human body is that it can reduce the amount of markers and manual annotation required for faithful reproduction of human motion.

As demonstrated in Shotton et al. [4], synthetic models (such as that proposed in Chapter 5) can be used to train discriminative classifiers for real-time, body driven gameplay. While this thesis is more focused on refined shape estimation rather than coarse pose estimation, the use of models for a system like the Microsoft Xbox suggests the power of having a forward modeling process for training discriminative systems.

1.4.2 Ergonomics

There is a long history of human body model usage for ergonomic design; as early as 1985, 30 such models existed for this purpose [5]. Automotive, cockpit, and workstation design are currently among the more popular applications of body models to product design [5]. A typical goal might be to design a driver's seat to comfortably fit people from a population with corresponding shape variation.

Models for these purposes include Jack [6], RAMSIS [7], SAFEWORK [7], and SAMMIE [8].

1.4.3 Health

While obesity poses a threat to life expectancy [9] and quality of life [10], weight and BMI alone do not indicate fat distribution. For example, waist circumference is a valuable predictor of heart disease [11], independent of weight. There is great promise in augmenting traditional measures (such as weight and body mass index, or BMI) with more precise shape estimates [12]. Monitoring of shape from image or depth data (as in Chapter 4) could augment other measurements at the doctor's office, and could provide motivation at home to exercise.

1.4.4 Clothing retail

Clothing returns are a big problem for online retailers. As of 2013, more than half of online retailers in the clothing sector have returns over 25 percent [13]. By collecting shape estimates for customers, it should be possible to ensure better expected fits and thereby lower return rates. These shape estimates could be collected from image data.

1.5 Challenges and Solutions

The challenges of inference include those presented by observations and those presented by underlying body state. These challenges, and our strategies to address them, are as follows:

Observations contain noise and missing data, which add ambiguity to the inference process. We address this with rich forward models that estimate (and factor out) many parameters in order to estimate the variables of interest. For example, in Chapter 4, many ancillary parameters such as lighting, albedo, and camera parameters are adjusted to better estimate shape; in Chapter 3, marker placement is estimated to better estimate shape and pose.

Body state has many degrees of freedom, and presents a large search space. We address this with regularization (which shrinks the search space) and with gra-

dient based methods which converge to an optimum more quickly than gradient-free methods. While gradient-based methods are subject to local minima, our focus on refinement means that a combination of good initialization and regularization keeps estimates on track.

1.6 Contributions

The contributions of this work concern fitting simulated models to observations, as depicted in Figure 1-3. While the scope of the chapters differ by observation type (markers, images, or registrations) and by desired output (animation, body shape, or body model), they all illustrate techniques for precise state estimation with a human body model. These contributions will now be discussed in more detail.

1.6.1 Motion and Shape from Sparse Markers

Our first contribution is a method for extracting body state from conventional motion capture markers. As shown in Figure 1-4, 3D marker positions are used to guide body estimation. More specifically, intrinsic body shape, pose, dynamic soft tissue motion, and marker placement are all estimated from sparse marker positions and coarse marker placement. A statistical body model is critical to the estimation of all these parameters from only a sparse set of markers: such a model helps to regularize pose, regularize shape, and accurately predict the geometry between the markers over time.

Our methods are evaluated on two large motion capture databases, and the effects of varying numbers of markers is also investigated. A key component is marker placement refinement: because marker locations on the skin are difficult to replicate exactly across different people and sessions, we refine the marker locations using the motion capture sequence itself. The result is a lifelike animation. This contribution is detailed in Chapter 3.

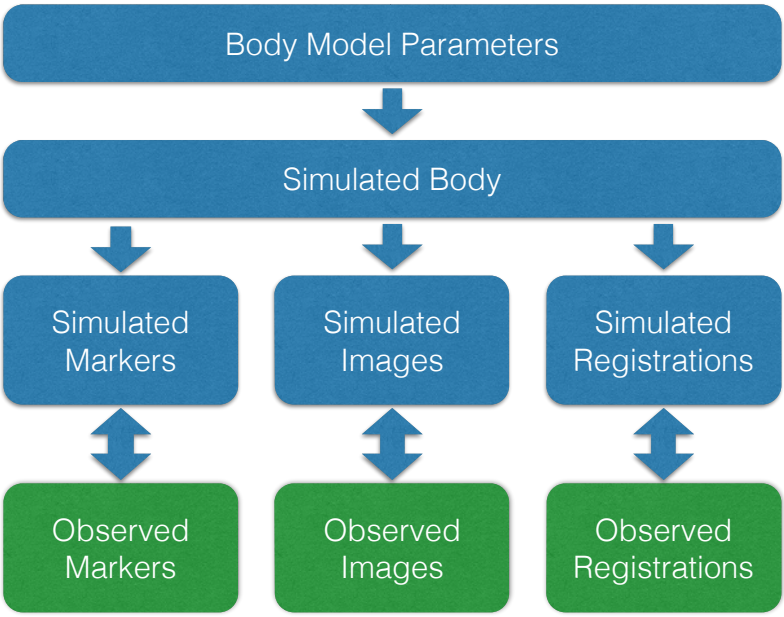


Figure 1-3: Data-driven body models are used to fit models to markers (Chapter 3), images (Chapter 4), and registrations (Chapter 5). While Chapter 5 focuses on learning a body model, the other two chapters focus on shape and pose inference.

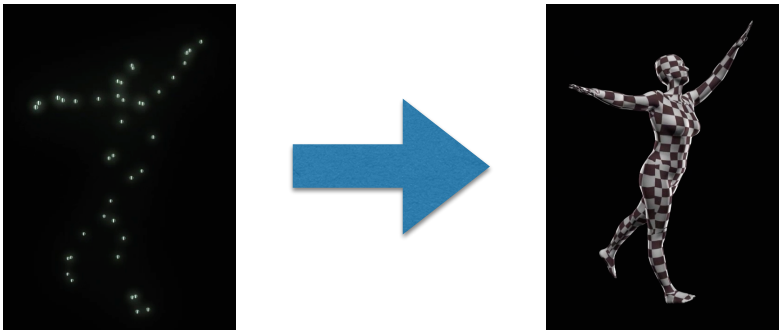


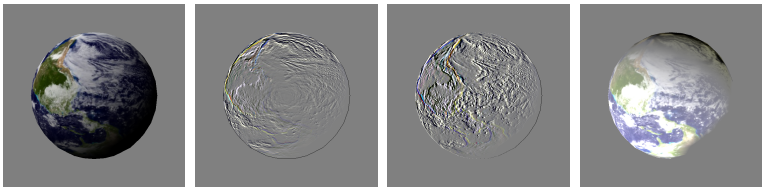
Figure 1-4: In Chapter 3, the body surface is predicted from marker locations. It works because we use a statistically trained body model to estimate body shape, pose, soft tissue deformation, and marker placement, all from the marker data.

1.6.2 Differentiable Rendering for Body Shape Estimation

Next, we show how concise patterns for parameter estimation can be achieved with a *differentiable renderer*. Whereas a typical renderer provides only pixels, a differentiable renderer also produces derivatives with respect to its parameters. This enables illumination, camera parameters, albedo, and geometry to be refined from depth or color images. We release a general framework dedicated towards the flexible estimation of model-based parameters for depth and color images. Its reusability is valuable for solving new problems, and the provided gradients are useful in computer vision problems characterized by the refinement of a large number of parameters. Rendering and derivatives are illustrated in Figure 1-5.

In Chapter 4, we describe this framework and its application to body state estimation. Body shapes and measurements are estimated for 23 subjects from color and range observations. Accuracy is compared to another method: 3D fitting to laser scans.

A statistically-learned human body model is critical our chosen application in two ways. First, the body model maps body shape parameters to geometry, which can have its rendering compared against observed images. And second, statistically



(a) Rendering (b) Sensitivity to focal length (c) Sensitivity to translation in the "x" direction (d) Sensitivity to light position in the "y" direction

Figure 1-5: Differentiable rendering produces not only the rendering in (a), but also per-pixel derivatives such as those shown in (b), (c), and (d). Differentiable rendering helps estimate parameters with gradient-based methods.

learned regularizations on body model parameters help to keep predicted bodies within the natural space of human shapes. The body model therefore serves both to match observables and to regularize inferred results.

1.6.3 A Skinned Multi-Person Body Model

Our final contribution is a new body model that is comparable with (and often superior to) existing state-of-the-art body models in speed, fidelity, and compatibility. Rotation matrix elements are used to drive pose-dependent blendshapes, which is shown to work well with linear blend skinning. As shown in Figure 1.6.3, our Skinned Multi-Person Linear model (or SMPL for short) can produce many body shapes, and its compatibility and speed lend it practicality. The details of its formulation, training, and comparison against another state-of-the-art body model can be found in Chapter 5. This model is freely available for research purposes.

1.7 Thesis Outline

This remainder of this document is organized as follows.

Chapter 2: Related Work: We describe previous work both in building body models and in using them for shape inference.

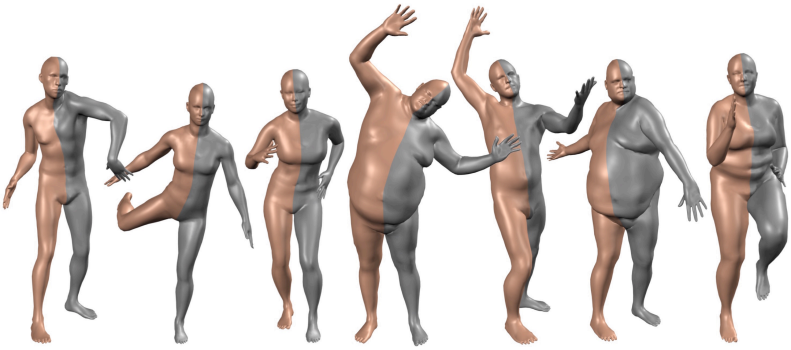


Figure 1-6: The SMPL body model produces meshes from pose and shape parameters, and can represent a wide variety of human bodies. Here, meshes produced by SMPL (in brown) are superimposed with ground-truth body registrations (in gray).

Chapter 3: Shape Estimation from Sparse Markers: Methods are introduced for the acquisition of static body shape, dynamic body changes, and pose from sparse markers.

Chapter 4: Shape Estimation from RGB and Range Images: A differentiable renderer is described and applied to the estimation of body shape from color and range images.

Chapter 5: A Skinned Multi-Person Linear Model: A body model is introduced which is competitive with state-of-the-art alternatives in fidelity, compatibility, and runtime performance.

Chapter 6: Conclusion: Considerations about "inverse graphics," the limits of modeling, and future work are discussed.

1.8 Published Papers

Matthew M. Loper, Naureen Mahmood, and Michael J. Black. "MoSh: Motion and Shape Capture from Sparse Markers". In: *ACM Trans. Graph., (Proc. SIGGRAPH Asia)* 33.6 (Nov. 2014), 220:1–220:13. DOI: 10.1145/2661229.2661273. URL:

http://doi.acm.org/10.1145/2661229.2661273

Matthew M. Loper and Michael J. Black. "OpenDR: An Approximate Differentiable Renderer". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Vol. 8695. Lecture Notes in Computer Science. Heidelberg: Springer, 2014, pp. 154–169. ISBN: 978-3-319-10583-3. DOI: 10.1007/978-3-319-10584-0_11

Matthew M. Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 34.6 (2015). DOI: 10.1145/2816795.2818132. URL: http://doi.acm.org/10.1145/2816795.2818132

1.9 Related Papers

David A. Hirshberg, Matthew Loper, Eric Rachlin, and Michael J. Black. "Coregistration: Simultaneous Alignment and Modeling of Articulated 3D Shape". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid. Vol. 7577. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 242–255. ISBN: 978-3-642-33782-6. DOI: 10.1007/978-3-642-33783-3_18

Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. "FAUST: Dataset and evaluation for 3D mesh registration". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Columbus, Ohio, USA, June 2014

Javier Romero, Matthew Loper, and Michael J. Black. "FlowCap: 2D Human Pose from Optical Flow". In: *German Conference on Pattern Recognition (GCPR)*. 2015

Federica Bogo, Michael J. Black, Matthew Loper, and Javier Romero. "Detailed Full-Body Reconstructions of Moving People from Monocular RGB-D Sequences". In: *International Conference on Computer Vision (ICCV)*. Dec. 2015

1.10 Related Patents

M. Loper, N. Mahmood, and M. Black, *Motion and Shape Capture from Sparse Markers*, U.S. Provisional Patent Application No. 61/930,711.

M. Black, O. Freifeld, P. Guan, M. Loper, A. Weiss, *Parameterized model of 2d articulated human shape*, US (13/696,676). Apr 19, 2012.

M. Black, A. Balan, A. Weiss, L. Sigal, M. Loper, T. St Clair, *Method and apparatus for estimating body shape*, US (12/541,898), May 6, 2010.

Chapter 2

Related Work

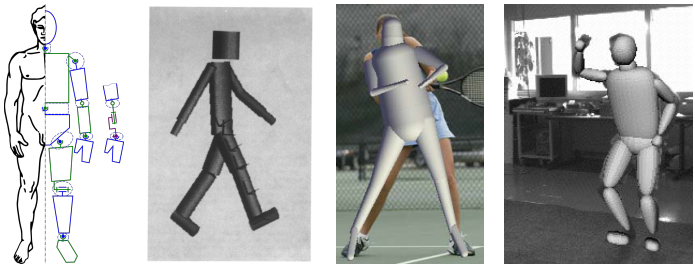
This thesis contributes to both work on human modeling and work on human body state estimation. Therefore, we first review previous work in modeling, followed by previous work in model-driven inference.

2.1 Human Body Modeling

Marr and Nishihara define *shape* as the physical surface geometry of an object [21]. We define a *human body model* as a mapping from parameters to human-like shapes. Human body models have a long history; early models resembled stick-figures, and were used in coarse pose recovery from images. Later models added realism and more detailed shape variation, and were able to help infer more details. In this section, we examine how body models have changed over the years.

2.1.1 Models Using Geometric Primitives

Some of the earliest models of the body were composed from geometric primitives. In early work by Hanavan [22], a mathematical model of the human body was constructed from simple polygonal shapes. Its anthropometric and inertial properties were guided by the measurement of 25 subjects. The model was intended to help the thruster design of space vehicle maneuvering systems.



(a) Kinematic chain (NASA) (b) Cylindrical model by Hogg [23] (c) Conic model by Lee [24] (d) Superquadric model by Sminchisescu [25]

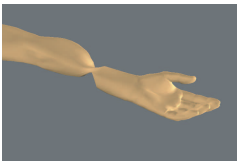
Figure 2-1: Basic geometric primitives have been used for modeling bodies and recovering pose.

Marr and Nishihara proposed geometric primitives as compact representations of observable objects [21]. Since then, cylinders [23, 26, 27, 28], cones [29, 24, 30, 31, 32, 33], and superquadrics [34, 25] have been successfully used to help infer the state of bodies from images. A sampling of these primitive-based models are shown in Figure 2-1.

Most of these models use the abstraction of a *kinematic chain*: rigid bodies connected by links, as shown in Figure 2-1. Human pose may then be expressed as relative rotations of these links. While realism was improved upon by later models, the characterization of pose as a set of relative angles remains useful in the models of today, and in the models used in this thesis.

2.1.2 Artist-Driven Skinned Models

The animation industry has driven much of the quest for realism in human body modeling. An especially vexing problem is modeling of body joints: while a kinematic chain can be used to drive surfaces, surfaces near two or more bones can be difficult to model. The use of *skinning*, or moving surfaces according to a weighted combination of bones in a kinematic chain, has been accepted in the animation community as one useful paradigm for controlling the relationship between an articulated skeleton and an animated surface.



(a) LBS collapse (reprinted from [37])

(b) Dual Quaternion Blend Skinning [36]

(c) Skins from Examples [37]

Figure 2-2: Various works have tried to address the artifacts of LBS.

The earliest skinning method was Linear Blend Skinning (LBS), which assigns transformations to bones, and transforms vertices according to a linear combination of these transformations [35]. Because this implies linear combinations of rotation matrices, the method suffers [36] from volume shrinkage (also known as "candy-wrapper" effects) at the joints as shown in Figure 2-2.

Dual quaternion blend skinning [36] was one attempt to address this problem. Mohr and Gleicher [37] address joint collapse by automatically adding more joints to a kinematic chain; this avoids collapse by making sure no single joint ever rotates enough to collapse. Examples of these methods are shown in Figure 2-2.

These skinning methods alone cannot model surface deformation with the precision required by the animation industry. To address this, Lewis et al [35] proposed example-based interpolation to allow an artist to manually apply corrections to skinning methods and to improve expressiveness.

2.1.3 Data-Driven Models

Data-driven models can be trained to closely fit large numbers of examples in a statistical manner, bypassing the subjectivity of artists. The earliest data-driven model may be from Kakadiaris et al. [38], in which a body model and its segmentation were estimated from three orthogonal views. The shape of the model was image-driven. This model was later used for human tracking [39, 34].

More realism was obtained with the work of Allen et al. [40], in which a space of body shapes was learned from registrations to 250 laser scans. Most attention

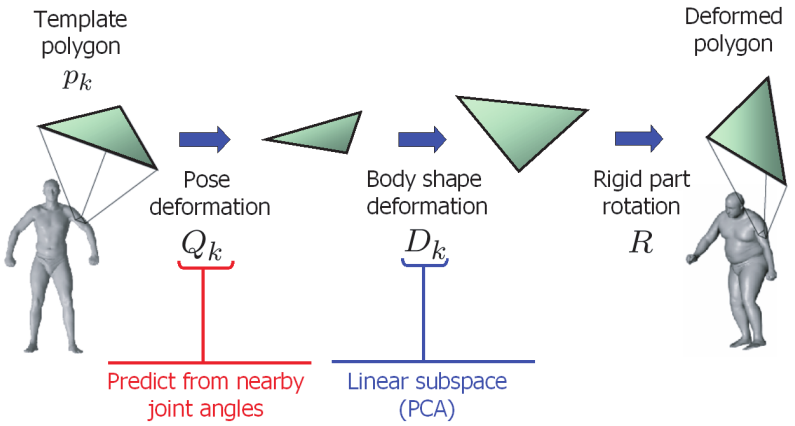


Figure 2-3: In SCAPE, vertices are unstitched into edges; these edges are first deformed according to shape and pose parameters, and finally restitched into a connected mesh. (Reprinted from [42])

was given to the shape space, with the pose-dependent deformations limited by unrealistic stock skinning methods. Allen's later work [41] corrected this deficiency by incorporating example-based shape and pose deformations, such that pose deformations depend explicitly on shape. In contrast to the methods described next, Allen's models are position-based: they are trained to reproduce global vertex positions, and store they offsets to vertex positions.

In 2004, ideas were introduced [43, 44] for representing a surface with local differential properties. In contrast to the position-based methods described above, differential methods first apply transformations to very local patches (on the scale of a triangle). Because the locally deformed patches often do not fit together precisely, least-squares methods are typically used to stitch the patches into a coherent surface. Representing changes to a mesh by local differential properties (instead of with vertex offsets) can preserve global smoothness while allowing local editing; a review of such representations is in [45].

One body model using differential methods is SCAPE [46]; its pipeline is shown

in Figure 2-3.

2.2 Shape Estimation with Body Models

We now review the literature on shape estimation with the use of human body models. An overview of modern pose estimation methods is found in [47]; our focus here is not on pose, but rather on intrinsic shape estimation, or the estimation of aspects of shape which are specific to an individual.

Table 2.1 illustrates many of the shape estimation methods used to date. Data acquisition, features, and search methods are all critical components of the estimation process.

Data acquisition. Data for automatic human shape estimation is optically acquired: laser scans, depth images, marker data, and standard color images all originate with a camera. Because of the ubiquity of color images and the challenges of uncontrolled environments, earlier methods acquired shape from multiple color cameras in a controlled setting. Later methods either used monocular images, incorporated depth data, used images over time and/or worked better in uncontrolled settings.

Features. Silhouettes [48, 49, 50, 51, 52] have been used by many works to obtain shape estimates. Silhouettes rely on background subtraction methods, which fare poorly in uncontrolled settings, however. Edges, which do not rely on segmentation, are another popular feature [53, 49]. Raw image intensity [49, 54, 50] and depth camera image values [52, 54, 55] are also valuable, especially when the quality of initial parameters and the body model are sufficiently good. Finally, dense 3D marker positions have also been used to reconstruct personal shape [56].

Search. Shape estimation is like any optimization problem: given observables, the goal is to search for the best possible characterization of hidden state (eg shape). As such, shape estimation depends on many design decisions; these include initialization, stochastic vs. non-stochastic search, whether gradients are used, and whether only one most likely answer is desired (as opposed to a posterior distribution).

To place this thesis in the context of previous work, this thesis is focused on

Authors	Search	Features	Notes
Carranza et al., 2003 [57]	Powell	silhouettes	
Park et al., 2006 [58]	LSQ	markers	
Sigal et al., 2007 [59]	MoE+PF	silhouettes	[a]
Bălan et al., 2007 [48]	annealed PF	silhouettes	
Bălan et al., 2007 [50]	annealed PF	shading, shadows	
Park et al., 2008 [56]	LSQ	markers	
Bălan et al., 2008 [51]	simplex	silhouttes, skin	
Guan et al., 2009 [49]	simplex	color	
Hasler et al., 2009 [60]	ICP	laser scans	[b]
Hasler et al., 2010 [61]	ICP	silhouettes	[c]
Weiss et al., 2011 [52]	gradient-based	silhouettes, depth	
Chen et al., 2013 [62]	ICP	depth image	
Loper et al., 2014 [14]	LSQ	markers	
Loper et al., 2014 [15]	gradient-based	depth, color	

MoE = Mixture of Experts
ICP = Iterative closest point
PF = particle filtering
LSQ = least squares

[a] Features: radial distance functions and shape contexts.
[b] shape under clothing from laser scans
[c] clicked points required

Table 2.1: Previous work on human body shape estimation has varied by search strategy, feature choices, and type of observable.

precise shape estimation with differentiated models. To that end, generative models and regularization both play strong roles in this thesis, and discriminative methods do not. Discriminative methods are critical for pose estimation from ambiguous data when weak priors are available, but here we have either a strong data term (as with MoSh in Chapter 3) or a strong prior (as with OpenDR in Chapter 4).

Chapter 3

Shape Estimation from Sparse Markers

3.1 Introduction

While marker-based motion capture (mocap) is widely used to animate human characters in films and games, it is also widely criticized as producing lifeless and unnatural motions. We argue that this is the result of "indirecting" through a skeleton that acts as a proxy for the human movement. In standard mocap, visible 3D markers on the body surface are used to infer the unobserved skeleton. This skeleton is then used to animate a 3D model and what is rendered is the visible body surface. While typical protocols place markers on parts of the body that move as rigidly as possible, soft-tissue motion always affects surface marker motion. Since non-rigid motions of surface markers are treated as noise, subtle information about body motion is lost in the process of going from the non-rigid body surface to the rigid, articulated, skeleton representation. We argue that these non-rigid marker motions are not noise, but rather correspond to subtle surface motions that are important for realistic animation.

We present a new method called MoSh (for **Mo**tion and **Sh**ape capture) that replaces the skeleton with a 3D parametric body model. Given a standard marker set, MoSh simultaneously estimates the marker locations on a proxy 3D body model, estimates the body shape, and recovers the articulated body pose. By allowing body

shape to vary over time, MoSh is able to capture the non-rigid motion of soft tissue. Previous work on the mocap of such motions relies on large marker sets [58, 56]. In contrast, we show that significant soft tissue motion is present in small marker sets and that capturing it results in more nuanced and lifelike animations. MoSh also recovers qualitatively and metrically accurate body shapes from small numbers of markers; Fig. 1 shows body shapes and poses recovered with 67 markers and compares the body shapes with 3D scans. While fine details are missing, MoSh enables users of standard mocap to obtain reasonable 3D body shapes from markers alone.

The basic version of MoSh has five core components. 1) MoSh uses a parametric 3D body model that realistically represents a wide range of natural body shapes, poses, and pose-dependent deformations. For this we use a learned statistical body model based on SCAPE [46]. 2) Marker placement on the human body varies across subjects and sessions, consequently we do not assume that the exact marker placement is known. Instead, a key contribution of MoSh is that it solves for the observed marker locations relative to the 3D body model. 3) MoSh also simultaneously solves for the 3D body shape of the person that best explains the observed 3D mocap marker data. 4) Steps 2 and 3 above require that we also simultaneously solve for 3D body pose. Components 2–4 are all embodied in a single objective function and we optimize this for a subset of the mocap sequence. 5) In a second stage, MoSh uses the computed body shape and marker locations on the body, to estimate body pose throughout a mocap session.

This basic method produces appealing animations but the assumption of a single body shape across the session does not account for the *dynamics of soft tissue*; for example, the jiggling of fat during jumping. Currently there are no practical technologies for easily capturing these soft-tissue motions. Previous methods have used large marker sets [58] but these are time consuming to apply, difficult to label, and suffer from occlusion. These methods also do not apply to archival data. Video-based surface capture methods offer the potential for even greater realism [63, 64] but are not yet mature and are not widely adopted. To capture soft-tissue deformation, we allow the body shape to change over time to better fit the marker motions. Our

solution uses a low-dimensional shape model to make it practical and penalizes deviations from the fixed body shape estimated without soft-tissue deformation. We make an assumption that these deformations can be approximated *within the space of static human body shape variations*; that is, we model the soft-tissue deformations of an individual effectively by changing their identity. Given a sufficiently rich space of body shape variation, this works surprisingly well.

While we can estimate body shape and pose from standard marker sets and archival mocap sequences, we go further to design *additional marker sets* with greater or fewer markers. Using a principled objective function, and a training set of 3D body meshes, we evaluate the effect of different marker sets on the accuracy of body shape and pose capture. While the standard 47-marker set that is often used for motion capture (e.g. in the CMU dataset) works surprisingly well for recovering both shape and pose, we find that an expanded set, with 20 additional markers, captures more soft tissue motion.

We validate the method with nearly 800 mocap sequences. Since no body scanner or other hardware is required, MoSh can be applied to archival mocap data. To demonstrate this we reconstruct gender, shape, and motion of 39 subjects in the CMU mocap dataset using 47 markers. The resulting animations are nuanced and lifelike and the body shapes qualitatively match reference video. For quantitative evaluation, we scanned twenty subjects with widely different body shapes and performed MoSh with different numbers of markers.

MoSh can be used directly for animation or as a reference for animators. In the accompanying video we show that we can change the body shape to retarget the mocap sequence to new bodies (cf. [46]). This transfer works for any character with the same topology as our body model. We align several cartoon characters to our mesh and then animate them without the labor-intensive process of developing a rigged model or retargeting the skeletal motions. The animations include the transfer of soft tissue motions and we show further how these motions can be magnified to produce interesting animations with exaggerated soft-tissue dynamics.

In summary, the main contribution of MoSh is that it provides a fully automated

method for "mining" lifelike body shape, pose, and soft-tissue motions from sparse marker sets. This makes MoSh appropriate for processing archival mocap. By using the same (or slightly augmented) marker sets, MoSh complements, existing marker-based mocap in that animators can extract standard skeletal models from the markers, MoSh meshes, or both.

3.2 Prior work

There is an extensive literature on (and commercial solutions for) estimating skeleton proxies from marker sets. Since MoSh does not use a skeleton, we do not review these methods here. Instead, we focus on several key themes in the literature that more directly relate to our work: fitting models to sparse markers, dense marker sets, and surface capture.

From Markers to Models. To get body shape from sparse markers, one needs a model of body shape to constrain the problem. There have been several previous approaches. Allen et al. [40] learn a model of body shape variation in a fixed pose from 3D training scans. Anguelov et al. [46] go further to learn a model that captures both body shape and non-rigid pose deformation.

Allen et al. show that one can approximately recover an unknown 3D human shape from a sparse set of 74 landmarks. They do this only for a fixed pose since their model does not represent pose variation. Importantly the landmarks are perfect and known; that is, they have the 3D points on the mesh they want to recover and do not need to estimate their location on the mesh. Unlike MoSh this does not address the problem of estimating body shape and pose from mocap markers alone.

