University of Alberta

Image-based Capture and Modeling of Dynamic Human Motion and Appearance

by

Neil Aylon Charles Birkbeck

A thesis submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science

©Neil Aylon Charles Birkbeck
Fall 2011
Edmonton, Alberta

Abstract

Photo-realistic renderings of humans are required for real-time graphics applications, and accurate human models are useful in applications such as model-based tracking. Non-rigid deformations of humans, e.g., deforming cloth and muscle bulging, are hard to model geometrically and are inefficient to simulate. Such deformations are often dependent on the subjects kinematic pose. In this thesis, an image-based pipeline is used to acquire a compact model of these non-rigid deformations. The model is capable of generating photo-realistic renderings of deformation and appearance effects from novel animation and viewpoint.

The compact model of non-rigid deformations is distilled from a multi-view training sequence exhibiting the desired pose-dependent deformations. The geometric deformations are modeled with a pose-dependent geometry attached to a kinematic skeleton, and the appearance is modeled with a pose- and view-dependent appearance basis. Pose-dependent effects not encoded in the geometry are represented in the appearance, and view-dependent appearance compensates for inaccurate geometries and specular effects.

In acquisition, a base geometry is recovered from a static multi-view image sequence. For human geometries, a two camera turntable-based acquisition is proposed. The acquisition interleaves tracking and silhouette refinement to account for unintentional motion of the subject.

The coarse motion of the base geometry is tracked in a separate training sequence using a common tracking formulation for linear blend skinned meshes. The energy formulation combines silhouette or intensity-based data terms with pose prior and smoothness terms. Local optimization with GPU acceleration gives near-real time results on intensity-based terms.

The recovery of fine-scale geometric deformations necessary to build the model is studied in situations with a few or non-overlapping views. For a moving monocular camera, a simple constant velocity constraint is shown to enable the reconstruction of both dense scene flow and structure in a variational formulation. This simple constraint is generalized to a multi-view setting, where long range flow is represented with a temporal motion basis layered on top of a geometric proxy surface.

The complete compact model is demonstrated on several examples, including modeling of cloth deformation on arms, transferring the appearance of wrinkles on pants to novel walk cycles, and applications of free-viewpoint compression.

Acknowledgements

This thesis would not have been possible without the persistence, patience, and support of my supervisor Martin Jägersand and co-advisor Dana Cobzaş. Despite my own stubbornness, their direction and guidance has led me down several positive career and life paths that I would not have taken on my own. However, these opportunities would not have existed if Keith Yerex hadn't recommended me for the initial summer research assistant-ship, which put me in the vision lab almost 8 years ago.

In addition, I would like to thank all of the labmates with whom I have had the pleasure of discussing and sharing problems, including Parisa Mosayebi, Azad Shademan, Alejandro Hernandez-Herdocia, Dave Lovi, Karteek Poporui, Howard Chung, Cameron Upright, and Adam Rachmielowski. Thanks for putting up with blue curtains in the lab, and for helping me acquire data when necessary. Experiments with the WAM robot would not have been possible without the support of Alejandro and Azad: thanks for allowing my last minute requests to take over your workstations, even when I'm sure you had your own writing/work to deal with.

Skate-breaks with Adam R. and Keith Y., and games of Quake 3 were necessary stress relievers (or time killers), and will remain as some of my best memories of grad school. Thanks to Adam Harrison and James Neufeld for all of the shared coffees, and to Steven Eliuk for the lunchtime discussions over nachos, and to everyone that ever came out for the Friday nights at the Garneau.

I express my sincere gratitude to the staff at the department of computing science, who have recommended me for several awards and who have put up with me almost always leaving deadlines to the last (or past the last) minute—this dissertation notwithstanding. This work would not have been possible without these awards, including the financial scholarships from NSERC, the QE-II, and the dissertation scholarship.

I would like to thank my thesis committee (John Bowman, Herb Yang, Csaba Szepesvári, Edmond Boyer, and Adrian Hilton) for reading and commenting on my candidacy document and this thesis. Your comments were uplifting, your questions were insightful, and your corrections/suggestions have helped improve the quality of this manuscript.

Thanks to Michal Sofka and S.Kevin Zhou for giving me the opportunity to intern at Siemens Corporate Research. It was a great experience and provided me with the necessary motivation to finish this dissertation. Debbi Nichols, thanks for letting me share your house with you during this time.

To my friends outside of university: Jason Dublanko, you are still my favorite person to run into on campus—somehow we've always had a way of finding one another; Dave Bloom, the U.S. trip will always remind me of the midway point of this dissertation as it happened right after my candidacy. Walter Manning, I cannot think of any other single person who has had a stronger influence on my outlook, tastes, and personality, than you. Susanne Lerohl and Jason Baumle, among other things, thank you for the introduction to reiki course. Those Sundays were a welcome break of relaxation. And to Nathan Matthews, thanks for many of the books which line my shelves, the bike I rode for several years, and for connecting everyone almost religiously on those weekend skate sessions. But most of all, thanks for being the most genuine, fun loving, best story-telling person I have ever met. Your memory will live on in me forever. Rest in peace, friend.

I would also like to thank my parents (Al and Donnagay) and family (Ryan, Amy, Madison & Brooklyn) for their support and encouragement (not to mention all of the Sunday night family dinners with left-overs that often saved us from cooking well into the week). I would also like to thank Ryan & Amy for the accordion; it has turned out to be one of my favorite hobbies despite the initial dust that it collected. Also, to Leslie's family (Colin, Laurie, Nick & Crystal), thank you for all of the shared holidays, visits, and lottery tosses.

And finally, to my wife Leslie, who has had to put up with countless physical and emotional absences. Thank you for the late night discussions, for looking after zig, for making supper when it was my turn, for advising me when to take breaks, and for supporting the decisions I have made. But mostly, thanks for putting up with me when things weren't working out.

Contents

List of Figures

List of Tables

List of Symbols

$\mathbf{E}_i, \mathbf{E}$	4×4 Euclidean transformation matrix	14
$\mathbf{t}_i$	Translation vector	14
$\mathbf{R}_i, \mathbf{R}$	Rotation matrix	14
Π	Projective division operator	14
$\mathbf{P}, \mathbf{P}_i$	Camera projection matrix, a 3×4 matrix	14
$\mathbf{K}, \mathbf{K}_i$	Camera internal parameters, a 3×3 matrix	14
f_x, f_y	Camera focal lengths	14
p_x, p_y	Camera principle point	14
ζ	Camera skew	14
$w_{i,j}$	Skinning weight for vertex i in bone j	24
$\mathbf{v}_i$	Transformed vertex i	24
$\hat{\mathbf{v}}_i$	Rest position for vertex i	24
$\mathbf{T}_j$	Transformation matrix for bone/link j	24
f_{uv}	The uv-parameterization.	48
G	The base geometry	48
$\#_{\text{points}}$	The number of vertices in the base geometry	48
$\triangle_k$	A triangle in the geometry	48
$P(i)$	Parent of bone i	48
θ_i	Angles for a bone i	49
$\boldsymbol{\Theta}$	Concatentation of all joint angles	49
$\mathbf{M}_i(\boldsymbol{\Theta})$	Animated bone matrix.	49
$\mathbf{d}_j$	Deformation offset of vertex j	50
H_i	A set of vertices making up a part of the base mesh	50
$\alpha^G_{i,j}$	Membership weight for vertex j in part i	50
$\phi_i(x)$	Mapping from index in concatenated geometry vector to vertex index	50
$\mathbf{W}^G_i$	Geometry deformation linear basis for mesh part i	51
$\mathbf{h}^G_{i,l}$	Latent variables of geometry basis for mesh part i, frame l	51
$\bar{\mathbf{e}}^G_i$	Mean displacement of geometry for mesh part i	51
$\alpha^A_{i,j}$	Membership weight for texel j in part i	51
$\mathbf{W}^A_i$	Color attribute linear basis (e.g., from PCA)	52
$\mathbf{h}^A_{i,l}$	Latent variables for part i, frame l	52
$\bar{\mathbf{e}}^A_i$	Mean texture for part i	52
$\mathbf{l}_i$	View direction for a part	52
$I_{i,t}$	Image from camera i at time t.	52
C	Number of cameras in a capture setup	52
T	Number of time frames in a sequence	52
$\mathcal{W}_{i,t}$	Texture warp for image i at time t	53
$\mathbf{e}_{i,j,t}$	Concatenated vector of all attributes for part i, view j, a time t	53
$\mathbf{p}_i$	The joint angle parameters that influence the geometry of part i.	54
$\boldsymbol{\lambda}_{i,t}$	Radial basis function weights	55
$\mathbf{p}^A_i$	The joint angle parameters that influence appearance of part i.	56
α_{pose}	The contribution of pose to a combined pose and view RBF interpolator	56
$\boldsymbol{\Theta}_{1:T}$	The concatenated vector of all joint angles over all frames	65

E	A cost function	67
α_{kin}	Kinematic term in the tracking used for static reconstruction.	67
α_{smooth}	Smoothness term in the tracking used for static reconstruction.	67
$P^{\text{sil}}_{i,t}(G)$	The projected silhouette of the geometry, G	67
$\mathcal{T}$	A texture image	69
tw	The texture width	69
th	The texture height	69
$\alpha_i(x, y)$	The alpha value of a matted texture image	71
$\partial S_{i,t}$	The boundary contour of a binary silhouette, S_i	93
$R(\mathbf{v}_j)$	A reference color for vertex $\mathbf{v}_j$	94
$\wp_j(\mathbf{x}, \mathbf{d})$	The back-backprojection function for a point $\mathbf{x}$ with disparity d.	128
$\underset{j\rightarrow i}{W}$	Warp for backprojection of a point from image j and projected into image i	128
Ψ_*	A robust function used in flow estimation	129
$\mathcal{B}_k(t)$	A temporal basis function	139
$\Pi_i(\mathbf{x})$	A shorthand for projection of a point into image i (e.g., $\Pi(\mathbf{P}_i\mathbf{x})$)	141
$\lambda_k(u, v)$	The modulation coefficients used in the basis constrained flow estimation	143

Nomenclature

BRDF	Bidirectional reflectance distribution function
VDTM	View-dependent texture mapping
AABB	Axis-aligned bounding box
AAM	Active appearance model
CSP	Colored surface point
FEM	Finite element method
IBR	Image-based rendering
LBS	Linear blend skinning
MI	Marching-intersections
MM	Morphable model
PCA	Principle component analysis
RBF	Radial basis functions
SFS	Shape-from-silhouette
TPS	Thin-plate spline
WAM	Whole arm manipulator
XOR	Exclusive or
MoCap	Human motion capture

1 Introduction

Accurate models of human appearance, geometry, and deformation are important for both computer graphics and computer vision applications. In graphics, realistic renderings and animations of humans is of the utmost importance due to both the ubiquity of humans in graphics applications and the sensitivity of our visual system to small inconsistencies in simulated humans. In computer vision, human models can be used for applications like model-based motion capture or interactive games.

Static human geometry can be modeled manually or through laser scanners, and these meshes can be coupled and animated through the use of kinematic skeletons. However, non-rigid dynamic surface deformations, such as cloth or muscle motion, are tedious to model manually and are often simulated with computationally intensive physical simulations. Such dynamic simulation can be too slow for real-time applications. Instead, a data-based approach, where cloth motion is captured from visual observations, can deliver realistic deformations at a lower computational cost. Existing data-based approaches for capturing such non-rigid motion either ignore the appearance of the clothing or only allow the captured deformations to be played back under the same animation.

In this thesis, the focus is on developing image-based techniques to capture non-rigid deformations and to model the deformations and appearance as pose-dependent (see Figures 1.1 and 1.2). Image-based techniques have a long history of being used to acquire a representation of the real world when the geometry and motion is too complex (or tedious) to model manually or when the dynamical simulations are too slow to simulate in real-time. In this thesis, the end result is realistic renderings of deformations associated with human movement, such as muscle bulging, skin wrinkling, and cloth deformation, that are derived directly from a set of input images. The deformation model extends image-based rendering techniques. However, in addition to allowing photo-realistic views from novel viewpoints, the acquired model is also capable of generating photo-realistic renderings of a subject undergoing novel animation. Rendering under novel animation allows deformations acquired from the input images to be transferred to new animations, giving realistic deformations on unseen animations, which is an ideal situation for graphics applications.

1.1 Motivation

An image-based model of pose- and view-dependency is capable of representing pose-dependent effects including cloth motion and muscle bulging on articulated humans or non-rigid pose-dependent deformation of arbitrary objects, such as a cushion being pushed. A key use of such a model is graphics applications, real-time (e.g., games, virtual reality) or otherwise (e.g., movies). However,

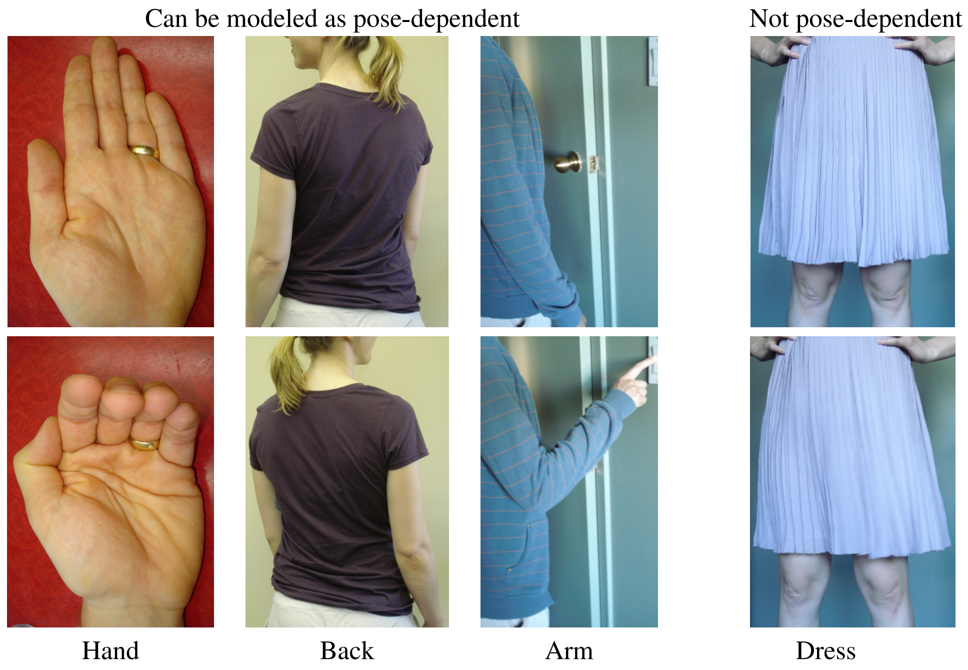


Figure 1.1: Skin wrinkles and wrinkles of tighter fitting garments can often be modeled as pose-dependent. Here, the *pose* refers to bending of the fingers, twisting of the back, and kinematic pose of the arm. Loose fitting clothing, such as dresses, that exhibit dynamic effects (e.g, waving) cannot be modeled as pose-dependent.

outside the application of graphics, such a photo-realistic model of pose-dependent geometry also has computer vision applications in, for example, vision-based tracking. In tracking, a more accurate generative model better agrees with input images, allowing for better tracking.

As dynamic humans play an important role in graphical applications, a large amount of work has gone into creating realistic, efficient models of humans. Staples in computer graphics, such as linear blend skinning, provide a quick way to animate a static model through the use of an underlying skeleton; regions influenced by skeletal components are determined by a set of per-vertex weights, which can be determined automatically (e.g., [25]). Such methods provide a geometric model that can be efficiently animated from the motion of a skeleton, but, alone, they are incapable of modeling the non-linear geometric deformations exhibited on real humans. Other graphics researchers have proposed methods for providing more realistic muscle deformations via the use of several geometric models in different poses [148, 221, 8]. Yet these methods require manual modelers or laser scanners.

Instead of manually modeling muscle deformation, dynamic effects, such as jiggling of muscles, can be simulated as an elastic material attached to the bone [216, 48, 117, 105]. Layers of cloth on a human subject are also often modeled through a physical simulation [23, 24, 42, 58, 169, 170]. Traditionally, a full simulation of the high resolution meshes necessary to achieve cloth folding and buckling was too slow for real-time applications, but recent advances in physical solvers are

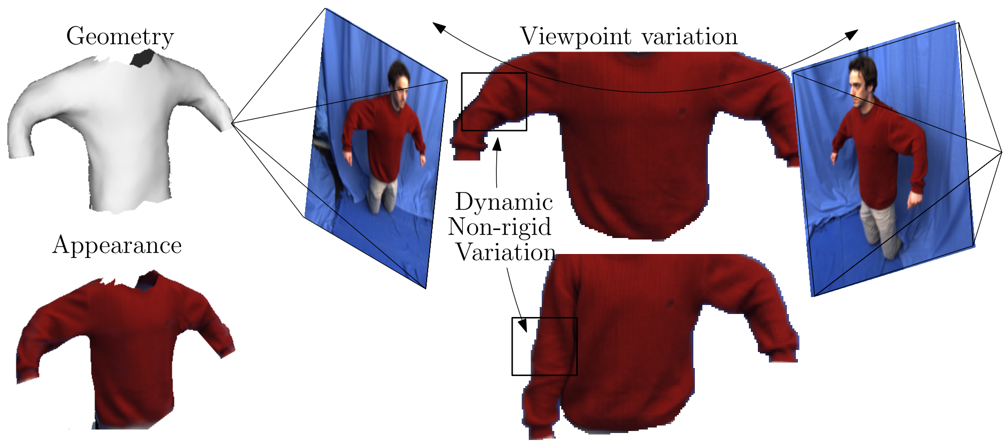


Figure 1.2: A graphics model is typically made up of 3D geometry that defines the surface, and the appearance component that describes the color or reflectance on the 3D geometry (left). A *static* geometry defines the surface of the subject in a fixed pose. When the model is reconstructed from images, an inexact geometry will require a view-dependent appearance to generate realistic renderings between input views. Motion of the human model results in *dynamic* deformation in the geometry or appearance (e.g., wrinkles). In this thesis, this dynamic variation is modeled as a function of the kinematic pose (i.e., joint angles) of the human. Such deformation is said to be pose-dependent.

resolving this limitation [169]. Even so, cloth simulation can still be time consuming and typically requires an artist to tweak the cloth parameters to achieve realistic clothing. The latter limitation can be overcome by learning real cloth parameters from examples [265, 190].

However, in many cases a computationally intensive cloth simulation is unnecessary. For example, several researchers have noticed the kinematic nature of tighter fitting clothing [130, 266, 273]. In other words, wrinkles and folds on clothing are often a function of the kinematic pose. In this case, a data-based approach can be taken, where cloth folds are simulated or captured for various poses and at run-time are predicted as a function of pose [273, 266]. Such approaches model the folds explicitly in a dense geometry and do not model appearance variation.

The above methods all model dynamic motion through manual models, laser scanned data, or physical simulations. These models focus on geometry alone, which is enough in applications like virtual clothing [212, 152], but often a representation of color or appearance is needed. The traditional image-based rendering methods, such as view-dependent texture mapping [73], are best suited for modeling view variation of static scenes. These image-based models could be combined with standard non-linear geometric deformations to produce a pose-dependent model that can be reanimated. However, in this way, the appearance or texture representation could only encode view variation, and could not account for effects like cloth shifting, meaning that geometric deformations would need to be accurate—a condition that is hard to guarantee from image-based acquisition. In order to achieve high levels of realism without imposing too much emphasis on the geometric component, the appearance component should model both variation due to viewpoint as well as some

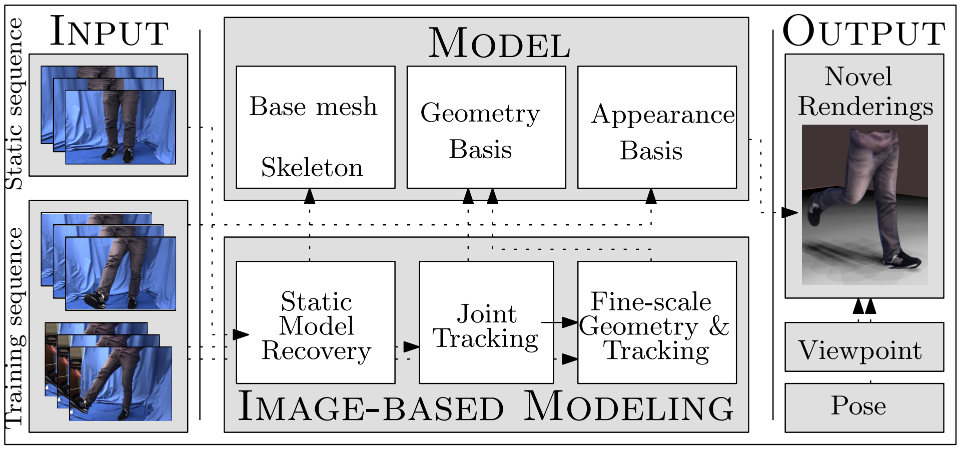


Figure 1.3: An overview of the model components and the acquisition pipeline. For acquisition, two input sequences are used: a *static sequence* is used to obtain the base geometry, and a *training sequence* is used to model the dynamic effects. The purpose of the model is to compactly represent pose- and view-dependent variations for the purpose of rendering photo-realistic images in novel poses from novel viewpoints.

residual variation due to cloth motion or muscle bulging.

Some of the key benefits of using vision to acquire realistic models of humans are that the sensors are relatively low cost, the appearance is acquired simultaneously with geometry, and photo-realistic detail is attainable from the direct use of images in renderings. Furthermore, since vision-based acquisition is markerless and data-based, any deformations due to pose can be handled in the same framework. This is in contrast to physical simulations, which often require a different physical model and simulation framework for either muscle or clothing.

Many multi-camera vision-based approaches have already attempted similar tasks of human modeling from images. Methods like free-viewpoint video are capable of recovering a model that can be rendered from a novel viewpoint for any time in the input sequence. Some methods even capture an accurate time-varying unlit surface model allowing novel illumination [243, 50, 260]. Others even recover the joint angles along with surface deformation, allowing a mapping from pose to deformed shape [203]. These acquisition methods demonstrate that it is possible to model fine-scale surface deformations and joint angles from multi-view video. However, in most methods the surface deformation is simply captured for the purpose of being played back, and, unlike the methods proposed in this thesis, cannot be transferred to novel animations.

1.2 Design Goals and Assumptions

The main goal of this work is to *model complex appearance of non-rigid motion on human subjects using image-based techniques*. A primary requirement for such a model of appearance/geometry is the ability to *achieve realistic renderings* of a subject for any given kinematic pose and view. Ideally, to be useful in all graphical applications, it should be possible to also re-illuminate the model under arbitrary lighting. Other secondary requirements on the model are that it *should be compact* (in terms of space and memory requirements), and *rendering should be efficient* in order to be used in real-time applications.

One of the primary benefits of image-based modeling methods is their accessibility—anyone with a camera can try them out. Unfortunately, for the capture of pose-dependent geometric deformation and view-dependent appearance variation, a single camera is not always sufficient. At a minimum, multiple camera views will be required to acquire complete coverage of the non-rigid surface motion from different viewpoints. Of course, these camera views could be accompanied by consumer level structured light depth sensors (e.g., Microsoft's Kinect) to help extract the surface geometry. Again, multiple depth sensors may be required to obtain complete coverage, but multiple depth sensors may interfere with one another if their field of views overlap. Alternatively, time-multiplexed illumination can also be used to provide dense depth from each viewpoint via photometric stereo [260]. Such elaborately engineered acquisition setups with many cameras and depth sensors will make obtaining input easier, but the acquisition setup becomes more expensive, more elite, and less accessible.

Instead, as secondary goals, *the acquisition pipeline in this thesis has been designed to keep the acquisition strictly image-based* and, for accessibility, *to require as few cameras as necessary*. In order to achieve this secondary goal, intra-viewpoint observations (i.e., observations and correspondences from the same viewpoint) are utilized when possible to obtain surface structure.

In order to acquire the model, it is assumed that there are two multi-view input sequences available (Fig. 1.3). A *static sequence* is used to acquire a vision-based model of the human subject in a fixed pose. The reconstructed model consists of a mesh geometry with a simple appearance model. The multi-view *training sequence* observes the training motions that exhibits the poses, surface deformations, and appearance variation to be encoded by the compact model. This training sequence needs to be structured such that any desired appearance changes due to pose changes are observed in the input. The model will be limited to deformation and appearance effects that are actually pose dependent. Other dynamic effects, such as motion due to wind, are outside the scope of this work.

Finally, it is also assumed that a kinematic skeleton is available for the subject or object being captured. The skeleton must be aligned to the input model by manually specifying joint locations in a 3D user interface. The requirement of a skeleton may seem like a limitation of the proposed model, but even non-rigid surface motion, like loose clothing, can be approximately modeled by a kinematic skeleton [119]. Furthermore, the kinematic structure allows a natural definition of pose, i.e., via the skeletal joint angles, that can be used to drive the pose-dependent geometry and appearance animation. This is in contrast to a mesh only representation (i.e., without a skeleton), where the *pose* would have to be inferred from the mesh configuration. Other advantages of using a skeleton are the ease at which new animations can be transferred and edited on the model, and the strong prior it provides during acquisition.

The proposed model achieves the above design goals by building on existing graphics and vision techniques. Specifically, the geometric variation due to cloth movement or muscle bulging is modeled with a linear basis layered on top of a geometry attached to a kinematic skeleton. This geometric variation is modeled as a function of the recovered joint angles and other abstract interpolators. Appearance variation due to both view direction and cloth shifting not captured by the geometry is modeled in a similar way as image-based rendering methods model appearance.

1.3 Thesis Contributions

In order to build an instance of the model, the acquisition pipeline distills the input images through the following steps. First, an approximate geometry is acquired from the static multi-view sequence. This geometry is tracked to recover the pose of the subject over the training sequence using cues from the multi-view training sequence. Fine-scale geometric deformations are then recovered from the training sequence using methods that focus on using only a few views. The model can then be built from the tracked joint angles and associated geometric deformations and appearance observations. The contributions of the proposed model and acquisition pipeline are as follows:

Development of a compact pose- and view-dependent model

- *A novel compact model of pose-dependent geometry and pose- and view-dependent appearance is proposed.* The model consists of an approximate surface geometry coupled to a kinematic skeleton. Surface deformations and appearance changes in the training motion sequence are encoded using a linear appearance basis and a linear geometry basis. The low dimensional subspaces model variation over a small set of geometric parts that cover the surface of the object. Each part shares a common set of influencing variables (e.g., a subset of the complete kinematic pose). As appearance and geometry are assumed to be dependent on these influencing parameters, rendering a new viewpoint and pose is then treated as a scattered data interpolation problem. In other words, the correct geometric deformation and appearance for the novel pose and viewpoint are generated from the compact linear basis by interpolation over the observations in the training sequence. The encoding of pose-dependent deformations can be transferred to new motions and rendered from novel viewpoints.

- *Pose- and view-based appearance interpolation.* Synthesizing new images from the compact model requires sparse data interpolation to obtain both the displaced geometry and the surface texture. Appearance is dependent on both viewpoint and pose. Although it is natural to interpolate over a combined pose and viewpoint vector (possibly with different weightings of their contribution), in this thesis the task is broken down and preference is given to the pose coordinates over the view. First, all view coefficients for each viewpoint are interpolated based on pose, and then the resulting view coefficients are interpolated by viewpoint. Such a scheme allows the designer to choose appropriate methods for interpolating over pose and view spaces independently. In practice, a bilinear approach often works well.

- *The proposed model is validated experimentally.* Through a series of experiments, the proposed model is validated qualitatively and quantitatively. Novel animations from novel viewpoint are demonstrated for a human face, an upper body, and a lower body. The experiments are used to identify several guidelines for setting up the camera configurations, designing the training sequence, and choosing model parameters.

Static human geometric capture from few views

- *A two view turntable-based acquisition for human geometries that uses tracking to compensate for unintended human motion is developed.* The foundation of the compact model is a static geometry. As the focus of this thesis is on modeling dynamic humans, a low-cost two camera acquisition method is proposed to obtain a static human geometry. Traditional turntable-based acquisition is inappropriate for humans as the subject typically unintentionally moves while standing on a turntable. Such unaccounted motion causes image-based reconstruction to fail. To compensate for this unintended motion, an iterative tracking and refinement algorithm is proposed: unintended motion is tracked with an approximate geometry, and the geometry is then refined using the recovered motion. The tracking procedure combines silhouette cues, joint smoothness, and kinematic constraints. The geometric refinement is based solely on silhouettes.

- *Experimental evaluation of texturing methods on human geometries reconstructed from turntable sequences.* The texture methods include a basic weighted average, a multi-band blend, and a super-resolution approach. Experimentally, the multi-band blend is demonstrated to give the best visual quality on the reconstructed models.

Visual tracking of skinned meshes

- *A common energy framework for tracking skinned meshes.* In order to build the compact model from images, the kinematic joint angles of the base object need to be recovered from a training sequence of images. Through a consistent energy formulation, two silhouette data objectives, an exclusive-or term and a closest point scheme, and an intensity data objective are combined with pose priors and smoothness terms to track a skinned geometry. To avoid slow global optimizers, the focus is instead on efficient local optimization. For this reason, minimization of each of the data energies is discussed. Optimizations include the use of analytic derivatives and GPU acceleration when possible. GPU acceleration allows for near-real time tracking with the intensity data objective.

- *Experimental evaluation of several combinations of data objectives for human tracking.* Experimentally, it is demonstrated that the local optimization methods work well in most cases.

Of the two silhouette objectives the exclusive-or is more robust to silhouette errors but the closest-point scheme is faster. The pose-priors also overcome noisy inputs but are not always necessary.

Dense non-rigid temporal surface modeling with few views

- *A method for recovering 3D scene flow and scene structure from a single moving camera undergoing known motion is proposed.* The compact model requires dense temporal surface displacements and surface correspondences to be recovered from the training sequence. These dense temporal surface displacements and correspondences are hard to obtain with conventional methods when there is little overlap between views. In a restricted version of the problem, where there is a single camera moving with known motion, we propose the use of a constant velocity constraint to obtain both dense surface displacements and dense 3D surface correspondences (i.e., flow). This constant velocity constraint is embedded into an image-based variational depth and displacement estimation process which is solved using geometric multi-grid methods.

- *Basis-constrained surface flow on a moving proxy surface*. The above approach represents the surface and flow relative to a reference image. Such a representation is awkward when dealing with multiple, possibly non-overlapping image views. This limitation is overcome by generalizing the above formulation to a scene-based representation, where 3D scene flow is represented relative to a moving proxy surface. Coarse tracked motion of the proxy surface is taken into consideration, meaning that in ideal conditions the dense surface structure and flow can be obtained from a single view. Furthermore, instead of using a simple constant velocity motion model, temporal motion is represented with a more general low-dimensional linear basis. The use of a low-dimensional basis imposes temporal constraints on the flow, making it less sensitive to noise, and reduces the number of variables in the estimation process.

1.4 Organization and Reading Guideline

The foundations for the proposed research, which have been laid out by computer graphics and vision communities working towards achieving similar goals, are presented in Chapter 2. The body of this thesis is then presented in three parts. The first part contains a description of the compact model (Chapter 3).

The second part contains the contributions specific to the phases of the acquisition pipeline (Chapters 4, 5, & 6). In these three chapters, the phases of the acquisition pipeline are decomposed into independent problems with applicability that extend beyond the scope of the proposed compact model. The multi-view vision-based acquisition pipeline starts with capturing a static model. Chapter 4 details a two camera turntable-based solution for acquiring the static geometry of a full scale human. The next stage of the acquisition pipeline is to track the model through the input training

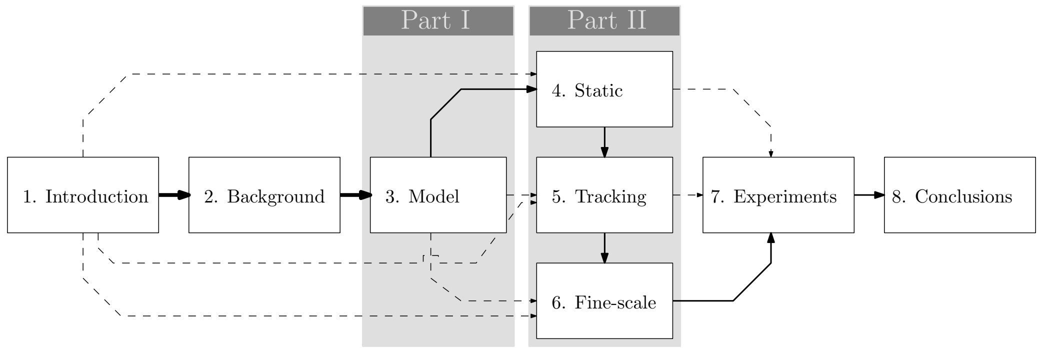


Figure 1.4: The thesis should be read linearly, but the body chapters are independent and can be read separately.

sequence. This tracking problem is formulated on a skinned geometry using multiple image cues in Chapter 5. The final component necessary to build the model is obtaining the fine-scale surface deformations and temporal correspondences. In Chapter 6, the fine-scale temporal correspondence problem is posed when only a few input views are available.

The final part of the thesis body contains experiments that validate the proposed model (Chapter 7). These experiments utilize the acquisition pipeline to obtain image-based models that are animated through novel view and pose conditions.

Although the thesis is meant to be read linearly, the interested reader can skip unrelated chapters (Fig. 1.4). The description of notations used in this thesis should be read (beginning of Chapter 2), but the related work portion of Chapter 2 can be bypassed for those familiar with the vision and graphics literature. The body chapters in the second part can be read independently, and do not depend much on the details specific to the compact model (Chapter 3). The final experimental results (Chapter 7) can be understood without reading Part II, although some of the experimental methodology will rely on the output of methods in Part II.

1.5 Publications

1.5.1 Refereed Conferences

Neil Birkbeck, Dana Cobzas, and Martin Jagersand. Object centered stereo: Displacement map estimation using texture and shading. In *Third International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'06)*, pages 790-797, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

Martin Jagersand, Neil Birkbeck, and Dana Cobzas. A Three-tier Hierarchical Model for Capturing and Rendering of 3D Geometry and Appearance from 2D Images. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'08)*, 2008.

Neil Birkbeck, Dana Cobzas, and Martin Jagersand. Tracking human joint motion for turntable-based static model reconstruction. In *IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM '09)*, pages 1825-1832, 2009.

Neil Birkbeck, Dana Cobzas, and Martin Jagersand. Monocular depth and scene flow under constant velocity. In *International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'10)*, 2010.

Neil Birkbeck, Dana Cobzas, and Martin Jagersand. Basis constrained 3D scene flow on a dynamic proxy. To appear in *International Conference on Computer Vision (ICCV'11)*, 2011.

1.5.2 Book Chapters

Martin Jagersand, Neil Birkbeck, and Dana Cobzas. View-Dependent Texturing using Linear Basis. In *Image and Geometry Processing for 3D Cinematography*, Ronfard, R. and Taubin, T. Eds., Springer-Verlag, 2010.

2 Background & Related Work

Topics in vision and graphics relevant to this thesis range from vision-based shape/appearance capture to the rendering and modeling of dynamic articulated models. In this chapter, these fields and their relevance to this thesis are reviewed (see Figure 2.1 for an overview). As images are the main source of input for the acquisition of the proposed compact model, the notation and imaging model are introduced first (Section 2.1), followed by image-cues (e.g., silhouettes, texture) and their usage in general multi-view static model reconstruction (Section 2.2). The multi-view methods that are specific to the context of reconstructing human geometry are also discussed.

In the proposed model, the time varying deformations also need to be reconstructed from a dynamic sequence of the human. These deformations are measured relative to the underling gross motion of an articulated figure or skeleton model, which also needs to be recovered from the input sequence. Again, vision-based cues are used to acquire this gross motion, and a review of the existing methods for markerless vision-based capture is provided (Section 2.4.2). As many of these vision-based tracking methods are model-based, the discussion of these methods is postponed until after the review of human graphics models (Section 2.3). In the review on graphics models, both static aspects (e.g., fixed geometry) and the dynamic models used to model deforming skin are discussed. The example-based dynamic models take several input geometries at different poses as input and map interpolation parameters (e.g., joint angles) to smoothly interpolate between the different geometries. These methods provide the backbone for the underlying deformation model used in the proposed project.

In order to render novel images of an acquired model, a model of appearance is built to complement the geometry. As a final topic, image-based and parametric methods for modeling appearance are reviewed (Section 2.5). In the proposed model, the existing image-based appearance modeling methods are adapted and extended to our application of an articulated human, where we consider the parameterization of appearance as a function of both kinematic configurations and view direction.

2.1 Terminology, Notation & Preliminaries

This section discusses some basic terminology (§2.1.1), notations (§2.1.2), and the camera model (§2.1.3) used in this thesis.

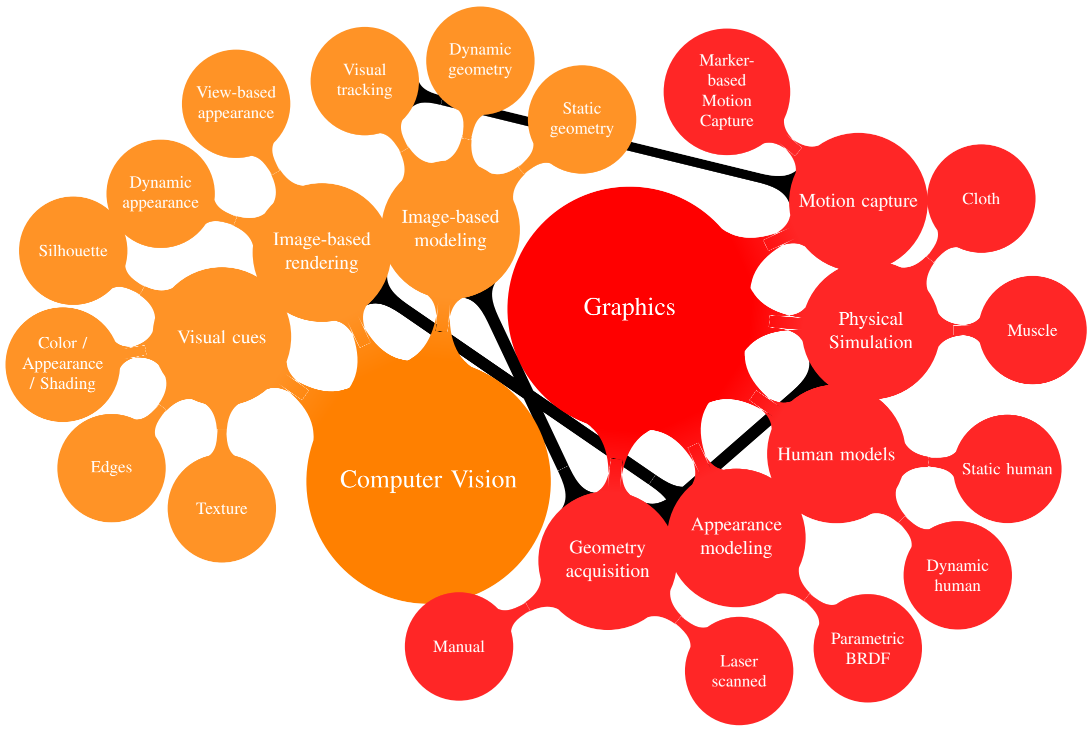


Figure 2.1: Vision methods, using images and derived cues, have applications to all the components of the proposed project. Each of these components also overlaps with the field of computer graphics, often providing similar results through different means. The relevant topics discussed in this section are highlighted, with proximity to their appropriate field.