Anguelov et al. [46] show how to animate a SCAPE model from motion capture markers. Their method requires a 3D scan of the subject with the markers on their body. This scan is used for two purposes. First it is used to estimate the 3D shape model of the person; this shape is then held fixed. Second the scanned markers are used to establish correspondence between the scan and the mocap markers. These limitations mean that the approach cannot work on archival mocap data and that a

user needs both a 3D body scanner and a mocap system.

It is important to note that Anguelov et al. did not solve the problem addressed by MoSh. They fit a SCAPE model to a 3D body *scan* (what they call shape completion) and with *known marker locations*, animate the model from mocap markers. We go beyond their work to estimate the body shape from *only the sparse mocap markers* without the use of any scan and without knowing their precise location on the body. We do this by simultaneously solving for the marker locations, the shape of the body and the pose using a single objective function and optimization method. Unlike [46], MoSh is fully automatic and applicable to archival data.

We also go beyond previous work to define new marker sets and evaluate the effect of these on reconstruction accuracy. This provides a guide for practitioners to choose appropriate marker sets.

Dynamics of Soft Tissue. Unlike MoSh, the above work does not address the capture of soft tissue motion. Interestingly, much of the attention paid to soft-tissue motion in the mocap community (particularly within biomechanics) actually focuses on minimizing the effects of soft tissue dynamics [65]. Soft tissue motion means the markers move relative to the bones and this reduces the accuracy of the estimated skeletal models. For animation, we argue that such soft tissue motions are actually critical to making a character look alive.

Dense Marker Sets. To capture soft-tissue motion, previous work has used large, dense, marker sets. Park and Hodgins [58] use 350 markers to recover skin deformation; in the process, they deform a subject-specific model to the markers and estimate missing marker locations. In later work [56], they use a large (400-450) marker set for $\approx 10,000$ frames of activity to create a subject-specific model; this model can then be used to recover pose for the same subject in later sessions with a sparse marker set. In these works, the authors visualize soft-tissue deformations on characters resembling the mocap actor. Here we transfer soft-tissue deformations to more stylized characters.

Hong et al. [66] use 200 markers on the shoulder complex and a data driven approach to infer a model of shoulder articulation. While dense markers can capture

rich shape and deformation information, they are not practical for many applications. Placing the markers is time consuming and a large number of markers may limit movement. With these large sets, additional challenges emerge in dealing with inevitable occlusions and marker identification.

Recent work captures skin deformations using a dense set of markers or patterns painted on the body [18, 67]. The work is similar to Park and Hodgins but uses computer vision methods rather than standard mocap markers.

Our work differs in that it conforms to standard mocap practice and is backwards-compatible with existing sparse marker sets. The goal of MoSh is to get more out of sparse markers.

Surface Capture. At the other extreme from sparse markers are methods that capture full 3D meshes at every time instant [63, 64]; this can be conceived of as a very dense marker set. Still other methods use a scan of the person and then deform it throughout a sequence [68, 69]. Existing methods for surface capture rely on multi-camera computer vision algorithms that are computationally expensive compared with commercial marker-based systems. These methods are most applicable to capturing complex surfaces like clothing or breathing [70] that are difficult to parametrize. In the case of body shape, we find that, together with a parametric body model, a small marker set is already very powerful.

In a related approach, de Aguiar et al. [71] use an intermediate template that is animated in a traditional way from mocap markers. They then transfer the template motion to a more complex mesh. Like MoSh this method is motivated by standard practice but it still indirects through a crude proxy, rather than solving directly for shape and pose from markers.

Attribute Capture. The idea that markers contain information about body shape is not new. Livne et al. [72] use motion capture data to extract socially meaningful attributes, such as gender, age, mental state and personality traits by applying 3D pose tracking to human motion. This work shows that a sparse marker set contains rich information about people and their bodies. MoSh takes a different approach by using the sparse marker data to extract faithful 3D body shape. Like Livne et al.,

we show that gender can be estimated from markers. Beyond this, we suspect that the full 3D body model can be used to extract additional attributes.

Motion Magnification. There has been recent work on magnifying small motions in video sequences [73, 74, 75] but less work on magnifying 3D motions. In part this may be because capturing 3D surface motions is difficult. Other work exaggerates mocap skeletal motions using mocap data [76]. In [77] they develop methods for spatially localized modeling of deformations and show that these deformations can be edited and exaggerated. In [78] they edit body shape to exaggerate it but do not model or amplify non-rigid soft-tissue dynamics. While the exaggeration of facial motion has received some attention, we think ours is the first work to use only sparse marker sets to extract full-body soft-tissue motion for exaggeration.

In summary, MoSh occupies a unique position – it estimates 3D body shape and deformation using existing mocap marker sets. MoSh produces animated bodies directly from mocap markers with a realism that would be time consuming to achieve with standard rigging and skeleton-based methods.

3.3 Body Model

Extracting body shape from sparse markers is clearly an ill-posed problem; an infinite number of bodies could explain the same marker data. To infer the most likely body we must have a model of human shape that captures the correlations in body shape within the population. For this we use a learned body model that is similar to SCAPE [46]. It should be noted however that any mesh model could be used, as long as (1) it allows shape and pose variation, and (2) is differentiable with respect to its parameters.

Our body model is a function that returns a triangulated mesh with 10,777 vertices, and is parameterized by a global translation center γ, a vector of pose parameters, θ, a mean shape, μ, and a vector of shape parameters, β. Shape is defined in terms of deformations applied to the triangles of a base template mesh. The surface of the body is described as $S(\beta, \theta, \gamma)$, with the coordinates of vertex k notated

$S_k(\beta, \theta, \gamma)$. The body mesh is segmented into parts and each part can undergo a rotation defined by θ. The pose parameters θ consist of 19 angle-axis vectors, whereby length indicates the amount of rotation. Like SCAPE, the function $S(\cdot)$ includes pose-dependent non-rigid deformations that are learned from bodies in a wide range of poses. Body shape is approximated by the mean shape and a linear combination of shape basis vectors; β is a vector of these linear coefficients. This shape basis is learned from deformations of training body shapes using principal component analysis (PCA). In what follows, we represent body shape using 100 principal components.

We train the body shape model from 3803 CAESAR scans of people in an upright pose (approximately 2103 women and 1700 men from the US and EU datasets) [79]. The pose-dependent component of the model is learned from 1832 scans of 78 people (41 women and 37 men) in a wide range of poses. The scans are aligned using the technique in [17]. Since the model is trained from an extensive set of scans, it is able to realistically capture a wide range shapes and poses. For details of SCAPE, the reader is referred to [46].

Note that we train three body shape models: separate models for men and women, plus a gender neutral model. If we know the gender of the subject, we use the appropriate model. If not, we fit the gender-neutral model, infer the gender, and then use a gender-specific model as described below.

3.4 Markers on the Body and in the World

Mocap markers extend from the human body to varying degrees and are placed on the body manually. Precise placement can be difficult, particularly on heavy subjects where fat makes it difficult to palpate boney locations. The result is that we cannot expect to know the exact marker locations in advance. The first step of MoSh solves for the marker locations, relative to a template body mesh, for a given mocap sequence (or collection of sequences for one subject).

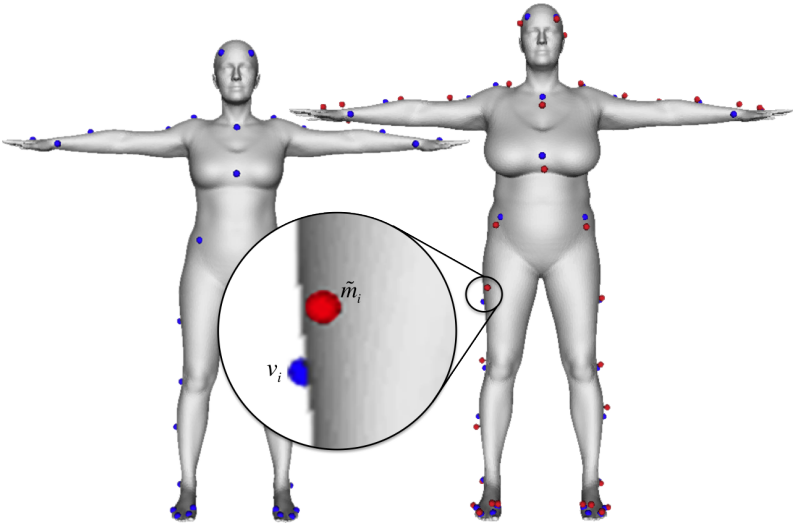


Figure 3-1: **Optimizing shape and markers.** Left: initial guess of markers, v_i, on the template shape in the canonical pose (blue). Right: Shape and marker locations after optimization. Optimized marker locations, $\tilde{m}_i$, are shown in red. Note that they have moved (see inset).

3.4.1 Defining a Marker Set

We assume that we know the number of markers and their approximate location relative to a reference template mesh. The only manual part of MoSh occurs if a user wants to use a new marker set. In this case they need to identify a template vertex for each marker. Notationally, we say a user creates a mapping $h(i)$ from marker indices, i, to vertex indices on the template. Each marker requires the user-specification of an expected distance d_i from the marker center to the skin surface. Both the location and the distance can be approximate since we optimize these for each subject.

To parameterize marker locations with respect to the body, we introduce a latent coordinate system that contains markers and our body model in a neutral pose, γ_0, θ_0, as in Fig. 3-1 (left). The purpose of this latent coordinate system is to model the relationship between the body surface and the markers in a pose-independent,

translation-independent, fashion. This relationship is then transferred to meshes in observed mocap frames.

We then denote the default position of the markers, v_i, as

$$v_i(\beta) \equiv S_{h(i)}(\beta, \theta_0, \gamma_0) + d_i N_{h(i)}(\beta, \theta_0, \gamma_0), \tag{3.1}$$

where $N_k(\beta, \theta, \gamma)$ indicates the vertex normal for index k given body model parameters. Thus $v_i(\beta)$ is the position of the model vertex, offset by a user-prescribed distance, d_i, from the surface, in the latent coordinate system, corresponding to marker i. These are illustrated as blue balls in Fig. 3-1.

Defining the marker set needs to be done once and then it is used for any subject captured with that marker set. For example, we did this once for the 47-marker Vicon set and used this for all mocap sequences in the CMU database.

3.4.2 Parameterizing Markers

The default markers, v_i, are approximate and below we optimize to solve for the body shape, β, and the actual location of the latent markers, $\tilde{m}_i$, for a given subject and mocap sequence. Let $\tilde{M}$ denote the collection of latent markers. Notationally, we use i to indicate marker number and t to indicate the mocap sequence frame number. Observed markers are denoted $m_{i,t}$ individually and M_t together. From a collection of M_t we estimate the latent markers $\tilde{M}$, shown as red balls in Fig. 3-1.

To that end, we define a function $\hat{m}(\tilde{m}_i, \beta, \theta_t, \gamma_t)$ that maps latent markers to the world given a particular shape, pose, and location of the body. We call these "simulated markers". Intuitively, we want to solve for the shape, pose, body location, and latent marker locations $\tilde{m}_i$ such that, when projected into the mocap sequence, the simulated markers match the observed markers M_t.

This requires a mapping from local surface geometry to a 3D marker position that can be transferred from the latent coordinate system to the observed markers resulting from different poses. We represent a marker position in an orthonormal basis defined by its nearest triangle in the latent coordinate system. We define that

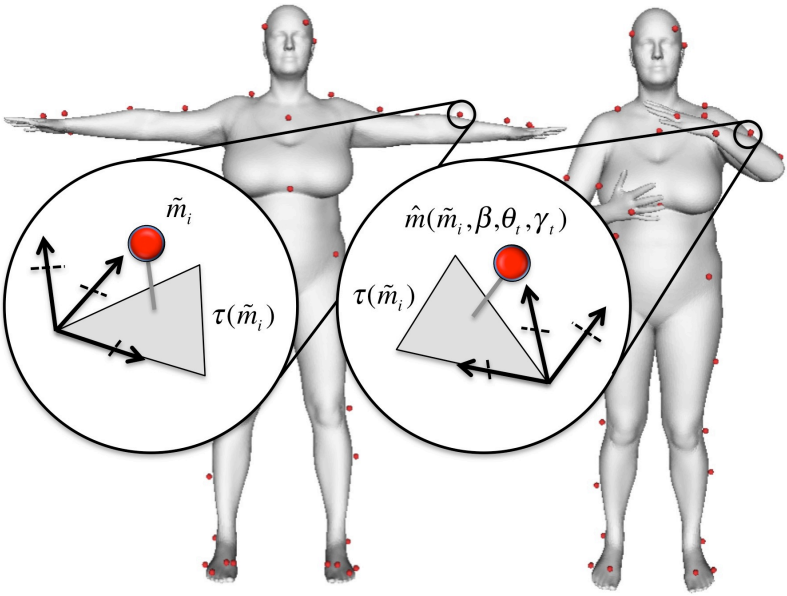


Figure 3-2: **Marker transformations.** In the latent coordinate space (left) we project a marker, $\tilde{m}_i$ into a basis defined by the nearest vertex: specifically by its normal, an arbitrary normalized edge, and the cross product between them. This provides a pose invariant representation for the marker. When the body pose changes (right), we then compute the location of the marker, $\hat{m}(\tilde{m}_i, \beta, \theta_t, \gamma_t)$, in the observed frame.

basis by three vectors: the triangle normal, one of the triangle's normalized edges, and the cross product between those two. This is geometrically depicted in Fig. 3-2 (left).

We denote the rigid transformation matrix that projects $\tilde{m}$ into the basis for closest triangle $\tau(\tilde{m})$ in the mesh, as $B_{\tau(\tilde{m})}(\beta, \theta, \gamma)$. We then define a simulated marker position $\hat{m}(\cdot)$ as

$$\hat{m}^*(\tilde{m}, \beta, \theta_t, \gamma_t) = B_{\tau(\tilde{m})}(\beta, \theta_t, \gamma_t) B^{-1}_{\tau(\tilde{m})}(\beta, \theta_0, \gamma_0) \tilde{m}^* \tag{3.2}$$

where $\tilde{m}^* = [\tilde{m}^T, 1]^T$ and $\hat{m}^*(\cdot) = [\hat{m}(\cdot)^T, 1]^T$ denote the marker locations in homo-

geneous coordinates. Equation 3.2 can be seen as having two steps. First, the matrix $B^{-1}_{\tau(\tilde{m})}(\beta, \theta_0, \gamma_0)$ transforms $\tilde{m}^*$ from a 3D *latent*-space position into a coordinate vector in the space of its local basis. In the second step, $B_{\tau(\tilde{m})}(\beta, \theta_t, \gamma_t)$ maps this coordinate vector into a 3D *observed*-space position, $\hat{m}^*(\cdot)$, defined by the specific position and pose, γ_t, θ_t. This is illustrated in Fig. 3-2 (right).

With the marker parameterization defined, we next define the objective functions we use to estimate marker positions, shape, pose, and nonrigid motion.

3.5 Objectives

Let sequences of body pose $\theta_{1..n}$, and position $\gamma_{1..n}$, with n time instants be denoted as Θ and Γ respectively. We wish to estimate the latent markers $\tilde{M}$, poses Θ, body locations Γ, and body shape β, such that the simulated markers $\hat{m}(\cdot)$, match the observed markers $m_{i,t}$. To do so we define an objective function with several terms.

The data term, E_D, is the sum of squared distances between simulated and observed landmarks:

$$E_D(\tilde{M}, \beta, \Theta, \Gamma) = \sum_{i,t} ||\hat{m}(\tilde{m}_i, \beta, \theta_t, \gamma_t) - m_{i,t}||^2. \tag{3.3}$$

Note that distances are measured in *cm*.

A surface distance energy term, E_S, encourages markers to keep a prescribed distance from the body surface in the latent coordinate system. Let $r(x, S)$ denote the signed distance of a 3D location x to the surface S. Then

$$E_S(\beta, \tilde{M}) = \sum_i ||r(\tilde{m}_i, S(\beta, \theta_0, \gamma_0)) - d_i||^2. \tag{3.4}$$

Since the marker locations are roughly known to begin with, we penalize estimated latent markers if they deviate from this. The energy term E_I regularizes the adjusted

marker towards its original position

$$E_I(\beta, \tilde{M}) = \sum_i ||\tilde{m}_i - v_i(\beta)||^2. \tag{3.5}$$

We also define pose and shape priors to regularize the estimation of body shape and pose. These are modeled as Gaussian, with their statistics $\mu_\beta, \mu_\theta, \Sigma_\beta, \Sigma_\theta$ computed from the pose and shape training data used to train our body model. We regularize β and θ_t by penalizing the squared Mahalanobis distance from the mean shape and pose:

$$E_\beta(\beta) = (\beta - \mu_\beta)^T \Sigma_\beta^{-1} (\beta - \mu_\beta) \tag{3.6}$$

$$E_\theta(\Theta) = \sum_t (\theta_t - \mu_\theta)^T \Sigma_\theta^{-1} (\theta_t - \mu_\theta). \tag{3.7}$$

We also add a velocity constancy term E_u that helps to smooth marker noise by a small amount:

$$E_u(\Theta) = \sum_{t=2}^{n} ||\theta_t - 2\theta_{t-1} + \theta_{t-2}||^2. \tag{3.8}$$

Our objective in total is the sum of these terms, each weighted by its own weight, λ:

$$E(\tilde{M}, \beta, \Theta, \Gamma) = \sum_{\omega \in \{D,S,\theta,\beta,I,u\}} \lambda_\omega E_\omega(\cdot). \tag{3.9}$$

3.6 Optimization

The objective function above is quite general and it enables us to solve a variety of problems depending on what we minimize and what we hold constant. In all cases, optimization uses Powell's dogleg method [80], with Gauss-Newton Hessian approximation. The gradients of the objective function are computed with algorithmic differentiation [81], which applies the chain rule to the objective function; for this we use an auto-differentiation package called Chumpy [82]. Only the differentiation of the body model $S_k(\beta, \theta, \gamma)$ and the signed mesh distance $r(x, S)$ were done by hand, to improve runtime performance.

There are two main optimization processes. The first estimates time-independent

parameters (body shape β and marker placements $\tilde{M}$), while the second estimates time-dependent parameters $\Theta = \{\theta_1 \dots \theta_n\}$, $\Gamma = \{\gamma_1 \dots \gamma_n\}$.

Body Shape and Latent Markers. For a given mocap sequence (or set of sequences for the same subject), optimization always starts by estimating the latent marker locations $\tilde{M}$, body shape β, poses Θ, and body positions Γ for a subset of the frames. The latent marker locations and the body shape are assumed to be time independent and can be estimated once for the entire sequence (or set of sequences).

Notably, the transformation from latent to observed coordinate systems is continuously re-estimated during the optimization of marker placement. The assignment of nearest neighbors, the local basis itself, and the coefficients relating a marker to that basis undergo continual adjustment to allow refinement of the relationship between markers and the body surface.

The λ values in Eq. 3.9 are: $\lambda_D = 0.75$, $\lambda_S = 100.0$, $\lambda_I = 0.25$, $\lambda_\beta = 1.0$, $\lambda_\theta = 0.25$, $\lambda_u = 0$.

The λ values were initialized to normalize each term by an estimate of its expected value at the end of the optimization; in particular, the distance-based λ values (λ_D, λ_S, λ_I) have interpretations as inverse variances with units of $\frac{1}{cm^2}$. These λ values were then empirically refined.

The velocity term is not used in this stage ($\lambda_u = 0$) because we are optimizing over random disconnected frames.

To help avoid local optima, the optimization is run in six stages, starting with strong regularization and then gradually decreasing this. Specifically, the regularization weights $\{\lambda_\theta, \lambda_\beta, \lambda_I\}$ are lowered from being multiplied by 40, then by 20, 10, 4, 2, and finally 1. Note that these regularization terms are linear and quadratic in contrast to the data term, which is non-linear. Similar to graduated non-convexity schemes, by increasing the regularization weights we make the objective function more convex, potentially helping the optimization avoid local optima during early stages of the process. In practice we found this to work well.

Computational cost increases with the number of frames used to estimate the

parameters since each frame requires its own pose θ_t. For efficiency we perform this optimization using a randomly selected subset of mocap time instants. We ran experiments with different numbers of randomly chosen frames and saw little improvement with more than 12 frames. Consequently we use 12 random frames for all experiments here.

Pose. Motion capture now becomes the problem of estimating the pose of the body, θ_t, and body position, γ_t, at each time instant given the known body shape and latent markers. We initialize the optimization at frame t with the solution at $t-1$ if it is available and then a short optimization is run for each time step.

For pose estimation, the λ values are now: $\lambda_D = 0.75$, $\lambda_S = 0$, $\lambda_I = 0$, $\lambda_\beta = 0$, $\lambda_\theta = 1.0$, $\lambda_u = 6.25$. Note that we now employ the velocity smoothness term, λ_u. A weight of zero means that this term is not used and the corresponding parameters are not optimized. Specifically, we do not optimize the marker locations or body shape. We do however use a pose prior, $\lambda_\theta = 1.0$, to penalize unlikely poses. Here we do not used the staged regularization because the optimization begins close to the minimum and converges quickly.

Pose and Soft Tissue Motion. In the optimization above we assume body shape and latent marker locations do not change. To capture soft tissue motions we now allow the body shape to vary across the sequence while keeping the marker transformation fixed. We still denote β as a shape estimated in the first stage, but now denote the time-varying deviations in shape from β as $B = \{\beta_1 \dots \beta_n\}$, such that a person's shape at time t is now $\beta + \beta_t$.

To regularize the β_t, we add one additional energy term to Eq. 3.9:

$$E_\Delta(B) = \sum_t ||\beta_t||^2 \tag{3.10}$$

and set λ_Δ to 0.25, adding $\lambda_\Delta E_\Delta(\cdot)$ in Eq. 3.9. This term allows body shape to change over time while regularizing it to not deviate too much from the person's "intrinsic shape", β.

While our body shape training set does not contain examples of soft tissue dynamics, it does capture many shape variations across the population. These are exploited to capture soft tissue deformations during motion. Someone inhaling, for example, might look like a different person with a higher chest or a bigger stomach. When someone jumps up and down, their chest changes in ways that resemble the chests of other people. It is interesting, and perhaps surprising, that the shape variations between people can be used to approximate the shape variation of an individual due to dynamics. Presumably there are soft-tissue deformations that cannot be explained this way but, given sufficiently many training body shapes, and sufficiently many principal components, we posit that a wide range of such deformations are representable. We suspect, however, that training shapes specific to soft-tissue deformations could be used to learn a more concise model. Note further that we do not *model* dynamics of soft tissue, we only approximate what is present in the mocap marker data.

Since standard marker sets are designed for estimating a skeleton, the markers are mostly placed on rigid body structures to minimize soft tissue motion. This is another reason why existing mocap methods lack nuance. Consequently *to capture soft tissue dynamics, we want just the opposite;* we must have markers on the soft tissue. We consider this below.

Run Time. Shape and marker estimation requires about 7 minutes. Pose estimation without soft tissue estimation takes about 1 second per frame; pose estimation with soft tissue estimation requires about 2 seconds per frame.

3.7 Marker Selection

Body shape estimation from motion capture depends on the number and placement of markers; here we propose a method for constructing a new marker set to improve body surface reconstruction. To be practical a marker set must be simple, make sense to the technician applying it, be repeatable across subjects, and take into account self occlusion, self contact, and the impact on subject movement. Consequently we start

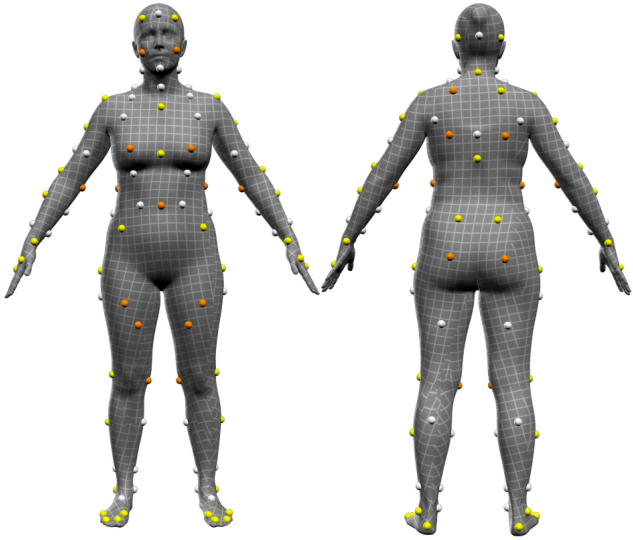


Figure 3-3: **Marker sets.** The union of all markers illustrates the 114 possible markers we considered. Yellow markers correspond to a standard 47-marker Vicon set. The 20 orange markers were found to improve shape estimation the most. The union of yellow and orange markers corresponds to our 67-marker set used for capturing shape and soft-tissue motion. White markers were deemed redundant and were not used.

with a standard marker set and propose additional symmetrical marker locations for a total of 114 candidate markers (Fig. 3-3).

We then evaluate these putative markers to determine how important the different markers are for shape recovery. For this we use a set of 165 meshes of 5 females of different shapes in a variety of poses selected from the FAUST dataset [18]. A template mesh is aligned to each of the 3D scans resulting in a set of registered meshes, R^z, $z = 1 \ldots 165$, in which all vertices are in correspondence across the 165 instances. We associate our 114 markers with vertices of the template and then estimate body shape from different subsets of the markers. We evaluate the accuracy of the result in terms of the Euclidean distance between the vertices of the estimated and true mesh. Specifically we compute the root mean squared error (RMSE) over all the vertices

(including the subset used for fitting) for all meshes.

More formally, given a maximum number of markers, c, we seek a subset, T, of the mesh vertices, A, that enables the most accurate estimation of body shape. This subset T is the one that minimizes a cost $E_M(T)$; that is

$$T^* = \underset{T \subseteq A, |T|=c}{\arg\min}\ E_M(T). \tag{3.11}$$

Notationally, we will now abbreviate body model parameters $\{\beta, \theta, \gamma\}$ as P. We will also denote vertex k of registered mesh z as R_k^z. The best parameters $P^*(\{R_j^z | j \in T\})$, given access only to subset T of the vertices for registered mesh z, are defined as

$$P^*(\{R_j^z | j \in T\}) = \arg\min_P \sum_{i \in T} ||S_i(P) - R_i^z||^2. \tag{3.12}$$

The cost of choosing subset T takes into account the distance between all vertices $i \in A$ across all the registered meshes $z \in Z = \{1 \dots 165\}$

$$E_M(T) = \sum_{i \in A, z \in Z} ||S_i(P^*(\{R_j^z | j \in T\})) - R_i^z||^2. \tag{3.13}$$

Note that the RMSE is $(E_M(T)/(|A||Z|))^{1/2}$.

Evaluating all possible subsets of 114 markers is infeasible so we take a greedy approach. If we currently have N markers, we remove one, evaluate the cost for the $N-1$ possible sets, and select the deleted marker that produces the lowest error. We remove this marker and repeat.

Figure 3-3 shows all 114 putative markers. The standard 47-marker set is in yellow. White and orange markers correspond to the set of additional markers that we considered. Using the greedy method, we found that the white markers were not as useful for estimating shape as the orange ones. Figure 3-4 shows a plot of the RMSE for different numbers of markers. Note that here we start with the 47-marker set and subtract markers from it and add markers to it. Surprisingly one can remove markers from the standard set and still obtain reasonable shape estimates down to about 25

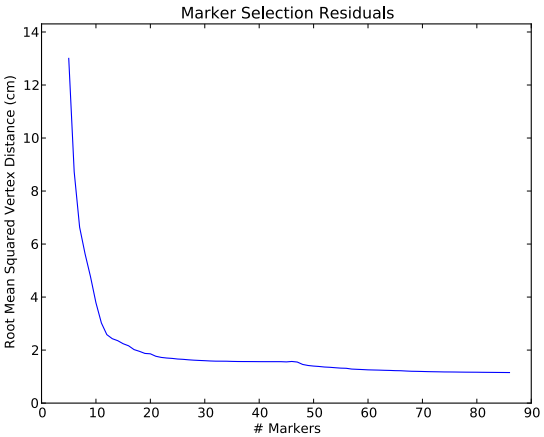


Figure 3-4: **Marker selection residuals.** The plot shows the mesh shape reconstruction error as a function of marker count.

markers. We decided to keep the original set and add the 20 additional (orange) markers. The addition of markers to the 47 results in a noticeable decrease in RMSE. Note that we could obtain similar error to our set of 67 with fewer markers by dropping some of the original 47. To enable comparison with CMU results, however, we decided to preserve the 47 and add to this set.

3.8 Results

3.8.1 Quantitative Shape Analysis

We evaluate the first stage of optimization, which computes the body shape and marker locations. To compare estimated body shapes to real ones, we scanned 20 subjects using a high-resolution 3D body scanner (3dMD LLC, Atlanta, GA). Before scanning, all subjects gave informed written consent. Additionally, 10 of the subjects were professional models who signed modeling contracts that allow us to release their full scan data.

We also used a Vicon mocap system (Vicon Motion Systems Ltd, Oxford, UK) to capture subjects with 89 markers. The 89 markers were selected using the marker optimization analysis from the full set of 114 evaluated in Sec. 3.7. We use at most 67 markers for shape and pose estimation; unused markers prove valuable to evaluate held-out marker error. In all cases we used the optimization with soft-tissue deformation. We processed, and evaluate error using, a total of 73 mocap sequences.

Our goal is to estimate a body shape that minimizes 3D body shape reconstruction error. We measure this error in two different ways: as held-out marker error and as mesh registration error. Held-out marker error reveals how well we can predict marker locations that were not used by the optimization: for example, if we use 47 of the markers to estimate the body shape then we use the remaining markers to estimate held-out error. As shown in Fig. 3-5 (right), the mean distance for held-out markers drops to approximately 3.4cm when we use 67 markers. Note that these errors include deviations in placing markers on a subject, which can easily exceed a centimeter. Specifically, when we estimate shape from a subset of markers, we do not optimize the placement of the held-out markers. So this error combines human placement error with errors in soft-tissue motion of the held-out markers that are not predicted by the subset used for fitting.

After about 25 markers the improvement is very gradual. This is interesting because it suggests that small marker sets can give good estimates of body shape. Note that this evaluation uses all 73 mocap sequences and hence evaluates how well MoSh explains marker motions due to changes in both shape and pose.

Example 3D scans of several subjects are shown in Fig. 3-6 (row 1). For each subject we align a template mesh to the scan and this template mesh has the same topology as the MoSh body model (Fig. 3-6 row two); this produces a registered mesh that we use for evaluation. Note that the registered meshes faithfully represent the scans and conform to the mesh topology of our model but do not have holes. Registration error is a measure of how well we can explain a subject's registered mesh in terms of average vertex-to-vertex mesh distance. Recovered body shapes using 67 markers are shown in Fig. 3-6 row three. Here we pose the MoSh result in the same

pose as the scan. Given that MoSh results in a shape vector β, we adjust $\{\theta, \gamma\}$ for a body model to minimize model-to-registration distance. The heat map in the bottom row of Fig. 3-6 shows the distance from the MoSh shape to the registered mesh, illustrating how well MoSh approximates the shape from 67 markers.

This registration error is shown in Fig. 3-5 (left). Registration error behaves much like held-out marker error, except it is uniformly smaller. Unlike the held-out experiment, here we only need to explain shape and not both pose and shape. Shape estimates are obtained from 12 mocap frames and are well constrained.

While large marker sets like those used in [58] certainly contain more information, we see in Fig. 3-5 (left) diminishing returns with larger marker sets. The ideal number of markers is likely related to the resolution of the mesh.

To give some insight into what these numbers mean, Fig. 3-7 shows body shape for one subject reconstructed using different numbers of markers. Here we selected markers based on our greedy evaluation strategy. What is surprising is that with only 10 markers, we get a shape that roughly captures the person's size. Note that the registration error decreases as we add more markers; the numerical results show the registration error in m.

For the 10 models, scans, aligned meshes, mocap sequences, and MoSh fits are provided for research purposes here:

`http://ps.is.tuebingen.mpg.de/project/MoSh`

This data allows others to estimate shape from the same sequences and compare with both the ground truth shape and our results.

3.8.2 Archival Mocap (CMU)

While we do not have ground truth shape for the CMU dataset, we can evaluate results qualitatively. A visual inspection of shape recovery from CMU can be seen in Fig. 3-8, where video frames are shown above the bodies and poses estimated from 47 standard markers. To be clear, MoSh does not use this video frame; we show it here only for a visual evaluation of rough shape. Since the CMU dataset has no anthropometric data, a quantitative evaluation is not possible.

3.8.3 Gender Estimation

For the above CMU results we used sequences for which the gender of the subject could be determined using accompanying video footage. Next we ask whether we can estimate gender from the markers automatically (cf. [72]). We use a linear support vector machine to predict gender from body model parameters. First we fit a gender-neutral body model to all subjects in the CAESAR dataset to obtain linear shape coefficients. We then train the SVM to predict known gender given the shape parameters. We then evaluate gender classification on body shape parameters estimated by MoSh from the CMU dataset with the gender-neutral body model. For the 39 subjects with known gender we correctly predicted it 89.7% of the time; this is comparable to [72], which is not surprising since both methods rely on essentially the same kind of marker data.

3.8.4 Pose Estimation Results

Given our estimate of intrinsic shape, β, and the marker locations, $\tilde{M}$, we now optimize the pose across a mocap sequence. We compute the pose for 39 subjects across 722 different mocap sequences in the CMU dataset. Figure 3-9 shows some representative frames from some representative sequences in the CMU dataset. Even with 47 markers we can capture some soft tissue deformation and the results shown here allow body shape deformation over time. The visual nuance of pose reconstruction is difficult to illustrate in a static image but is apparent in the **accompanying video**. Note that this is fully automatic.

The best way to evaluate accuracy of pose and shape together is in terms of held out marker error. For this we used 20 subjects and 73 mocap sequences acquired with our extended marker set. We use 67 markers for estimation and 22 to compute held-out error. This error is $3.4cm$ and corresponds to the rightmost point on the right plot in Fig. 3-5 (right).

With a small marker set, noise in any one marker can have an impact. In the shape estimation stage, the shape and marker placement are estimated from many

poses, so variation in any individual marker should not unduly harm shape or marker placement estimation. During pose estimation, velocity constancy helps reduce the effect of single marker noise. Future work should address methods to automatically detect and downweight missing markers or markers that have moved.

3.9 Soft Tissue Deformation Results

Our body model was learned to represent both shape and pose-dependent deformations from registered meshes of static subjects. Many other subtle body shape deformations were not explicitly learned by our model, including static muscle contraction, breathing, gravity, external forces, and dynamics. What we show is that the space of body shapes learned from different people captures variations in shape that can approximate soft tissue motions. Note that we do not *model* the dynamics of soft tissue. We only fit the effects of such motions that are apparent in the marker data.

Figure 3-10 shows examples from several sequences. We show the estimated body shape with a single body shape, β, per subject (left image in each pair) and the results allowing deviations, β_t, from this shape (right image in each pair). Note the markers on the chest and belly. Red are the simulated markers predicted by our model and green are the observed markers. With changing body shape, we more accurately fit the markers undergoing soft-tissue deformation. This is not surprising, but what is important is that the shape remains "natural" and continues to look like the person.

Numerically we see the mean observed marker error go down from 0.79cm to 0.62cm with dynamics. Again this is not surprising since we are allowing the shape to deform to fit these markers. We also tested held out marker error; these are markers that were not used to estimate shape. Here too we see the mean error go from 3.41cm to 3.39cm. This is not a significant improvement, but rather a validation that fitting the soft-tissue motion does not hurt held-out marker error. This confirms our subjective impression that the body shape does not deform unnaturally and the non-rigid motions, away from the tracked markers, reflect realistic body deformations. While, of course, we cannot capture fine ripples with a sparse set of markers, it is

surprising how much realistic deformation MoSh can estimate.

See the **accompanying video** for better visualizations and more results. In the video one sees the observed markers "swimming" around relative to the estimated shape when we do not model dynamics. There we also compare 47 markers with our 67-marker set and find that the extra markers placed on the soft tissue are important.

3.9.1 Exaggerated Soft-Tissue Deformation

Our soft tissue deformations correspond to directions in the space of human body shapes. We can vary the amount of deformation along these directions to either attenuate or amplify the effect. Specifically we magnify the 3D motion by multiplying β_t by a user-specified constant to exaggerate the soft tissue deformations.

This is difficult to show in print but the **video** shows examples of the same sequence with different levels of exaggeration. We found that we could magnify the deformations by a factor of 1.5 or 2 while retaining something like natural motion. Pushing the exaggeration by a factor of 4 sometimes produce interesting effects and, other times, unnatural body shapes.

This tool could be useful to animators to produce reference material since it highlights how soft tissue deforms. It could also be used to create new effects that exaggerate human actions but in a way that is based on physically realistic deformations.

3.9.2 Soft-Tissue Retargeting

An important use of skeletal mocap data is the retargeting of motion to a new character; the same can be done with MoSh. Consider the stylized characters in Fig. 3-11 that were downloaded from the Internet. For each character, we deform our template towards the character using regularized registration, initialized by hand-clicked correspondences. To model shape deformations from this character mesh, we simply recenter our PCA model of body shape by replacing our original mean shape, μ, with the character's template deformations. The soft tissue deformation coefficients, β_t, are then simply applied to this new mean shape. We also directly apply the estimated

translation, γ_t, and MoSh part rotations, θ_t, to the parts of the new character along with the learned non-rigid pose-dependent shape deformations. This produces plausible animations. Note that, to get realistic soft-tissue transfer, we use human actors with body shapes that resemble the stylized character; see Fig. 3-11. Of course, these deformations can also be exaggerated.

3.10 Conclusion and Discussion

MoSh addresses a key criticism of existing motion capture methods. By estimating a changing body shape over time from sparse markers, MoSh captures detailed non-rigid motions of the body that produce lifelike animations. MoSh is completely compatible with existing industry-standard mocap systems. It can be used alone or in conjunction with traditional skeletal mocap since no information is lost and MoSh can use exactly the same markers as current systems. Our hope is that MoSh breathes new life into old mocap datasets and provides an easily adopted tool that extends the value of existing investments in marker-based mocap.

There are several current limitations that present interesting directions for future work. For example, we need to roughly know the marker set and we also assume the markers are in correspondence. We can correct for some mislabeled markers but we still assume a largely labeled dataset. Establishing correspondence and cleaning markers sets is a time consuming part of current mocap practices. It would be interesting to leverage the body model to try to solve these problems automatically. For example, we could also use our simulated markers to detect when a marker is missing or has moved. If a marker moves between sessions we could then update its location on the fly. We could also estimate the noise in each marker independently and take this into account during pose and shape estimation. The estimated body pose could also be used to create a virtual marker sequence that could replace the original. This would provide a principled way of fixing occlusions. Simulating a different set might be useful for methods that extract skeletal data from markers.

The quality of MoSh output is very dependent on the quality of the body model

that is used. If our model cannot represent a pose realistically, then the output of MoSh will have artifacts. We observed this for a few poses, for example, both arms pointed forward, elbows straight and palms together. This suggests our pose training set should be augmented with new poses.

An interesting direction for future work would be to use other types of body models. For example, it should be possible to replace our model with one that uses linear blend skinning and corrective blend shapes.

Our method for evaluating new marker sets could be used to construct sets to capture specific types of non-rigid deformations such as breathing. If we had 3D mesh *sequences* we could extend our analysis to select marker sets directly relevant for capturing soft-tissue motion. We did not evaluate which poses are most effective for estimating body shape; we simply chose 12 at random. Jointly optimizing the marker set and the poses could make a mocap system a more effective "body scanner;" the body scanning protocol would involve attaching the markers and having the subject assume the prescribed poses.

Our soft-tissue motions are approximations based on sparse markers but result in dense deformations. Since it is easy to acquire the data, it would be interesting to use these to train a more physical model of how soft tissue moves. That is, possibly we could leverage MoSh to learn a more sophisticated body shape model with dynamics. This could allow generalization of soft-tissue motions to new body shapes and movements.

We plan to extend our body model and MoSh methods to include the motion of feet, hands and faces. We think this is relatively straightforward but likely requires a more sophisticated pose prior model than the Gaussian one used here. It may be possible to extend these ideas further for capturing clothing or to couple our marker-based analysis with video or range data. Finally, we are also working on speeding up processing using a multi-resolution model to enable the use of MoSh in virtual production.

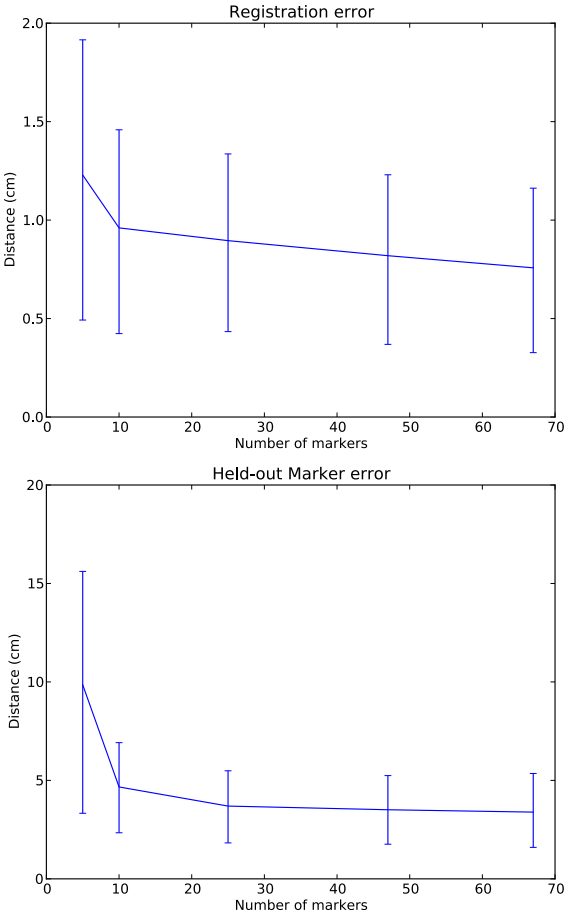


Figure 3-5: **Effects of marker number on reconstruction error.** The mean and standard deviations of distance residuals indicate how the marker number affects reconstruction. *Top:* Shape reconstruction error. This is computed as the mean absolute distance between the true body shape (as represented by the alignment of the template to a scan) and the body shape estimated by MoSh reposed to match the registered mesh. *Bottom:* Held-out marker error across all sequences. This measures errors in both shape and pose but is inflated by marker placement error and marker movement. In both plots, 68.2% ($\pm 1\sigma$) of the residuals are contained between the error bars.

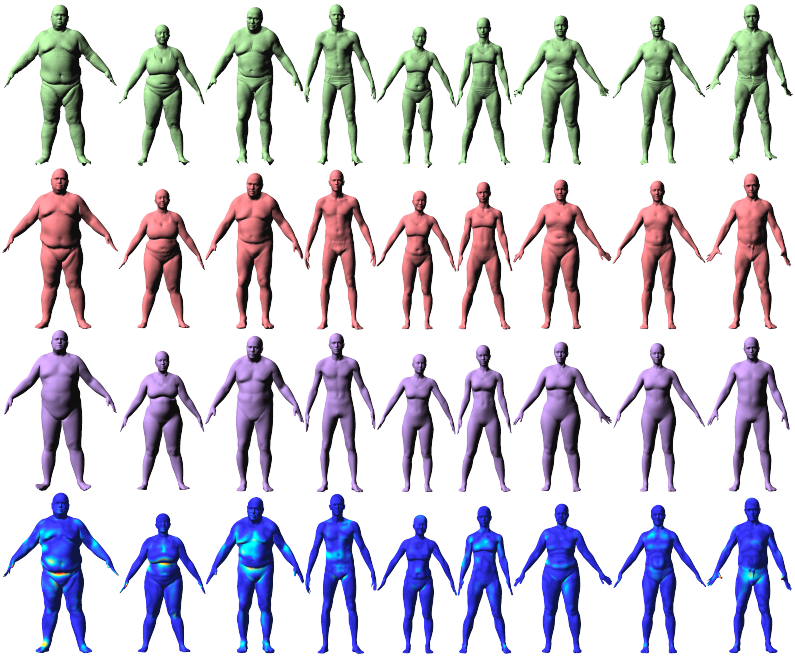


Figure 3-6: **Shape reconstruction.** First row: raw 3D scans from a high-resolution scanner. Second row: registered meshes obtained by precisely aligning a template mesh, with the same topology as our model, to the scans. These registered meshes faithfully capture the body shape and are used for our quantitative analysis. Third row: our model with shape, β, estimated from only 67 markers. Here we estimate the pose, θ, of our model to match the registered meshes to facilitate comparison. Bottom row: Distance between second and third rows. The heat map shows Euclidean distance from the registered mesh to the nearest point on the surface of the body estimated by MoSh; blue means zero and red means ≥ 4 cm.

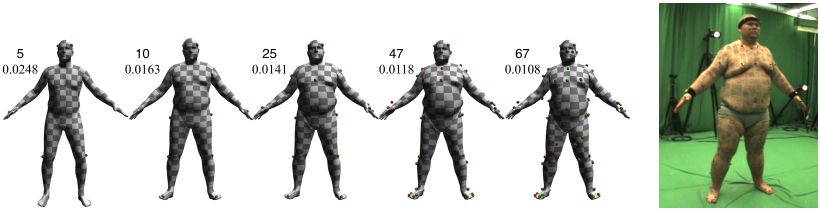


Figure 3-7: **Shape from markers.** We show the effect of the number of markers (5, 10, 25, 47, 67) on the registration error (in m) of the estimated shape. Far right: reference image of the subject.

Figure 3-8: **CMU bodies.** Extracted shapes (bottom) and reference images (top) for several CMU subjects. Shape and pose is computed with MoSh using 47 Vicon markers only.

Figure 3-9: **CMU mocap.** Example meshes extracted from the CMU mocap dataset and representative frames from the animation. All shapes and poses are estimated automatically using only 47 markers. See **accompanying video** to see these and other results for CMU.

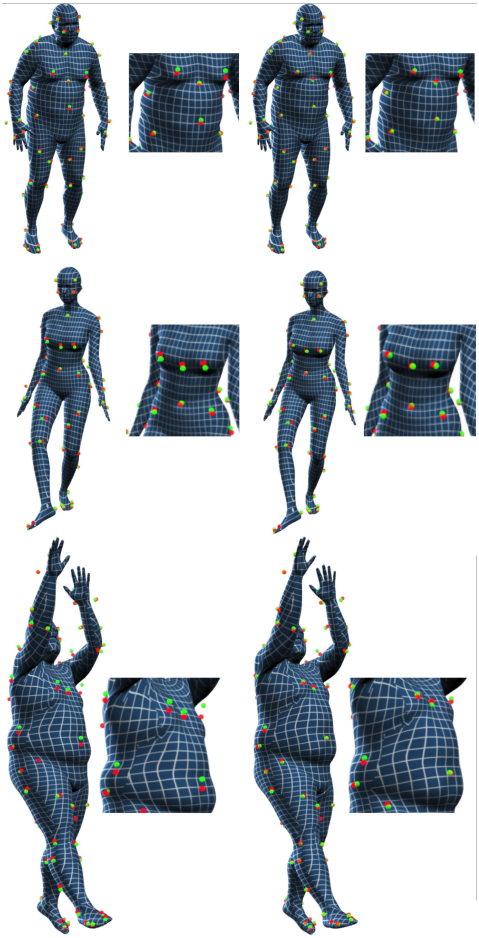


Figure 3-10: **Motion of soft tissue.** Some representative samples are shown. In each pair, the left image is without modeling dynamics (body shape fixed) and the right with with dynamics (body shape varying). Each image shows the full body and a detail region. Green balls correspond to the mocap markers. Red balls correspond to the simulated marker locations. Allowing body shape to change over time better captures soft tissue deformations. Note that, with dynamics, the predicted markers much more closely match the observed markers.

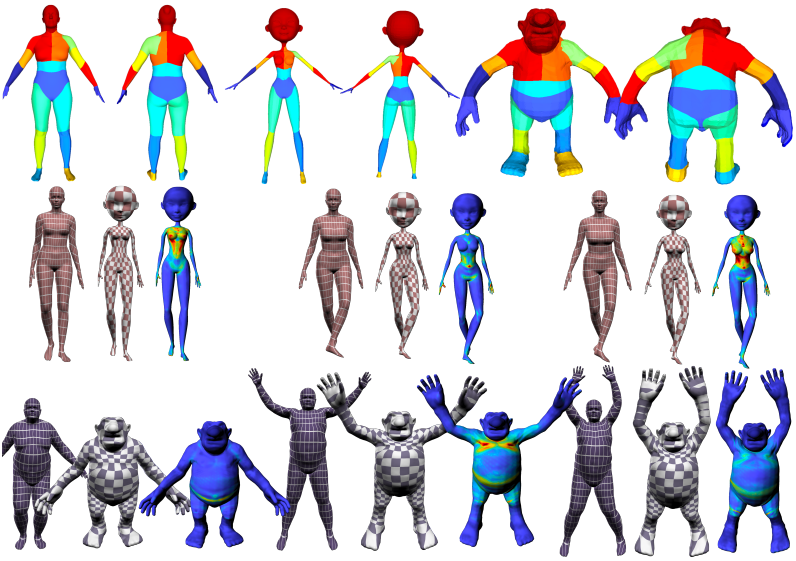


Figure 3-11: **Retargeting soft-tissue motions.** Top row: Body part segmentation for human and stylized characters. Middle row: retargeting pose and soft-tissue motion of an actor (left) to a stylized female character (middle), with heat maps (right) illustrating the percentage of soft-tissue deformation; blue means zero and red means $\geq$ 20 percent deformation. Bottom row: retargeting to another stylized character. See **accompanying video** to visualize the soft-tissue motions.

Chapter 4

Shape Estimation from RGB and Range Images

4.1 Introduction

Computer vision as *analysis by synthesis* has a long tradition [83, 84] and remains central to a wide class of generative methods. In this top-down approach, vision is formulated as the search for parameters of a model that is *rendered* to produce an image (or features of an image), which is then compared with image pixels (or features). The model can take many forms of varying realism but, when the model and rendering process are designed to produce realistic images, this process is often called *inverse graphics* [85, 86]. In a sense, the approach tries to reverse-engineer the physical process that produced an image of the world.

We define an observation function $f(\Theta)$ as the forward rendering process that depends on the parameters Θ. The simplest optimization would solve for the parameters minimizing the difference between the rendered and observed image intensities, $E(\Theta) = \|f(\Theta) - I\|^2$. Of course, we will specify much more sophisticated functions, including robust penalties and priors, but the basic idea remains – minimize the difference between the synthesized and observed data. While much has been written about this process and many methods fall under this rubric, few methods literally adopt the inverse graphics approach. High dimensionality makes optimizing an objective like

the one above a challenge; renderers have a large output space, and realistic renderers require a large input parameter space. Fundamentally, the forward rendering function is complex, and optimization methods that include it are often purpose-built with great effort. Put succinctly, *graphics renderers are not usually built to be inverted.*

Here we fully embrace the view of vision as inverse graphics and propose a framework to make it more practical. Realistic graphics engines are available for rendering the forward process and many discriminative approaches exist to recover scene properties directly from images. Neither explicitly models how the observables (pixels or features) smoothly change with model parameters. These derivatives are essential for optimization of high-dimensional problems and constructing these derivatives by hand for each application is onerous. Here we describe a general framework based on differentiating the render. We define a *differentiable renderer (DR)* as a process that (1) supplies pixels as a function of model parameters to simulate a physical imaging system and (2) supplies derivatives of the pixel values with respect to those parameters. To be practical, the DR also has to be fast; this means it must have hardware support. Consequently we work directly with OpenGL. Because we make it publicly available, we call our framework *OpenDR* (`http://open-dr.org`).

Since many methods formulate generative models and differentiate them, why has there been no general DR framework until now? Maybe it is because rendering seems like it is not differentiable. At some level this is true, but the question is whether it matters in practice. All renderers are approximate and our DR is no exception. We describe our approximations in Sections 4.3 and 4.4 and argue that, in practice, "approximately differentiable" is actually very useful.

Our goal is not rendering, but inverse rendering: we wish to specify and minimize an objective, in which the renderer is only one part. To that end, our DR is built upon a new autodifferentiation framework, called Chumpy, in Python that makes programming compact and relatively easy. Our public autodiff framework makes it easy to extend the basic features of OpenDR to address specific problems. For example, instead of specifying input geometry as vertices, one might parameterize the vertices in a shape space; or in the output, one might want a Laplacian pyramid of pixels, or

edges, or moments, instead of the raw pixel values. While autodifferentiation does not remove the need to write these functions, it does remove the need to differentiate them by hand.

Using this we define the OpenDR framework that supports a wide range of real problems in computer vision. The OpenDR framework provides a compact and efficient way of expressing computer vision problems without having to worry about how to differentiate them. This is the first publicly-available framework for differentiating the image generation process.

To evaluate the OpenDR, and to illustrate how to use it, we present two examples. The first is a simple "hello world" example, which serves to illustrate the basic ideas of the OpenDR. The second, more complex, example involves fitting an articulated and deformable model of 3D human body shape to image and range data from a Kinect. Here we optimize 3D body shape, pose, lighting, albedo, and camera parameters. This is a complex and rich generative model and optimizing it would generally be challenging; with OpenDR, it is straightforward to express and optimize.

While differentiating the rendering process does not solve the computer vision problem, it does address the important problem of local refinement of model parameters. We see this as piece of the solution that is synergistic with stochastic approaches for probabilistic programming [87]. We have no claim of novelty around vision as inverse graphics. Our novelty is in making it practical and easy to solve a fairly wide class of such problems. We believe the OpenDR is the first generally available solution for differentiable rendering and it will enable people to push the analysis-by-synthesis approach further.

4.2 Related Work

The view of vision as *inverse graphics* is nearly as old as the field itself [85]. It appears in the work of Grenander on *analysis by synthesis* [83], in physics-based approaches [88], in regularization theory [89, 90], and even as a model for human perception [91, 84, 92]. This approach plays an important role in Bayesian models and today

the two notions are tightly coupled [93]. In the standard Bayesian formulation, the likelihood function specifies the forward rendering process, while the prior constrains (or regularizes) the space of models or parameters [93]. Typically the likelihood does not involve an actual render in the standard graphics sense. In graphics, "inverse rendering" typically refers to recovering the illumination, reflectance, and material properties from an image (e.g. the estimation of BRDFs); see [94] for a review. When we talk about inverting the rendering process we mean something more general, involving the recovery of object shape, camera parameters, motion, and illumination.

The theory of inverse graphics is well established, but what is missing is the direct connection between rendering and optimization from images. Graphics is about synthesis. Inference is about going from observations to models (or parameters). *Differentiable rendering* connects these in a concrete way by explicitly relating changes in the observed image with changes in the model parameters.

Stochastic Search and Probabilistic Programming. Our work is similar philosophy to Mansinghka et al. [87]. They show how to write simple probabilistic graphics programs that describe the generative model of a scene and how this relates to image observations. They then use automatic and approximate stochastic inference methods to infer the parameters of the scene model from observations. While we share the goal of automatically inverting graphics models of scenes, our work is different and complimentary. They address the stochastic search problem while we address the deterministic refinement problem. While stochastic sampling is a good way to get close to a solution, it is typically not a good way to refine a solution. A full solution is likely to incorporate both of these elements of search and refinement, where the refinement stage can use richer models, deterministic optimization, achieve high accuracy, and be more efficient.

Our work goes beyond [87] in other ways. They exploit a very general but computationally inefficient Metropolis-Hastings sampler for inference that will not scale well to more complex problems. While their work starts from the premise of doing inference with a generic graphics rendering engine, they do not cope with 3D shape,

illumination, 3D occlusion, reflectance, and camera calibration; that is, they do no render graphics scenes as we typically think of them. None of this is to diminish the importance of that work, which lays out a framework for probabilistic scene inference. This is part of a more general trend in probabilistic programming where one defines the generative graphical model and lets a generic solver do the inference [95, 96, 97]. Our goal is similar but for deterministic inference. Like them we offer a simple programming framework in which to express complex models.