2.1.1 Terminology

Throughout this thesis several terms are used in the specific context of modeling humans from images. The following definitions will be helpful to the reader who is not familiar with this topic.

- *Geometry:* In the context of human modeling, the geometry is the 3D geometric representation of the surface of the object, for example, a triangulated mesh.
- *Appearance:* In general, the term appearance refers to what an object looks like. In computer graphics and image-based modeling, the appearance of an object refers to the distribution of colors or reflectance properties on the surface of an object.
- *View-dependent:* A quantity that can be represented as a function of viewpoint. For example, specular highlights on a shiny object vary as the viewpoint changes, so the appearance of a shiny object is said to be view-dependent.
- *Pose-dependent:* A quantity that can be represented as a function of kinematic pose. For example, the wrinkles on a finger change as the kinematic pose of the finger changes. Such wrinkling is said to be pose-dependent as it varies as a function of pose.

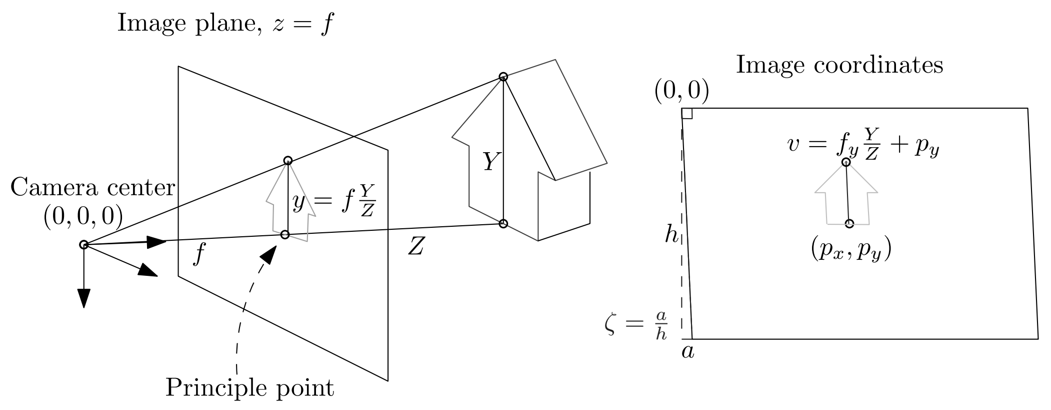


Figure 2.2: The geometry of pinhole projection. The projection equation can easily be derived from similar triangles (left). The scale of an object on the image plane is related to the focal length f. Eq. 2.5 allows for non-square pixels using possibly different focal lengths, (f_x, f_y). The principle point, (p_x, p_y), is the projection of the camera z-axis onto the image coordinates. ζ is often zero, but when non-zero allows for skewed image pixels.

- *Pose- and view-dependent:* A quantity that can be represented as being dependent on both pose and viewpoint.
- *Static:* A quantity that does not vary. For example, we would refer to the geometry of a glass coffee mug as being static. In human modeling, a static geometry refers to a geometric model of a person in a fixed pose. However, we also talk about *models of static variation*. These are models of the variation of a population of static geometries (e.g., a set of humans in a fixed pose). With respect to appearance, a static appearance model is a model of appearance that is not dependent on viewpoint or pose.
- *Dynamic:* A quantity that varies as a function of some other quantity. In the context of human modeling, a dynamic model represents, for example, how the human geometry changes as a function of pose. With respect to appearance, a dynamic appearance is an appearance that is dependent on some quantity, for example, a view-dependent appearance.

2.1.2 Notation

Throughout this work, images will be denoted with the letter I. Single subscripts, I_i, index over either time or viewpoint, and double subscripts, $I_{i,t}$, index over viewpoint, i, and time, t. Silhouettes of an object are also often provided as input and will be denoted with the letter S (e.g., S_i).

Matrices will be in upper-case bold (e.g., $\mathbf{P}, \mathbf{E}$), and vectors will be lower case bold $\mathbf{x}$. The transpose of a matrix $\mathbf{X}$ is written as $\mathbf{X}^\intercal$ and the pseudo-inverse of a matrix, $\mathbf{X}$, is written as $\mathbf{X}^\dagger$. The notation $[\mathbf{X}]_j$ is used to index row j of the matrix $\mathbf{X}$ as a row vector. A concatenation of N vectors (or matrices) will be denoted $\mathbf{x}_{1:N}$.

Matrix transformations of 3D points will be represented in homogeneous coordinates, meaning

3D transformations are 4×4 matrices, $\mathbf{E}$:

$$\mathbf{x}' = [x', y', z', 1]^\intercal = \mathbf{E}[x, y, z, 1]^\intercal = \mathbf{x}. \tag{2.1}$$

The 3D transformation matrices are often Euclidean transforms. In such cases, the transformation can be represented as a 3×3 rotation, $\mathbf{R}$, and a 3×1 translation $\mathbf{t}$:

$$\mathbf{E} = \left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right]. \tag{2.2}$$

The inverse Euclidean transformation is given by

$$\mathbf{E}^{-1} = \left[\begin{array}{c|c} \mathbf{R}^\intercal & -\mathbf{R}^\intercal\mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right]. \tag{2.3}$$

2.1.3 Camera Modeling

A 3D homogeneous point, $\mathbf{x} = [X, Y, Z, 1]^\intercal$, is related to its 2D image projection, $\mathbf{u}$, through a matrix multiplication with the 3×4 camera matrix, $\mathbf{P}$, followed by a perspective division, $\Pi : \mathbb{R}^3 \mapsto \mathbb{R}^2$:

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\hat{x}}{\hat{z}} \\ \frac{\hat{y}}{\hat{z}} \end{bmatrix} = \Pi\left(\begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix}\right) = \Pi\left(\mathbf{P}\mathbf{x}\right). \tag{2.4}$$

The camera matrix can be decomposed into a 3×3 internal component, $\mathbf{K}$ and a 3D Euclidean transform $\mathbf{E}$ [107]:

$$\mathbf{P} = \mathbf{K}\mathbb{I}_{3\times4}\mathbf{E} = \underbrace{\begin{bmatrix} f_x & \zeta & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}}_{\text{Internals}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{\mathbb{I}_{3\times4}} \underbrace{\left[\begin{array}{c|c} \mathbf{R} & \mathbf{t} \\ \hline \mathbf{0} & 1 \end{array}\right]}_{\text{Externals}} = \mathbf{K}[\mathbf{R}|\mathbf{t}]. \tag{2.5}$$

The internal matrix contains the focal length (f_x, f_y), the skew, ζ, and the principle point (p_x, p_y) (see Figure 2.2). $\mathbf{E}$ is the Euclidean transformation that takes points from a world coordinate frame to the camera coordinate frame. A camera is said to be calibrated when its $\mathbf{P}$ matrix is known.

Distortion model

The camera model in Eq. 2.5 maps 3D straight lines onto straight 2D projected lines. Real lenses tend to have radial distortion. Radial distortion is modeled as follows [287]. Let $r = \hat{u}^2 + \hat{v}^2$ be the radial distance of the normalized point in the camera frame:

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \Pi([\mathbf{R}|\mathbf{t}]\mathbf{x}). \tag{2.6}$$

The distorted image point is given by

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = f_{\text{dist}}(\hat{u}, \hat{v}) = \mathbf{K} \begin{bmatrix} \hat{u}(1 + \alpha_1 r^2 + \alpha_2 r^4) \\ \hat{v}(1 + \alpha_1 r^2 + \alpha_2 r^4) \\ 1 \end{bmatrix}, \tag{2.7}$$

where α_1 and α_2 are coefficients of the distortion model.

Silhouettes	**Stereo**		
Shape-from-silhouettes	**Indirectly maximize photoconsistency**		
Representation	Two-frame stereo	$\longrightarrow$ Fuse depth maps	
Voxel-based	Multiplexed photometric stereo		
Marching intersection	**Directly maximize photoconsistency**		
Exact polyhedral mesh	**Representation**	**Photoconsistency**	**Optimization**
	Voxel-based	Color-variance	Carving, graph cuts
	Displacement map	Intensity difference	Belief propagation
	Mesh, Level set	Cross-corrrelation	Gradient descent

Table 2.1: Multi-view methods for reconstruction often rely on silhouette cues or use stereo methods to obtain a 3D model. Silhouette methods compute an approximation to the *visual hull*, and typically vary by model representation. Stereo-based methods either merge depth maps from binocular views or try to directly refine a surface to be consistent with the images. The latter methods vary by model representation, matching cost, and optimization method.

Once the camera internals and distortion coefficients are known (e.g., through calibration [287]), the distorted image can easily be rectified so that straight lines are straight by a simple resampling:

$$I_{\text{undist}}(\mathbf{u}) = I_{\text{dist}}(f_{\text{dist}}(\mathbf{K}^{-1}\mathbf{u})). \tag{2.8}$$

Therefore, the rest of this thesis assumes the input images are undistorted and uses the simple camera model of Eq. 2.5.

2.2 Image-based Modeling

For several decades a central research area in the vision community has been 3D geometrical reconstruction from 2D imagery. In the case of single view imagery, common cues are shading or texture variation. Single image, single view reconstruction usually leads to an ill-posed problem, limited to specific classes of objects (e.g., uniform Lambertian reflectance), and the reconstructed geometry is often only recovered up to some ambiguity (e.g., bas-relief in the case of unknown illumination [31]). The problem becomes well-posed when photometric variation is observed in multiple images taken from the same viewpoint but under different illumination, like in photometric stereo [275]. The photometric variation, combined with known illumination, provides per-pixel constraints on the surface normals that can be integrated to recover a per-pixel depth map.

2.2.1 Multi-view Reconstruction

Multi-view methods, where the images stem from either a single moving camera observing a static scene over time or multiple cameras observing the same scene, are often more practical, with applications in robot navigation and automated modeling. Classical structure and motion algorithms [107], which automatically recover camera pose/parameters from the consistent motion of sparsely identified image features, have now matured enough to give stable pose for image sets taken of static geometry from different cameras on different dates at various times of day [223].

Techniques for multi-view reconstruction rely on silhouettes and/or stereo cues and vary on model representation and optimization method (Table 2.1). One of the most commonly studied configurations is that of binocular stereo reconstruction. Specifically, research has been devoted to the special case where two image sensors are separated by a horizontal displacement and their principle axis lies in the same plane; in this configuration a 3D scene point will project to the same scan-line but will have a different position on that line. This relative difference in the horizontal coordinate is known as the disparity, and it is related to the inverse depth. One image (left or right) is often chosen as the reference image, and the goal is to recover the disparity map, $d(x, y)$, storing the disparity for each pixel in the reference image. Algorithms are developed for this standard configuration, but a general pair of images can be warped into this configuration [93, 185].

A common assumption is that a scene point will project to the same image color in both images, meaning that the following condition on the unknown disparity should hold: $I_r(x + d(x, y), y) = I_l(x, y)$. To overcome sensor noise and problems in textureless regions, the similarity can be measured over a window. For example, a sum of squared difference score (SSD) integrated over a window,

$$\sum_{(u,v)\in\mathcal{N}(x,y)} \|I_r(x + u + d(x, y), y + v) - I_l(x + u, y + v)\|^2,$$

where $\mathcal{N}(x, y)$ is a neighborhood around pixel location (x, y). Alternatively, regularity can be imposed on the recovered map. Other windowing scores (e.g., sum of absolute difference, correlation), specific regularization and optimization methods, as well as a classification and comparison of existing methods is discussed by Scharstein and Szeliski [205].

An extra view, e.g., a trinocular setup, can give more robust results [283]. Both binocular and trinocular stereo are image centered reconstructions, as the depth is represented for a single view. Given a sequence of several images, a more scene-centric reconstruction can be obtained from a sequence of several images (when the calibration is known) by merging several individual, and often independent, disparity maps obtained from neighboring views (e.g., [281]). A common problem with this approach is enforcing consistency between the recovered disparity maps, which are sometimes recovered independently.

For scenes consisting of a single object it is often more convenient to represent and recover the geometry in a fixed world coordinate system, as opposed to depth from a particular view. In the most direct extension, the depth can be represented as an offset from a mesh in the world coordinates (e.g., [262, 33]). Other object-based representations include level sets, meshes, or voxels. In a controlled multi-camera environment for acquisition of human shape, the actor volume is typically limited by the intersection of camera viewing frustums, making an object-based representation and the related reconstruction methods ideal.

In fact, for objects it is useful to consider the silhouette in addition to color information. Given several silhouettes of an object from known viewpoints it is possible to restrict the recovered geometry to lie within the volume created by intersecting the backprojected cones defined by these

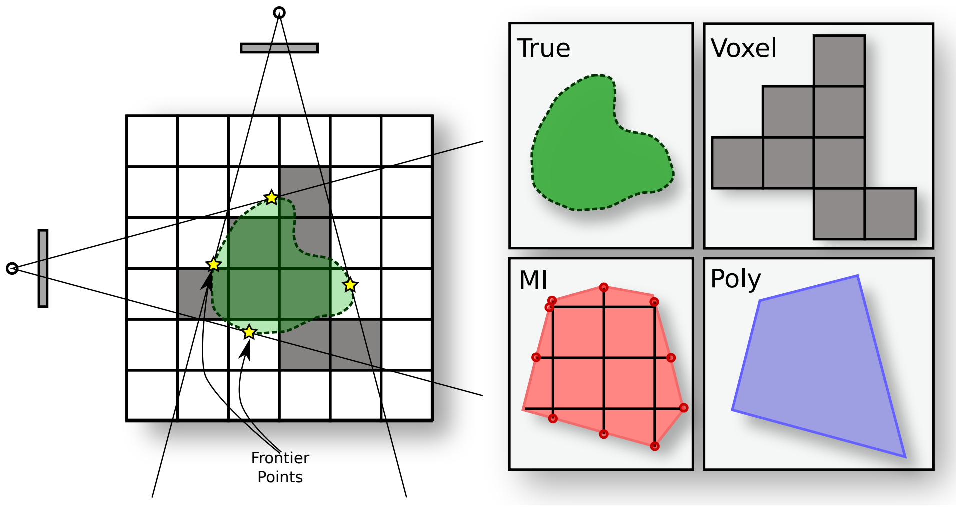


Figure 2.3: A two image planar example of SFS. SFS algorithms approximate the *True* volume. Frontier points (or rims) are points on the object that project to the silhouette boundary on the image.

silhouettes. Theoretically, if it were possible to image the object from every potential viewpoint the limiting volume would be what Laurentini defined as the *visual hull* [139]. In practice, shape-from-silhouette (SFS) uses a set of silhouettes to limit a maximal volume capable of producing the silhouettes (Figure 2.3). Several approaches exist to compute the SFS. The volumetric cones delimited by the image silhouettes can be backprojected and intersected in 3D, or the extruded silhouette edges from one image can be projected onto the other images where they are intersected with the 2D polygon that delimits the silhouette [159]. Volumetric octree occupancy [239], and other novel data structures, such as Marching-Intersections (MI) that represents the volume by three sets of orthogonal rays marking entry/exit of the volume, have been used [193, 240]. These latter approaches are easily converted to a typical graphics triangulated mesh through the marching cubes algorithm [150].

As the silhouette of an object can be reliably extracted using background subtraction (or blue screening), SFS is often the starting point for desktop vision-based capture [28] as well the driving component of several vision-based tracking methods, as we will see in Section 2.4. Although it provides a tight constraint on many objects, SFS cannot recover concavities that are not observable in an input image silhouette.

The use of color intensity information can be used to further refine the object. Using a volumetric voxel representation, *voxel coloring* [211] and *space carving* [136] iteratively peel off voxels that project to inconsistent image colors. Similar to the visual hull, Kutulakos and Seitz present a theoretical bound on these types of reconstructions called the photo hull [136]: the maximal object that is photo-consistent with a set of images. For a Lambertian object observed from several views, each point on the true surface should project to a set of consistent colors in the input images that observe it. The variance on these observed colors from different images should be low, and a voxel can be

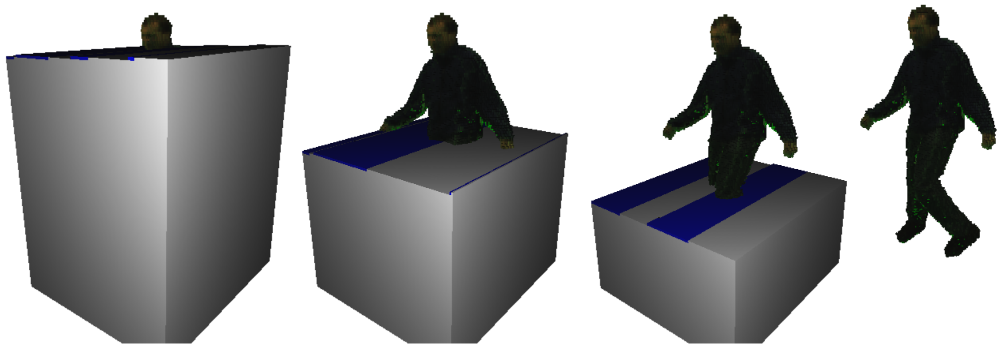


Figure 2.4: An example of voxel coloring when the cameras are all above the scene, the voxels can be traversed in a downward direction.

declared photo-consistent if this variance falls below a predefined threshold. As visibility of a surface point depends on the surface itself, efficient methods that only test photo-consistency once for each voxel with a single pass through the volume are possible if the voxels can be ordered by depth from all cameras (requires a scene that does not intersect convex hull of cameras, see Fig. 2.4 for an example) [211]. Multiple passes (or sweeps) through the volume [136] and efficient bookkeeping of visibility can be used to handle arbitrary camera configurations [65]. Other photo-consistency tests have been introduced to overcome specular highlights [35, 279], and some are developed to be discriminative in captures that include illumination variation [268]. A common problem with voxel carving has been the choice of threshold: if the threshold is too high not enough of the surface gets carved; if it is too low holes get carved into the object (i.e., voxels behind the surface are not photo-consistent, so once the true surface is carved the holes can potentially erode through the object).

In addition to their use in visual hull and photo hull computations, the voxel representation has also been used in integrating unreliable stereo information from subsets of views. Three-dimensional point information from dense stereo can be collected into 3D voxels and used for discarding noisy measurements [14]. Similarly, the dense stereo information can be used to vote for a surface. Matsumoto *et al*. use dense information to cast a vote for the volume behind the image-based depth map for camera views. The final reconstruction can be extracted by thresholding the voted volume [158].

Faugeras and Keriven formulated the multi-view reconstruction problem in the variational framework, where the desired surface minimized a cost functional [85]. The Euler-Lagrange equations of the functional were derived, a level-set was used to represent their surface (see Fig. 2.5), and the minimum of their cost function was found by evolving an initial surface using the level-set methods of Osher and Sethian [177]. Their cost functional used a cross-correlation measure that was computed over the surface using image pairs after pre-warping via a homography defined by the surface normal and surface point. The variational formulation has since been extended with various cost functions to allow for non-Lambertian surfaces [224, 122, 123] and to segment foreground from

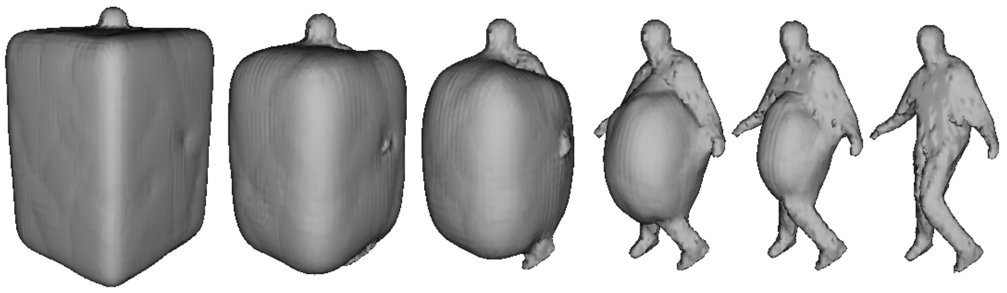


Figure 2.5: An example of level set evolution for capturing a human model.

background while simultaneously obtaining the reconstructed shape [280, 121].