Recently Jampani et al. [98] define a generic sampler for solving inverse graphics problems. They use discriminative methods (bottom up) to inform the sampler and improve efficiency. Their motivation is similar to ours in that they want to enable inverse graphics solutions with simple generic optimization methods. Their goal differs however in that they seek a full posterior distribution over model parameters, while we seek a local optimum. In general, their method is complimentary to ours and the methods could be combined.

Differentiating Graphics Models. Of course we are not the first to formulate a generative graphics model for a vision problem, differentiate it, and solve for the model parameters. This is a tried-and-true approach in computer vision. In previous work, however, this is done as a "one off" solution and differentiating the model is typically labor intensive. For a given model of the scene and particular image features, one defines an observation error function and differentiates this with respect to the model parameters. Solutions obtained for one model are not necessarily easily applied to another model. Some prominent examples follow.

Face modeling: Blanz and Vetter [99] define a detailed generative model of human faces and do analysis by synthesis to invert the model. Their model includes 3D face shape, model texture, camera pose, ambient lighting, and directional lighting. Given model parameters they synthesize a realistic face image and compare it with image pixels using sum-of-squared differences. They explicitly compute derivatives of their objective function and use a stochastic gradient descent method for computational reasons and to help avoid local optima.

3D shape estimation: Jalobeanu et al. [100] estimate underlying parameters (lighting, albedo, and geometry) of a 3D planetary surface with the use of a differentiated rendering process. They point out the importance of accurate rendering of the image and the derivatives and work in *object space* to determine visibilities for each pixel using computational geometry. Like us, they define a differentiable rendering process but with a focus on Bayesian inference.

Smelyansky et al. [101] define a "fractional derivative renderer" and use it to compute camera parameters and surface shape together in a stereo reconstruction problem. Like [100], they use geometric modeling to account for the fractional contributions of different surfaces to a pixel. While accurate, such a purely geometric approach is potentially slow.

Bastian [102] also argues that working in object space avoids problems of working with pixels and, in particular, that occlusions are a problem for differentiable rendering. He suggests super-sampling the image as one solution to approximate a differentiable render. Instead he uses MCMC sampling and suggests that sampling could be used in conjunction with a differentiable renderer to avoid problems due to occlusion. See also [103], which addresses similar issues in image modeling with a continuous image representation.

It is important to remember that any render only produces an approximation of the scene. Consequently any differentiable render will only produce approximations of the derivatives. This is true whether one works in object space or pixel space. The question is how good is the approximation and how practical is it to obtain? We argue below that pixel space provides the better tradeoff.

Human pose and shape: Sminchisecu [104] formulates the articulated 3D human tracking problem from monocular video. He defines a generative model of edges, silhouettes and optical flow and derives approximations of these that are differentiable. In [105] Sminchisescu and Telea define a generic programming framework in which ones specifies models and relates these to image observations. This framework does not automatically differentiate the rendering process.

de La Gorce et al. [106] recover pose, shape, texture, and lighting position in a hand

tracking application. They formulate the problem as a forward graphics synthesis problem and then differentiate it, paying special attention to obtaining derivatives at object boundaries; we adopt a similar approach. Weiss et al. [52] estimate both human pose and shape using range data from Kinect and an edge term corresponding to the boundary of the human body. They formulate a differentiable silhouette edge term and mention that it is sometimes not differentiable, but that this occurs at only finitely many points, which can be ignored.

The above methods all render a model of the world and differentiate some image error with respect to the model parameters. Despite the fact that they all can be seen as inverse rendering, in each case the authors formulate an objective and then devise a way to approximately differentiate it. Our key insight is that, instead of differentiating each problem, we *differentiate the render*. Then any problem that can be posed as rendering is, by construction, (approximately) differentiable. To formulate a new problem, one writes down the forward process (as expressed by the rendering system), the derivatives are given automatically, and optimization is performed by one of several local optimization methods. This approach of differentiating the rendering process provides a general solution to many problems in computer vision.

4.3 Defining our Forward Process

Let $f(\Theta)$ be the rendering function, where Θ is a collection of all parameters used to create the image. Here we factor Θ into vertex locations V, camera parameters C, and per-vertex brightness A: therefore $\Theta = \{V, C, A\}$. Inverse graphics is inherently approximate, and it is important to establish our approximations in both the forward process and its differentiation. Our forward model makes the following approximations:

Appearance (A): Per-pixel surface appearance is modeled as product of mipmapped texture and per-vertex brightness, such that brightness combines the effects of reflectance and lighting. Spherical harmonics and point light sources are available as part of OpenDR; other direct lighting models are easy to construct. Global illumi-

nation, which includes interreflection and all the complex effects of lighting, is not explicitly supported.

Geometry (V): We assume a 3D scene to be approximated by triangles, parameterized by vertices V, with the option of a background image (or depth image for the depth renderer) to be placed behind the geometry. There is no explicit limit on the number of objects, and the DR does not even "know" whether it is rendering one or more objects; its currency is triangles, not objects.

Camera (C): We approximate continuous pixel intensities by their sampled central value. We use the pinhole-plus-distortion camera projection model from OpenCV. Its primary difference compared with other projections is in the details of the image distortion model [107], which are in turn derived from [108].

Our approximations are close to those made by modern graphics pipelines. One important exception is appearance: modern graphics pipelines support per-pixel assignment on surfaces according to user-defined functions, whereas here we support *per-vertex* user-defined functions (with colors interpolated between vertices). While we also support texture mapping, we do not yet support differentiation with respect to intensity values on the texture map. Unlike de La Gorce [106], we do not support derivatives with respect to texture; whereas they use bilinear interpolation, we would require trilinear interpolation because of our use of mipmapping. This is future work.

We emphasize that, if the OpenDR proves useful, users will hopefully expand it, relaxing many of these assumptions. Here we describe the initial release.

4.4 Differentiating our Forward Process

To describe the partial derivatives of the forward process, we introduce U as an intermediate variable indicating 2D projected vertex coordinate positions. Differentiation follows the chain rule as illustrated in Fig. 4-1. Our derivatives may be grouped into the effects of appearance ($\frac{\partial f}{\partial A}$), and changes in projected coordinates ($\frac{\partial U}{\partial C}$ and $\frac{\partial U}{\partial V}$), and the effects of image-space deformation ($\frac{\partial f}{\partial U}$).

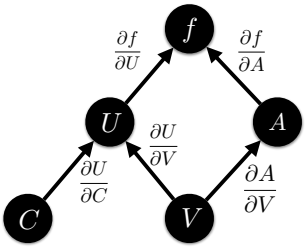


Figure 4-1: Partial derivative structure of the renderer.

4.4.1 Differentiating Appearance

Pixels projected by geometry are colored by the product of texture T and appearance A; therefore $\frac{\partial f}{\partial A}$ can be quickly found by rendering the texture-mapped geometry with per-vertex colors set to 1.0, and weighting the contribution of surrounding vertices by rendered barycentric coordinates. Partials $\frac{\partial A}{\partial V}$ may be zero (if only ambient color is required), may be assigned to built-in spherical harmonics or point light sources, or may be defined directly by the user.

4.4.2 Differentiating Projection

Image values relate to 3D coordinates and camera calibration parameters via 2D coordinates; that is, where U indicates 2D coordinates of vertices,

$$\frac{\partial f}{\partial V} = \frac{\partial f}{\partial U}\frac{\partial U}{\partial V}, \tag{4.1}$$

$$\frac{\partial f}{\partial C} = \frac{\partial f}{\partial U}\frac{\partial U}{\partial C}. \tag{4.2}$$

Partials $\frac{\partial U}{\partial V}$ and $\frac{\partial U}{\partial C}$ are straightforward, as projection is well-defined. Conveniently, OpenCV provides $\frac{\partial U}{\partial C}$ and $\frac{\partial U}{\partial V}$ directly.

4.4.3 Differentiating Intensity with Respect to 2D Image Coordinates

In order to estimate $\frac{\partial f}{\partial U}$, we first segment our pixels into occlusion boundary pixels and interior pixels, as inspired by [106]. The change induced by boundary pixels is primarily due to the replacement of one surface with another, whereas the change induced by interior pixels relates to the image-space projected translation of the surface patch. The assignment of boundary pixels is obtained with a rendering pass by identifying pixels on edges which (a) pass a depth test (performed by the renderer) and (b) join triangles with opposing normals: one triangle facing towards the camera, one facing away. We consider three classifications for a pixel: interior, interior/boundary, and many-boundary.

Interior: a pixel contains no occlusion boundaries. Because appearance is a product of interpolated texture and interpolated color, intensity changes are piecewise smooth with respect to geometry changes. For interior pixels, we use the image-space first-order Taylor expansion approach adopted by [109]. To understand this approach, consider a patch translating right in image space by a pixel: each pixel becomes replaced by its lefthand neighbor, which is similar to the application of a Sobel filter. Importantly, we do not allow this filtering to cross or include boundary pixels (a case not handled by [109] because occlusion was not modeled).

Specifically, on pixels not neighboring an occlusion boundary, we perform horizontal filtering with the kernel $\frac{1}{2}[-1, 0, 1]$. On pixels neighboring an occlusion boundary on the left, we use $[0, -1, 1]$ for horizontal filtering; with pixels neighboring occlusion boundaries on the right, we use $[-1, 1, 0]$; and with occlusion boundaries on both sides we approximate derivatives as being zero. With vertical filtering, we use the same kernels transposed.

Interior/Boundary: a pixel is intersected by one occlusion boundary. For the interior/boundary case, we use image-space filtering with kernel $\frac{1}{2}[-1, 0, 1]$ and its transpose. This approximates one difference (that between the foreground boundary and the surface behind it) with another (that between the foreground boundary and

a pixel neighboring the surface behind it). Instead of "peeking" behind an occluding boundary, we are using a neighboring pixel as a surrogate and assuming that the difference is not too great. In practical terms, the boundary gradient is almost always much larger than the gradient of the occluded background surface patch, and therefore dominates the direction taken during optimization.

Many-Boundary: more than one occlusion boundary is present in a pixel. While object space methods provide exact derivatives for such pixels at the expense of modeling all the geometry, we treat this as an interior/boundary case. This is justified because very few pixels are affected by this scenario and because the exact object-space computation would be prohibitively expensive.

To summarize, the most significant approximation of the differentiation process occurs boundary pixels where we approximate one difference (nearby pixel minus occluded pixel) with another (nearby pixel minus almost-occluded pixel). We find this works in practice, but it is important to recognize that better approximations are possible [106].

As an implementation detail, our approach requires one render pass when a raw rendered image is requested, and an additional three passes (for boundary identification, triangle identification, and barycentric coordinates) when derivatives are requested. Each pass requires read back from the GPU.

4.4.4 Software Foundation

Flexibility is critical to the generality of a differentiable renderer; custom functions should be easy to design without requiring differentiation by hand. To that end, we use automatic differentiation [110] to compute derivatives given only a specification of the forward process, without resorting to finite differencing methods. As part of the OpenDR release we include a new automatic differentiation framework (Chumpy). This framework is essentially Numpy [111], which is a numerical package in Python, made differentiable. By sharing much of the API of Numpy, this allows the forward specification of problems with a popular API. This in turn allows the forward specification of models not part of the renderer, and allows upper layers of the renderer

to be specified minimally. Although alternative auto-differentiation frameworks were considered [112, 113, 114], we wrap Numpy for its ease-of-use. Our overall system depends on Numpy [111], Scipy [115], and OpenCV [107].

4.5 Programming in OpenDR: Hello World

First we illustrate construction of a renderer with a texture-mapped 3D mesh of Earth. In Sec. 4.3, we introduced f as a function of $\{V, A, U\}$; in Fig. 4-2, V, A, U and f are constructed in turn. While we use spherical harmonics and a static set of vertices, anything expressible in Chumpy can be assigned to these variables, as long the dimensions make sense: given N vertices, then V and A must be $N \times 3$, and U must be $N \times 2$.

```
from opendr.simple import *
w, h = 320, 240

import numpy as np
m = load_mesh('nasa_earth.obj')

# Create V, A, U, f: geometry, brightness, camera, renderer
V = ch.array(m.v)
A = SphericalHarmonics(vn=VertNormals(v=V, f=m.f),
                       components=[3.,1.,0.,0.,0.,0.,0.,0.,0.],
                       light_color=ch.ones(3))
U = ProjectPoints(v=V, f=[300,300.], c=[w/2.,h/2.], k=ch.zeros(5),
                  t=ch.zeros(3), rt=ch.zeros(3))
f = TexturedRenderer(vc=A, camera=U, f=m.f, bgcolor=[0.,0.,0.],
                     texture_image=m.texture_image, vt=m.vt, ft=m.ft,
                     frustum={'width':w, 'height':h, 'near':1,'far':20})
```

Figure 4-2: Constructing a renderer in OpenDR.

Figure 4-3 shows the code for optimizing a model of Earth to match image evidence. We reparameterize V with translation and rotation, express the error to be minimized as a difference between Gaussian pyramids, and find a local minimum of the energy function with simultaneous optimization of translation, rotation, and light parameters. Note that a Gaussian pyramid can be written as a linear filtering operation and is therefore simply differentiable. The process is visualized in Fig. 4-4.

```
# Parameterize the vertices
translation, rotation = ch.array([0,0,4]), ch.zeros(3)
f.v = translation + V.dot(Rodrigues(rotation))

# Create the energy
difference = f - load_image('earth_observed.jpg')
E = gaussian_pyramid(difference, n_levels=6, normalization='SSE')

# Minimize the energy
light_parms = A.components
ch.minimize(E, x0=[translation])
ch.minimize(E, x0=[translation, rotation, light_parms])
```

Figure 4-3: Minimizing an objective function given image evidence. The derivatives from the renderer are used by the minimize method. Including a translation-only stage typically speeds convergence.

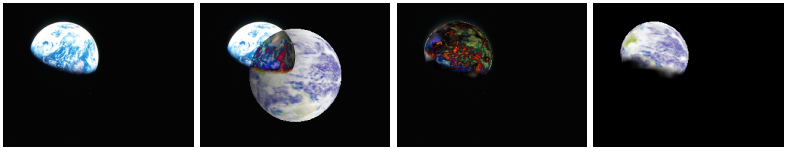


Figure 4-4: Illustration of optimization in Figure 4-3. In order: observed image of earth, initial absolute difference between the rendered and observed image intensities, final difference, final result.

In this example, there is only one object; but as mentioned in Sec. 4.3, there is no obvious limit to the number of objects, because geometry is just a collection of triangles whose vertices are driven by a user's parameterization. Triangle face connectivity is required but may be disjoint.

```
rn = TexturedRenderer(...)
edge_image = rn[:,1:,:] - rn[:,:-1,:]
ch.minimize(ch.sum((edge_image - my_edge_image)**2.),
            x0=[rn.v], method='bfgs')
```

Figure 4-5: Optimizing a function of the rendered image to match a function of image evidence. Here the function is an edge filter.

Image pixels are only one quantity of interest. Any differentiable operation applied to an image can be applied to the render and hence we can minimize the difference between functions of images. Figure 4-5 illustrates how to minimize the difference

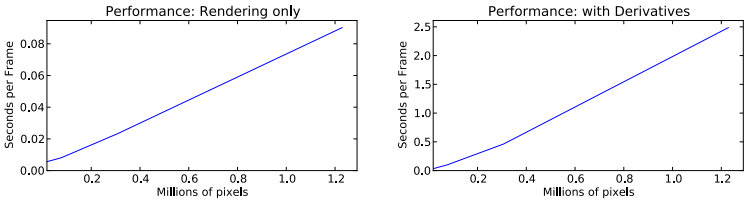


Figure 4-6: **Rendering performance versus resolution.** For reference, 640x480 is 0.3 million pixels. Left: with rendering only. Right: with rendering and derivatives.

between image edges and rendered edges. For more examples, the `opendr.demo()` function, in the software release, shows rendering of image moments, silhouettes, and boundaries, all with derivatives with respect to inputs.

4.6 Experiments

Run-time depends on many user-specific decisions, including the number of pixels, triangles, underlying parameters and model structure. Figure 4-6 illustrates the effects of resolution on run-time in a simple scenario on a 3.0 GHz 8-core 2013 Mac Pro. We render a subdivided tetrahedron with 1024 triangles, lit by spherical harmonics. The mesh is parameterized by translation and rotation, and timings are according to those 6 parameters. The figure illustrates the overhead associated with differentiable rendering.

Finite differences on original parameters are sometimes faster to compute than analytic differences. In the experiment shown in Fig. 4-6, at 640x480, it is 1.75 times faster to compute forward finite differencing on 6 parameters than to find analytic derivatives according to our approach. However, if derivatives with respect to all 514 vertices are required, then forward finite differencing becomes approximately 80 times slower than computing derivatives with our approach.

More importantly, the correct finite differencing epsilon is pixel-dependent. Figure 4-7 shows that the correct epsilon for finite-differencing can be spatially varying: the chosen epsilon is too small for some pixels and too large for others.

Figure 4-7: **Differentiable rendering versus finite differencing.** Left: a rotating quadrilateral. Middle: OpenDR's predicted change in pixel values with respect to in-plane rotation. Right: finite differences recorded with a change to in-plane rotation.

4.6.1 Body Shape from Kinect

We now address a body measurement estimation problem using the Kinect as an input device. In an analysis-by-synthesis approach, many parameters must be estimated to effectively explain the image and depth evidence. We effectively estimate thousands of parameters (per-vertex albedo being the biggest contributor) by minimizing the contribution of over a million residuals; this would be impractical with derivative-free methods.

Subjects were asked to form an A-pose or T-pose in two views separated by 45 degrees; then a capture was performed without the subject in view. This generates three depth and three color images, with most of the state, except pose, assumed constant across the two views.

Our variables and observables are as follows:

- **Latent variables:** lighting parameters A_L, per-vertex albedo A_C, color camera translation T, and body parameters B: therefore $\Theta = \{A_L, A_C, T, B\}$.
- **Observables:** depth images $D_{1...n}$ and color images $I_{1...n}$, $n = 3$.

Appearance, A, is modeled here as a product of per-vertex albedo, A_C, and spherical harmonics parameterized by A_L: $A = A_C H(A_L, V)$, where $H(A_L, V)$ gives one brightness to each vertex according to the surface normal. Vertices are generated by a BlendSCAPE model [17], controlled by pose parameters $P_{1..n}$ (each of n views has a slightly different pose) and shape parameters S (shared across views) which we concatenate to form B.

To use depth and color together, we must know the precise extrinsic relationship between the sensors; due to manufacturing variance, the camera axes are not perfectly aligned. Instead of using a pre-calibration step, we pose the camera translation estimation as part of the optimization, using the human body itself to find the translation, T, between color and depth cameras.

Our data terms includes a color term E_C, a depth term E_D, and feet-to-floor contact term E_F. Our regularization terms include a pose prior E_P, a shape prior E_S (both Gaussian), and smoothness prior E_Q on per-vertex albedo:

$$E = E_C + E_D + E_F + E_P + E_S + E_Q. \tag{4.3}$$

The color term accumulates per-pixel error over images

$$E_C(I, A_L, A_C, T, B) = \sum_i \sum_u \|I_{iu} - \tilde{I}_{iu}(A_L, A_C, T, B)\|^2 \tag{4.4}$$

where $\tilde{I}_{ui}$ is the simulated pixel intensity of image-space position u for view i.

The depth term is similar but, due to sensor noise, is formulated robustly

$$E_D(D, T, B) = \sum_i \sum_u \|D_{iu} - \tilde{D}_{iu}(T, B)\|^\rho \tag{4.5}$$

where the parameter ρ is adjusted from 2 to 1 over the course of an optimization.

The floor term E_F minimizes differences between foot vertices of the model and the ground

$$E_F(D, B) = \sum_k \|r(B, D_b, k)\|^2 \tag{4.6}$$

where $r(B, D_b, k)$ indicates the distance between model footpad vertex k and a mesh D_b constructed from the background shot,

The albedo smoothness term E_Q penalizes squared differences between the log albedo of neighboring mesh vertices

$$E_Q = \sum_e \|\log(b(e, 0)) - \log(b(e, 1))\|^2 \tag{4.7}$$

where $b(e, 0)$ denotes the albedo of the first vertex on edge e, and $b(e, 1)$ denotes the albedo of the other vertex on edge e.

Finally, shape and pose parameter priors, $E_S(S)$ and $E_P(P)$, penalize the squared Mahalanobis distance from the mean body shape and pose learned during BlendSCAPE training.

Initialization for the position of the simulated body could be up to a meter away from the real body and still achieve convergence. Without the use of Gaussian pyramids or background images, initialization would require more precision (while we did not use it, initialization could be obtained with the pose information available from the Kinect API).

Male and female body models were each trained from approximately 2000 scans from the CAESAR [79] dataset. This dataset comes with anthropometric measurements for each subject; similar to [116], we use regularized linear regression to predict measurements from our underlying body shape parameters. To evaluate accuracy of the recovered body models, we measured RMSE and percentage of explained variance of our predictions as shown in Fig. 4-8. For comparison, Fig. 4-8 also shows the accuracy of estimating measurements directly from 3803 meshes accurately registered to the CAESAR laser scans. Although these two settings (23 subjects by Kinect and 3803 subjects by laser scan) differ in both subjects and method, and we do not expect Kinect scans to be as accurate, Fig. 4-8 provides an indication of how well the Kinect-based method works.

Figure 4-9 shows some representative results from our Kinect fitter. While foot posture is not always correct, the effects of geometry, lighting and appearance are generally well-estimated. Obtaining this result was made significantly easier with a platform that includes a differentiable renderer and a set of building blocks to compose around it.

Each fit took around 7 minutes on a 3.0 GHz 8-core 2013 Mac Pro.

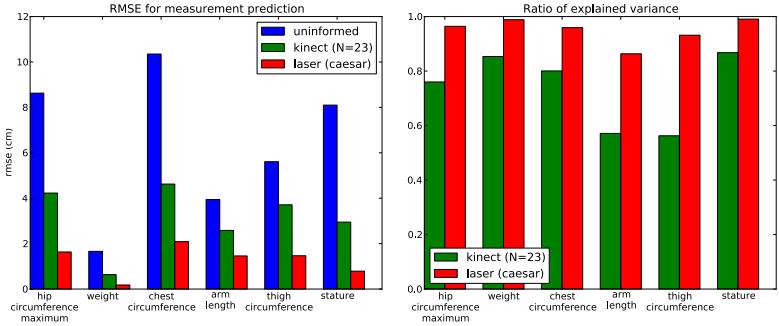


Figure 4-8: Accuracy of measurement prediction for Kinect-based fitting compared to measurements from CAESAR scans or guessing the mean (uninformed). Left: root mean squared error (RMSE) in cm. Right: percentage of explained variance.

4.7 Conclusions

Many problems in computer vision have been solved by effectively differentiating through the rendering process. This is not new. What is new is that we provide an easy to use framework for both renderer differentiation and objective formulation. This makes it easy in Python to define a forward model and optimize it. We have demonstrated this with a challenging problem of body shape estimation from image and range data. By releasing the OpenDR with an open-source license (see `http://open-dr.org`), we hope to create a community that is using and contributing to this effort. The hope is that the this will push forward research on vision as inverse graphics by providing tools to make working on this easier.

Differentiable rendering has its limitations. When using differences between RGB Gaussian pyramids, the fundamental issue is overlap: if a simulated and observed object have no overlap in the pyramid, the simulated object will not record a gradient towards the observed one. One can use functions of the pixels that have no such overlap restriction (e.g. moments) to address this but the fundamental limitation is one of visibility: a real observed feature will not pull on simulated features that are entirely occluded because of the state of the renderer.

Consequently, differentiable rendering is only one piece of the puzzle: we believe that informed sampling [98] and probabilistic graphics programming [87] are also essential to a serious application of inverse rendering. Despite this, we hope many will benefit from the OpenDR platform.

Future exploration may include increasing image realism by incorporating global illumination. It may also include more features of modern rendering pipelines (for example, differentiation through a fragment shader). We are also interested in the construction of an "integratable renderer" for posterior estimation; although standard sampling methods can be used to approximate such an integral, there may be graphics-related techniques to integrate in a more direct fashion within limited domains.

Acknowledgements.

We would like to thank Eric Rachlin for discussions about Chumpy and Gerard Pons-Moll for proofreading.

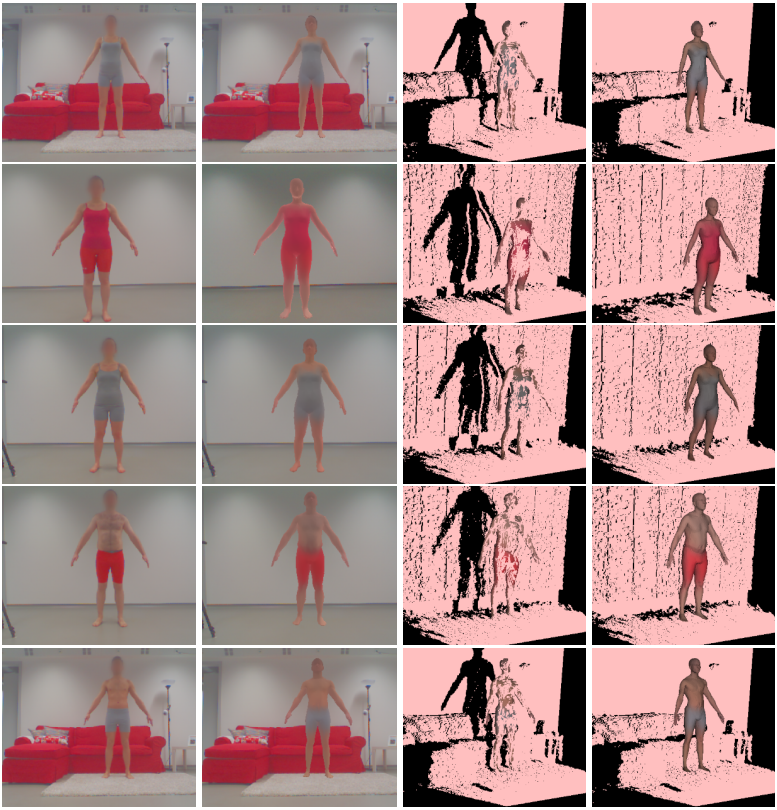


Figure 4-9: Reconstruction of subjects (each subject is a row). **First column:** original captured images, with faces blurred for anonymity. **Second column:** simulated images after convergence. **Third column:** captured point cloud together with estimated body model. **Fourth column:** estimated body shown on background point cloud.

Chapter 5

A Skinned Multi-Person Linear Model

5.1 Introduction

Our goal is to create realistic animated human bodies that can represent different body shapes and deforms naturally with pose. We want such models to be fast to render, easy to deploy, and compatible with existing rendering engines. The commercial approach commonly involves hand rigging a mesh and manually sculpting blend shapes to correct problems with traditional skinning methods. Many blend shapes are typically needed and the manual effort required to build them is large. As an alternative, the research community has focused on learning statistical body models from example scans of different bodies in a varied set of poses. While promising, these approaches are not compatible with existing graphics software and rendering engines that use standard skinning methods.

Our goal is to automatically learn a model of the body that is both realistic and compatible with existing graphics software. To that end, we describe a "Skinned Multi-Person Linear" (SMPL) model of the human body that can realistically represent a wide range of human body shapes, can be posed with natural pose-dependent deformations, is efficient to animate, and is compatible with existing rendering engines (Fig. 5-1).

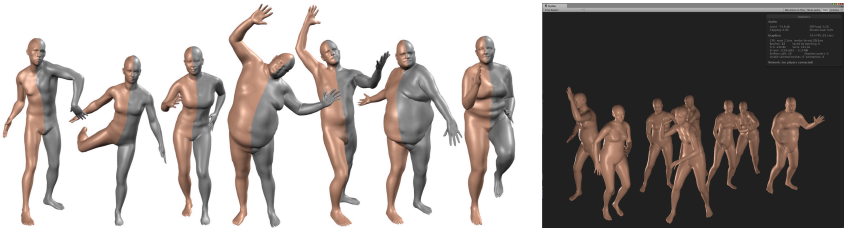


Figure 5-1: SMPL is a realistic learned model of human body shape and pose that is compatible with existing rendering engines, allows animator control, and is available for research purposes. (left) SMPL model (orange) fit to ground truth 3D meshes (gray). (right) Unity 5.0 game engine screenshot showing bodies from the CAESAR dataset animated in real time.