Explicit mesh-based methods provide an alternative to the level-set representation. Similar surface evolution equations can be performed on a mesh [79], or the cost function can be explicitly defined over the triangles of the mesh [88]. Hernández Esteban and Schmitt use a mesh-based representation to extract a surface from a voxel volume that contains a voting score for a surface at each point [83, 84]. A voxel receives a vote for the surface if the results of several pair-wise stereo scores from a camera concur at the voxel. The mesh moves along a smoothed gradient of the volume subject to silhouette and smoothness constraints.

Graph-cut methods adapted from disparity and two-view stereo are also an effective method to find a global minimum to similar cost functionals as the level set formulation. With an initial approximate surface (e.g., using SFS), increasing depth layers can be defined by consecutively peeling away the volume [261]. With an appropriate neighborhood definition, the continuous cost functional can be approximated by a graph where edge weights correspond to the photo-consistency. Nodes on the outer layer are connected to a source element, and nodes on a maximal depth layer connected to a sink node, each connection with infinite weight. Edges between neighboring nodes encode the photo-consistency cost of the surface at the midpoint between the nodes. A graph-cut separating the source from the sink defines a surface, and the corresponding cost of the cut is equivalent to the approximate value of the functional. Finding the surface is equivalent to finding the minimum cut of the graph. Visibility of the surface must be approximated when evaluating the cost for a point. To reduce the preference for a small surface, Vogiatzis *et al.* include a volume preserving term that penalizes volumetric deviation from the initial volume [261]. Such a cost is incorporated in the graph structure by a constant weight connecting each node to the sink.

Another possible alternative to the level set or graph cut formulations is to represent the surface as a binary characteristic function, where a value of 1 (resp. 0) indicates inside (resp. outside) the surface [131, 132]. Through this representation, the multi-view stereo problem can be formulated as finding the characteristic function that minimizes the photo-consistency integrated over its surface [131]. The minimization is performed by a convex relaxation that converts the discrete indicator function to a continuous range of $[0, 1]$. The minimization proceeds as a gradient descent of the photo-consistency function. Silhouette constraints are easily incorporated into this minimization by constraining the integrated characteristic function along silhouette pixels to be greater than 1. The use of silhouette constraints means that no extra volume preserving term is needed.

The above discussion provides a coarse overview of the methods and representations for multi-view scene reconstruction, but it is not meant to be a complete survey of the topic. More recent, ongoing comparisons of the current state of the art are available [210].

2.2.2 Human Reconstruction

We now consider how these general multi-view methods have been used in the context of reconstructing geometry of a human subject. Commercial full body laser scanners serve as alternatives to vision-based methods, but the scanners are generally expensive. For example, Cyberware™'s *Whole Body X* Scanner starts at $200,000 USD [68][1]. The scanner is capable of acquiring a model of static geometry in less than 20s [68]. Again, vision-based methods are potentially less expensive, and are better suited to the reconstruction of dynamic motions as a multi-camera setup can ideally capture an accurate geometry for every frame. Below, we consider methods that are only concerned with obtaining geometry, and postpone the discussion of methods that obtain a dense temporal correspondence to Section 2.4.2.

Some of the reconstruction methods above, including those that focus on shape from silhouette, often demonstrate the use on a human subject in a multi-view studio [159, 86, 160, 271], and many are capable of operating in real-time. View-dependent texturing (discussed in more detail in Section 2.5) is used to render novel viewpoints of the reconstruction. Such methods allow playback of the reconstruction at the discrete time steps where images were taken. Vedula *et al.* [255] show how voxel coloring and 3D scene flow (the 3D motion of scene points) can be used to render between time steps. The main focus of these methods, and other stereo-based reconstruction of arbitrary dynamic scenes (e.g., [289, 140]), is primarily the re-rendering of the same motion from novel viewpoints.

For the methods that focus on multi-view studios for human capture, recent work has explored the design decisions required when building such a studio [227]. Typical issues involve choosing an appropriate background, picking cameras and lenses, designing the lighting environment, choosing the camera configuration, and calibrating the cameras. The camera configuration is one of the most important decisions. Starck *et al.* suggests 16 views on a ring with one overhead view sufficient to achieve accuracy within 1 mm [227].

The multi-view studios can be used to obtain reconstructions of static or dynamic human models. For example, to recover a static human model, Cheung *et al.* align temporal multi-view silhouette information and use it to obtain a more accurate model when the scene object undergoes rigid motion [56]. This technique is used to obtain a more refined human geometry model by capturing a

[1]Accessed from http://www.cyberware.com/pricing/domesticPriceList.html on May 15, 2011

subject rotating on a turntable in front of several cameras [57], with the turntable motion initially unknown. The silhouette cues are integrated over time by finding the rigid transformation that the model undergoes between frames. Colored surface points (CSP), which are frontier points with color information, in frame t must lie within the silhouettes at time $t+1$ after being transformed by the Euclidean transformation between the frames. Similarly, the color information of the transformed CSPs at time t in image $t+1$ must be in agreement. The same holds for the CSPs from $t+1$ when projected into image I_t using the inverse transformation. A cost function measuring discrepancy from these conditions is used to find the transformation of the scene from time t to $t+1$, for each time frame of the sequence. The idea of accumulating multiple observations of a static geometry over time is also used in our static model initialization (Chapter 4), but we do this with fewer views.

Fitting a static model need not reconstruct an arbitrary geometry, and instead parameters of template human models can also be adjusted to fit multi-view images (e.g., [3, 225]). For example, Starck *et al.* fit a template model, consisting of a mesh attached to a skeleton, to as few as two silhouettes [225]. Joints are first manually positioned in the input views, and a 2D shape interpolation method is used to constrain the motion of the projected silhouette vertices towards the input silhouette. The mesh is regularized with a smoothness force that acts to preserve the original distance of neighboring vertices.

When considering dynamic scenes, silhouette cues have also been used to refine and improve dynamic human geometries over time [203, 238, 259, 95, 226]. For example, Sand *et al.* refine a human geometry using silhouettes over time, but in contrast to this thesis they use motion capture data attached to a template mesh to register silhouette data from a temporal multi-view sequence into the coordinate frame of each body part [203]. Furthermore, they do not model appearance, and their subject is suited in tight fitting clothing. However, similar to this thesis, they attempt to recover a model of geometry deformation as a function of local joint angles, a topic that is investigated in more detail in Section 2.3.2. Another common way to utilize the sparse silhouette constraints is to use them in conjunction with an approximate tracked input geometry [259, 95]. Assuming that the model can be tracked through the multi-view input sequence, at each time instant the deformed reference model is pulled toward the input silhouettes through a sparse set of silhouette constraints. The sparse constraints are diffused over the mesh in a way that ensures details in the original mesh are preserved.

In addition to silhouette terms, stereo terms can also be used to refine dynamic human geometries. These refinements have been applied to time-independent geometries [230] or to tracked templated meshes [231, 245]. In a model-based approach, Theobalt *et al.* use a similar approach to combine time varying silhouette cues with color consistency cues to refine the geometry of a deforming human model [245]. The deformations are fit over a series of consecutive frames to a few vertices that are determined to be inconsistent with the images. They also provide a method to transfer recovered motion from a captured human to a laser scanned geometry, but the deformations due to joint motion are not transferable to new animations.

A mesh with a consistent topology (e.g., in the template model-based approaches), can be useful in our intended application of modeling deformations over time. Starck *et al.* illustrate how model-free representations, initially with disparate topology can be brought into alignment [228]. Assuming a genus-0 surface [2], at each frame a dense mesh reconstructed using *generalized voxel coloring* is brought into alignment with a sphere by simplifying the input mesh, subdividing the resulting mesh and resampling the 3D positions of the geometry and color information in spherical coordinates. The spherical representation gives a 2D parameterization of the 3D geometry and appearance maps; subsequent frames of these 2D maps are brought into alignment using appearance and geometric cues. The authors exploit the resulting correlation between geometry and appearance by treating the sequence of geometry and appearance as movies and using video encoding. The assumption of a genus-0 surface, and more refined reconstruction algorithms have been used in the same framework [234]. Specifically, generalized voxel coloring can be replaced with a graph-cut optimization that uses surface line features and frontier points to constrain the resulting surface, allowing a reconstruction without a volume preserving constraint [229] (volume preservation was necessary in earlier graph cut volumetric approaches [261]).

Time-multiplexed photometric stereo has also been used with high speed multi-view acquisition to obtain accurate and dense time varying 3D geometries of human subjects [260]. Vlasic *et al.* use a total of 7 lighting conditions to illuminate a sphere covered in the same Lambertian clothing as the subject. This reference is used to relate a series of illumination conditions of the subject to a per-pixel normal. During capture, the high speed cameras capture the subject sequentially illuminated by the 7 light conditions. The frames are brought into alignment through optic flow, and the normal maps can be computed from the reference sphere for each series of 7 images. The individual normal maps are integrated into depth maps for each view and combined with the visual hull to achieve a time-varying geometry that is not in temporal correspondence. Although the method recovers detailed geometries, it is limited by the hardware setup and the requirement of the subject to wear a single type of clothing.

2.2.3 Discussion

Multi-view vision reconstructions have reached a point where they rival active sensors in geometric quality under some circumstances. Some of the fundamental approaches and cues for solving this problem and their use in reconstructing human geometry were reviewed. Silhouette-based reconstruction alone has been a popular method for object approximation, is used extensively to reconstruct both static and dynamic scene geometry, and is often used as a starting point for further refinement. For this reason, this thesis relies heavily on SFS for both the static initialization of a geometry and to obtain time-dependent geometry variations. Similar to the intended application,

[2]A genus-0 surface is a surface that is topologically equivalent to a sphere.

		Dynamic Variation	
	Static Variation	*Skeleton coupling*	*Example-based methods*
Input	Geometry from various subjects	Geometry, skeleton	Geometry in various poses, skeleton
Method	PCA, Sparse data interpolation	Skinning (LBS)	PCA, sparse data interpolation
Output	Low-dim. model of geom. variation	Geom. coupled to skeleton	Deforming geom. interpolated by pose
	Static & Dynamic Variation		
Input	Geometry from various subjects in various poses		
Method	PCA, sparse data interpolation		
Output	Low-dimensional model of inter-subject geometry variation & model of pose-dependent variation		

Table 2.2: Methods for modeling static variation are concerned with modeling variation across a population of subjects. Dynamic variation is concerned with the deformations of a geometry that occur when a skeleton is posed, either by simply connecting/coupling the geometry to the skeleton or by modeling muscle bulges and cloth wrinkles as a function of pose using a set of examples. Static & dynamic variation is concerned with combining the static and dynamic variation.

silhouette cues alone have been used to recover the geometry deformation as a function of joint angles, but tracking was not vision-based and tested only with tight fitting clothes [203]. In contrast, in this thesis, vision-based methods are used to recover the motion parameters and time-dependent geometry deformations. These image-based cues have been used in recovering dynamic deformations (e.g., [245]) but not in the context of recovering these deformations as a function of joint angles.

2.3 Human Graphics Models

This thesis deals with modeling deformations of a human that occur as their pose changes over time. The review of human modeling techniques by Magnenat-Thalmann *et al*. lists such deformations as dynamic [152]. These are in contrast to methods for modeling static variation of humans in a fixed pose, which are used to represent the inter-human variability. These methods of independently modeling static or dynamic variability can then be combined to model geometric variability across multiple subjects in addition to providing a model of deformation that occurs due to pose (i.e., static & dynamic). Table 2.2 provides an overview of the domains of the static, dynamic, and combined static & dynamic methods. For completeness, the representative methods in each category are covered briefly below.

2.3.1 Modeling Static Human Variation

Although generating a static model that spans the inter-person variation is not the general focus of this work, the static human modeling techniques have interesting related applications. Static models of human variation take as input a set of detailed human laser scans (vision-based methods could be substituted for the geometry), fit a template model to the input data, and then perform some dimensionality reduction on the resulting geometry/skeleton, using for example PCA [9, 212, 213]. This results in a compact representation of human geometry, where a few PCA coefficients can be used to generate a particular instance of human geometry.

Using a template model has additional benefits: articulation parameters can be defined once on the template model and used to deform other geometries (deformation methods will be discussed in more detail in Section 2.3.2), and a single texture uv-parameterization of the mesh allows texture transfer between different models [9]. The construction of a low dimensional subspace representing geometry allows further applications, such as the reconstruction of an entire mesh from incomplete or sparse geometry by searching over the PCA coefficients [9]. In a similar fashion, Seo and Magnenat-Thalmann combine the input geometries with a number of anthropometric measurements, which are used as interpolators over the space of PCA coefficients [212]. A new model can then be generated from a set of body measurements.

The input scans typically consist of people in tight fitting clothing, but such a model could still serve as a good approximation of geometry of a clothed person in a vision application.

2.3.2 Dynamic Human Models

Again, to be useful in all applications, there must be a way to animate the static geometrical model. A set of distinct rigid geometries can be easily animated by attaching a kinematic chain with joints between the bodies. Such a model is readily animated but appears synthetic and mechanical. Early works in the graphics field focused on more natural motions for human models, where an outer skin layer is deformed by an underlying skeleton [51, 153, 133]. Some of these early, specialized approaches even allow for muscle bulging [51].

Many of the methods for modeling dynamic human motion deal with the coupling of a static model of geometry with a kinematic structure. The most common technique for this coupling is linear blend skinning (LBS). Linear blend skinning, also known as skeleton subspace deformation or vertex blending, is simple, efficient, supported by programmable graphics hardware accelerators, and is capable of deforming a manifold by the motion of a kinematic structure or even just a general set of transformations. Instead of assigning a mesh vertex to a single articulated link, LBS transforms a vertex using a convex combination of a set of links (i.e., bones) [4]. Letting $\hat{\mathbf{v}}_i$ denote the position of the i^{th} mesh vertex in a rest position, LBS deforms the vertex as:

$$\mathbf{v}_i(\theta) = \sum_{j \in J_i} w_{i,j} \mathbf{T}_j(\theta) \hat{\mathbf{v}}_i, \tag{2.9}$$

where J_i is a set of links that affect vertex i, and the weighting coefficients $w_{i,j}$ satisfy $\sum_{j \in J_i} w_{i,j} = 1$, and $w_{i,j} \geq 0$. The transformation $\mathbf{T}_j$ usually takes on the form $\mathbf{T}_j(\theta) = \mathbf{M}_j(\theta)\mathbf{M}_j^{-1}(\mathbf{0})$, where $\mathbf{M}_j^{-1}(\mathbf{0})$ indicates the rest transformation of the link j, and $\mathbf{M}_j(\theta)$ denotes the animated transformation of the link as a function of the animation parameters θ.

Modeling packages, (e.g., Blender) allow artists to paint the vertex weights, $w_{i,j}$, onto a mesh as though they were colors. Automated methods, such as a simple scheme of assigning based on proximity, or more elaborate methods for assigning smoothly varying weights by heat diffusion also exist [25]. Figure 2.6 shows an example of an articulated arm modeled with a set of geometric primitives (deformed by a single link) and a single connected skin that is deformed with LBS.

There are several well known problems with LBS: the *"candy wrapper"* effect of collapse when

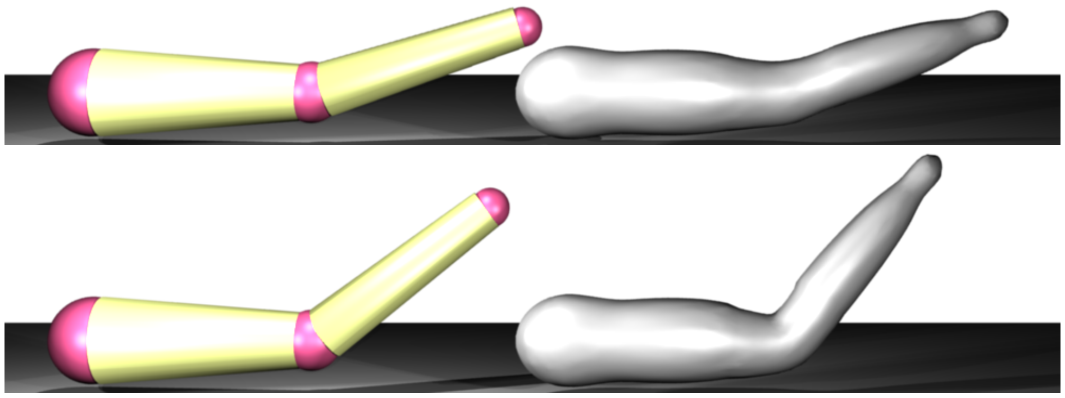


Figure 2.6: An example of geometric primitives, spheres and truncated cones, rigidly attached to an armature consisting of two kinematic links/bones (left), and a single manifold geometry with linear blend skinning (right) rendered in Blender.

a limb has too much roll, and the problem of volume loss at bending joints [168]. Cordier and Magnenat-Thalmann combine the transformation matrices in a different manner to alleviate the first of these problems [63]. Instead of using a linear combination, Cordier and Magnenat-Thalmann use Alexa's method of combining transformations [6], where the matrices are transformed into the tangent space of SE(3) using the matrix logarithm, combined with addition, and transformed back to matrices using the matrix exponential: $\mathbf{v}_i(\theta) = \exp(\sum_{j \in J_i} w_{i,j} \log(T_j(\theta)))\hat{\mathbf{v}}_i$. Other methods for interpolation include spherical blend skinning, where a rotation center is chosen for each set of influencing bones and the interpolated rotational component is constrained to be a unit quaternion [127]. Kavan *et al.* point out shortcomings of these alternative interpolation methods, and suggest using dual quaternions to represent and interpolate Euclidean transformations, which overcomes the aforementioned problems, is computationally efficient, and requires no extra input data (e.g., it uses the same data as LBS) [125].

The skinning techniques, regardless of interpolation method, are a simple way to deform an attached mesh by means of skeleton motion. Although the skinning weight parameters do allow some non-rigid motion of the mesh, they alone are not sufficient to reproduce a rich set of deformations (e.g., muscle bulging). A common approach in building a model with a richer set of deformations is to take a set of example geometries at different poses and treat the problem as a scattered data interpolation [148, 221, 8, 135, 269]. In the case of artist edited data, where the geometry is in full correspondence, these example-based approaches annotate the input mesh with a point in the interpolation space, which may contain joint angles and other abstract parameters, such as gender [148, 221]. The interpolation problem is then to reproduce a deformed geometry from any point in the interpolation space. Radial-basis functions (RBF) are commonly used to perform the scattered interpolation, and the deformations are stored as vertex displacements in the rest position [148, 221]. For kinematic objects, like human hands, it is impractical to densely sample all possible combinatorial pose spaces. In order to deal with a sparse sampling, typically the vertices that are attached to a bone are only influenced by a few neighboring joints. For example, in the weighted pose-space deformation used for modeling deformations on a human hand, each vertex uses an independent weighted combination of joint differences to influence the per-vertex interpolation [135]. These methods can also run in real-time on the GPU [192].

In the case geometric data is acquired from a range scanner or vision-based geometry capture, the input geometries must first be mapped to a consistent geometry [8, 135]. Allen *et al.* used a set of range scans of the upper torso of a human in different poses as example geometries [8]. Noting the combinatorial nature of human motion, each part (e.g., forearm, upper arm, and torso) was captured separately for one side of the subject only. Known marker positions were used to obtain a coarse articulated alignment, and the dense geometry scans of corresponding parts were brought into dense correspondence by computing a subdivision displacement map from a base surface to the dense geometry. Interpolation between the captured aligned geometries was obtained by two components: a k-nearest neighbor interpolation over the displacement maps of the individual parts weighted by inverse distance of pose, and an interpolation to ensure a smoothly varying surface between parts (e.g., at the elbow, shoulder).

In their *eigen-skin* method, Kry *et al.* use a similar example-based interpolation and demonstrate it for a human hand model where the deformations are computed by simulating a physical model of the hand [134]. Instead of using a displacement map, the deformations are measured by a displacement vector in the rest pose of a skinned mesh. Each joint is moved independently in the input data, and the affected subset of vertices and their displacements are measured and mapped to rest pose space (e.g., by undoing the transformation of the current pose). The displacement vectors for a joint are assumed to belong to a low dimensional subspace, which is approximated using PCA, and radial basis functions are again used to interpolate the coefficients of the deformation in this lower dimensional space. As the deformation is represented in the rest pose, the skinning transformation is subsequently applied. The net effect of all joints on a vertex is computed as a sum of the displacement from the affecting joints. They show how their approach can be implemented efficiently on current consumer graphics hardware.

Instead of a scattered data approach, some example-based approaches have instead focused on using a similar machinery as skinning to model the deformations. Mohr and Gleicher demonstrate that the candy wrapper problem can be overcome by inserting an extra joint in the affected link that performs half the rotation [168]. Additionally, they suggest a method for modeling the dynamic deformations not handled directly by LBS, such as muscle deformations. The non-linear vertex deformations that occur on a kinematic link are assumed to be a function of the adjacent link orientations. Four new transformation joints that scale up or down in a direction perpendicular to that of the limb are added. Using a set of topologically equivalent models illustrating such deformations with corresponding kinematic locations as input, they insert these extra joints and estimate the rest positions of the vertex positions and corresponding weights of all the bones. This optimization pro-

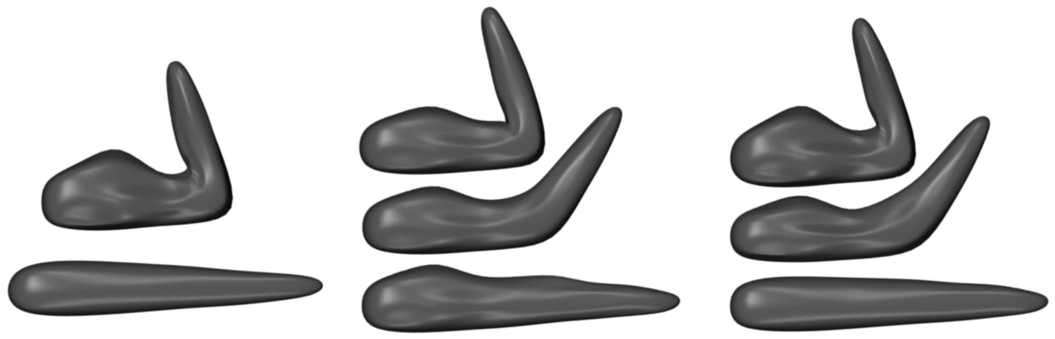


Figure 2.7: Two sample input meshes (left) with an exaggerated bulging muscle. The effects of fitting linear skinned weights averages the bump over the two states; Mohr and Gleicher's approach of extra bones faithfully represents the bulging muscle (right) [168].

cess will assign non-zero weights to the portions of the skin that deform in such a manner, allowing realistic muscle bulges to appear in their model deformation (see Figure 2.7 for an example). The authors note that an advantage of such an approach is that it uses the same mechanism as LBS.

Another candidate, termed multi-weight enveloping, was proposed by Wang and Phillips [267]. Instead of using a single weight for a bone transformation per vertex, as in Eq. 2.9, their example-based approach assumes that each vertex can have an independent weight for each element of the affine transformation matrix of the bone. Some precautions are taken to avoid over-fitting, but the extra degrees of freedom allow for a better representation of the example deformations.

Augmenting a skeleton with auxiliary joints could also be used to drive more general deformations (e.g., cloth motion), where the auxiliary joint pose is driven by some underlying animation/interpolation process [119, 126]. James and Twigg demonstrate that the skinning approach can even be used to model deformation of non-kinematic structured objects such as deforming cloth [119, 126]. From a set of examples, they cluster triangles that undergo common transformations, and use these clusters to find an underlying set of transformations. Vertices are assumed to be affected by a fixed number of these transformations; the corresponding skinning weights are found by minimizing the residual between the predicted/skinned vertices and the input data. The remaining residual is modeled using an eigen-skin approach.

2.3.3 Static & Dynamic Models

If body variation parameters are treated as abstract parameters in interpolation space, then the example-based approaches discussed above can be used to achieve a static & dynamic human model. For example, Sloan *et al.* show how their example-based interpolation approach can be used to model dynamic effects for a multi-gender human arm where the interpolation is a function of arm joint angles and gender [221].

Others have focused more specifically on modeling inter-human geometrical variability while being capable of achieving realistic deformation during motion [12, 19, 10]. These models often share the same benefits as the static models (e.g., texture transfer, new models from few parameters, etc.), as well as the potential for deformation transfer. The SCAPE model of Anguelov *et al.* models the pose specific and inter-personal variability as deformations on the local coordinates of triangles [12]. Using a set of scans of a single subject in several poses, which are automatically registered and topologically equivalent, deformation of the vertices of a triangle $\mathbf{v}_{i,1}^j, \mathbf{v}_{i,2}^j, \mathbf{v}_{i,3}^j$ in pose j from a chosen rest pose, say $j = 0$, is modeled as:

$$(\mathbf{v}_{i,k}^j - \mathbf{v}_{i,1}^j) = \mathbf{R}_{\mathbf{J(i)}} \mathbf{Q}_{ij} (\mathbf{v}_{i,k}^0 - \mathbf{v}_{i,1}^0), k \in \{2, 3\}$$

where $\mathbf{R}_{\mathbf{J(i)}}$ is the rotational motion of the rigid part that triangle i belongs, and $\mathbf{Q}_{ij}$ is a $3x3$ linear transformation modeling pose deformation of the triangle. The transformation matrices $\mathbf{Q}_{ij}$ are fit to the data with a smoothness constraint on the transformations of neighboring triangles as the problem is under-constrained. Given a set of rotation and deformation matrices, the mesh must be reconstructed from this differential representation by finding coordinates that best agree with the applied deformation (translation of the entire model is a free parameter). Linear regression is used to map the 6 parameters of the relative orientation of neighboring body parts (represented in axis angle format) to the triangle deformation matrix, e.g., $\mathbf{Q}_i(\delta\mathbf{R}_{\mathbf{J(i)}}^+, \delta\mathbf{R}_{\mathbf{J(i)}}^-)$. This gives a dynamic model that interpolates deformations over any pose of the object.

Interpersonal variation is modeled as a linear transformation that occurs between the rotational and pose dependent transformations. Using a set of laser scans that span the inter person variation (similar to what was used in the static models), this linear transformation is fit to the data using a smoothness constraint between neighboring triangles using the deformation model from the single subject. A compact representation of the concatenated vector of parameters (there are $9 \times ntri$ per input) is obtained through PCA; any shape can be represented by a handful of basis coefficients. Due to the pose-deformation model, given a set of PCA shape coefficients, a model complete with pose-dependent deformations can be generated. The authors show the applicability of such a model to filling missing data and to generating a complete mesh given only a few correspondences (e.g., correspondences could be obtained from laser scanning) [12].

An alternative method, proposed by Allen *et al.* [10], criticizes the SCAPE model for using the same pose-dependent deformations for all body types. Instead of using only a single model with a dense set of pose information, they use a single model with a dense sampling of poses (as was done with SCAPE), several models with a few poses, and more scans with only two poses. Their model is based on LBS with a *corrective skinning* technique, where they use a template mesh and initialize a global set of skinning weights. Pose-dependent variation is modeled using radial basis functions to map joint angles to vertex displacements in the rest pose (as was discussed in Section 2.3.2). The key angles for the RBFs are at fixed locations, and each character is allowed to have its own unique pose-dependent vertex displacements. In total, a character is represented by a set of bone lengths, the rest position of the vertices of the mesh, and its pose dependent variation displacements. These parameters are assumed to have been generated from some low dimensional latent variables. As their input data is inhomogeneous, they describe a complicated optimization process to recover the global model parameters (e.g., skinning weights), the non-orthogonal basis vectors of their linear latent variable model, and the latent variables for the input data. They used only 3 latent variables (i.e., 3 basis vectors and a mean) to model the combined body and pose variation in their input data.

The dynamic and static models serve as a useful tool for computer vision applications. The 3D parameterized model can be fit to images [19], and can be used to recover the body shape underneath clothing [20]. Similar bilinear static & dynamic models have been used to find the pose of a subject and the camera pose from a single image [108]. The 3D SCAPE model and other static & dynamic models can be decomposed into 2D models for person detection in 2D [87] and human pose estimation when clothed [103].

2.3.4 Physical Simulations

Physical simulation plays a central part in creating realistic and interactive virtual worlds (see [171] for a more complete review). Two central areas relating to humans are the physical modeling of muscle/tissue and clothing.

Muscle and Tissue Simulation

The dynamic methods of Section 2.3.2 provided a way to deform an external skin using a kinematic skeleton. Muscle bulging could be approximated by procedural or example-based methods. In this thesis, the proposed model is also capable of modeling muscle deformation as purely kinematic. However, real tissue can also exhibit dynamic effects such as jiggling. Although modeling these types of deformation are outside the scope of this thesis, a few relevant methods are briefly reviewed below.

The Finite Element Method (FEM) is often used to add elastic vibrations into muscle deformation [117, 216, 105, 48]. For example, Capell *et al.* use the FEM to simulate the equations of motion in a volume surrounding the mesh [48]. As FEM integration can be slow, the dynamic simulation is constrained by the skeletal motion, and for efficiency the dynamic equations are linearized and solved on overlapping parts of the object.

An alternative approach to accelerate the simulation is to use a simpler dynamical model. Instead of using the FEM method, Shi *et al.* uses a simple surface-based mass spring model on the object surface [216]. The unintuitive parameters of their dynamic model (e.g., damping of internal springs) can be fit to the results of a traditional FEM simulation. In an even simpler dynamic framework, Park and Hodgins use an independent damped spring model for the dynamics of surface points [180]. In their case, the force acting on a surface point is approximated as a weighted combination of the angular and linear velocity of the part to which it is attached. The parameters of the model (the weights contributing to forces and the spring coefficients) are fit to a sparse set of motion capture points on a real skin surface. The vertex displacements from the dynamic model are accumulated

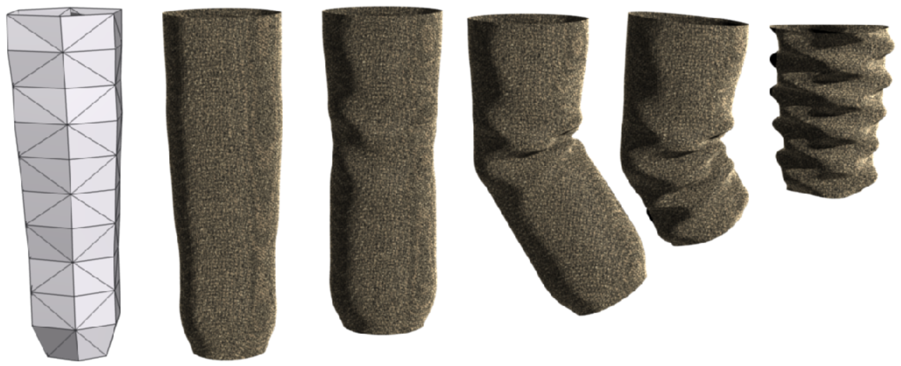


Figure 2.8: Modeling cloth buckling using a low resolution mesh (left). Shrinking of the base mesh is used to procedurally deform diamond patterns on the base mesh [74]. A higher resolution mesh is splined over the procedurally deformed base mesh. The end result exhibits wrinkles, but the wrinkles look artificial (especially the compressed mesh on the far right).

with a set of pose-dependent vertex displacements that are modeled through a linear regressor on the adjacent joint angles.

Cloth Modeling and Simulation

For the most part, the human geometric models use a human model that is either unclothed or covered in tight fitting clothing that deforms directly with the model. In the case of the methods modeling static variation, it is not clear how well the methods would work if the subjects were clothed. Clothing could cause potential problems in aligning the geometries. Dimensionality of the subspace might need to be increased to adequately represent input meshes, and interpolated elements not close to input scans may exhibit strange artifacts.

With respect to modeling the dynamic aspect of clothing and wrinkling, one option is to use a procedural approach [138, 106], for example, by modeling buckling and folding as a function of the deformation of triangles [74]. However, such methods typically look artificial (see Figure 2.8). Another alternative is to use the example-based methods to model clothing effects as pose-dependent. The benefit of example-based approaches are that they can be art directed: a modeler designs the wrinkles for a few poses, and these deformations are interpolated across animations [66, 130]. Instead of manual modeling, the input deformations can even be acquired from vision-based cloth capture [273].

Another alternative is to physically simulate the cloth animations on top of an underlying collision geometry. The above techniques for modeling a static or dynamic human geometry could be used to obtain this collision geometry. The realistic simulations involve modeling the cloth as a mass-spring system, complete with shear, stretch, and bend forces [23, 24, 42, 58]. In applications where real-time is not an issue, such as the film industry, full dynamical simulation with fine tessellation of the mass-spring system and collision detection is feasible. Unfortunately, in real-time graphics applications processing dedicated to cloth simulation is finite, meaning the resolution of the cloth often needs to be limited and collision needs to be approximated. More recent methods help overcome these issues through hierarchical solvers to achieve real-time simulations of complicated phenomenon, such as draping of dresses [169].

Several other approaches avoid the high costs of a dense simulation at runtime by combining dense pre-simulated cloth motion to a low resolution simulation to ensure real-time results. For example, Cordier and Magnenat-Thalmann use the pre-simulated data to limit the candidates for potential collision, and also use it to interpolate the fine-scale folds between the coarse regions that are actually simulated in real-time [63]. In a similar approach as the example-based dynamic human methods of Section 2.3.2 , Wang *et al.* model tighter fitting clothing as pose-dependent (or kinematic), and transfer the high resolution pre-simulated cloth folds to a low resolution simulation [266]. In their case, the pre-simulated database of clothing wrinkles and folds is generated by moving a human subject from a rest pose to a variety of poses and physically simulating the cloth deformations that occur at those poses.

In a similar approach, Müller and Chentanez use a separate simulation of a *wrinkle mesh* that is attached to a coarse real-time simulation [170]. The higher resolution *wrinkle mesh*, is constrained to lie within a prespecified radius of the coarse mesh, and collision detection is avoided by adding a constraint that keeps the wrinkle mesh on the positive normal direction of the coarse surface to which it is attached.

The design parameters for cloth simulation (coefficients related to spring stiffness and damping), are often hand tuned, but calibrated acquisition methods have been proposed for extracting these parameters from cloth samples [32, 265]. The cloth parameters have even been fit to multi-view sequences of a subject performing arbitrary motion. For example, Stoll *et al.* infer a human geometry underneath the recovered multi-view geometry and fit the parameters of a cloth model to the multi-view silhouettes and extracted SIFT feature points [235]. In their application, the objective is not to recover exact parameters, only to reconstruct clothing that fits the input images and looks realistic.

2.3.5 Discussion

In this section, several methods for modeling graphical human models have been reviewed. Linear blend skinning is the simplest way to attach a surface to a kinematic hierarchy, and although a number of alternatives have been presented to help overcome some of its limitations, it remains a commonly used representation. The compact model of appearance and geometric variation proposed in this thesis is built upon this linear blend skinned mesh, primarily because it is expressive (for both kinematic, and non-kinematic objects), and the derivatives of the surface motion with respect to the joint angles are easy and efficient to derive, meaning it can easily be used for vision-based tracking.

Through the example-based methods, which interpolate a set of deformed example meshes at specific poses, it is possible to layer pose-dependent deformations, such as muscle bulging, on the skinned mesh. Furthermore, many researchers have noted that clothing is often kinematic [266, 130] and have used example-based methods to also model cloth deformation. This pose-based modeling of clothing and muscle helps overcome the computational requirement of physical simulations by precomputing or capturing the cloth deformation. In this thesis, a similar example-based method is used to model deformation variation, with the main difference being the acquisition from video of unmarked surfaces, instead of through simulation or acquisition of specifically designed marked clothing.

2.4 Motion Capture

Human motion capture (MoCap), here defined loosely as the acquisition of kinematic joint angles over time, has become an integral part of entertainment applications including computer gaming and computer animation in feature films. Acquiring the motion data from actors ensures realistic animations while reducing the manual burden on animators.

The direct output of motion capture is time varying joint angles, which are typically tied directly to a kinematic specification (i.e., the joint configuration and link-lengths of the kinematic structure are essential for playback). Several research topics related to MoCap focus on the usability and reusability of the data, such as *motion transfer* which seeks to map data acquired on one subject to other subjects/characters. Here the focus is on the methods of obtaining the joint angles, which are commonly broken into two categories: marker-based and markerless.

2.4.1 Marker-based Motion Capture

Marker-based motion capture utilizes special markers, such as retro-reflectors, that are attached to the subject who typically wears tight fitting clothing. Many commercial products take this approach, for example Vicon [258] and PTI's Visualeyez [191]. The special markers are specifically placed, their position is easily extracted, and the temporal correspondence is used to obtain 3D feature point tracks. This data can then be related to the motion of an underlying skeleton.

Commercial systems that employ markers are often costly, and the restriction to the use of tight fitting suits limits their use in recovering general clothing deformations. As we are interested in directly capturing clothing deformation, the ability to suit the actor in plain clothes is essential. Furthermore, the artificial markers could produce visual artifacts in the recovered appearance and would require an extra clean-up phase to remove the markers.

2.4.2 Markerless Motion Capture

Joint-angle capture

In contrast to marker-based methods, markerless methods do not introduce artificial markers onto the actor but rather seek to infer the human motion from natural image features such as edges, silhouettes, or texture. Markerless tracking of humans is an important field of study where many

Representation	**Cues**	**Priors**	**Optimization**
Truncated cones [75]	Color [219, 217]	Joint limits	Algorithmic [163]
Super-quadrics [100]	Optic flow [124]	Online smoothing [197]	Local [244, 245]
Meta-balls [184]	Stereo [75, 184, 167]	Dynamic prior [44]	Global [18, 217]
Meshes [72, 175]	Silhouettes [199, 245]	Data-based [252, 43]	Local & global [95]

Table 2.3: Components (and some representative options) that make up a markerless model-based human joint tracking implementation.

approaches have been proposed. The most general of these approaches attempt to track human motion in single-view camera sequences [222, 44, 54, 252, 217, 40]. As monocular video sequences are extremely common (e.g., existing video footage, television broadcasts, etc.), the monocular methods would offer the greatest level of applicability. Unfortunately, recovering the pose of a human from a single view is often ambiguous, complicated by the number of degrees of freedom and self occlusions. Many of the tracking ideas directly extend to multi-view reconstructions, and the addition of more cameras has been shown to reduce some of the ambiguities. In this thesis, it is assumed that acquisition takes place in a controlled environment where multi-view methods are utilized for dense geometric reconstruction, so this review is focused on multi-view model-based tracking. Alternative approaches can use machine learning to learn a more direct mapping from images to pose (see [188]).