Traditional methods model how vertices are related to an underlying skeleton structure. Basic linear blend skinning (LBS) models are the most widely used, are supported by all game engines, and are efficient to render. Unfortunately they produce unrealistic deformations at joints including the well-known "taffy" and "bowtie" effects (see Fig. 5-2). Tremendous work has gone into skinning methods that ameliorate these effects [35, 117, 118, 119, 36]. There has also been a lot of work on learning highly realistic body models from data [120, 46, 121, 122, 123, 62]. These methods can capture the body shape of many people as well as non-rigid deformations due to pose. The most realistic approaches are arguably based on triangle deformations [46, 62, 122, 124]. Despite the above research, existing models either lack realism, do not work with existing packages, do not represent a wide variety of body shapes, are not compatible with standard graphics pipelines, or require significant manual labor.

In contrast to the previous approaches, a key goal of our work is to make the body model as simple and standard as possible so that it can be widely used, while, at the same time, keeping the realism of deformation-based models learned from data. Specifically we learn blend shapes to correct for the limitations of standard skinning. Different blend shapes for identity and pose are additively combined with a rest template before being transformed by blend skinning. A key component of our approach is that we formulate the pose blend shapes as a linear function of the

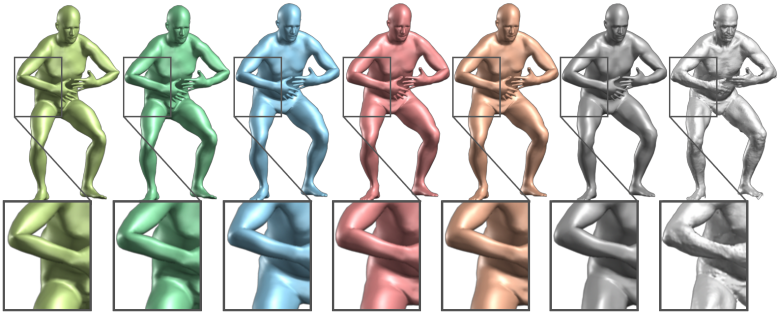


Figure 5-2: **Models compared with ground truth.** This figure defines the color coding used in this chapter. The far right (light gray) mesh is a 3D scan. Next to it (dark gray) is a registered mesh with the same topology as our model. We ask how well different models can approximate this registration. From left to right: (light green) Linear blend skinning (LBS), (dark green) Dual-quaternion blend skinning (DQBS), (blue) BlendSCAPE, (red) SMPL-LBS, (orange) SMPL-DQBS. The zoomed regions highlight differences between the models at the subject's right elbow and hip. LBS and DQBS produce serious artifacts at the knees, elbows, shoulders and hips. BlendSCAPE and both SMPL models do similarly well at fitting the data.

elements of the part rotation matrices. This formulation is different from previous methods [120, 119, 117] and makes training and animating with the blend shapes simple. Because the elements of rotation matrices are bounded, so are the resulting deformations, helping our model generalize better.

Our formulation admits an objective function that penalizes the per-vertex disparities between registered meshes and our model, enabling training from data. To learn how people deform with pose, we use 1786 high-resolution 3D scans of different subjects in a wide variety of poses. We align our template mesh to each scan to create a training set. We optimize the blend weights, pose-dependent blend shapes, the mean template shape (rest pose), and a regressor from shape to joint locations to minimize the vertex error of the model on the training set. This joint regressor predicts the location of the joints as a function of the body shape and is critical to animating realistic pose-dependent deformations for any body shape. All parameters are automatically estimated from the aligned scans.

We learn linear models of male and female body shape from the CAESAR dataset [79]

(approximately 2000 scans per gender) using principal component analysis (PCA). We first register a template mesh to each scan and pose normalize the data, which is critical when learning a vertex-based shape model. The resulting principal components become body shape blend shapes.

We train the SMPL model in various forms and compare it quantitatively to a BlendSCAPE model [125] trained with exactly the same data. We evaluate the models both qualitatively with animations and quantitatively using meshes that were not used for training. We fit SMPL and BlendSCAPE to these meshes and then compare the vertex errors. Two main variants of SMPL are explored, one using linear blend skinning (LBS) and the other with Dual-Quaternion blend skinning (DQBS); see Fig. 5-2. These variants are described in more detail in Section 5.3. The surprise is that a vertex-based, skinned, model such as SMPL is actually more accurate than a deformation-based model like BlendSCAPE trained on the same data. The test meshes are available for research purposes so others can quantitatively compare to SMPL.

SMPL models can be animated significantly faster than real time on a CPU using standard rendering engines. Consequently SMPL addresses an open problem in the field; it makes a realistic learned model accessible to animators. The SMPL base template is designed with animation in mind; it has a low-polygon count, a simple vertex topology, clean quad structure, a standard rig, and reasonable face and hand detail (though we do not rig the hands or face here). SMPL can be represented as an Autodesk Filmbox (FBX) file that can be imported into animation systems. We make the SMPL model available for research purposes and provide scripts to drive our model in Maya, Blender, Unreal Engine and Unity.

5.2 Related Work

Linear blend skinning and blend shapes are widely used throughout the animation industry. While the research community has proposed many novel models of articulated body shape that produce high-quality results, they are not compatible with industry

practice. Many authors have tried to bring these worlds together with varying degrees of success as we summarize below.

Blend Skinning. Skeleton subspace deformation methods, also known as blend skinning, attach the surface of a mesh to an underlying skeletal structure. Each vertex in the mesh surface is transformed using a weighted influence of its neighboring bones. This influence can be defined linearly as in Linear Blend Skinning (LBS). The problems of LBS have been widely published and the literature is dense with generic methods that attempt to fix these, such as quaternion or dual-quaternion skinning, spherical skinning, etc. (e.g. [117, 118, 36, 126, 127]). Generic methods, however, often produce unnatural results and here we focus on learning to correct the limitations of blend skinning, regardless of the particular formulation.

Auto-rigging. There is a great deal of work on automatically rigging LBS models (e.g. [128, 129, 130, 131]) and commercial solutions exist. Most relevant here are methods that take a collection of meshes and infer the bones as well as the joints and blend weights (e.g. [132]). Such methods do not address the common problems of LBS models because they do not learn corrective blend shapes. Models created from sequences of meshes (e.g. [128]) may not generalize well to new poses and motions. Here, we assume the kinematic structure is known, though the approach could be extended to also learn this using the methods above.

The key limitation of the above methods is that the models do not span a space of body shapes. Miller et al. [133] partially address this by auto-rigging using a database of pre-rigged models. They formulate rigging and skinning as the process of transferring and adapting skinning weights from known models to a new model. Their method does not generate blend shapes, produces standard LBS artifacts, and does not minimize a clear objective function.

Blend shapes. To address the shortcomings of basic blend skinning, the pose space deformation model (PSD) [35] defines deformations (as vertex displacements) relative to a base shape, where these deformations are a function of articulated pose. This is the key formulation that is largely followed by later approaches and is also referred to as "scattered data interpolation" and "corrective enveloping" [134]. We

take an approach more similar to weighted pose-space deformation (WPSD) [135, 136], which defines the corrections in a rest pose and then applies a standard skinning equation (e.g. LBS). The idea is to define corrective shapes (sculpts) for specific key poses, so that when added to the base shape and transformed by blend skinning, produce the right shape. Typically one finds the distance (in pose space) to the exemplar poses and uses a function, e.g. a Radial Basis (RBF) kernel [35], to weight the exemplars non-linearly based on distance. The sculpted blend shapes are then weighted and linearly combined.

These approaches are all based on computing weighted distances to exemplar shapes. Consequently, these methods require computation of the distances and weights at runtime to obtain the corrective blend shape. For a given animation (e.g. in a video game) these weights are often defined in advance based on the poses and "baked" into the model. Game engines apply the baked-in weights to the blend shapes. The sculpting process is typically done by an artist and then only for poses that will be used in the animation.

Learning pose models. Allen et al. [137] use this PSD approach but rather than hand-sculpt the corrections, learn them from registered 3D scans. Their work focuses primarily on modeling the torso and arms of individuals, rather than whole bodies of a population. They store deformations of key poses and interpolate between them. When at, or close to, a stored shape, these methods are effectively perfect. They do not tend to generalize well to new poses, requiring dense training data. It is not clear how many such shapes would be necessary to model the full range of articulated human pose. As the complexity of the model increases, so does the complexity of controlling all these shapes and how they interact.

To address this, Kry et al. [138] learn a low-dimensional PCA basis for each joint's deformations. Pose-dependent deformations are described in terms of the coefficients of the basis vectors. Kavan et al. [139] use example meshes generated using a non-linear skinning method to construct linear approximations. James and Twigg [140] combine the idea of learning the bones (non-rigid, affine bones) and skinning weights directly from registered meshes. For blend shapes they use an approach similar to

[138].

Another way to address the limitations of blend skinning is through "multi-weight enveloping" (MWE) [117]. Rather than weight each vertex by a weighted combination of the bone transformation matrices, MWE learns weights for the *elements* of these matrices. This increases the capacity of the model (more parameters). Like [140] they overparameterize the bone transformations to give more expressive power and then use PCA to remove unneeded degrees of freedom. Their experiments typically involve user interaction and the MWE approach is not supported by current game engines.

Merry et al. [119] find MWE to be overparameterized, because it allows vertices to deform differently depending on rotation in the global coordinate system. Their Animation Space model reduces the parameterization at minimal loss of representational power, while also showing computational efficiency on par with LBS.

Another alternative is proposed by Mohr and Gleicher [37] who learn an efficient linear and realistic model from example meshes. To deal with the problems of LBS, however, they introduce extra "bones" to capture effects like muscle bulging. These extra bones increase complexity, are non-physical, and are non-intuitive for artists. Our blend shapes are simpler, more intuitive, more practical, and offer greater realism. Similarly, Wang et al. [127] introduce joints related to surface deformation. Their rotational regression approach uses deformation gradients, which then must be converted to a vertex representation.

Learning pose and shape models. The above methods focus on learning poseable single-shape models. Our goal, however, is to have realistic poseable models that cover the space of human shape variation. Early methods use PCA to characterize a space of human body shapes [40, 141] but do not model how body shape changes with pose. The most successful class of models are based on SCAPE [46] and represent body shape and pose-dependent shape in terms of triangle deformations rather than vertex displacements [62, 121, 142, 125, 124]. These methods learn statistical models of shape variation from training scans containing different body shapes and poses. Triangle deformations provide allow the composition of different

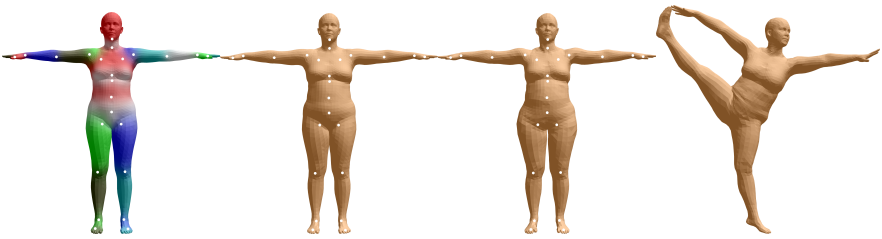
transformations such as body shape variation, rigid part rotation, and pose-dependent deformation. Weber et al. [143] present an approach that has properties of SCAPE but blends this with example shapes. These models are not consistent with existing animation software.

Hasler et al. [122] learn two linear blend rigs: one for pose and one for body shape. To represent shape change, they introduce abstract "bones" that control the shape change of the vertices. Animating a character of a particular shape involves manipulating the shape and pose bones. They learn a base mesh and blend weights but not blend shapes. Consequently the model lacks realism.

What we would like is a vertex-based model that has the expressive power of the triangle deformation models so that it can capture a whole range of natural shapes and poses. Allen et al. [120] formulate such a model. For a given base body shape, they define a standard LBS model with scattered/exemplar PSD to model pose deformations (using RBFs). They greedily define "key angles" at which to represent corrective blend shapes and they hold these fixed across all body shapes. A given body shape is parameterized by the vertices of the rest pose, corrective blend shapes (at the key angles), and bone lengths; these comprise a "character vector." Given different character vectors for different bodies they learn a low-dimensional latent space that lets them generalize character vectors to new body shapes; they learn these parameters from data. Their model is more complex than ours, has fewer parameters, and is learned from much less data. A more detailed analysis of how this method compares to SMPL is presented in Sec. 5.6.

5.3 Model Formulation

Our Skinned Multi-Person Linear model (SMPL) is illustrated in Fig. 5-3. Like SCAPE, the SMPL model decomposes body shape into identity-dependent shape and non-rigid pose-dependent shape; unlike SCAPE, we take a vertex-based skinning approach that uses corrective blend shapes. A single blend shape is represented as a vector of concatenated vertex offsets. We begin with an artist-created mesh with



(a) $\bar{\mathbf{T}}$, $\mathcal{W}$ (b) $\bar{\mathbf{T}} + B_S(\vec{\beta})$, $J(\vec{\beta})$ (c) $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$ (d) $W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

Figure 5-3: **SMPL model.** (a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blendshape contribution only; vertex and joint locations are linear in shape vector $\vec{\beta}$. (c) With the addition of of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by dual quaternion skinning for the split pose.

$N = 6890$ vertices and $K = 23$ joints. The mesh has the same topology for men and women, spatially varying resolution, a clean quad structure, a segmentation into parts, initial blend weights, and a skeletal rig. The part segmentation and initial blend weights are shown in Fig. 5-7.

Following standard skinning practice, the model is defined by a mean template shape represented by a vector of N concatenated vertices $\bar{\mathbf{T}} \in \mathbb{R}^{3N}$ in the zero pose, $\vec{\theta}^*$; a set of blend weights, $\mathcal{W} \in \mathbb{R}^{N \times K}$, (Fig. 5-3(a)); a blend shape function, $B_S(\vec{\beta}) : \mathbb{R}^{|\vec{\beta}|} \mapsto \mathbb{R}^{3N}$, that takes as input a vector of shape parameters, $\vec{\beta}$, (Fig. 5-3(b)) and outputs a blend shape sculpting the subject identity; a function to predict K joint locations (white dots in Fig. 5-3(b)), $J(\vec{\beta}) : \mathbb{R}^{|\vec{\beta}|} \mapsto \mathbb{R}^{3K}$ as a function of shape parameters, $\vec{\beta}$; and a pose-dependent blend shape function, $B_P(\vec{\theta}) : \mathbb{R}^{|\vec{\theta}|} \mapsto \mathbb{R}^{3N}$, that takes as input a vector of pose parameters, $\vec{\theta}$, and accounts for the effects of pose-dependent deformations (Fig. 5-3(c)). The corrective blend shapes of these functions are added together in the rest pose as illustrated in (Fig. 5-3(c)). Finally, a standard blend skinning function $W(\cdot)$ (linear or dual-quaternion) is applied to rotate the vertices around the estimated joint centers with smoothing defined by the blend weights. The result is a model, $M(\vec{\beta}, \vec{\theta}; \Phi) : \mathbb{R}^{|\vec{\theta}| \times |\vec{\beta}|} \mapsto \mathbb{R}^{3N}$, that maps shape and pose parameters to vertices (Fig. 5-3(d)). Here Φ represents the learned model parameters

described below.

Below we will use both LBS and DQBS skinning methods. In general the skinning method can be thought of as a generic black box. Given a particular skinning method our goal is to learn Φ to correct for limitations of the method so as to model training meshes. Note that the learned pose blend shapes both correct errors caused by the blend skinning function and static soft-tissue deformations caused by changes in pose.

Below we describe each term in the model. For convenience, a notational summary is provided in Table 5.1 in the Appendix.

Blend skinning. Blend skinning deforms a surface according to the movement of a skeleton. The surface is typically defined by a set of vertices, and the skeleton is defined by its endpoint positions and its parent-child structure (or kinematic tree structure). A depiction of such a skeleton, and the surface it controls, is in Figure 5-4.

While the idea that "limbs attach to bones" is intuitive, the model by which they do so varies based on the choice of skinning method. For both DQBS and LBS, the template surface is deformed according to linearly blended transformations. LBS and DQBS encode these transformations differently, however: with LBS, matrices are blended, whereas with DQBS, quaternion parameters are blended. The most important consequence of this is that blending rigid DQBS-encoded transformations always results in rigid transformations, whereas blending rigid LBS-encoded transformations can result in non-rigid transformations including which include scale and shear. Put another way, rigid transformations used by LBS are not closed under addition, whereas those used by DQBS are closed under addition. As a practical matter, this means that certain artifacts (such as volumetric collapse at the joints, and the so-called "candy wrapper" effect) exhibited by LBS are avoided with DQBS, as described by Kavan [36].

We now present notation which is invariant to the choice of LBS or DQBS. Meshes and blend shapes are vectors of vertices represented by bold capital letters (e.g. $\mathbf{X}$) and lowercase bold letters (e.g. $\mathbf{x}_i \in \mathbb{R}^3$) are vectors representing a particular vertex. The vertices are sometimes represented in homogeneous coordinates. We use the

(a) Skeletal rig in green, and one relative rotation $\vec{\omega}_i$ illustrated in white.

(b) Surface driven by skeletal rig.

Figure 5-4: **Rig-driven deformation.** Joint angles control the relative orientation of bone segments; the bone segments, in turn, affect the computed surface.

same notation for a vertex whether it is in standard or homogeneous coordinates as it should always be clear from the context which form is needed.

The pose of the body is defined by a standard skeletal rig, where $\vec{\omega}_k \in \mathbb{R}^3$ denotes the axis-angle representation of the relative rotation of part k with respect to its parent in the kinematic tree. Our rig has $K = 23$ joints, hence a pose $\vec{\theta} = [\vec{\omega}_0^T, \ldots, \vec{\omega}_K^T]^T$ is defined by $|\vec{\theta}| = 3 \times 23 + 3 = 72$ parameters; i.e. 3 for each part plus 3 for the root orientation. Let $\bar{\omega} = \frac{\vec{\omega}}{\|\vec{\omega}\|}$ denote the unit norm axis of rotation.

Having described notation for skinning which is invariant to the choice of LBS or DQBS, we next summarize the workings of LBS and DQBS in turn.

Linear blend skinning. The axis angle for every joint j is transformed to a rotation matrix using the *Rodrigues formula*

$$\exp(\vec{\omega}_j) = \mathcal{I} + \widehat{\bar{\omega}}_j \sin(\|\vec{\omega}_j\|) + \widehat{\bar{\omega}}_j^2 \cos(\|\vec{\omega}_j\|) \tag{5.1}$$

where $\widehat{\bar{\omega}}$ is the skew symmetric matrix of the 3-vector $\bar{\omega}$ and $\mathcal{I}$ is the 3×3 iden-

tity matrix. Using this, the standard linear blend skinning function $W(\bar{\mathbf{T}}, \mathbf{J}, \vec{\theta}, \mathcal{W}) : \mathbb{R}^{3N\times 3K\times|\vec{\theta}|\times|\mathcal{W}|} \mapsto \mathbb{R}^{3N}$ takes vertices in the rest pose, $\bar{\mathbf{T}}$, joint locations, $\mathbf{J}$, a pose, $\vec{\theta}$, and the blend weights, $\mathcal{W}$, and returns the posed vertices. Each vertex $\bar{\mathbf{t}}_i$ in $\bar{\mathbf{T}}$ is transformed into $\bar{\mathbf{t}}'_i$ (both column vectors in homogeneous coordinates) as

$$\bar{\mathbf{t}}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, \mathbf{J}) \bar{\mathbf{t}}_i \tag{5.2}$$

$$G'_k(\vec{\theta}, \mathbf{J}) = G_k(\vec{\theta}, \mathbf{J}) G_k(\vec{\theta}^*, \mathbf{J})^{-1} \tag{5.3}$$

$$G_k(\vec{\theta}, \mathbf{J}) = \prod_{j\in A(k)} \left[\begin{array}{c|c} \exp(\vec{\omega}_j) & \mathbf{j}_j \\ \hline \vec{0} & 1 \end{array}\right] \tag{5.4}$$

where $w_{k,i}$ is an element of the blend weight matrix $\mathcal{W}$, representing how much the rotation of part k effects the vertex i, $\exp(\vec{\theta}_j)$ is the local 3×3 rotation matrix corresponding to joint j, $G_k(\vec{\theta}, \mathbf{J})$ is the world transformation of joint k, and $G'_k(\vec{\theta}, \mathbf{J})$ is the same transformation after removing the transformation due to the rest pose, $\vec{\theta}^*$. Each 3-element vector in $\mathbf{J}$ corresponding to a single joint center, j, is denoted $\mathbf{j}_j$. Finally, $A(k)$ denotes the ordered set of joint ancestors of joint k. Note, for compatibility with existing rendering engines, we assume $\mathcal{W}$ is sparse and allow at most four parts to influence a vertex.

Many methods have modified equation (5.2) to make skinning more expressive. For example MWE [117] replaces $G_k(\vec{\theta}, \mathbf{J})$ with a more general affine transformation matrix and replaces the scalar weight with a separate weight for every element of the transformation matrix. Such changes are expressive but are not compatible with existing animation systems.

Dual quaternion blend skinning. Using dual quaternion blend skinning implies a change in the behavior of the skinning function $W(\cdot)$. A dual quaternion can be interpreted as a rigid transformation, and is the "dual number" counterpart of a single quaternion. With ε being a dual unit satisfying $\varepsilon^2 = 0$, a dual quaternion $\hat{q}$ can be considered as the sum of two conventional quaternions q_0 and q_ε:

$$\hat{q} = q_0 + q_\varepsilon \varepsilon$$

Whereas q_0 is a conventional quaternion representing rotation, q_ε stores the product of translation and rotation. We will not write out all the operations of dual quaternions (and conversions to and from other types of transformations) in detail here, as Kavan [36] provides an excellent summary. But it is worth re-emphasizing that when the dual quaternion parameters are linearly blended, the result is always a rigid transformation, whereas blending matrices with LBS rarely results in a rigid transformation and often includes scale and shear as part of the transformation.

Incorporation of pose and shape. To maintain compatibility, we keep the basic skinning function and instead modify the template in an additive way and learn a function to predict joint locations. Our model, $M(\vec{\beta}, \vec{\theta}; \Phi)$ is then

$$M(\vec{\beta}, \vec{\theta}) = W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W}) \tag{5.5}$$

$$T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta}) \tag{5.6}$$

where $B_S(\vec{\beta})$ and $B_P(\vec{\theta})$ are vectors of vertices representing offsets from the template. We refer to these as shape and pose blend shapes respectively.

Given this definition, and assuming LBS, a vertex $\bar{\mathbf{t}}_i$ is transformed according to

$$\bar{\mathbf{t}}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, J(\vec{\beta}))(\bar{\mathbf{t}}_i + \mathbf{b}_{S,i}(\vec{\beta}) + \mathbf{b}_{P,i}(\vec{\theta})) \tag{5.7}$$

where $\mathbf{b}_{S,i}(\vec{\beta}), \mathbf{b}_{P,i}(\vec{\theta})$ are vertices in $B_S(\vec{\beta})$ and $B_P(\vec{\theta})$ respectively and represent the shape and pose blend shape offsets for the vertex $\bar{\mathbf{t}}_i$. Hence, the joint centers are now a function of body shape and the template mesh that is deformed by blend skinning is now a function of both pose and shape. Below we describe each term in detail.

Shape blend shapes. The body shapes of different people are represented by a linear function B_S

$$B_S(\vec{\beta}; \mathcal{S}) = \sum_{n=1}^{|\vec{\beta}|} \beta_n \mathbf{S}_n \tag{5.8}$$

where $\vec{\beta} = [\beta_1, \ldots, \beta_{|\vec{\beta}|}]^T$, $|\vec{\beta}|$ is the number of linear shape coefficients, and the $\mathbf{S}_n \in \mathbb{R}^{3N}$ represent orthonormal principal components of shape displacements. Let

$\mathcal{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_{|\vec{\beta}|}] \in \mathbb{R}^{3N \times |\vec{\beta}|}$ be the matrix of all such shape displacements. Then the linear function $B_S(\vec{\beta}; \mathcal{S})$ is fully defined by the matrix $\mathcal{S}$, which is learned from registered training meshes, see Sec. 5.4.

Notationally, the values to the right of a semicolon represent learned parameters, while those on the left are parameters set by an animator. For notational convenience, we often omit the learned parameters when they are not explicitly being optimized in training.

Figure 5-3(b) illustrates the application of these shape blend shapes to the template $\bar{\mathbf{T}}$ to produce a new body shape.

Pose blend shapes. We denote as $R : \mathbb{R}^{|\vec{\theta}|} \mapsto \mathbb{R}^{9K}$ a function that maps a pose vector $\vec{\theta}$ to a vector of concatenated part relative rotation matrices, $\exp(\vec{\omega})$. Given that our rig has 23 joints, $R(\vec{\theta})$ is a vector of length $(23 \times 9 = 207)$. Elements of $R(\vec{\theta})$ are functions of sines and cosines (Eq. (5.1)) of joint angles and therefore $R(\vec{\theta})$ is non-linear with $\vec{\theta}$.

Our formulation differs from previous work in that we define the effect of the pose blend shapes to be linear in $R^*(\vec{\theta}) = (R(\vec{\theta}) - R(\vec{\theta}^*))$, where $\vec{\theta}^*$ denotes the rest pose. Let $R_n(\vec{\theta})$ denote the n^{th} element of $R(\vec{\theta})$, then the vertex deviations from the rest template are

$$B_P(\vec{\theta}; \mathcal{P}) = \sum_{n=1}^{9K} (R_n(\vec{\theta}) - R_n(\vec{\theta}^*))\mathbf{P}_n, \tag{5.9}$$

where the blend shapes, $\mathbf{P}_n \in \mathbb{R}^{3N}$, are again vectors of vertex displacements. Here $\mathcal{P} = [\mathbf{P}_1, \ldots, \mathbf{P}_{9K}] \in \mathbb{R}^{3N \times 9K}$ is a matrix of all 207 pose blend shapes. In this way, the pose blend shape function $B_P(\vec{\theta}; \mathcal{P})$ is fully defined by the matrix $\mathcal{P}$, which we learn in Sec. 5.4.

Note that subtracting the rest pose rotation vector, $R(\vec{\theta}^*)$, guarantees that the contribution of the pose blend shapes is zero in the rest pose, which is important for animation.

Joint locations. Different body shapes have different joint locations. Each joint is represented by its 3D location in the rest pose. It is critical that these are accurate, otherwise there will be artifacts when the model is posed using the skinning equation.

For that reason, we define the joints as a function of the body shape, $\vec{\beta}$,

$$J(\vec{\beta}; \mathcal{J}, \bar{\mathbf{T}}, \mathcal{S}) = \mathcal{J}(\bar{\mathbf{T}} + B_S(\vec{\beta}; \mathcal{S})) \tag{5.10}$$

where $\mathcal{J}$ is a matrix that transforms rest vertices into rest joints. We learn the regression matrix, $\mathcal{J}$, from examples of different people in many poses, as part of our overall model learning in Sec. 5.4. This matrix models which mesh vertices are important and how to combine them to estimate the joint locations.

SMPL model. We can now specify the full set of model parameters of the SMPL model as $\Phi = \left\{\bar{\mathbf{T}}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\right\}$. We describe how to learn these in Sec. 5.4. Once learned they are held fixed and new body shapes and poses are created and animated by varying $\vec{\beta}$ and $\vec{\theta}$ respectively.