The literature for model-based multi-view markerless motion capture is extensive. Such methods seek to align a prior model of the human with corresponding image features. Multi-view model-based tracking implementations are distinguished by the choice of model representation, the type of image cue used, the motion or pose prior (if any), and the type of optimization (Table 2.3). For model representation, typically a set of simple geometric primitives (truncated cones [75], super-quadrics [237, 100, 222], meta-balls [184], or triangulated meshes [72, 245, 175]) to a kinematic chain. These primitives are often rigidly connected to a single kinematic link, but others have used the more traditional graphics approach of linear blend skinning (see Section 2.3.2) to give a smoothly varying mesh near joint angles [175, 128]. Some implementations even take into account a geometric model of clothing [198]. Model parameters, including initial joint positions, kinematic link lengths, and in some cases the shape or scale of the attached geometric representations, are often initialized (manually or in some cases automatically) in the first frame of the sequence. With few exceptions (e.g., [245, 71]) these parameters are fixed throughout the tracking phase.

The corresponding image features include texture/color/appearance [219, 217], optic-flow [124], edges [222], and silhouettes. Each of these features can be used directly to drive the tracking. For example, Ju *et al*. use the parameters of a planar patch model to parameterize the observed optic flow [124]. Edges and silhouettes can also be used directly; a common application typically extracts edges/silhouettes from the input images and formulates a cost function that aligns model edges with image edges [100]. Minimizing the cost function gives the parameters of the kinematic model that minimizes the distance of a set of projected model edge points from their corresponding image edge.

The correspondence between model and image edges is approximate, where the closest edge point is declared the corresponding point for a given configuration. For efficiency, these methods often make use of a distance map (chamfer map) [19, 100, 18], which encodes the minimum distance to an image-edge for every location in the image domain. With the distance map, which can be computed efficiently [70], the exact correspondence between model point and silhouette/edge is no longer needed as the minimum distance can be looked up. A similar cost function can be defined in 3D, where the distance is computed from the model point to the closest silhouette image ray [199]. Yet another direct use of silhouette information is to formulate the cost function as the sum of pixel-wise XORs of the image and model silhouettes (obtained by rendering the model to an image) [244, 245]. This XOR cost can be weighted by the chamfer map for less sensitivity to erroneous silhouettes [54].

Other information derived from the 2D image features can also be used to drive the tracking. For example, individual silhouettes can be combined using shape-from-silhouettes (Section 2.2), giving an approximate enclosing 3D geometry. Analogous to the driving force with silhouettes, tracking seeks to minimize the distance between model points and their corresponding points on the approximate visual hull. Again, this correspondence is in general unknown so an Iterated Closest Point (ICP) approach is taken. Incorrect correspondence can lead to mistracks. Surface color information, accumulated from image observations of the model surface over time, can be used to penalize matching to visual hull points with inconsistent color [128]. Surface normal information can also be used to prune poor matching points; Ogawara *et al.* do this by combining a term that measures Euclidean distance between a model and visual hull point with a term that measures normal deviation between the points [175].

Three-dimensional point features derived from stereo matching techniques can also be incorporated into the tracking [75, 184, 167]. These stereo features can be interpreted as the combination of texture/edge information from multiple views, and they may be either obtained from dedicated stereo rigs or from the closest neighbors in a capture environment.

Typically, the silhouette of an object is extracted by an independent process that has some model of the background, and uses deviation from this to segment an image. As the silhouette and its derived information usually provide strong constraints on the pose of a human, some have investigated the joint estimation of both the silhouette and the pose of a human actor [38, 195]. For example, Rosenhahn *et al.* interleave the pose estimation with the silhouette extraction. Silhouette cues are used to find the pose of model that aligns with current silhouettes, which is then interleaved with a refinement of the silhouette segmentation that uses the projected model silhouette as a segmentation prior [195]. The simultaneous estimation of segmentation while tracking allows for practical applications including vision-based tracking of humans interacting with machines [196] and motion capture in outdoor environments from moving user-held cameras [110].

Silhouettes alone are a powerful cue, but combination of silhouettes with other cues can be used to resolve ambiguous configurations [96, 244, 22]. For example, Gall *et al.* use both a silhouette and

	Two-time	**Multi-time**
Single-view	Optic flow [45]	Linked flow [236], basis-constrained [99]
Two-view	Binocular scene flow [114]	Patch-based long range [78]
Multi-view	Scene-flow [256, 186]	Align per-time reconstructions [233, 253]
		Pattern clothing [209, 273]
		Markerless garment capture [37, 90]
		Spatial & temporal matching [64]
Intra-view structure		Template geometry [201]
		Non-rigid structure and motion [39]

Table 2.4: A breakdown of methods for solving non-rigid dense motion estimation by number of views and number of times used in the method. These techniques are reviewed roughly by row. The simplest case is 2D non-rigid optic flow from a single view. The multi-view and multi-time approaches have the advantage that multi-view stereo can be run at each time instant. The intra-view techniques, which may be multi-view or single-view, try to also make use of intra-view observations to improve structure.

a color distribution term on body parts [96]. Theobalt *et al.* use silhouettes to get an initial registration of the body parts, and then refine the motion parameters so that they agree with 3D flow fields that bring the image intensities of the rendered model into alignment with the previous frame [244]. Gall *et al.* use SIFT-like features extracted from the reference model along with silhouette cues in an ICP-like minimization framework to reduce tracking drift [94]. Instead of matching between features extracted on a reference pose, SIFT features can also be extracted and matched between adjacent frames [71]. In these cases, the silhouette features complement the typical sparse SIFT features.

Regardless of the image cue used in the tracking, it is also possible to constrain the recovered motion through the use of pose and motion priors. Joint limits can be used to avoid some unnatural poses; online smoothing [197], dynamical motion priors [44], and motion priors derived from motion capture databases [252, 218, 43] can also be used to avoid unlikely pose and motion trajectories.

For the most part, the optimization of the cost function is dependent on the features used, and some methods are purely algorithmic [163]. Direct approaches capitalize on the hierarchical nature of the human body, and use a hierarchical optimization that first optimizes the parameters of the root, and subsequently continues down the kinematic chains [244, 245, 175].

Bayesian and multi-modal formulations typically use a particle filtering approach [18, 217, 219, 77, 222, 218]. The accuracy and noise of particle filtering methods can be reduced if followed by a subsequent local minimization [96]. More recently, a combined global and local approach has been advocated by Gall *et al.* [95]. In the local-global approach, the local optimization is used to track when possible. If the local tracker loses track (on any limb), a global particle filtering approach is used to recover the lost limbs. The fast local method succeeds most of the time, with the exception of fast motion or noisy inputs, meaning the slow global tracking is used infrequently.

Dense motion capture

The above discusson considered motion capture for the purpose of recovering the coarse joint angle motion. For analysis and re-animation, it is often necessary to obtain finer-scale motion, such as the correspondence of surfaces over time. With a denser set of input markers, the same marker-based methods as in Section 2.4.1 can be used to obtain these dense correspondences (e.g., [179]). Such dense marker-based methods suffer the same limitations as the marker-based motion capture systems.

Obtaining dense temporal correspondence from images has been formulated in different settings, from the single-view cases with two time frames to multi-view setups with long range video. These formulations are reviewed here according to Table 2.4, with the review leading up to the methods for dense multi-view long range correspondences and closing with methods that utilize intra-view observations to obtain estimates of structure.

Using images, obtaining dense correspondences can be seen as a similar correspondence problem as multi-view stereo, except the correspondences are now obtained over time instead of over view. The simplest case is optical flow, where the 2D pixel motion is recovered from two adjacent time frames. A common approach to obtain optical flow is to find the scalar fields, u and v, that minimize a variational energy integrated over the image domain, Ω [45]:

$$E_{flow}(u,v) = \iint_{\Omega} |I_2(x+u(x,y), y+v(x,y)) - I_1(x,y)|^2 + \alpha_{\text{smooth}}(|\nabla u|^2 + |\nabla v|^2)dxdy \quad (2.10)$$

The first term of the variational energy has a data cost, which seeks to find a displacement that gives similar intensity as the undisplaced point, and a smoothness term weighted by α_{smooth}, which ensures neighboring pixels have similar displacements. The displacements are recovered by solving the Euler-Lagrange equations of Eq. 2.10. The minimization typically requires several linearizations of the data term and is often solved on multiple levels of a Gaussian pyramid.

Optical flow is a restricted example, where 2D motion is obtained and linked between neighboring temporally adjacent frames. Longer range correspondences can be obtained by linking the observations over several frames [236] or by using a temporal basis to parameterize the flow over time [99, 173]. However, 2D optical flow is not enough and must be related to the 3D scene motion in order to track dense surfaces in 3D.

Scene flow, the 3D flow of scenes, is the 3D analog of 2D optical flow. To reconstruct scene flow, constraints between differential motion of the surface and its differential motion in the images are used to link the intra-camera optical flow to the 3D surface flow [256]. The geometry can be reconstructed using any of the multi-view vision representations (e.g., voxels or level sets). Instead of this decoupled reconstruction, which first reconstructs a geometry and then uses optic flow to obtain 3D flow, better results can be obtained by simultaneously estimating the scene flow with the structure. For example, it is possible to solve for 3D flow directly, without requiring individual 2D flow fields [186]. Other improved estimates of structure and flow can be obtained by clustering

3D regions and modeling motion with parametric flow [285], by coupling depth and flow in the estimation framework [114, 270, 26], or by propagating covariances from depth and optic-flow estimation into the 3D flow estimates [149].

In fact, the use of temporal information through scene flow has been shown to improve geometric reconstructions [257, 49, 285]; in other words, even when the correspondences between temporal geometries are not obtained, temporal constraints improve reconstructions. For example, in the level-set stereo framework temporal regularization can be used [154], or, in the case of stereo, aggregation over a skewed window in time [284].

Unfortunately, scene flow methods often only relate the geometries between adjacent time steps. Longer range tracks have been established using patches [78] or through spherical parameterization of scene geometry and a matching function that encodes both geometric and color distances [232]. However, in the latter case the tracking does not take into account known coarse motion, and the per-frame reconstructed geometry does not utilize temporal cues for refinement.

Another approach to obtaining dense 3D correspondences over long sequences is to first reconstruct the geometry at each time instant and then to use image and geometric feature matching to relate these geometries to one another. For example, when enough input views are available, a silhouette- and photo-consistent geometry could be reconstructed in each frame [112, 229]. Appearance-based keypoints extracted from input images, such as SIFT [151] or SURF [29], identified in the input images can be backprojected onto the model, and geometric features (such as extrema of the geodesic distances) can be used to obtain sparse correspondence between these disparate geometries [233, 253, 72]. Starck and Hilton have robustly solved this keypoint matching problem on geometries through a markov random field formulation [233], and the sparse feature matches can be extrapolated to the entire surface, e.g., through a differential coordinate deformation framework as done by Varanasi *et al*. [253]. Alternatively, the surface tracking can be accomplished through geometric cues only [264, 263, 47], for example, by iteratively deforming a reference mesh using only correspondences between local surface patches [47]. The geometry need not be reconstructed independently each frame, but in contrast, an initial accurate model can be brought into alignment and refined with input images through a similar key-point driven approach [71]. These approaches essentially provide a way to perform surface tracking without the use of a kinematic skeleton. They do not make use of the known coarse tracking and provide no way to improve geometry when limited views are available.

One common application of dense motion capture is the acquisition of garment motion. Methods for garment capture have similar goals of acquiring dense correspondence. To simplify the reconstruction process, a template of the clothing or garment can be used to fill holes in stereo data [190, 37], physical simulations can be used to constrain the tracking [109], and color-coded patterns can be used to ease correspondence finding [209, 273]. The use of color-coded patterns is impractical for our application, as we also care to model the appearance.

To acquire garments, Bradley *et al*. use multi-view stereo from 16 input views to obtain dense per-frame geometries that are linked over time [37]. The noisy input geometries from stereo are parameterized by minimizing the stretch relative to a simple base mesh through the use of the geometrical holes that appear at the neck, arms, and base of the garment. The input geometries are then represented as the vertex motion of template geometry (computed by inflating a flat image of the garment). Later work has illustrated how to improve the geometry of captured wrinkles on the garment [187]. However, the accuracy of the tangential motion on the surface is limited as the temporal correspondence is only obtained from parameterization relative to the base mesh that uses only 4 geometrical correspondence points.

Another mesh-based alternative that does make use of dense image-based cues was proposed by Furukawa and Ponce for highly textured clothing and faces [89, 90]. In their formulation, the surface motion normal to the mesh vertices is first optimized using multi-view stereo, and tangential motion is subsequently optimized by taking into account temporal image frames. A more global approach to deforming and tracking a mesh through a multi-view sequence was proposed by Courchay *et al*. [64]. In their case, the mesh surface motion is recovered by minimizing an energy that used data costs due to stereo and flow (possibly over several time frames), and both spatial and temporal smoothness. Due to its complexity, the method is slow (taking 24 hours for some sequences).

Most of the above methods rely on a large number of cameras (e.g., more than 8 for portions of the object). Although temporal information is sometimes used to improve the geometry, the intra-camera observations are not fully exploited. In the case when there are only a few viewpoints, it is essential to utilize the intra-camera observations, for example, by using structure and motion on almost rigid surface parts within each viewpoint [250]. Ideally, the non-rigid dense surface structure and motion can be reconstructed from a single view, which is possible when the template geometry is known [201, 202], for example, by using a planar inextensible triangulated mesh [202]. Assumptions about maximizing rigidity have also been used to incrementally improve an approximate sparse geometry undergoing non-rigid motion [251]. Similarly, non-rigid structure and motion techniques [248, 39] are capable of recovering the structure and deformation model (as well as orthographic camera motion) given known correspondences in a single view. The deformation of the surface vertices is factored into a low rank motion basis that is modulated for each point (i.e., the shape component). This low rank motion basis can be extracted from reliable point tracks and can be used to extract long-range temporal motion of other (unreliable) points simply by finding the few coefficients that make up the shape component [247]. This empirically formed basis has also been used to reconstruct dense per-pixel surface motion using a variational formulation [99], where per-pixel shape coefficients (e.g., 2-3) are estimated with regularization for long sequences (e.g., 80 frames).

2.4.3 Discussion

Vision-based markerless motion capture is a viable alternative to the marker-based methods, with commercial systems already available (e.g., [176]). The primary advantage is that the subject can wear natural clothing, which is a requirement when the appearance of the subject (and clothing) are to be modeled.

Further, keeping a vision-based approach implies the acquisition of our model will rely on vision-based human geometry tracking. The above discussed cues, such as silhouette or appearance cues, and optimization methods are directly transferrable to the approach discused in Chapter 5. As many model-based methods are successful using simple local optimization, the focus on this thesis is a tracking framework that is both efficient, uses both silhouette and appearance cues, and operates on linear blend skinned meshes.

For recovering fine-scale displacements, or temporal correspondences, there are also many viable solutions. The main limitation of many dense temporal correspondence methods is that they only recover two-time correspondence. Methods for obtaining longer range correspondence often require a large number of viewpoints to obtain a dense geometry at each frame or simply provide a way to perform a parameter-free coarse tracking. Instead, in this thesis, the focus is on trying to obtain these dense correspondences by leveraging the coarse-scale joint angle tracking when fewer views are available.

2.5 Rendering & Appearance Modeling

In order to generate photo-realistic images of humans, some model of appearance is necessary to complement the geometry. In this section, the basic theory of reflectance in computer graphics is reviewed. Complicated reflectance or scene geometry can be tedious to model for some natural objects, such as hair, fur, and foliage, to name a few. Image-based approaches seek to replace some or all of the geometrical/reflectance model with images or information derived from images (see Figure 2.9). These image-based rendering (IBR) approaches are first reviewed for static scenes, and then the application of traditional and image-based methods is discussed for dynamic human scenes.

2.5.1 Static Scenes

The traditional graphics approach to generate high quality photo-realistic renderings requires a highly detailed geometry, an accurate model of surface reflectance, and a realistic model of lighting. When geometry and surface normals are available, the color of a surface point observed from a view direction is a function of the incoming light and the surface reflectance properties of the point [4]:

$$I_\lambda(\theta_v, \phi_v) = \int_{-\pi}^{\pi} \int_{0}^{\frac{\pi}{2}} f_\lambda(\theta_v, \phi_v, \theta_l, \phi_l) L_\lambda(\theta_l, \phi_l) \cos(\phi_l) \sin(\phi_l) d\phi_l d\theta_l \tag{2.11}$$

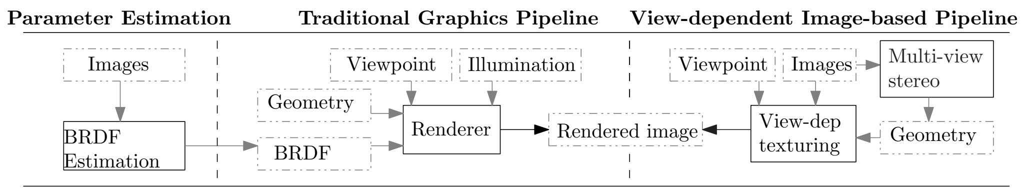


Figure 2.9: The traditional graphics pipeline requires viewpoint, illumination, surface geometry, and a model of reflectance to render an image. Parameterized models of reflectance (e.g., Phong) can be estimated through image-based methods and used as input to the traditional pipeline. On the other hand, an example image-based method uses input images directly to produce approximate geometries and then applies the input images as textures in a view-dependent manner to achieve realistic renderings.

where θ_v, ϕ_v represent the view/outgoing direction, and θ_l, ϕ_l is an incoming light direction, $L_\lambda(\theta_l, \phi_l)$ is the light arriving at the surface, and f_λ is the bidirectional reflectance distribution function (BRDF). The BRDF models the amount of light that is reflected from the incoming direction to the outgoing direction. The dependence on wavelength, λ, is usually approximated with a BRDF for the red, green, and blue channels.

The BRDF is often modeled with some parameterized model, such as the single parameter Lambertian model for a matte/diffuse surface where BRDF is a constant, or others with parameters to also model specular highlights, such as Phong [181], Lafortune [137], or Torrance-Sparrow [246]. Recovering the BRDFs of real objects has been studied in the literature [206, 200, 143, 161, 98, 13, 115, 204], and for a particular parameterized model the process often uses several images with varying illumination and view directions (both of which are calibrated) and requires an optimization over the parameters of the model that best produce the images under the same conditions. In the case of objects with spatially varying BRDF, some form of clustering by BRDF is useful to obtain more observations for each fitted BRDF [143]. In other words, the specular highlight on specular surfaces is not always observed on each surface point in enough images to reliably estimate the parameters independently for each point on the surface.

The reliance on controlled laboratory conditions often make these approaches impractical. To complement a model acquired from image-based methods, some approaches simply try to recover a single texture map. These methods acquire a lit version of the texture by choosing the best input image to texture a triangle. The projected area of a triangle in a viewpoint is often the criterion to select the best image used to texture a given triangle [27, 11, 141, 120]. The selection of image texture for each triangle can also be cast into a discrete labeling framework, where the objective is to choose a label for each triangle such that a global energy is minimized. Each triangle can be labeled one of N possible labels for N images. The energy tries to select labels that maximize the projected area of a triangle in an input image, while, a competing smoothness term, prefers neighboring triangles to have similar colors along their shared edge [11]. Finally, to avoid discontinuities across triangles, a multi-band blending approach can be used to blend different frequency bands [27, 11].

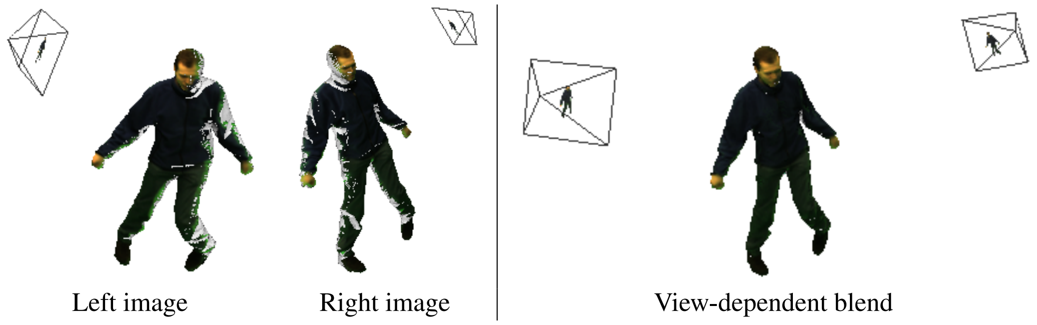


Figure 2.10: An example of view-dependent texture mapping with projective textures. Each camera, left and right, can texture portions of the object. The VDTM blend uses the camera information to blend between the two, allowing filling in of missing regions (rendered using crane data from MIT [259]).

In the multi-band blending approaches, a Laplacian pyramid is created with the lowest resolution a Gaussian blurred texture. When the input images are blended across triangles to create the single texture, the lower frequency bands are blended over a larger spatial extent, whereas higher frequency components are typically sampled from the best labeled image.

In many image-based rendering applications the surface geometry is either not known or imperfect, and a single texture is inadequate. Instead, the goal is to use multiple input images to re-render photo-realistic views of the scene from novel viewpoints. Each input image gives a sample of the so called plenoptic function [1], $g_\lambda(x, y, z, \phi, \theta, t)$, which relates a 3D ray (parameterized by position and angle) at a particular time, t, to a scene color. Image-based rendering can then be thought of as a sparse data interpolation problem. In the simplest case, a spherical panorama represents a full sample of the plenoptic function at a point in time, and transitions between multiple panoramas at different places in the scene can give a crude approximation of a 3D world [52].

For object-based capture, the 5D spatial plenoptic function has been sampled in a region of space using a 4D parameterization where a ray is parameterized by its coordinates in two intersecting parallel planes enclosing the scene [102, 146]. When a more accurate geometry is available, the image colors (and corresponding rays) can be mapped to directions on the local hemisphere on the surface of the object; new images can be reconstructed by computing which surface point a novel pixel observes and interpolating the sample colors for the corresponding direction. Such a representation is called the *surface light field* [274].This is similar to application of Eq. 2.11, but note that this is now only a function of view direction so illumination at the time of capture is coupled with the BRDF. The surface light field can again be parameterized in 4D: two angles for the view direction and two spatial coordinates for the parameterization of the surface [53].

Instead of representing the light field directly on the surface of the object, the View-Dependent Texture Mapping (VDTM) of Debevec *et al.* use a small set of input images with similar view directions as the desired viewpoint along with projective texture mapping to approximate the surface light field [73] (see Fig. 2.10). More compact representations of the input images have been demonstrated

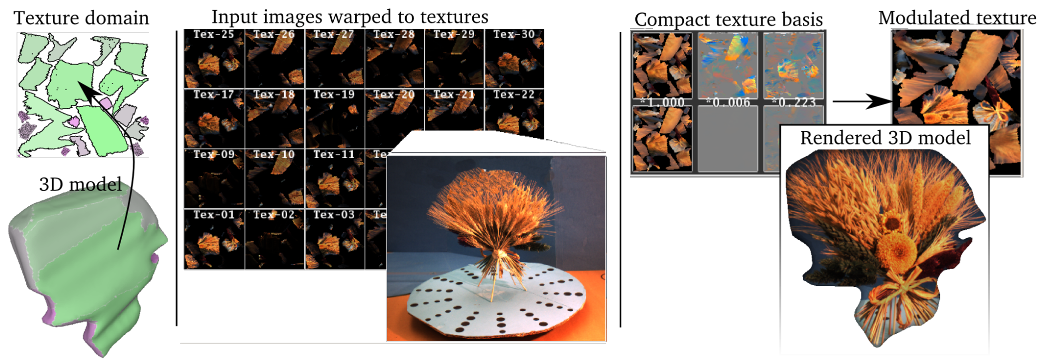


Figure 2.11: A VDTM can be represented more compactly through a linear projection when the model has a texture parameterization. Images are warped to the texture domain, the linear basis is extracted, and rendering simply modulates the basis with view-dependent coefficients [116].

when a uv-mapping of the geometry is available: input images are warped to the uv-space creating texture images by projection of the geometry onto the input images, a compact representation of the space spanned by the texture maps is obtained (e.g., with PCA), and sparse data interpolation is used to generate the correct texture map for a given view [61, 174] (see Fig. 2.11 for an example). The surface light field can be compressed in a similar manner, but the region of support for the PCA is performed individually in the 1-ring neighborhood of each vertex [53] instead of the entire texture domain [61].

The downside to the surface light field and VDTM methods is that they capture a lit model of the surface: re-rendering of the captured object will always appear under the same lighting conditions, and these conditions may not agree with the desired illumination in a novel re-rendering. It is possible to extend the above approaches by using input imagery that also samples illumination directions, and uses both illumination and view direction as interpolators of the input data [254].

Even when sampling the illumination direction, the image-based representations can contain artifacts resulting from semi-transparent or fine features that are poorly approximated by an opaque geometric proxy. Matusik *et al.* introduced the opacity hull as a method to overcome these deficiencies [162]. Multiple images of the object with varying background (provided by a programmable monitor) are used to extract an alpha matte for multiple input views. Their method is also endowed with various illumination conditions, but the entire capture requires an elaborate capture setup with programmable and manually activated components (e.g., turntable, backdrops, and lamps).

The above applications use image-based rendering to model view or light variation, but they can also be used to model temporal variation [208, 118]. For example, in *video textures*, image-based rendering can be used to transition and create infinite non-repeating loops of repetitive scenes like flowing water [208]. In one of the few methods that combines both appearance and geometry, James and Fatahalian model appearance and geometric variation that occurs as a function of simple world disturbances [118]. In their case, a PCA model is used to compress the appearance model, which mostly encodes light variation due to changes in the geometry.

2.5.2 Appearance of Dynamic Human Subjects

There exist a few image-based rendering methods capable of rendering novel viewpoints of dynamic scenes without the use of scene geometry or camera positions (e.g., [155, 276]). However, in many image-based modeling methods, and in this thesis, a 3D geometry and camera pose is available. In such cases, the view-dependent approaches discussed above can be simply extended in time.

For example, a common application for the vision-based tracking methods as well as the time varying model reconstructed sequences is to achieve a free-viewpoint video of the sequence [244, 230, 289, 164]. In such cases, a simple time ignorant application of view-dependent texture mapping or multi-texturing is common. At each time instant multi-view blending of the input images is used to achieve a photo-realistic rendering from new viewpoints, allowing free view navigation through the captured sequence during playback. Such methods do not require a consistent uv-parameterization, nor does the mesh need to be consistent over time and may be computed independently at each time step. A disadvantage is that no temporal consistency is exploited to utilize image information from previous or subsequent frames in the re-rendering process. Furthermore, such an approach requires the input images at any time step to be available when rendering, so the entire video sequence needs to be available (video compression is possible). Texturing in this manner (without a consistent uv-parameterization) does not easily generalize to sequences where a new motion is transferred to the model.

Through animated imposters, image-based methods have also been used as a method to speed up rendering of virtual humans in graphics worlds [15, 241]. With an existing graphics model, the space of viewpoint variation over an animation sequence can be rendered onto a single quadrilateral; novel renderings need only interpolate the stored imagery for a given animation-frame and view-direction. An elaborate real-world implementation by Chabert *et al.* uses an extensive image-based system for acquiring a similar representation of dynamic human appearance from several viewpoints [50]. In their system, a human walks on a treadmill that rotates on a turntable, while several cameras arranged in a vertical line capture video. During the sequence, a set of hemispherical basis light configurations are activated. The input data can be used to generate novel renderings of the human at different times in their captured sequence under novel viewpoint and lighting. The results are photo-realistic, but the data can not easily be re-used for different animations of the model.

Theobalt *et al.* propose a free-viewpoint approach that removes light variation from the captured sequence [243]. Their method recovers a dynamic human appearance model capable of re-illumination using a consistent surface parameterization, time-dependent surface normals, and a time-dependent uv-coordinate shift to account for cloth motion. In addition to the multiple cameras, the capture room also contains two calibrated lights. A subject is first captured in a reference pose, with arms out, rotating on the spot. A predefined geometry is fit to the sequence, and the image sequences are warped to uv-space. Misalignment of image-features due to inaccurate geometry is compensated for by correcting the camera image so that model features of a texel align with a projected image of the best camera (most frontal) on the model rendered in the view of the current camera (computed using optical flow between each pair of reprojected images). This idea of warping to adjust images for IBR due to inaccurate geometry and poor calibration has also been explored more recently in terms of static models in *floating textures* [80]. From this initialization sequence, a per texel Phong or Lafortune BRDF model is fit using the image samples and known illumination, where the specular component is constrained to be the same over clustered surface regions with similar color. A dynamic sequence of the subject in arbitrary motion is then tracked (using similar vision-based methods discussed in Section 2.4.2), and the multi-image camera streams are warped to uv-space, where optical flow is again used to compute texture flow from the reference sequence—this 2D flow represents cloth motion over time. Finally, this dynamic sequence is partitioned into consecutive frames over which the surface normal for each texel in the chunk is refined to agree with the image sequences—where color samples take into account the cloth deformation parameters. The complete model allows for novel relighting of the dynamic sequence under novel viewpoint. Rendering ensures that each surface point uses the dynamic normal map and indexes the BRDF parameters using the cloth offset. The authors do not attempt to model the deformations (normal and cloth offsets) as a function of abstract interpolators such as joint angles, implying that the dynamic component of the model cannot be transfered to new animations.

The above applications of image-based appearance modeling considered modeling static or dynamic scenes where variation was either due to light, view direction, or time. Similar appearance models can also be used to model variation across subjects. For example, in the Active Appearance Model (AAM) [62], a statistical 2D model of shape variation is combined with a low-dimensional linear appearance basis to model variation across subjects. The model is often used to model faces for detection, recognition, or tracking applications. The 3D extension to AAM is the Morphable Model (MM) [183, 182, 34]. The 3D MMs is also often used for facial modeling, and can be used to model pose/expression variation within a subject [182] or variation across subjects [183]. The approach of Pighin *et al.* [183], even considers view- and expression-dependent renderings for MMs, as is done in this thesis, but their model is for non-kinematic geometries.

2.5.3 Discussion

The above review has illustrated how appearance is often modeled as a view-dependent interpolation problem, where the appearance is view-dependent due to underlying surface reflectance or due to incorrect geometry. The image-based view-dependent rendering approach has been used in a variety of contexts, including exploring captured photos of static scenes, such as Microsoft's PhotoSynth [223], and for visualizing real-world dynamic scenes captured by uncoordinated hand-held video [21].

In this thesis, such view-dependent modeling of appearance is essential. In addition to using a compact linear basis for modeling appearance variation due to view direction, appearance changes due to joint motion are modeled in a similar way. The motivation for using appearance in this way is analogous to its use for modeling view variation. During an image-based capture it is unlikely that fine-scale cloth deformations will always be modeled correctly in the geometry. To ensure photo-realistic renderings, such residual image variation will be modeled in the texture.

Unlike the existing methods, such as free-viewpoint video, which can only play back a pre-captured sequence, our appearance model can then be transfered to new motion sequences and rendered under new view conditions. This makes the model more practical for graphical applications where the animation is driven by, for example, user interaction.

For a finite camera observing a subject, the view-direction will vary continuously over the surface. In order to account for this view variation, the proposed compact model decomposes the object geometry into overlapping parts that are attached to different kinematic bones. The compact model of appearance then uses these parts as regions of support in the PCA modeling. This approach allows the view-variation to change over each part and strikes a balance between the fine-scale support region used to represent surface light fields (e.g., [53]) and the coarse-scale support region used in dynamic textures (e.g., [61]).

2.6 Summary

As identified above, this thesis has components that overlap several fields in vision and graphics research. The end goal is to recover a combined model of dynamic appearance and geometrical deformation of human motion from images.

The example-based methods have been recently exploited as an efficient alternative to physical simulations for modeling pose-dependent cloth movement. Unfortunately, the input is typically obtained from textured cloth, simulated data, or artist edited data. In order to model appearance, view-dependent methods are often a requirement to achieve realistic renderings from sparse multi-view studios [227], but the acquisition methods typically produce a model that can only play back the captured motion. In order to model dynamic appearance and geometric deformation using convenient image-based capture, the proposed model adapts components from the graphics literature for modeling geometric deformation, specifically the example-based methods, and combines them with the image-based rendering approaches for modeling appearance. Furthermore, this thesis also address issues of efficiently representing appearance on an articulated skeleton.

In the vision-based acquisition, multi-view cues for both reconstruction and tracking are necessary. For the static model, the use of turntables is exploited to acquire a human model during an initialization phase. Turntables are common for object capture, and have been used for human geometries [57, 243]. In Chapter 4 the focus is on obtaining a decent model using only a few cameras.

There are already many multi-view systems in place for tracking (e.g., [96, 245]) and acquiring deformations of human performances, many of which acquire a deforming surface over time (e.g., [245, 235, 95, 227]). Most of these focus on playback of the same motion under similar illumination, while some generalize the acquired data to novel illumination (e.g., [243]) and novel animation (e.g., [235, 203]). The pipeline presented in this thesis is similar to these works for recovering the dynamic motion, although we try to utilize the coarse joint motion to enable the dense reconstruction.

3 A Compact Model of Human Deformation and Appearance

In this chapter, a compact model capable of representing pose-dependent geometric variation and pose- and view-dependent appearance variation of kinematic objects (including humans) from images is proposed (§3.1). Unlike existing approaches that consider modeling both appearance and geometric variation (e.g., [62, 183, 182, 34]), the presented model builds upon dynamic methods for representing articulated human motion (§2.3.2), and it uses a base geometry coupled to a kinematic hierarchy with a linear displacement basis to model surface displacements. The appearance component leverages the image-based rendering techniques (e.g., [61, 174]) to model two types of variation: residual pose-dependent variation not represented in the geometry, and appearance variation due to view-dependent surface reflectance or inaccurate geometry. The pose- and view-dependent components, modeled separately over regions of the object, give a model that is compact, richly expressive, efficient to render, and targeted to image-based acquisition.

This combined model is designed to represent appearance and geometric deformations exhibited in video sequences. The details on reconstructing an instance of this model from a training multi-view video sequence (given geometric correspondences and kinematic joint angles) are also presented in this Chapter (§3.2). The reconstructed model can then be used to synthesize images with the following properties: 1) the renderings exhibit the geometric and appearance deformations present in the input sequence, and 2) the images can be rendered under novel pose and observed from a novel viewpoint. The synthesis of novel images is treated as a sparse data interpolation problem. Specifically, input geometric deformations and appearance changes are interpolated as a function of both pose and viewpoint (§3.3).

3.1 Model Components

In this section, the components of the proposed model are described, and the mathematical notations used for these components are introduced. The complete pose- and view-dependent model is a heterogeneous coarse to fine representation with three components (see Figure 3.1). The coarsest component, a base geometry and coupled kinematic skeleton (§3.1.1), models the gross kinematic deformation and provides an approximate proxy for rendering. This coarsest component includes a decomposition of the mesh into roughly rigid parts that are used by the fine-scale geometric and appearance components. The second component, a pose-dependent geometry (§3.1.2), models surface

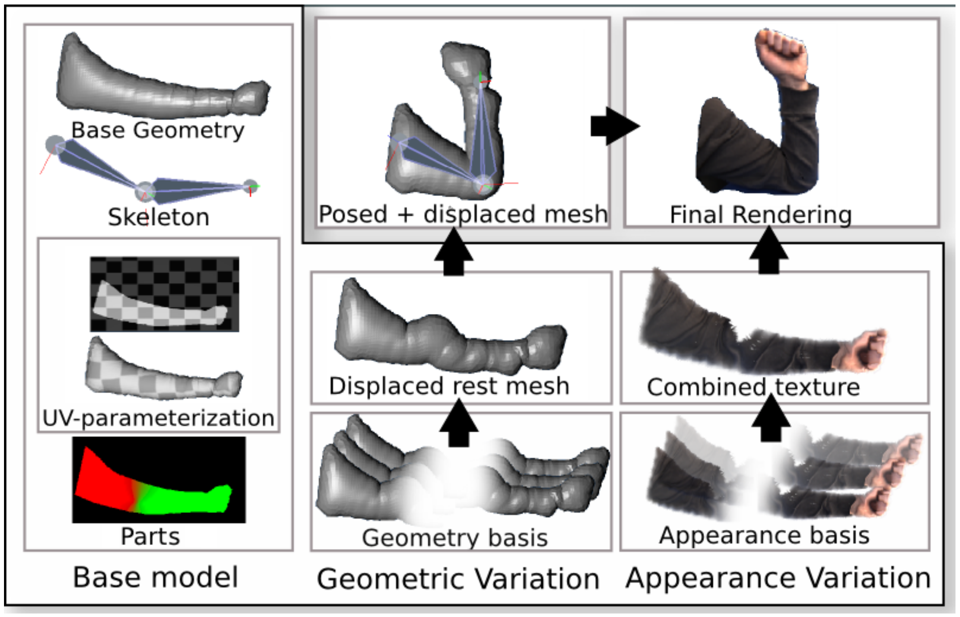


Figure 3.1: The base model consists of a skeleton, a base geometry with uv-coordinates, and a set of mesh parts. Each part has its own linear basis for geometry and appearance. Rendering for a novel pose and view direction is done by finding the appropriate modulation coefficients for the geometry and appearance.

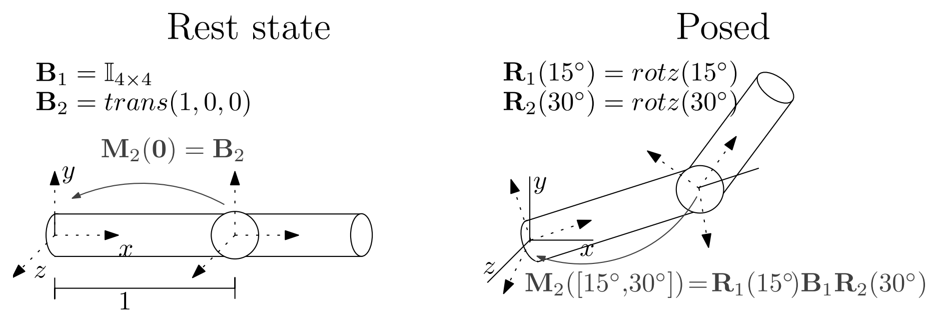
geometric deformation on top of the base geometry through a low dimensional linear basis. The finest pixel-scale component is a pose- and view-dependent appearance representation that models the surface appearance at a pixel level (§3.1.3). This pixel-scale component is responsible for modeling both view-dependent effects and residual geometric motion due to approximately reconstructed surface deformations using a low-dimensional texture basis. The expressiveness of the appearance and geometric components of the combined model is determined by the low dimensional linear basis used to represent these components (§3.1.4).

3.1.1 Base Model

The base model consists of a triangulated mesh and an articulated skeleton to which this mesh is attached. The geometry, $G = (\{\hat{\mathbf{v}}_j\}_{i=1}^{\#\text{points}}, \{w_{j,i}\}, \{\triangle_k\})$, consists of a set of vertices in rest pose, $\{\hat{\mathbf{v}}_j\}$, the corresponding skinning weights, $\{w_{j,i}\}$, and a set of triangles that index the vertices, $\{\triangle_k\}$, where $\triangle_k \in (\mathbb{Z}^+)^3$. Each triangle has corresponding texture coordinates, giving a uv-parameterization of the mesh. The uv-parameterization, $f_{uv} : (\Omega \subset \mathbb{R}^2) \mapsto G$, is a bijective mapping from the 2D texture domain, Ω, to the surface of the 3D geometry.