Then the SMPL model is finally defined as $M(\vec{\beta}, \vec{\theta}; \Phi) =$

$$W\left(T_P(\vec{\beta}, \vec{\theta}; \bar{\mathbf{T}}, \mathcal{S}, \mathcal{P}), J(\vec{\beta}; \mathcal{J}, \bar{\mathbf{T}}, \mathcal{S}), \vec{\theta}, \mathcal{W}\right) \tag{5.11}$$

and hence (assuming LBS) each vertex is transformed as

$$\mathbf{t}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, J(\vec{\beta}; \mathcal{J}, \bar{\mathbf{T}}, \mathcal{S})) \mathbf{t}_{P,i}(\vec{\beta}, \vec{\theta}; \bar{\mathbf{T}}, \mathcal{S}, \mathcal{P}) \tag{5.12}$$

where $\mathbf{t}_{P,i}(\vec{\beta}, \vec{\theta}; \bar{\mathbf{T}}, \mathcal{S}, \mathcal{P}) =$

$$\bar{\mathbf{t}}_i + \sum_{m=1}^{|\vec{\beta}|} \beta_m \mathbf{s}_{m,i} + \sum_{n=1}^{9K} (R_n(\vec{\theta}) - R_n(\vec{\theta^*})) \mathbf{p}_{n,i} \tag{5.13}$$

represents the vertex i after applying the blend shapes and where $\mathbf{s}_{m,i}, \mathbf{p}_{n,i} \in \mathbb{R}^3$ are the elements of the shape and pose blend shapes corresponding to template vertex $\bar{\mathbf{t}}_i$.

Below we experiment with both LBS and DQBS and train the parameters for each. We refer to these models as SMPL-LBS and SMPL-DQBS; SMPL-DQBS is our default model, and we use SMPL as shorthand to mean SMPL-DQBS.

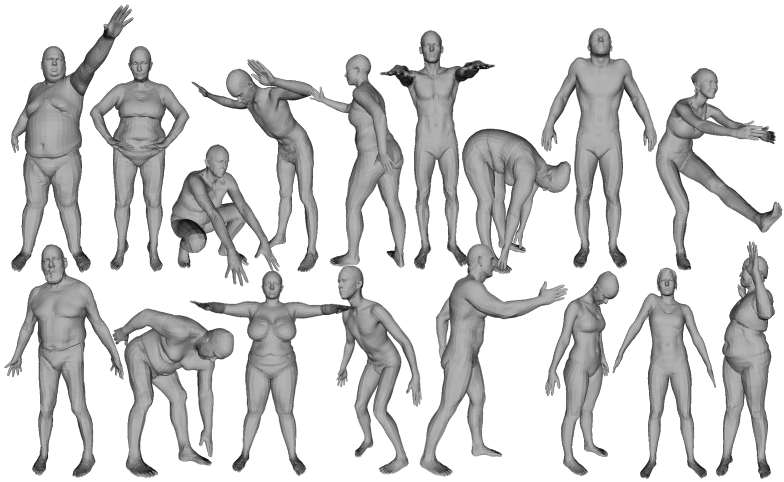


Figure 5-5: Sample registrations from the multipose dataset.

5.4 Training

We train the SMPL model parameters to minimize reconstruction error on two datasets. Each dataset contains meshes with the same topology as our template that have been aligned to high-resolution 3D scans using [144]; we call these aligned meshes "registrations." The *multi-pose* dataset consists of 1786 registrations of 40 individuals (891 registrations spanning 20 females, and 895 registrations spanning 20 males); a sampling is shown in Fig. 5-5. The *multi-shape* dataset consists of registrations to the CAESAR dataset [79], totaling 1700 registrations for males and 2100 for females; a few examples are shown in Fig. 5-6. We denote the j^{th} mesh in the multi-pose dataset as $\mathbf{V}_j^P$ and the j^{th} mesh in the multi-shape dataset as $\mathbf{V}_j^S$.

Our goal is to train the parameters $\Phi = \left\{\bar{\mathbf{T}}, \mathcal{W}, \mathcal{S}, \mathcal{J}, \mathcal{P}\right\}$ to minimize vertex reconstruction error on the datasets. Because our model decomposes shape and pose, we train these separately, simplifying optimization. We first train $\{\mathcal{J}, \mathcal{W}, \mathcal{P}\}$ using our multi-pose dataset and then train $\{\bar{\mathbf{T}}, \mathcal{S}\}$ using our multi-shape dataset. We train separate models for men and women (i.e. Φ_m and Φ_f).

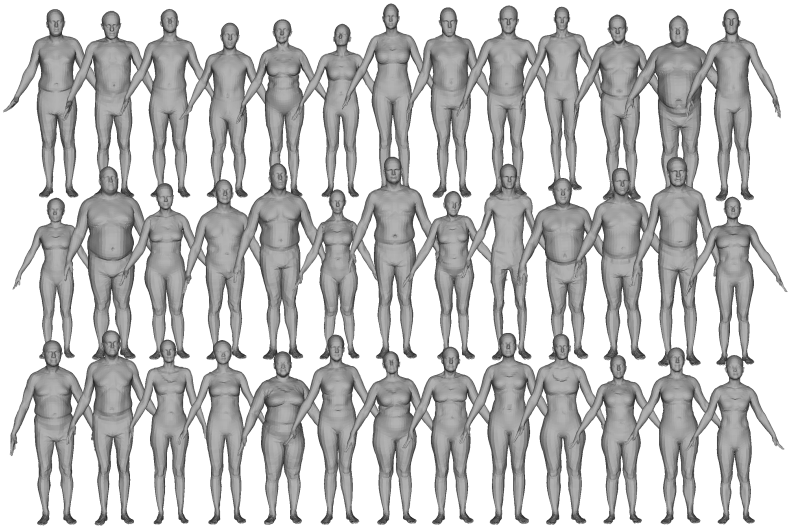


Figure 5-6: Sample registrations from the multishape dataset.

5.4.1 Pose Parameter Training

We first use the multi-pose dataset to train $\{\mathcal{J}, \mathcal{W}, \mathcal{P}\}$. To this end, we need to compute the rest templates, $\hat{\mathbf{T}}_i^P$, and joint locations, $\hat{\mathbf{J}}_i^P$, for each subject, i, as well as the pose parameters, $\vec{\theta}_j$, for each registration, j, in the dataset. We alternate between optimizing registration specific parameters $\vec{\theta}_j$, subject-specific parameters $\{\hat{\mathbf{T}}_i^P, \hat{\mathbf{J}}_i^P\}$, and global parameters $\{\mathcal{W}, \mathcal{P}\}$. We then learn the matrix, $\mathcal{J}$, to regress from subject-specific vertex locations, $\hat{\mathbf{T}}_i^P$, to subject-specific joint locations, $\hat{\mathbf{J}}_i^P$. To achieve all this, we minimize an objective function consisting of a data term, E_D, and several regularization terms $\{E_J, E_Y, E_P, E_W\}$ defined below.

The data term penalizes the squared Euclidean distance between registration ver-

tices and model vertices

$$E_D(\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \mathcal{W}, \mathcal{P}, \Theta) = \sum_{j=1}^{P_{\text{reg}}} ||\mathbf{V}_j^P - W(\hat{\mathbf{T}}_{s(j)}^P + B_P(\vec{\theta}_j; \mathcal{P}), \hat{\mathbf{J}}_{s(j)}^P, \vec{\theta}_j, \mathcal{W})||^2$$

where $\Theta = \left\{\vec{\theta}_1, \ldots, \vec{\theta}_{P_{\text{reg}}}\right\}$, $s(j)$ is the subject index corresponding to registration j, P_{reg} are the number of meshes in the pose trainings set, $\hat{\mathbf{T}}^P = \{\hat{\mathbf{T}}_i^P\}_{i=1}^{P_{\text{subj}}}$, $\hat{\mathbf{J}}^P = \{\hat{\mathbf{J}}_i^P\}_{i=1}^{P_{\text{subj}}}$ are the sets of all rest poses and joints, and P_{subj} is the number of subjects in the pose training set.

We estimate $207 \times 3 \times 6890 = 4,278,690$ parameters for the pose blend shapes, $\mathcal{P}$, $4 \times 3 \times 6890 = 82,680$ parameters for the skinning weights, $\mathcal{W}$, and $3 \times 6890 \times 23 \times 3 = 1,426,230$ for the joint regressor matrix, $\mathcal{J}$. To make the estimation well behaved, we regularize by making several assumptions. A symmetry regularization term, E_Y, penalizes left-right asymmetry for $\hat{\mathbf{J}}^P$ and $\hat{\mathbf{T}}^P$

$$E_Y(\hat{\mathbf{J}}^P, \hat{\mathbf{T}}^P) = \sum_{i=1}^{P_{\text{subj}}} \lambda_U ||\hat{\mathbf{J}}_i^P - U(\hat{\mathbf{J}}_i^P)||^2 + ||\hat{\mathbf{T}}_i^P - U(\hat{\mathbf{T}}_i^P)||^2,$$

where $\lambda_U = 100$, and where $U(\mathbf{T})$ finds a mirror image of vertices $\mathbf{T}$, by flipping across the sagittal plane and swapping symmetric vertices. This term encourages symmetric template meshes and, more importantly, symmetric joint locations. Joints are unobserved variables and along the spine they are particularly difficult to localize. While models trained without the symmetry term produce reasonable results, enforcing symmetry produces models that are visually more intuitive for animation.

Our model is hand segmented into 24 parts (Fig. 5-7). We use this segmentation to compute an initial estimate of the joint centers and a regressor $\mathcal{J}_I$ from vertices to these centers. This regressor computes the initial joints by taking the average of the ring of vertices connecting two parts. When estimating the joints for each subject we

regularize them to be close to this initial prediction:

$$E_J(\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P) = \sum_{i=1}^{P_{\text{subj}}} ||\mathcal{J}_I \hat{\mathbf{T}}_i^P - \hat{\mathbf{J}}_i^P||^2.$$

To help prevent overfitting of the pose-dependent blend shapes, we regularize them towards zero

$$E_P(\mathcal{P}) = ||\mathcal{P}||_F^2,$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Note that replacing the quadratic penalty with an L1 penalty would encourage greater sparsity of the blend shapes. We did not try this.

We also regularize the blend weights towards the initial weights, $\mathcal{W}_I$, shown in Fig. 5-7:

$$E_W(\mathcal{W}) = ||\mathcal{W} - \mathcal{W}_I||_F^2.$$

The initial weights are computed by simply diffusing the segmentation.

Altogether, the energy for training $\{\mathcal{W}, \mathcal{P}\}$ is as follows:

$$\begin{aligned} &E_*(\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \Theta, \mathcal{W}, \mathcal{P}) = \\ &\quad E_D + \lambda_Y E_Y + \lambda_J E_J + \lambda_P E_P + E_W, \end{aligned} \tag{5.14}$$

where $\lambda_Y = 100$, $\lambda_J = 100$ and $\lambda_P = 25$. These weights were set empirically. Our model has a large number of parameters and the regularization helps prevent overfitting. As the size of the training set grows, so does the strength of the data term, effectively reducing the influence of the regularization terms. Our experiments below with held-out test data suggest that the learned models are not overfit to the data and generalize well.

The overall optimization strategy is described in Sec. 5.4.3.

Joint regressor. Optimizing the above gives a template mesh and joint locations for each subject, but we want to predict joint locations for new subjects with new body

shapes. To that end, we learn the regressor matrix $\mathcal{J}$ to predict the training joints from the training bodies. We tried several regression strategies; what we found to work best, was to compute $\mathcal{J}$ using non-negative least squares [145] with the inclusion of a term that encourages the weights to add to one. This approach encourages sparsity of the vertices used to predict the joints. Making weights positive and add to one discourages predicting joints outside the surface. These constraints enforce the predictions to be in the convex hull of surface points. Figure 5-8 shows the non-zero elements of the regression matrix, illustrating that a sparse set of surface vertices are linearly combined to estimate the joint centers.

5.4.2 Shape Parameter Training

Our shape space is defined by a mean and principal shape directions $\{\bar{\mathbf{T}}, \mathcal{S}\}$. It is computed by running PCA on shape registrations from our multi-shape database after pose normalization. Pose normalization transforms a raw registration $\mathbf{V}_j^S$ into a registration, $\hat{\mathbf{T}}_j^S$, in the rest pose, $\vec{\theta}^*$. This normalization is critical to ensure that pose and shape are modeled separately.

To pose-normalize a registration, $\mathbf{V}_j^S$, we first have to estimate its pose. We denote $\hat{\mathbf{T}}_\mu^P$ and $\hat{\mathbf{J}}_\mu^P$ as the mean shape and mean joint locations from the multi-pose database respectively. Let $W_e(\hat{\mathbf{T}}_\mu^P, \hat{\mathbf{J}}_\mu^P, \vec{\theta}, \mathcal{W}), \mathbf{V}_{j,e}^S \in \mathbb{R}^3$ denote an edge of the model and of the registration respectively. An edge is obtained by subtracting a pair of neighboring vertices. To estimate the pose using an average generic shape $\hat{\mathbf{T}}_\mu^P$, we minimize the following sum of squared edge differences so that $\vec{\theta}_j =$

$$\arg\min_{\vec{\theta}} \sum_e ||W_e(\hat{\mathbf{T}}_\mu^P + B_P(\vec{\theta}; \mathcal{P}), \hat{\mathbf{J}}_\mu^P, \vec{\theta}, \mathcal{W}) - \mathbf{V}_{j,e}^S||^2, \tag{5.15}$$

where the sum is taken over all edges in the mesh. This allows us to get a good estimate of the pose without knowing the subject specific shape.

Once the pose $\vec{\theta}_j$ is known we solve for $\hat{\mathbf{T}}_j^S$ by minimizing

$$\hat{\mathbf{T}}_j^S = \arg\min_{\hat{\mathbf{T}}} ||W(\hat{\mathbf{T}} + B_P(\vec{\theta}_j; \mathcal{P}), \mathcal{J}\hat{\mathbf{T}}, \vec{\theta}_j, \mathcal{W}) - \mathbf{V}_j^S||^2.$$

This computes the shape that, when posed, matches the training registration. This shape is the pose-normalized shape.

We then run PCA on $\{\hat{\mathbf{T}}_j^S\}_{j=1}^{S_{\text{subj}}}$ to obtain $\{\bar{\mathbf{T}}, \mathcal{S}\}$. This procedure is designed to maximize the explained variance of vertex offsets in the rest pose, given a limited number of shape directions.

Note that the optimization of pose is critically important when building a shape basis from vertices. Without this step, pose variations of the subjects in the shape training dataset would be captured in the shape blend shapes. The resulting model would not be decomposed properly into shape and pose. Note also that this approach constrasts with SCAPE or BlendSCAPE, where PCA is performed in the space of per-triangle deformations. Unlike vertices, triangle deformations do not live in a Euclidean space [121]. Hence PCA on vertices is more principled and is consistent with the registration data term, which consists of squared vertex disparities.

Figure 5-9 visualizes the first three shape components. The figure also shows how the joint locations change with the changes in body shape. The joint positions are shown by the spheres and are computed from the surface meshes using the learned joint regression function. The lines connecting the joints across the standard deviations illustrate how the joint positions vary linearly with shape.

Figure 5-10 shows the relative cumulative variance of SMPL and BlendSCAPE. While SMPL requires many fewer PCs to account for the same percentage of overall variance, the variance is different in the two cases: one is variance in vertex locations and the other is variance in triangle deformations. Explained variance in deformations does not directly translate into explained variance in vertex locations. While this makes the two models difficult to compare precisely, triangle deformations have many more degrees of freedom and it is likely that there are many deformations that produce visually similar shapes. A model requiring fewer components is generally preferable.

5.4.3 Optimization summary

Pose parameters $\vec{\theta}_j$ in Eq. (5.14) are first initialized by minimizing the difference between the model and the registration edges, similar to Eq. (5.15) using an average

template shape. Then $\{\hat{\mathbf{T}}^P, \hat{\mathbf{J}}^P, \mathcal{W}, \mathcal{P}, \Theta\}$ are estimated in an alternating manner to minimize Eq. 5.14. We proceed to estimate $\mathcal{J}$ from $\{\hat{\mathbf{J}}^P, \hat{\mathbf{T}}^P\}$. We then run PCA on pose normalized subjects $\{\hat{\mathbf{T}}_j^S\}_{j=1}^{S_{\text{subj}}}$ to obtain $\{\bar{\mathbf{T}}, \mathcal{S}\}$. The final model is defined by $\{\mathcal{J}, \mathcal{W}, \mathcal{P}, \bar{\mathbf{T}}, \mathcal{S}\}$. Note that all training parameters except for $\{\bar{\mathbf{T}}, \mathcal{S}\}$ are found with gradient-based dogleg minimization [80]. Gradients are computed with automatic differentiation using the the Chumpy framework [15].

5.5 SMPL Evaluation

All training subjects gave prior informed written consent for their data to be used in creating statistical models for distribution. Registered meshes and identifiable subjects shown here are of professional models working under contract.

5.5.1 Quantitative Evaluation

We evaluate two types of error. *Model generalization* is the ability of the model to fit to meshes of new people and poses; this tests both shape and pose blend shapes. *Pose generalization* is the ability to generalize a shape of an individual to new poses of the same person; this primarily tests how well pose blend shapes correct skinning artifacts and pose-dependent deformations. Both are measured by mean absolute vertex-to-vertex distance between the model and test registrations. For this evaluation we use 120 registered meshes of four women and two men from the public Dyna dataset [146]. These meshes contain a variety of body shapes and poses. All meshes are in alignment with our template and none were used to train our models. Figure 5-11 (gray) shows four examples of these registered meshes.

We evaluate SMPL-LBS and SMPL-DQBS. We also compare these with a BlendSCAPE model [125] trained from exactly the same data as the SMPL models. The kinematic tree structure for SMPL and the BlendSCAPE model are the same: therefore we have the same number of pose parameters. We also compare the models using the same number of shape parameters.

To measure model generalization we first fit each model to each registered mesh,

optimizing over shape $\vec{\beta}$ and pose $\vec{\theta}$ to find the best fit in terms of squared vertex distances. Figure 5-11 shows fits of the SMPL-LBS (red) and BlendSCAPE (blue) models to the registered meshes. Both do a good job of fitting the data. The figure also shows how the model works. Illustrated are the estimated body shape, $\bar{\mathbf{T}}+B_S(\vec{\beta})$, and the effect of applying the pose blend shapes, $T_P(\vec{\beta},\vec{\theta})$.

For pose generalization, we take each indvidual, select one of the estimated body shapes from the generalization task, and then optimize the pose, $\vec{\theta}$, for each of the other meshes of that subject, keeping the body shape fixed. The assumption behind pose generalization is that, if a model is properly decomposed into pose and shape, then the model should be able to fit the same subject in a different pose, without readjusting shape parameters. Note that the pose blend shapes are trained to fit observed registrations. As such, they correct for problems of blend skinning and try to capture pose-dependent deformations. Since the pose blend shapes are not dependent on body shape, they capture something about the average deformations in the training set.

Figures 5-12 and 5-13 show the error of the SMPL models and BlendSCAPE as a function of the number of body shape coefficients used. The differences between SMPL and BlendSCAPE are small (on the order of 0.5mm) but SMPL is more accurate on average. Remarkably, SMPL-LBS and SMPL-DQBS are essentially identical in model generalization and SMPL-LBS is actually slightly better at pose generalization. This is surprising because the pose blend shapes have much more to correct with LBS. Possibly the simplicity of LBS helps with generalization to new poses. This analysis is important because it says that users can choose the simpler and faster LBS model over the DQBS model.

The plots also show how well standard LBS fits the test data. This corresponds to the SMPL-LBS model with no pose blend shapes. Not surprisingly, LBS produces much higher error than either BlendSCAPE or SMPL. LBS is not as bad in Fig. 5-12 because here the model can vary body shape parameters, effectively using changes in identity to try to explain deformations due to pose. Figure 5-13 uses a fixed body shape and consequently illustrates how LBS does not model pose-dependent

deformations realistically. Note that here we do not retrain a model specifically for LBS and expect such a model would be marginally more accurate.

5.5.2 Sparse SMPL

The pose blend shapes in SMPL are not sparse in the sense that a rotation of a part can influence any vertex of the mesh. With sufficient training data sparsity may emerge from data; *e.g.* the shoulder corrections will not be influenced by ankle motions. To make hand animation more intuitive, and to regularize the model to prevent long-range influences of joints, we can manually enforce sparsity. To this end, we trained a sparse version of SMPL by using the same sparsity pattern used for blend weights. In particular, we allow a vertex deviation to be influenced by at most 4 joints. Since every joint corresponds to a rotation matrix, the pose blend shape corresponding to any given vertex will be driven by 9×4 numbers as opposed to 9×23.

This model is referred to as SMPL-LBS-Sparse in Figs. 5-12 and 5-13. It is consistently less accurate than the regular SMPL-LBS model but may still be useful to animators. This suggests that SMPL-LBS is not overfit to the training data and that sparseness reduces the capacity of the model. The sparse model sometimes produces slight discontinuities at boundaries were vertices are influenced by different joint angles. Other strategies to enforce sparsity could be adopted, such as using an L1 prior or enforcing smoothness in the pose blend shapes. These approaches, however, would complicate the training process.

5.5.3 Visual Evaluation

Figure 5-14 illustrates the decomposition of shape parameters $\vec{\beta}$ and pose parameters $\vec{\theta}$ in SMPL. Pose is held constant from left to right across each row while varying the shape. Similarly, the shape of each person is held constant while varying the pose from top to bottom in each column. The bodies are reposed using poses from the CMU mocap database [147]. Note that the pose-dependent deformations look natural

through a wide range of poses, despite very different body shapes. This illustrates that the joint regression works well and that the blend shapes are general.

5.5.4 Run-time Performance

The run-time cost of SMPL is dominated by skinning and blend-shape multiplication. Many skinning implementations exist, and we do not claim to have the fastest. Performance of our own CPU-based implementation, and a comparison against BlendSCAPE, is shown in Fig. 5-15. The plot shows the time needed to generate the vertices. Note that our BlendSCAPE rendering makes use of multiple cores, while the SMPL rendering does not; this is why the system time for BlendSCAPE is higher than the wall-clock time. Note that here we are showing the cost of changing body shape. For most applications, this is done once and the shape is then held fixed. The cost of animating the mesh then comes from only the pose blend shapes; this cost corresponds to 0 shape coefficients.

For meshes with the same number of vertices, SCAPE will always be slower. In SMPL each blend shape is of size $3N$, requiring that many multiplications per shape. SCAPE uses triangle deformations with 9 elements per triangle and there are roughly twice as many triangles as vertices. This results in roughly a factor of 6 difference between SMPL and SCAPE in terms of basic multiplications.

5.5.5 Compatibility with Rendering Engines

Because SMPL is built on standard skinning, it is compatible with existing 3D animation software. In particular, for a given body shape, we generate the subject-specific rest-pose template mesh and skeleton (estimated joint locations) and we export SMPL as a rigged model with pose blend shapes in Autodesk's Filmbox (FBX) file format, giving cross-platform compatibility. The model loads as a typical rigged mesh and can be animated as usual in standard 3D animation software.

Pose blend weights can be precomputed, baked into the model, and exported as an animated FBX file. This kind of file can be loaded into animation packages and

played directly. We have tested the animated FBX files in Maya, Unity, and Blender.

Pose blend weights can also be computed on the fly given the pose, $\vec{\theta}_t$, at time t. To enable this, we provide scripts that take the joint angles and compute the pose blend weights. We have tested loading and animating SMPL in Maya 2013, 2014 and 2015. The animator can animate the model using any of the conventional animation methods typically used in Maya. We will provide a Python script that runs inside Maya to apply blend-shape corrections to an animated SMPL model. The pose blend shape values can be viewed and/or edited manually within Maya if desired. We have also tested SMPL in Unity. In SMPL, the blend weights range from -1 to +1 while in Unity they range form 0 to 1. Consequently, we scale and recenter our weights for compatibility. For runtime computation of pose blend shape coefficients, we provide a C# script that the user can attach to SMPL's mesh game object.

The SMPL model with shape and pose blend shapes, and the evaluation meshes, are available for research purposes at `http://smpl.is.tue.mpg.de`.

5.6 Discussion

Why does it work? First, good quality training data is important. Here we use thousands of high-quality registered template meshes. Importantly, the pose training data spans a range of body shapes enabling us to learn a good predictor of joint locations. Second, training all the parameters (template shape, blend weights, joint regressor, shape/pose blend shapes) to minimize vertex reconstruction error is important to obtain a good model. Here the simplicity of the model is an advantage as it enables training everything with large amounts of data.

In contrast to the scattered-data interpolation methods, we learn the blend shapes from a large set of training meshes covering the space of possible poses and learn a simpler function relating pose to blend-shape weights. In particular, our function is linear in the elements of the part rotation matrices. The larger support of the learned linear functions as opposed to radial basis functions allows the model to generalize to arbitrary poses; in addition the simple linear form makes it fast to animate in a

game engine without baking in the weights. Because elements of a rotation matrix are constrained, the model cannot "blow up" when generalizing outside the training set.

SMPL is an additive model in vertex space. In contrast, while SCAPE also factors deformations into shape and pose deformations, SCAPE multiplies the triangle deformations. With SCAPE a bigger person will have bigger pose-dependent deformations even though these deformations are not learned for different body shapes. Despite this, in our experiments, the SCAPE approach is less accurate at generalizing to new shapes.

Why is it more accurate than BlendSCAPE? Models based on the statistics of triangle deformations have dominated the recent literature [46, 62, 121, 142]. Such models are not trained to reproduce their training registrations directly. Instead, they are trained to reproduce the local deformations that produced those registrations. Part of the tractability of training these models comes from the ability to train deformations independently across triangles. As a result, long range distances and relationships are not preserved as well as local relationships between vertices. We speculate that an advantage of vertex based models (such as SMPL and [120]) is that they can be trained to minimize the mean squared error between the model and training vertices. Theoretically one could train a SCAPE model to minimize vertex error in global coordinates, but the inner loop of the optimization would involve solving a least-squares problem to reconstruct vertices from the deformations. This would significantly increase the cost of optimization and make it difficult to train the model with large amounts of data.

Why has it not been done before? While we think the SMPL model is a natural way to extend blend skinning, we are unaware of any previous published versions. Unfortunately, the obvious implementation makes the pose blend shapes a linear function of $\vec{\theta}$. This does not work. The key to SMPL's performance is to make the blendshapes a linear function of the elements of $R^*(\vec{\theta})$. This formulation, sufficient training data, and a good optimization strategy make it possible to learn the model.

The closest work to ours is the pioneering work of Allen et al. [120]. Their model is

more complex than ours, using radial basis functions for scattered data interpolation, shape-dependent pose deformations, and a fixed set of carrying angles. Consequently training it is also complex and requires a good initialization. They had limited data and difficulty with overfitting so they restricted their body shape PCA space. As a result, the model did not generalize well to new shapes and poses. Our simpler model lets us learn it from large datasets and having more data makes the simple model perform well.

Other features for driving pose blend shapes. We experimented with driving pose blendshapes linearly from other features, such as raw $\vec{\theta}$, simple polynomials of $\vec{\theta}$, and trigonometric functions (sin, cos) of $\vec{\theta}$. None of these performed as well as our proposed formulation. Using raw $\vec{\theta}$ has serious limitations because the values vary between -π and π. Imagine a twist of the neck (Fig. 5-16), which produces negative and postive angles about the vertical axis. Standard LBS will cause the neck to shrink as it rotates in either direction. To counteract this, we need a blend shape that increases the neck volume no matter which direction it rotates. Unfortunately, if the blendshapes are trained to expand during rightwards rotation (to counteract LBS shrinkage), they would contract during leftward rotation.

In general one can think of replacing the raw rotations with any functions of rotations and using these to weight the blend shapes. An exhaustive search is impossible and other features may work as well as our method; for example, we did not experiment with normalized quaternions.

Our pose blend shapes function is also very different from scattered data interpolation methods like WPSD [135, 136], which use a discrete number of poses and associated corrections are interpolated between them using RBFs. In practice, a large number of poses might be needed to cover the pose space well. This makes animation slow since the closest key poses have to be found at run time.

Limitations. The pose-dependent offsets of SMPL are not dependent on body shape. It is surprising how well SMPL works without this, but the general approach would likely not work if we were to model a space of nonrealistic animated characters in which body part scales vary widely, or a space of humans that includes infants and

adults. This limitation could be addressed by training a more general function that would take elements of $R^*(\vec{\theta})$ together with $\vec{\beta}$ to predict the blend shape coefficients.

As described, the SMPL model is a function of joint angles and shape parameters only: it does not model breathing, facial motion, muscle tension, or any changes independent of skeletal joint angles and overall shape. These could potentially be learned as additional additive blendshapes if the appropriate factored data is available (cf. [70]).

While we learn most model parameters, we do not learn them all. We manually define the segmentation of the template into parts, the topology of the mesh, and the zero pose. Theoretically these could also be learned but we expect only marginal improvements for significant effort.

Future work. SMPL uses 207 pose blend shapes. This could likely be reduced by performing PCA on the blend shapes. This would reduce the number of multiplications and consequently increase rendering speed. Also, here we fit our model to registered meshes but could fit SMPL to mocap marker data (cf. MoSh [14]), depth data, or video. We anticipate that optimizing the pose and shape of a SMPL-LBS model will be significantly faster than optimizing a SCAPE model of similar quality.

5.7 Conclusions

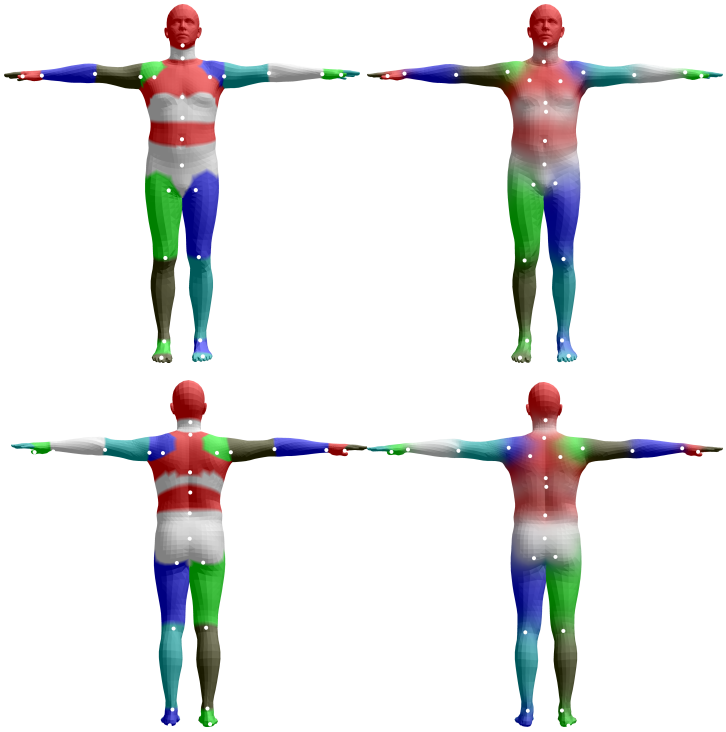
Our goal was to create a skeletally-driven human body model that could capture body shape and pose variation as well as, or better than, the best previous models while being compatible with existing graphics pipelines and software. To that end, SMPL uses standard skinning equations and defines body shape and pose blend shapes that modify the base mesh. We train the model on thousands of aligned scans of different people in different poses. The form of the model makes it possible to learn the parameters from large amounts of data while directly minimizing vertex reconstruction error. Specifically we learn the rest template, joint regressor, body shape model, and pose blend shapes. The surprising result is that, when BlendSCAPE and SMPL are trained on exactly the same data, the vertex-based model is more accurate and sig-

nificantly more efficient to render than the deformation-based model. Also surprising is that a relatively small set of learned blend shapes do as good a job of correcting the errors of LBS as they do for DQBS. SMPL can be exported as an FBX file and we make scripts available to animate the model in common rendering systems. This will allow anyone to realistically animate human bodies.

5.8 Appendix

5.8.1 Mathematical Notation

We summarize our notation here and in Table 5.1. Matrices $\mathcal{A} \in \mathbb{R}^{n \times m}$ are denoted with math calligraphic typeface. Vectors $\mathbf{A} \in \mathbb{R}^m$ are denoted with uppercase boldface, expect for the special case of 3-vectors, which are denoted with lower case $\mathbf{a} \in \mathbb{R}^3$ to distinguish a particular vertex from a vector of concatenated vertices. The notation $A() : \mathbb{R}^m \mapsto \mathbb{R}^n$ is used to denote a function that maps vectors in an m-dimensional space to vectors in n-dimensional space. Typically, indices are used as follows: j iterates over mesh registrations, k iterates over joint angles and i iterates over subjects, and t denotes time.



(a) Segmentation (b) Initialization $\mathcal{W}_I$

Figure 5-7: **Initialization of joints and blend weights.** Discrete part segmentation in (a) is diffused to obtain initial blend weights, $\mathcal{W}_I$, in (b). Initial joint centers are shown as white dots.

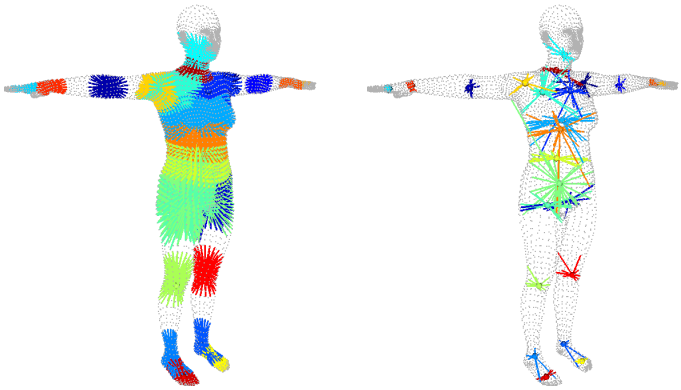
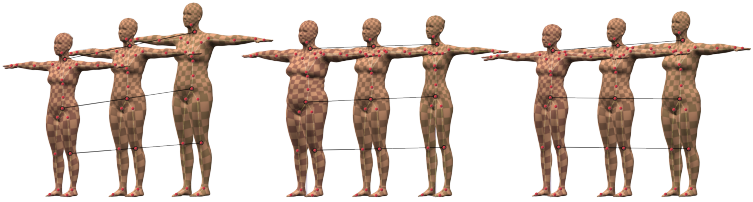


Figure 5-8: **Joint regression.** (left) Initialization. Joint locations can be influenced by locations on the surface, indicated by the colored lines. We assume that these influences are somewhat local. (right) Optimized. After optimization we find a sparse set of vertices and associated weights influencing each joint.



(a) PC 1 (b) PC 2 (c) PC 3

Figure 5-9: **Shape blend shapes.** The first three shape principal components of body shape are shown. PC1 and PC2 vary from -2 to +2 standard deviations from left to right, while PC3 varies from -5 to +5 standard deviations to make the shape variation more visible. Joint locations (red dots) vary as a function of body shape and are predicted using the learned regressor, $\mathcal{J}$.

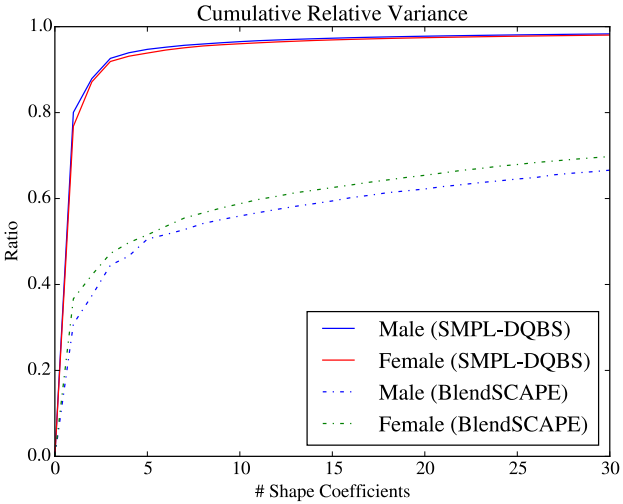


Figure 5-10: Cumulative relative variance of the CAESAR dataset explained as a function of the number of shape coefficients. For SMPL the variance is in vertex locations, while for BlendSCAPE it is in triangle deformations.

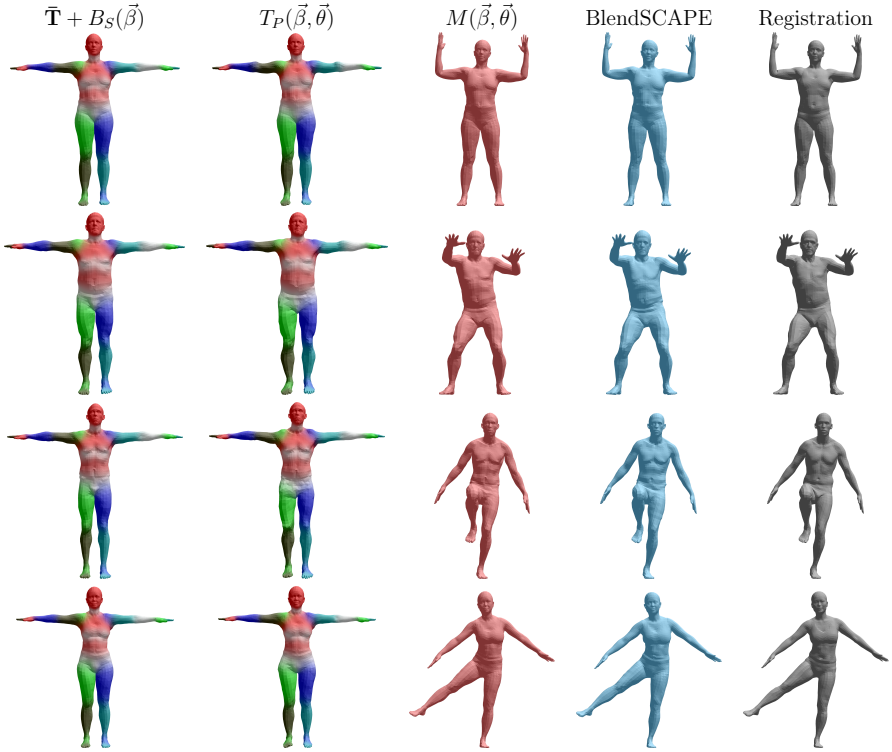


Figure 5-11: Model fitting with intermediate stages. We fit both BlendSCAPE (blue) and SMPL-LBS, $M(\vec{\beta}, \vec{\theta})$, (red) to registered meshes by optimizing pose and shape. $\bar{\mathbf{T}} + B_S(\vec{\beta})$ shows the estimated body shape and $T_P(\vec{\beta}, \vec{\theta})$ shows the effects of pose-dependent blend shapes. Here we show SMPL-LBS, because T_P shows more variation due to pose than SMPL-DQBS.

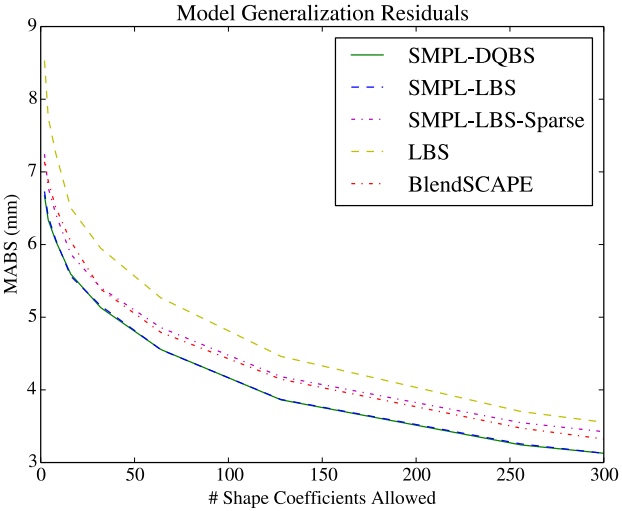


Figure 5-12: Model generalization indicates how well we can fit an independent registration. Mean absolute vertex error versus the number of shape coefficients used.

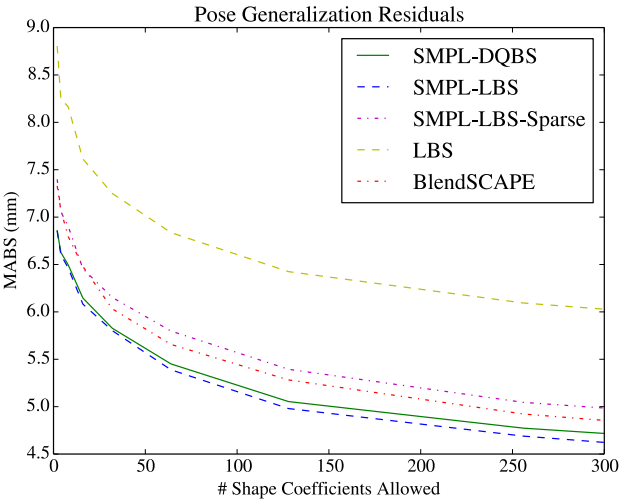


Figure 5-13: Pose generalization error indicates how well a fitted shape generalizes to new poses.

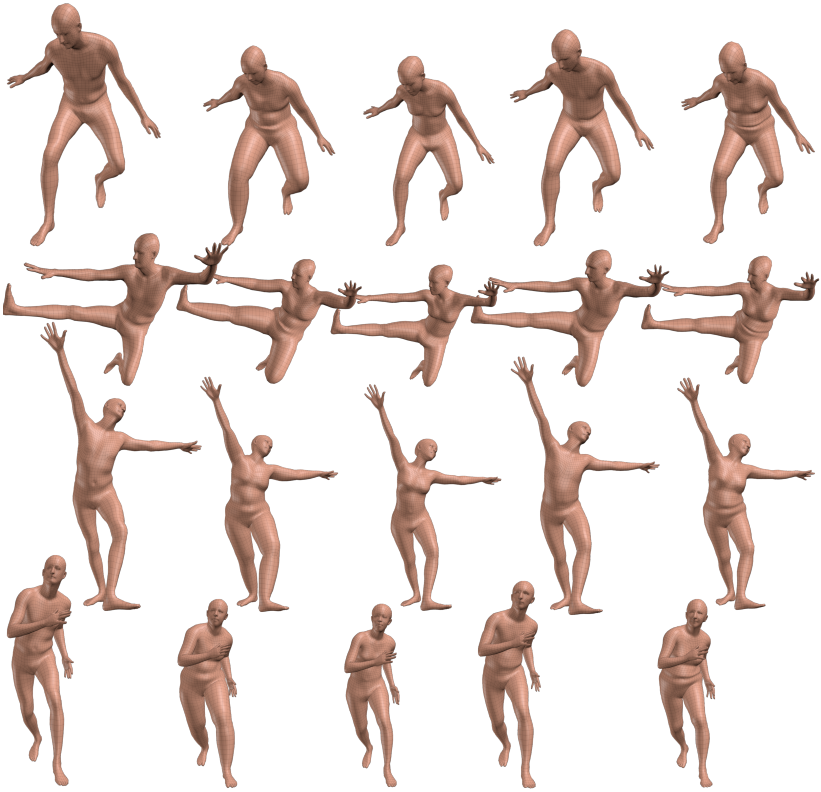


Figure 5-14: **Animating SMPL.** Decomposition of SMPL parameters into pose and shape: Shape parameters, $\vec{\beta}$, vary across different subjects from left to right, while pose parameters, $\vec{\theta}$, vary from top to bottom for each subject.

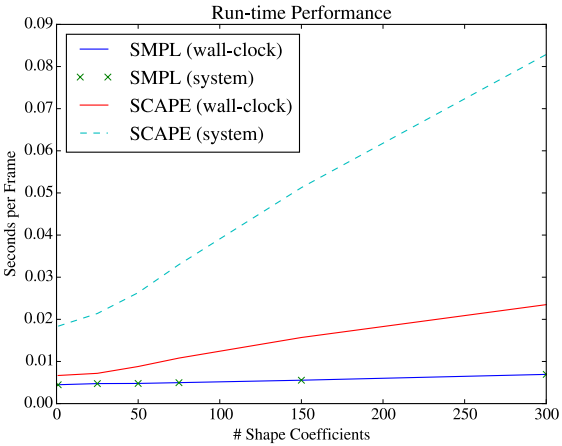
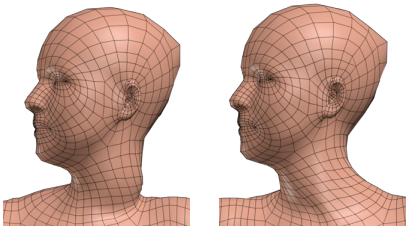


Figure 5-15: Performance of SMPL and BlendSCAPE vary with the number of body shape coefficients used. Performance shown here is from a 2014 Macbook Pro.



(a) Euler-angles (b) SMPL parameterization

Figure 5-16: **Parameterizing pose blend shapes.** (a) Pose blend shapes parameterized by Euler angles cause significant problems. (b) our proposed parameterization allows the head to rotate in either direction with natural deformations.

Table 5.1: Table of Notation

Model generation functions

W	$\equiv$	Skinning function
M	$\equiv$	SMPL function
B_P	$\equiv$	Pose blendshapes function
B_S	$\equiv$	Shape blendshapes function
J	$\equiv$	Joint regressor: Predicts joints from surface

Model input parameters (controls)

$\vec{\beta}$	$\equiv$	Shape parameters
$\vec{\theta}$	$\equiv$	Pose parameters
$\vec{\omega}$	$\equiv$	Scaled axis of rotation; the 3 pose parameters corresponding to a particular joint
$\vec{\theta^*}$	$\equiv$	Zero pose or rest pose; the effect of the pose blendshapes is zero for that pose

Model parameters (parameters learned)

$\mathcal{S}$	$\equiv$	Shape blendshapes
$\mathcal{P}$	$\equiv$	Pose blendshapes
$\mathcal{W}$	$\equiv$	Blendweights
$\mathcal{J}$	$\equiv$	Joint regressor matrix
$\bar{\mathbf{T}}$	$\equiv$	Mean shape of the template

Training data

$\mathbf{V}$	$\equiv$	A registration
$\mathbf{V}^P$	$\equiv$	Pose dataset registration
$\mathbf{V}^S$	$\equiv$	Shape dataset registration
$\hat{\mathbf{T}}^P$	$\equiv$	Pose dataset subject shape; body vertices in the template pose
$\hat{\mathbf{J}}^P$	$\equiv$	Pose dataset subject joint locations in the template pose
$\hat{\mathbf{T}}^P_\mu$	$\equiv$	Mean shape of a pose subject; body vertices in the template pose
$\hat{\mathbf{T}}^S$	$\equiv$	Shape dataset subject shape; body vertices in the template pose
$\hat{\mathbf{T}}^S_\mu$	$\equiv$	Mean shape of a subject in the shape dataset; body vertices in the template pose

Chapter 6

Conclusion

This thesis provides methods for human shape and pose estimation from three different types of observations. We explore the potential of a statistically motivated body model, and enrich the ability to solve for parameters with the generation of images, non-rigid deformations, marker positions, and registrations. Optimization methods for this are arguably very simple: Gauss-Newton gradient-based optimization, with a sum-of-squares data term, is used throughout this work. The focus is therefore not on search strategies but on the formulation and differentiation of good observation models. We now revisit the contributions of each work in turn.

The contribution of MoSh in Chapter 3 is to show that a small number of 3D markers can reveal a surprising amount of information about shape and nonrigid tissue motion. Traditional approaches estimate a skeleton from the data, and throw away valuable information about body shape and nonrigid jiggle. MoSh captures that information from archival data without high-resolution body scans. A crucial aspect of MoSh is marker placement parameterization and refinement: as it is impossible to place markers exactly, the data can help inform where markers are placed on the body, which in turn helps estimate shape, pose, and nonrigid tissue motion.

While OpenDR in Chapter 4 is applied to body shape estimation, the contribution of that work is more general. The key novelty is the proposal of a reusable differentiable renderer, which provides not only rendered images, but also the derivatives of the image process with respect to controlling parameters. This allows gradient-based

parameter estimation from images. Whereas many works have differentiated objective functions including some form of rendering, this work is the first we know of in which a differentiable renderer is built and distributed by itself.

Finally, the contribution of SMPL in Chapter 5 lies in the formulation and training of a statistically-motivated body model. We show that this model is superior to state-of-the art models in speed, accuracy, and compatibility. This model is unusual in that it neither uses radial basis function interpolation, nor does it use deformation transfer style constructions: instead, pose-dependent are driven directly from rotation matrix elements.

All of these works are accompanied by online materials intended to further the field: MoSh is released with ground-truth datasets, the SMPL model itself is available for research purposes, and OpenDR is freely available online under the MIT software license.

We now turn to possible enhancements and future directions for this work.

6.1 Future Directions

Improvements to realism would be a step forward for all the works presented. Specifically, it is interesting to consider ways to add realism without increasing the number of parameters (and generally the size of the search problem). MoSh, for example, could benefit from simulating infrared camera images (from which marker positions were derived) directly. Likewise, OpenDR would benefit from the modeling of shadows; ambient occlusion may be the easiest to model. Finally, the realism of SMPL may be improved with the addition of more physically motivated deformation.

Computational efficiency is another issue that bears further study. The performance of OpenDR and MoSh were both hampered by constant reallocation of sparse matrices, and both might benefit from moving to reverse-mode autodifferentiation from forward-mode autodifferentiation. The SMPL model may benefit either from compression of the pose-dependent blendshapes, either by inducing sparsity or via some form of dimensionality reduction such as PCA.

It is notable that (unlike many works in computer vision) no stochastic search or discriminative techniques are used in this thesis. While we find it encouraging how much is possible with gradient-based approaches alone, supplementing our methods with these techniques may help overcome the local minima that often hinder gradient-based methods.

Some parameters in these works were specified by hand, and the tuning of these parameters presented a real challenge. Ensuring generality of any candidate set of parameters means testing on large datasets; and because evaluation often includes qualitative perceptual criteria, evaluation can be time-consuming. Ways to address this in the future might include a more perceptually-oriented data term, the automatic learning of more parameters from more ground-truth data, or the inclusion of more physically (or statistically) motivated regularizations.

Bibliography

[1] Andrew Y. Ng and Michael I. Jordan. "On Discriminative vs. Generative Classifiers: A comparison of logistic regression and naive Bayes". In: *Advances in Neural Information Processing Systems 14*. Ed. by T.G. Dietterich, S. Becker, and Z. Ghahramani. MIT Press, 2002, pp. 841–848. URL: `http://papers.nips.cc/paper/2020-on-discriminative-vs-generative-classifiers-a-comparison-of-logistic-regression-and-naive-bayes.pdf`.

[2] Steve Preeg. *The Curious Face of Benjamin Button*. GRID '09, Stockholm, Sweden. 2009. URL: `https://vimeo.com/26718845`.

[3] Rob Bredow. "From mocap to movie: the making of'The Polar Express'". In: *International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2005 Courses: Los Angeles, California*. Vol. 2005. 2005.

[4] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. "Real-time human pose recognition in parts from single depth images". In: *Communications of the ACM* 56.1 (2013), pp. 116–124.

[5] Peter Blanchonette. *Jack human modelling tool: A review*. Tech. rep. DTIC Document, 2010.

[6] Cary B Phillips and Norman I Badler. "Jack: A toolkit for manipulating articulated figures". In: *Proceedings of the 1st annual ACM SIGGRAPH symposium on User Interface Software*. ACM. 1988, pp. 221–229.

[7] Don B Chaffin. *Digital human modeling for vehicle and workplace design*. 2001.

[8] J Mark Porter, Martin Freer, Keith Case, and Maurice C Bonney. "Computer aided ergonomics and workspace design". In: (1995).

[9] S Jay Olshansky, Douglas J Passaro, Ronald C Hershow, Jennifer Layden, Bruce A Carnes, Jacob Brody, Leonard Hayflick, Robert N Butler, David B Allison, and David S Ludwig. "A potential decline in life expectancy in the United States in the 21st century". In: *New England Journal of Medicine* 352.11 (2005), pp. 1138–1145.

[10] KR Fontaine and Ivan Barofsky. "Obesity and health-related quality of life". In: *Obesity reviews* 2.3 (2001), pp. 173–182.

[11] Rachel P Wildman, Paul Muntner, Kristi Reynolds, Aileen P McGinn, Swapnil Rajpathak, Judith Wylie-Rosett, and MaryFran R Sowers. "The obese without cardiometabolic risk factor clustering and the normal weight with cardiometabolic risk factor clustering: prevalence and correlates of 2 phenotypes among the US population (NHANES 1999-2004)". In: *Archives of internal medicine* 168.15 (2008), pp. 1617–1624.

[12] Jonathan CK Wells, Philip Treleaven, and Tim J Cole. "BMI compared with 3-dimensional body shape: the UK National Sizing Survey". In: *The American journal of clinical nutrition* 85.2 (2007), pp. 419–425.

[13] Sabine Pur, Erst Stahl, Michael Wittmann, Georg Wittmann, and Stefan Weinfurtner. *Retourenmanagement im Online-Handel — Das Beste daraus machen*. Tech. rep. Regensburg, Germany: ibi research GmbH, 2013.

[14] Matthew M. Loper, Naureen Mahmood, and Michael J. Black. "MoSh: Motion and Shape Capture from Sparse Markers". In: *ACM Trans. Graph., (Proc. SIGGRAPH Asia)* 33.6 (Nov. 2014), 220:1–220:13. DOI: 10.1145/2661229.2661273. URL: http://doi.acm.org/10.1145/2661229.2661273.

[15] Matthew M. Loper and Michael J. Black. "OpenDR: An Approximate Differentiable Renderer". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars. Vol. 8695. Lecture Notes in Computer Science. Heidelberg: Springer, 2014, pp. 154–169. ISBN: 978-3-319-10583-3. DOI: 10.1007/978-3-319-10584-0_11.

[16] Matthew M. Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. "SMPL: A Skinned Multi-Person Linear Model". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 34.6 (2015). DOI: 10.1145/2816795.2818132. URL: http://doi.acm.org/10.1145/2816795.2818132.

[17] David A. Hirshberg, Matthew Loper, Eric Rachlin, and Michael J. Black. "Coregistration: Simultaneous Alignment and Modeling of Articulated 3D Shape". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon, Svetlana Lazebnik, Pietro Perona, Yoichi Sato, and Cordelia Schmid. Vol. 7577. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, pp. 242–255. ISBN: 978-3-642-33782-6. DOI: 10.1007/978-3-642-33783-3_18.

[18] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. "FAUST: Dataset and evaluation for 3D mesh registration". In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. Columbus, Ohio, USA, June 2014.

[19] Javier Romero, Matthew Loper, and Michael J. Black. "FlowCap: 2D Human Pose from Optical Flow". In: *German Conference on Pattern Recognition (GCPR)*. 2015.

[20] Federica Bogo, Michael J. Black, Matthew Loper, and Javier Romero. "Detailed Full-Body Reconstructions of Moving People from Monocular RGB-D Sequences". In: *International Conference on Computer Vision (ICCV)*. Dec. 2015.

[21] D. Marr and H. K. Nishihara. "Representation and Recognition of the Spatial Organization of Three-Dimensional Shapes". English. In: *Proceedings of the Royal Society of London. Series B, Biological Sciences* 200.1140 (1978), pp. 269–294. ISSN: 00804649. URL: http://www.jstor.org/stable/77391.

[22] E.P. Hanavan. *A mathematical model of the human body*. A Mathematical Model of the Human Body v. 32, no. 3. Aerospace Medical Research Laboratories, Aerospace Medical Division, Air Force Systems Command, 1964. URL: http://books.google.de/books?id=7aslfUwMqKsC.

[23] David Hogg. "Model-based vision: a program to see a walking person". In: *Image and vision computing* 1.1 (1983), pp. 5–20.

[24] Mun Wai Lee and Isaac Cohen. "Proposal maps driven mcmc for estimating human body pose in static images". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 2. IEEE. 2004, pp. II–334.

[25] Cristian Sminchisescu and Bill Triggs. "Kinematic jump processes for monocular 3D human tracking". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2003, pp. I–69.

[26] Karl Rohr. "Towards model-based recognition of human movements in image sequences". In: *CVGIP: Image understanding* 59.1 (1994), pp. 94–115.

[27] Romer Rosales and Stan Sclaroff. "Inferring body pose without tracking body parts". In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. IEEE. 2000, pp. 721–727.

[28] Hedvig Sidenbladh, Michael J Black, and David J Fleet. "Stochastic tracking of 3D human figures using 2D image motion". In: *Computer Vision—ECCV 2000*. Springer, 2000, pp. 702–718.

[29] Jonathan Deutscher, Andrew Blake, and Ian Reid. "Articulated body motion capture by annealed particle filtering". In: *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*. Vol. 2. IEEE. 2000, pp. 126–133.

[30] Mun Wai Lee and Ram Nevatia. "Human pose tracking using multi-level structured models". In: *Computer Vision–ECCV 2006*. Springer, 2006, pp. 368–381.

[31] Stefan Wachter and Hans-Hellmut Nagel. "Tracking of persons in monocular image sequences". In: *Nonrigid and Articulated Motion Workshop, 1997. Proceedings., IEEE*. IEEE. 1997, pp. 2–9.

[32] Leonid Sigal, Sidharth Bhatia, Stefan Roth, Michael J Black, and Michael Isard. "Tracking loose-limbed people". In: *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*. Vol. 1. IEEE. 2004, pp. I–421.

[33] Leonid Sigal and Michael J Black. "Predicting 3d people from 2d pictures". In: *Articulated Motion and Deformable Objects*. Springer, 2006, pp. 185–195.

[34] Dariu M Gavrila and Larry S Davis. "3-D model-based tracking of humans in action: a multi-view approach". In: *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*. IEEE. 1996, pp. 73–80.

[35] J. P. Lewis, Matt Cordner, and Nickson Fong. "Pose Space Deformation: A Unified Approach to Shape Interpolation and Skeleton-driven Deformation". In: *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '00. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 165–172. ISBN: 1-58113-208-5. DOI: 10.1145/344779.344862. URL: http://dx.doi.org/10.1145/344779.344862.

[36] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. "Geometric skinning with approximate dual quaternion blending". In: *ACM Transactions on Graphics (TOG)* 27.4 (2008), 105:1–105:23.

[37] Alex Mohr and Michael Gleicher. "Building Efficient, Accurate Character Skins from Examples". In: *ACM SIGGRAPH 2003 Papers*. SIGGRAPH '03. San Diego, California: ACM, 2003, pp. 562–568. ISBN: 1-58113-709-5. DOI: 10.1145/1201775.882308. URL: http://doi.acm.org/10.1145/1201775.882308.

[38] Ioannis A Kakadiaris and Dimitri Metaxas. "3D human body model acquisition from multiple views". In: *Computer Vision, 1995. Proceedings., Fifth International Conference on*. IEEE. 1995, pp. 618–623.

[39] DM Gavrila, LS Davis, et al. "Towards 3-d model-based tracking and recognition of human movement: a multi-view approach". In: *International workshop on automatic face-and gesture-recognition*. Citeseer. 1995, pp. 272–277.

[40] B. Allen, B. Curless, and Z. Popović. "The space of human body shapes: Reconstruction and parameterization from range scans". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 22.3 (2003), pp. 587–594.

[41] Brett Allen, Brian Curless, Zoran Popović, and Aaron Hertzmann. "Learning a correlated model of identity and pose-dependent body shape variation for real-time synthesis". In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Vienna, Austria: Eurographics Association, 2006, pp. 147–156.

[42] Dragomir Anguelov. "Learning Models of Shape from Three-dimensional Range Data". AAI3197402. PhD thesis. Stanford, CA, USA, 2006. ISBN: 0-542-43099-1.

[43] Yizhou Yu, Kun Zhou, Dong Xu, Xiaohan Shi, Hujun Bao, Baining Guo, and Heung-Yeung Shum. "Mesh editing with poisson-based gradient field manipulation". In: *ACM Transactions on Graphics (TOG)*. Vol. 23. 3. ACM. 2004, pp. 644–651.

[44] Robert W Sumner and Jovan Popović. "Deformation transfer for triangle meshes". In: *ACM Transactions on Graphics (TOG)* 23.3 (2004), pp. 399–405.

[45] Mario Botsch and Olga Sorkine. "On linear variational surface deformation methods". In: *Visualization and Computer Graphics, IEEE Transactions on* 14.1 (2008), pp. 213–230.

[46] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. "SCAPE: Shape Completion and Animation of PEople". In: *ACM Trans. Graph. (Proc. SIGGRAPH* 24.3 (2005), pp. 408–416. ISSN: 0730-0301.

[47] Arjun Jain, Jonathan Tompson, Yann LeCun, and Christoph Bregler. "Modeep: A deep learning framework using motion features for human pose estimation". In: *arXiv preprint arXiv:1409.7963* (2014).

[48] Alexandru O Bălan, Leonid Sigal, Michael J Black, James E Davis, and Horst W Haussecker. "Detailed human shape and pose from images". In: *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE. 2007, pp. 1–8.

[49] P. Guan, A. Weiss, A. Bălan, and M. J. Black. "Estimating human shape and pose from a single image". In: *Int. Conf. on Computer Vision, ICCV*. Sept. 2009, pp. 1381–1388.

[50] Alexandru O Bălan, Michael J Black, Horst Haussecker, and Leonid Sigal. "Shining a light on human pose: On shadows, shading and the estimation of pose and shape". In: *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*. IEEE. 2007, pp. 1–8.

[51] Alexandru O Bălan and Michael J Black. "The naked truth: Estimating body shape under clothing". In: *Computer Vision–ECCV 2008*. Springer, 2008, pp. 15–29.

[52] A. Weiss, D. Hirshberg, and M.J. Black. "Home 3D body scans from noisy image and range data". In: *Int. Conf. on Computer Vision (ICCV)*. Barcelona: IEEE, Nov. 2011, pp. 1951–1958.

[53] Ioannis A Kakadiaris and Dimitris Metaxas. "Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection". In: *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR'96, 1996 IEEE Computer Society Conference on*. IEEE. 1996, pp. 81–87.

[54] Yan Cui, Will Chang, Tobias Nöll, and Didier Stricker. "KinectAvatar: fully automatic body capture using a single kinect". In: *Computer Vision-ACCV 2012 Workshops*. Springer. 2013, pp. 133–147.

[55] Will Chang and Matthias Zwicker. "Global registration of dynamic range scans for articulated model reconstruction". In: *ACM Transactions on Graphics (TOG)* 30.3 (2011), p. 26.

[56] Sang Il Park and Jessica K. Hodgins. "Data-driven Modeling of Skin and Muscle Deformation". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 27.3 (Aug. 2008), 96:1–96:6. ISSN: 0730-0301. DOI: 10.1145/1360612.1360695. URL: http://doi.acm.org/10.1145/1360612.1360695.

[57] Joel Carranza, Christian Theobalt, Marcus A Magnor, and Hans-Peter Seidel. "Free-viewpoint video of human actors". In: *ACM Transactions on Graphics (TOG)* 22.3 (2003), pp. 569–577.

[58] Sang Il Park and Jessica K. Hodgins. "Capturing and Animating Skin Deformation in Human Motion". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 25.3 (July 2006), pp. 881–889. ISSN: 0730-0301. DOI: 10.1145/1141911.1141970. URL: http://doi.acm.org/10.1145/1141911.1141970.

[59] Leonid Sigal, Alexandru Bălan, and Michael J Black. "Combined discriminative and generative articulated pose and non-rigid shape estimation". In: *Advances in neural information processing systems*. 2007, pp. 1337–1344.

[60] Nils Hasler, Carsten Stoll, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. "Estimating body shape of dressed humans". In: *Computers & Graphics* 33.3 (2009), pp. 211–216.

[61] N. Hasler, H. Ackermann, B. Rosenhahn, T. Thormählen, and H.-P. Seidel. "Multilinear pose and body shape estimation of dressed subjects from image sets". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. 2010, pp. 1823–1830. DOI: 10.1109/CVPR.2010.5539853.

[62] Yinpeng Chen, Zicheng Liu, and Zhengyou Zhang. "Tensor-Based Human Body Modeling". In: *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 105–112. DOI: 10.1109/CVPR.2013.21. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=196145.

[63] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. "Performance Capture from Sparse Multi-view Video". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 27.3 (Aug. 2008), 98:1–98:10. ISSN: 0730-0301. DOI: 10.1145/1360612.1360697. URL: http://doi.acm.org/10.1145/1360612.1360697.

[64] J. Stark and A. Hilton. "Surface capture for performance-based animation". In: *IEEE Computer Graphics and Applications* 27.3 (2007), pp. 21–31.

[65] Alberto Leardini, Lorenzo Chiari, Ugo Della Croce, and Aurelio Cappozzo. "Human movement analysis using stereophotogrammetry: Part 3. Soft tissue artifact assessment and compensation". In: *Gait & Posture* 21.2 (2005), pp. 212–225. ISSN: 0966-6362. DOI: http://dx.doi.org/10.1016/j.gaitpost.2004.05.002. URL: http://www.sciencedirect.com/science/article/pii/S0966636204000773.

[66] Q. Y. Hong, S. I. Park, and J. K. Hodgins. “A DataâĂŘ-driven Segmentation for the Shoulder Complex”. In: *Computer Graphics Forum* 29.2 (2010), pp. 537–544.

[67] 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* 32.2 (May 2013), pp. 285–294.

[68] E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. “Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture”. In: *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2007, pp. 1–8.

[69] Yebin Liu, J. Gall, C. Stoll, Qionghai Dai, Hans-Peter Seidel, and C. Theobalt. “Markerless Motion Capture of Multiple Characters Using Multiview Image Segmentation”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.11 (2013), pp. 2720–2735. ISSN: 0162-8828. DOI: http://doi.ieeecomputersociety.org/10.1109/TPAMI.2013.47.

[70] Aggeliki Tsoli, Naureen Mahmood, and Michael J. Black. “Breathing Life into Shape: Capturing, Modeling and Animating 3D Human Breathing”. In: *ACM Trans. Graph., (Proc. SIGGRAPH)* 33.4 (July 2014), 52:1–52:11. URL: http://dl.acm.org/citation.cfm?doid=2601097.2601225.

[71] E. de Aguiar, R. Zayer, C. Theobalt, H. P Seidel, and M. Magnor. “A Simple Framework for Natural Animation of Digitized Models”. In: *Computer Graphics and Image Processing, 2007. SIBGRAPI 2007. XX Brazilian Symposium on*. 2007, pp. 3–10. DOI: 10.1109/SIBGRAPI.2007.14.

[72] M. Livne, L. Sigal, N.F. Troje, and D.J. Fleet. “Human attributes from 3D pose tracking”. In: *Computer Vision and Image Understanding* 116.5 (2012), pp. 648–660.

[73] H. Wang, N. Xu, R. Raskar, and N. Ahuja. “Videoshop: A new framework for spatio-temporal video editing in gradient domain”. In: *Graph. Models* 69.1 (2007), pp. 57–70.

[74] Hao-Yu Wu, Michael Rubinstein, Eugene Shih, John Guttag, Frédo Durand, and William T. Freeman. “Eulerian Video Magnification for Revealing Subtle Changes in the World”. In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 31.4 (July 2012), 65:1–65:8.

[75] Neal Wadhwa, Michael Rubinstein, Frédo Durand, and William T. Freeman. “Phase-Based Video Motion Processing”. In: *ACM Trans. Graph., (Proc. SIGGRAPH)* 32.4 (July 2013), 80:1–80:10.

[76] Ji-yong Kwon and In-Kwon Lee. “Rubber-like Exaggeration for Character Animation”. In: *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications*. PG ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 18–26. ISBN: 0-7695-3009-5. DOI: 10.1109/PG.2007.58. URL: http://dx.doi.org/10.1109/PG.2007.58.

[77] Thomas Neumann, Kiran Varanasi, Stephan Wenger, Markus Wacker, Marcus Magnor, and Christian Theobalt. "Sparse Localized Deformation Components". In: *ACM Trans. Graph.* 32.6 (Nov. 2013), 179:1–179:10. ISSN: 0730-0301. DOI: 10.1145/2508363.2508417. URL: http://doi.acm.org/10.1145/2508363.2508417.

[78] Arjun Jain, Thorsten Thormählen, Hans-Peter Seidel, and Christian Theobalt. "MovieReshape: Tracking and Reshaping of Humans in Videos". In: *ACM Transactiosn on Graphics (Proc. SIGGRAPH)* 29.6 (Dec. 2010), 148:1–148:10. ISSN: 0730-0301. DOI: 10.1145/1882261.1866174. URL: http://doi.acm.org/10.1145/1882261.1866174.

[79] K. Robinette, S. Blackwell, H. Daanen, M. Boehmer, S. Fleming, T. Brill, D. Hoeferlin, and D. Burnsides. *Civilian American and European Surface Anthropometry Resource (CAESAR) Final Report*. Tech. rep. AFRL-HE-WP-TR-2002-0169. US Air Force Research Laboratory, 2002.

[80] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd. New York: Springer, 2006.

[81] Andreas Griewank and Andrea Walther. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Second. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2008. ISBN: 0898716594, 9780898716597.

[82] Matthew Loper. *Chumpy autodifferentiation library*. http://chumpy.org/. 2014.

[83] Ulf Grenander. *Lectures in Pattern Theory I, II and III: Pattern Analysis, Pattern Synthesis and Regular Structures*. Heidelberg-New York: Springer-Verlag, 1976–1981.

[84] D. Mumford. "Neuronal architectures for pattern-theoretic problems". In: *Large-scale neuronal theories of the brain*. Ed. by C. Koch and J. L. Davis. Bradford, 1994, pp. 125–152.

[85] B. G. Baumgart. *Geometric Modeling for Computer Vision*. Tech. rep. AI Lab Memo AIM-249. Stanford University, Oct. 1974.

[86] D. Terzopoulos. "Physically-based modeling: Past, present, and future". In: *ACM SIGGRAPH 89 Panel Proceedings* (1989), pp. 191–209.

[87] Vikash Mansinghka, Tejas D Kulkarni, Yura N Perov, and Josh Tenenbaum. "Approximate Bayesian Image Interpretation using Generative Probabilistic Graphics Programs". In: *Advances in Neural Information Processing Systems 26*. Ed. by C.J.C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger. 2013, pp. 1520–1528.

[88] B.K.P. Horn. "Understanding image intensities". In: *Artificial Intelligence* 8 (1977), pp. 201–231.

[89] M. Bertero, T. Poggio, and V. Torre. "Ill-posed Problems in Early Vision". In: *Proc. IEEE* 76.8 (Aug. 1988), pp. 869–889.

[90] D. Terzopoulos. "Regularization of Inverse Visual Problems Involving Discontinuities". In: *IEEE PAMI* 8.4 (1986), pp. 413–424.

[91] D. Kersten. “Inverse 3-D graphics: A metaphor for visual perception”. In: *Behavior Research Methods, Instruments & Computers* 29.1 (1997), pp. 37–46.

[92] Whitman Richards, ed. *Natural Computation*. Cambridge, MA: The MIT Press (A Bradford Book), 1988.

[93] David C. Knill and Whitman Richards, eds. *Perception as Bayesian Inference*. Cambridge, UK: The Press Syndicate of the University of Cambridge, 1996.

[94] Gustavo Patow and Xavier Pueyo. “A Survey of Inverse Rendering Problems”. In: *Computer Graphics Forum* 22.4 (2003), pp. 663–687.

[95] N. Goodman, V. Mansinghka, D. Roy, K. Bonawitz, and J. Tenenbaum. “Church: A language for generative models”. In: *Proc. Uncertainty in Artificial Intelligence (UAI)*. Ed. by David A. McAllester and Petri Myllymäki. July 2008, pp. 220–229.

[96] T. Minka, J. Winn, J. Guiver, and D. Knowles. *Infer.NET 2.4. Microsoft Research Cambridge*. `http://research.microsoft.com/infernet.`. 2010.

[97] F. Wood, J. W. van de Meent, and V. Mansinghka. “A New Approach to Probabilistic Programming Inference”. In: *Artificial Intelligence and Statistics*. 2014.

[98] Varun Jampani, Sebastian Nowozin, Matthew Loper, and Peter V. Gehler. “The Informed Sampler: A Discriminative Approach to Bayesian Inference in Generative Computer Vision Models”. In: *CoRR* abs/1402.0859 (2014). URL: `http://arxiv.org/abs/1402.0859`.

[99] Volker Blanz and Thomas Vetter. “A morphable model for the synthesis of 3D faces”. In: *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*. SIGGRAPH ’99. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194. ISBN: 0-201-48560-5.

[100] A. Jalobeanu, F.O. Kuehnel, and J.C. Stutz. “Modeling Images of Natural 3D Surfaces: Overview and Potential Applications”. In: *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW ’04. Conference on*. 2004, pp. 188–188. DOI: `10.1109/CVPR.2004.124`.

[101] V.N. Smelyansky, R.D. Morris, F.O. Kuehnel, D.A. Maluf, and P. Cheeseman. “Dramatic Improvements to Feature Based Stereo”. English. In: *Computer Vision — ECCV 2002*. Ed. by Anders Heyden, Gunnar Sparr, Mads Nielsen, and Peter Johansen. Vol. 2351. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, pp. 247–261. ISBN: 978-3-540-43744-4. DOI: `10.1007/3-540-47967-8_17`.

[102] J. W. Bastian. “Reconstructing 3D geometry from multiple images via inverse rendering”. PhD thesis. School of Computer Science, 2008.

[103] F. Viola, A. Fitzgibbon, and R. Cipolla. “A unifying resolution-independent formulation for early vision”. In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. 2012, pp. 494–501. DOI: `10.1109/CVPR.2012.6247713`.

[104] Cristian Sminchisescu. "Estimation algorithms for ambiguous visual models: Three dimensional human modeling and motion reconstruction in monocular video sequences". PhD thesis. Inst. National Polytechnique de Grenoble, July 2002.

[105] C. Sminchisescu and A. Telea. "A Framework for Generic State Estimation in Computer Vision Applications". In: *International Conference on Computer Vision Systems (ICVS)*. 2001, pp. 21–34.

[106] Martin de La Gorce, Nikos Paragios, and David J. Fleet. "Model-based hand tracking with texture, shading and self-occlusions". In: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008. DOI: http://dx.doi.org/10.1109/CVPR.2008.4587752.

[107] Gary Bradski and Adrian Kaehler. *Learning OpenCV*. O'Reilly Media Inc., 2008. URL: http://oreilly.com/catalog/9780596516130.

[108] J. Heikkila and O. Silven. "A four-step camera calibration procedure with implicit image correction". In: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. 1997, pp. 1106–1112. DOI: 10.1109/CVPR.1997.609468.

[109] Michael J. Jones and Tomaso Poggio. "Model-based matching by linear combinations of prototypes". In: *A.I. Memo 1583, MIT*. 1996, pp. 1357–1365.

[110] Andreas Griewank. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. Frontiers in Appl. Math. 19. Philadelphia, PA: SIAM, 2000. ISBN: 0–89871–451–6.

[111] Travis E. Oliphant. "Python for Scientific Computing". In: *Computing in Science and Engineering* 9.3 (2007), pp. 10–20.

[112] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. "Theano: a CPU and GPU Math Expression Compiler". In: *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Austin, TX, June 2010.

[113] Sebastian F. Walter. *PYADOLC 2.4.1*. 2012. URL: https://github.com/b45ch1/pyadolc.

[114] Abraham D. Lee. *ad: a Python package for first- and second-order automatic differentation*. 2012. URL: http://pythonhosted.org/ad/.

[115] Eric Jones, Travis Oliphant, Pearu Peterson, et al. *SciPy: Open source scientific tools for Python*. 2001–. URL: http://www.scipy.org/.

[116] Brett Allen, Brian Curless, and Zoran Popović. "The Space of Human Body Shapes: Reconstruction and Parameterization from Range Scans". In: *ACM Trans. Graph.* 22.3 (July 2003), pp. 587–594.

[117] Xiaohuan Corina Wang and Cary Phillips. "Multi-weight Enveloping: Least-squares Approximation Techniques for Skin Animation". In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '02. San Antonio, Texas: ACM, 2002, pp. 129–138. ISBN: 1-58113-573-4. DOI: 10.1145/545261.545283. URL: http://doi.acm.org/10.1145/545261.545283.

[118] Ladislav Kavan and Jiří Žára. "Spherical Blend Skinning: A Real-time Deformation of Articulated Models". In: *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*. I3D '05. Washington, District of Columbia: ACM, 2005, pp. 9–16. ISBN: 1-59593-013-2. DOI: 10.1145/1053427.1053429. URL: http://doi.acm.org/10.1145/1053427.1053429.

[119] Bruce Merry, Patrick Marais, and James Gain. "Animation Space: A Truly Linear Framework for Character Animation". In: *ACM Trans. Graph.* 25.4 (Oct. 2006), pp. 1400–1423. ISSN: 0730-0301. DOI: 10.1145/1183287.1183294. URL: http://doi.acm.org/10.1145/1183287.1183294.

[120] Brett Allen, Brian Curless, Zoran Popović, and Aaron Hertzmann. "Learning a Correlated Model of Identity and Pose-dependent Body Shape Variation for Real-time Synthesis". In: *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '06. Vienna, Austria: Eurographics Association, 2006, pp. 147–156. ISBN: 3-905673-34-7. URL: http://dl.acm.org/citation.cfm?id=1218064.1218084.

[121] Oren Freifeld and Michael J. Black. "Lie Bodies: A Manifold Representation of 3D Human Shape". In: *European Conf. on Computer Vision (ECCV)*. Ed. by A. Fitzgibbon et al. (Eds.) Part I, LNCS 7572. Springer-Verlag, Oct. 2012, pp. 1–14.

[122] Nils Hasler, Thorsten Thormählen, Bodo Rosenhahn, and Hans-Peter Seidel. "Learning Skeletons for Shape and Pose". In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. Washington, D.C.: ACM, 2010, pp. 23–30. ISBN: 978-1-60558-939-8. DOI: 10.1145/1730804.1730809. URL: http://doi.acm.org/10.1145/1730804.1730809.

[123] W. Chang and M. Zwicker. "Range Scan Registration Using Reduced Deformable Models". In: *Computer Graphics Forum* 28.2 (2009), pp. 447–456.

[124] Gerard Pons-Moll, Javier Romero, Naureen Mahmood, and Michael J. Black. "Dyna: A Model of Dynamic Human Shape in Motion". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH)* 34.4 (July 2015), 120:1–120:14.

[125] D. Hirshberg, M. Loper, E. Rachlin, and M.J. Black. "Coregistration: Simultaneous alignment and modeling of articulated 3D shape". In: *European Conf. on Computer Vision (ECCV)*. Ed. by A. Fitzgibbon et al. (Eds.) LNCS 7577, Part IV. Springer-Verlag, Oct. 2012, pp. 242–255.

[126] Binh Huy Le and Zhigang Deng. "Smooth Skinning Decomposition with Rigid Bones". In: *ACM Trans. Graph.* 31.6 (Nov. 2012), 199:1–199:10. ISSN: 0730-0301. DOI: 10.1145/2366145.2366218. URL: http://doi.acm.org/10.1145/2366145.2366218.

[127] Robert Y. Wang, Kari Pulli, and Jovan Popović. "Real-time Enveloping with Rotational Regression". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 26.3 (July 2007). ISSN: 0730-0301. DOI: 10.1145/1276377.1276468. URL: http://doi.acm.org/10.1145/1276377.1276468.

[128] Edilson De Aguiar, Christian Theobalt, Sebastian Thrun, and Hans-Peter Seidel. "Automatic Conversion of Mesh Animations into Skeleton-based Animations". In: *Computer Graphics Forum* 27.2 (2008), pp. 389–397.

[129] Ilya Baran and Jovan Popović. "Automatic Rigging and Animation of 3D Characters". In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: ACM, 2007. DOI: 10.1145/1275808.1276467. URL: http://doi.acm.org/10.1145/1275808.1276467.

[130] S. Corazza and E. Gambaretto. *Automatic generation of 3D character animation from 3D meshes*. US Patent 8,797,328. 2014. URL: http://www.google.com/patents/US8797328.

[131] S. Schaefer and C. Yuksel. "Example-based Skeleton Extraction". In: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. SGP '07. Barcelona, Spain: Eurographics Association, 2007, pp. 153–162. ISBN: 978-3-905673-46-3. URL: http://dl.acm.org/citation.cfm?id=1281991.1282013.

[132] Binh Huy Le and Zhigang Deng. "Robust and Accurate Skeletal Rigging from Mesh Sequences". In: *ACM Trans. Graph.* 33.4 (July 2014), 84:1–84:10. ISSN: 0730-0301. DOI: 10.1145/2601097.2601161. URL: http://doi.acm.org/10.1145/2601097.2601161.

[133] Christian Miller, Okan Arikan, and Don Fussell. "Frankenrigs: Building Character Rigs from Multiple Sources". In: *Proceedings of the 2010 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*. I3D '10. Washington, D.C.: ACM, 2010, pp. 31–38. ISBN: 978-1-60558-939-8. DOI: 10.1145/1730804.1730810. URL: http://doi.acm.org/10.1145/1730804.1730810.

[134] C. Rouet and J. Lewis. *Method and apparatus for creating lifelike digital representations of computer animated objects by providing corrective enveloping*. US Patent 5,883,638. 1999. URL: https://www.google.com/patents/US5883638.

[135] Tsuneya Kurihara and Natsuki Miyata. "Modeling Deformable Human Hands from Medical Images". In: *Proceedings of the 2004 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '04. Grenoble, France: Eurographics Association, 2004, pp. 355–363. ISBN: 3-905673-14-2. DOI: 10.1145/1028523.1028571. URL: http://dx.doi.org/10.1145/1028523.1028571.

[136] Taehyun Rhee, J.P. Lewis, and Ulrich Neumann. "Real-Time Weighted Pose-Space Deformation on the GPU". In: *EUROGRAPHICS* 25.3 (2006). Ed. by E. Gröller and L. Szirmay-Kalos.

[137] Brett Allen, Brian Curless, and Zoran Popović. "Articulated Body Deformation from Range Scan Data". In: *ACM Trans. Graph. (Proc. SIGGRAPH)* 21.3 (July 2002), pp. 612–619. ISSN: 0730-0301. DOI: 10.1145/566654.566626. URL: http://doi.acm.org/10.1145/566654.566626.

[138] Paul G. Kry, Doug L. James, and Dinesh K. Pai. "EigenSkin: Real Time Large Deformation Character Skinning in Hardware". In: *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '02. San Antonio, Texas: ACM, 2002, pp. 153–159. ISBN: 1-58113-573-4. DOI: 10.1145/545261.545286. URL: http://doi.acm.org/10.1145/545261.545286.

[139] Ladislav Kavan, Steven Collins, and Carol O'Sullivan. "Automatic Linearization of Nonlinear Skinning". In: *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games*. I3D '09. Boston, Massachusetts: ACM, 2009, pp. 49–56. ISBN: 978-1-60558-429-4. DOI: 10.1145/1507149.1507157. URL: http://doi.acm.org/10.1145/1507149.1507157.

[140] Doug L. James and Christopher D. Twigg. "Skinning Mesh Animations". In: *ACM Trans. Graph.* 24.3 (July 2005), pp. 399–407. ISSN: 0730-0301. DOI: 10.1145/1073204.1073206. URL: http://doi.acm.org/10.1145/1073204.1073206.

[141] Hyewon Seo, Frederic Cordier, and Nadia Magnenat-Thalmann. "Synthesizing Animatable Body Models with Parameterized Shape Modifications". In: *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. SCA '03. San Diego, California: Eurographics Association, 2003, pp. 120–125. ISBN: 1-58113-659-5. URL: http://dl.acm.org/citation.cfm?id=846276.846292.

[142] 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* 28.2 (2009), pp. 337–346.

[143] Ofir Weber, Olga Sorkine, Yaron Lipman, and Craig Gotsman. "Context-Aware Skeletal Shape Deformation". In: *Computer Graphics Forum* 26.3 (Sept. 2007), pp. 265–274.

[144] Federica Bogo, Javier Romero, Matthew Loper, and Michael J. Black. "FAUST: Dataset and Evaluation for 3D Mesh Registration". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE, 2014, pp. 3794–3801. DOI: 10.1109/CVPR.2014.491. URL: http://dx.doi.org/10.1109/CVPR.2014.491.

[145] Charles L. Lawson and Richard J. Hanson. *Solving least squares problems*. Classics in applied mathematics. SIAM : Society of industrial and applied mathematics. Philadelphia, PA: SIAM, 1995. ISBN: 0-89871-356-0. URL: http://opac.inria.fr/record=b1080804.

[146] *Dyna dataset*. http://dyna.is.tue.mpg.de/. Accessed: 2015-05-15. 2015.

[147] *CMU Graphics Lab Motion Capture Database*. http://mocap.cs.cmu.edu. Accessed: 2012-12-11. 2000.