Kinematic Skeleton

The kinematic skeleton is represented as a tree hierarchy of kinematic links, where each link i is positioned relative to its parent, $P(i)$, with a homogeneous transformation matrix $\mathbf{B}_i$. Each link also has a set of rotational (and possibly translational) freedoms, which are represented as transformation



Rest state
$\mathbf{B}_1 = \mathbb{I}_{4\times 4}$
$\mathbf{B}_2 = trans(1,0,0)$
$\mathbf{M}_2(\mathbf{0}) = \mathbf{B}_2$
y
x
z
1
Posed
$\mathbf{R}_1(15°) = rotz(15°)$
$\mathbf{R}_2(30°) = rotz(30°)$
y
x
z
$\mathbf{M}_2([15°,30°]) = \mathbf{R}_1(15°)\mathbf{B}_1\mathbf{R}_2(30°)$

Figure 3.2: An illustration of the notation for the kinematic skeleton. Here $rotz$ refers to a rotation around the z-axis and $trans$ is a translation matrix.

matrices, $\mathbf{R}_i(\boldsymbol{\theta}_i)$, $\boldsymbol{\theta}_i = [\theta_{i,1}, \theta_{i,2}, ..., \theta_{i,n_i}]$, where n_i is the number of freedoms in the transformation matrix, e.g., 3 for a ball and socket joint, and 1 for a hinge joint. See Figure 3.2 for a simple example. The transformation from the coordinates of a link to the root is recursively defined as

$$\mathbf{M}_i(\boldsymbol{\Theta}) = \mathbf{M}_{P(i)}(\boldsymbol{\Theta})\mathbf{B}_i\mathbf{R}_i(\boldsymbol{\theta}_i). \tag{3.1}$$

The shorthand $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ...]$ refers to all of the parameters that affect the skeleton, and $|\boldsymbol{\Theta}|$ refers to the total number of parameters. Each link is treated as a function of $\boldsymbol{\Theta}$; in practice its transformation is only affected by $\boldsymbol{\theta}_i$ and the parameters that affect any ancestor. The root link has no parent, meaning $\mathbf{M}_1(\boldsymbol{\Theta}) = \mathbf{B}_1\mathbf{R}_1(\boldsymbol{\theta}_1)$.

The base mesh is attached to the skeleton using linear blend skinning. For a particular skeleton, the mesh vertices move with linear combinations of attachments to individual links:

$$\mathbf{v}_j(\boldsymbol{\Theta}) = \sum_{i=1}^{|\boldsymbol{\Theta}|} w_{j,i}\mathbf{T}_i(\boldsymbol{\Theta})(\hat{\mathbf{v}}_j). \tag{3.2}$$

The transformation $\mathbf{T}_i(\boldsymbol{\Theta}) = \mathbf{M}_i(\boldsymbol{\Theta})\mathbf{M}_i(\mathbf{0})^{-1}$ is the relative transformation of link i under animation by the parameters $\boldsymbol{\Theta}$. Specifically, $\mathbf{M}_i(\mathbf{0})$ is the transformation from link i's coordinates to the root coordinates in rest pose, and $\mathbf{M}_i(\boldsymbol{\Theta})$ is the transformation when animated with parameters $\boldsymbol{\Theta}$.

The weights, $0 \leq w_{j,i} \leq 1$, determine the attachment of a vertex to the given link. Vertices are typically only attached to neighboring nearby joints, meaning most of $w_{j,i}$ are zero, and in practice a list of influencing bones and the corresponding weights are retained.

Mesh Parts

The final component of the base model is a set of possibly overlapping pieces or parts of the mesh. These are similar to the vertex supports of the eigen-skin [134] and the partitions used in morphable models [34]. Different regions of a model are influenced by different animation parameters (e.g., the angles of the left elbow only influence deformations in the left upper-arm and left forearm). To allow for this local region of influence, it is assumed that the model is separated into regions that are influenced by similar parameters.

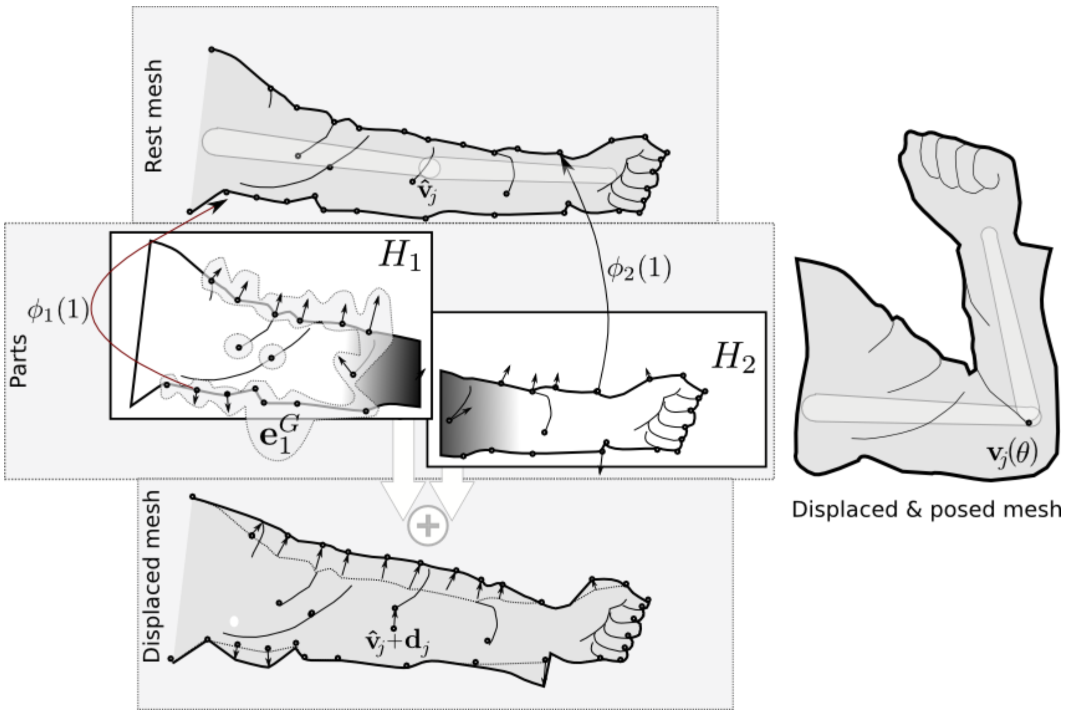


Figure 3.3: A two-link arm used to illustrate the notation. The arm has two parts, which overlap and have smooth weights. The displacements are modeled as pose-dependent for each part independently and combined to get a displaced mesh in a rest pose. The resulting mesh is then posed.

In order to blend at render time (discussed in §3.1.4), these mesh parts are allowed to overlap, with weights controlling the relative attachment of vertices to each region. For articulated humans, the mesh parts and these weights can be inferred from the skinning weights or they can be defined manually. Specifically, let $H_i = \{\phi_i(j) | j \in \{1, ..., |H_i|\}\}$ be a list of vertex indices for mesh part i, where $\phi_i(j)$ maps increasing indices onto the global vertex index (see Figure 3.3). Let $0 \leq \alpha_{i,j}^G \leq 1$ be the corresponding attachment of $\mathbf{v}_j$ to the part.

3.1.2 Pose-dependent Geometry

Given the above base model, adding pose-dependent variation to the skinned model is straightforward. Following the pose-space deformation approach [148], geometric variation is introduced in the rest/non-animated pose:

$$\mathbf{v}_j(\mathbf{\Theta}) = \sum_{i=1}^{|\mathbf{\Theta}|} w_{j,i} \mathbf{T}_i(\mathbf{\Theta})(\hat{\mathbf{v}}_j + \mathbf{d}_j). \tag{3.3}$$

The vertex displacements, $\mathbf{d}_j$, are assumed to be dependent on pose (although they could be arbitrary). As in the eigen-skin approach [134], the collection of displacements within a part are assumed to be well approximated by a linear subspace: for part, i, the concatenated displacements

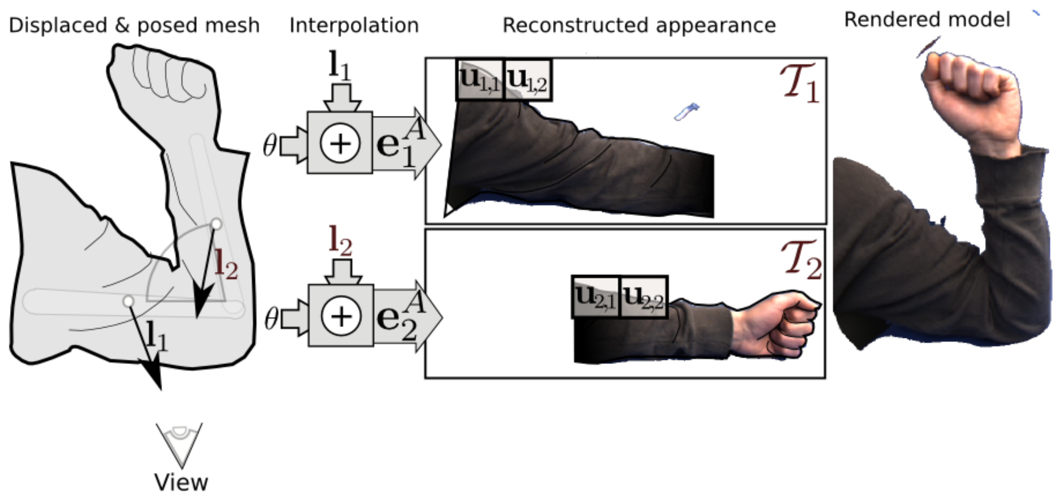


Figure 3.4: When texturing the two-link arm, the texture parts are created independently after interpolating modulation coefficients based on pose and view. The resulting textures are combined and applied as a texture map to get the final rendering

can be represented as

$$\mathbf{e}_i^G = [\mathbf{d}_{\phi_i(1)}^\intercal, \mathbf{d}_{\phi_i(2)}^\intercal, ...]^\intercal \simeq \mathbf{W}_i^G \mathbf{h}_i^G(\mathbf{\Theta}) + \bar{\mathbf{e}}_i^G. \tag{3.4}$$

The matrix $\mathbf{W}_i^G$ is the linear basis, $\mathbf{h}_i^G(\mathbf{\Theta})$ is a set of pose-dependent modulation coefficients, and $\bar{\mathbf{e}}_i^G$ is the mean displacement.

The per-vertex, final displacement is generated as a combination of the displacements of each of the parts to which it belongs, weighted by its attachment to that part:

$$\mathbf{d}_j = \sum_{i|j\in H_i} \alpha_{i,j}^G \mathbf{d}_{\phi_i(\phi_i^{-1}(j))}. \tag{3.5}$$

3.1.3 Pose- and View-dependent Appearance

The appearance is represented in the texture domain. For any surface point, $\mathbf{x} \in G$, the surface color, $R(\mathbf{x})$, is represented through a 2D texture image, $\mathcal{T}$. The texture image is a 2D array of square *tex*ture *el*ements (*texels*), each of which has a color value. The surface colors map to the texture image through the surface parameterization:

$$R(\mathbf{x}) = \mathcal{T}(f_{uv}^{-1}(\mathbf{x})). \tag{3.6}$$

The variation of appearance in the texture domain is modeled in an analogous manner to the geometric variation. Let $\mathbf{U}_i = [\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, ...]$ be the set of 2D texels that are influenced by part i. This is the set of texels, $\mathbf{u}$, such that $f_{uv}(\mathbf{u}) \in H_i$. Also, let, $\alpha_i^A(\mathbf{u})$, be the corresponding per-texel part weights obtained by linearly interpolating the vertex part weights. The texture colors from this part, $\mathcal{T}_i(\mathbf{u}_{i,j})$, are assumed to be generated from a low dimensional linear basis:

$$\mathbf{e}_i^A = [\mathcal{T}_i(\mathbf{u}_{i,1}), \mathcal{T}_i(\mathbf{u}_{i,2}), ...]^\intercal \simeq \mathbf{W}_i^A \mathbf{h}_i^A(\mathbf{\Theta}, \mathbf{l}_i) + \bar{\mathbf{e}}_i^A. \tag{3.7}$$

Where $\mathbf{W}_i^A$ is the appearance basis, $\bar{\mathbf{e}}_i^A$, is the mean texture, and $\mathbf{h}_i^A(\mathbf{\Theta}, \mathbf{l}_i)$ are the modulation coefficients or latent variables. These latent variables, are obtained from an interpolation process that takes into account both the influencing interpolating parameters (a subset of $\mathbf{\Theta}$) and also the view direction, $\mathbf{l}_i$ (§ 3.3). In this way, the appearance can model both pose-dependent variation (e.g., cloth shifting) and the view-dependency commonly handled by image-based rendering methods. Furthermore, as the appearance of an entire patch shares common modulation coefficients, the use of multiple patches on a kinematic object allows each patch to use different relative view directions (see Figure 3.4).

Finally, the combined appearance is approximated by a weighted combination of these components (using the per-texel part weights):

$$\mathcal{T}(\mathbf{u}) = \sum_{i|\mathbf{u}\in\mathbf{U}_i} \alpha_i^A(\mathbf{u}) \mathcal{T}_i(\mathbf{u}), \tag{3.8}$$

where $\sum_i \alpha_i^A(\mathbf{u}) = 1, \forall \mathbf{u}$.

3.1.4 Geometry and Appearance Synthesis

Rendering from a particular viewpoint and pose is straightforward. It is assumed that the modulation coefficients ($\mathbf{h}_i^G$ and $\mathbf{h}_i^A$) are related to the influencing parameters for each part (in our case by sparse data interpolation, discussed in Section 3.3). Displacements for each part are computed (Eq. 3.4) and then combined (Eq. 3.5). The appearance (or texture image) is generated in an analogous way (using Eq. 3.7 & 3.8), although the sparse data interpolation takes the viewpoint into account. The displaced geometry then undergoes the skinning transformation (Eq. 3.3), and the final model is rendered using traditional texture mapping.

3.2 Reconstruction from Images

The proposed model uses low-dimensional linear bases to model geometric and appearance variation. In practice, there is no analytical form of these low-dimensional subspaces; they must be estimated empirically from visual observations of the target. The empirical bases are extracted from a sequence of input images, $\{I_{i,t}\}_{t=1}^T$, from multiple cameras, $1 \leq i \leq C$, that observe the target object. It is important that the sequences contain motions of the target that are desirable to reproduce. For example, if it is desirable to represent clothing wrinkles on an arm as a function of the elbow, the input sequences should contain poses of the elbow bending.

Each camera is static and has known 3×4 calibration matrices:

$$\mathbf{P}_i = [\mathbf{K}_i|\mathbf{0}]\mathbf{E}_i. \tag{3.9}$$

Assume that the triangulated base-mesh with attached skeleton are given, that coarse joint angles, $\{\mathbf{\Theta}_t\}_{t=1}^T$, have been recovered, and that the dense geometric correspondences on the target object through the sequence are given (e.g., $\{\mathbf{v}_{j,t}\}_{t=1}^T$, where $\mathbf{v}_{j,t} = \mathbf{v}_j(\mathbf{\Theta}_t)$). Methods to obtain these inputs directly from the input video sequences are discussed in detail in Chapters 4, 5, & 6.

Given the displacements and corresponding joint angles, it is straightforward to extract the low dimensional geometric subspaces. The displacements are extracted by mapping $\mathbf{v}_{j,t}$ through the inverse of Eq. 3.3. For each part, the displacements for the associated vertices at each time t are extracted:

$$\mathbf{e}_{i,t}^G = [\mathbf{d}_{\phi_i(1),t}^\intercal, \mathbf{d}_{\phi_i(2),t}^\intercal, ...]^\intercal. \tag{3.10}$$

The basis, $\mathbf{W}_i^G$, and mean, $\bar{\mathbf{e}}_i^G$, are computed from the set of all observations by performing a PCA:

$$\mathbf{W}_i^G, \bar{\mathbf{e}}_i^G = PCA(\{\mathbf{e}_{i,t}^G\}_{t=1}^T). \tag{3.11}$$

Only the most important basis vectors, i.e., those describing most variation, are kept. Typically, the number of basis vectors used will be chosen such that they represent a large amount (e.g., 95%) of the variation in the training sequence. The projection of the input data for the low dimensional basis,

$$\hat{\mathbf{h}}_{i,t}^G = (\mathbf{W}_i^G)^\dagger(\mathbf{e}_{i,t}^G - \bar{\mathbf{e}}_i^G), \tag{3.12}$$

are used in the interpolation process (defined in § 3.3). Here † denotes the pseudo-inverse.[1]

The procedure is similar for the appearance. First, the texture image is computed for each input image,

$$\mathcal{T}_{j,t}(x,y) = (I_{j,t} \circ \mathcal{W}_{j,t})(x,y), \tag{3.13}$$

where $\mathcal{W}_{j,t} : \mathbb{R}^2 \mapsto \mathbb{R}^2$ is the piecewise linear texture warp that maps a texel location to its corresponding input image location using the mesh triangles and correspondences between the uv-coordinates and the projection of the vertices in image j. The texture warp takes occlusion into account and fills occluded areas with a weighted mean value. The weight for a pixel observation is zero for occluded pixels, otherwise it is proportional to the dot product of the surface normal and the view vector.

Finally, for each mesh part, the texels that are influenced by part i are extracted from each camera j at each time t to form an observation vector

$$\mathbf{e}_{i,j,t}^A = [\mathcal{T}_{j,t}(\mathbf{u}_{i,1}), \mathcal{T}_{j,t}(\mathbf{u}_{i,2}), ...]^\intercal, \tag{3.14}$$

and PCA is performed on the set of all observations giving:

$$\mathbf{W}_i^A, \bar{\mathbf{e}}_i^A = PCA(\{\mathbf{e}_{i,j,t}^A : 1 \leq t \leq T, 1 \leq j \leq C\}), \tag{3.15}$$

$$\hat{\mathbf{h}}_{i,j,t}^A = (\mathbf{W}_i^A)^\dagger(\mathbf{e}_{i,j,t}^A - \bar{\mathbf{e}}_i^A). \tag{3.16}$$

Again, only the first few salient basis vectors that explain a large amount of texture variation are retained.

[1]The columns of $\mathbf{W}_i^G$ are orthonormal, so the pseudo-inverse is the transpose.

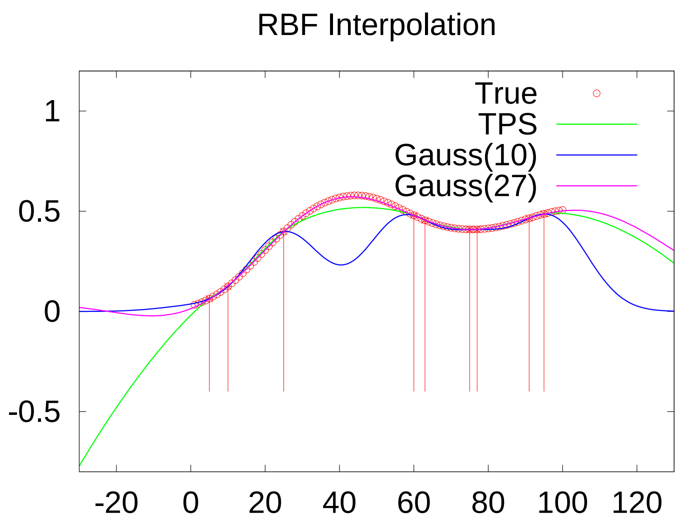


Figure 3.5: The choice of radial basis function interpolant is important. The true function is approximated with 9 sparse samples (vertical red lines). Local Gaussian kernels require that the σ be chosen. If the samples are not evenly spaced, $\sigma = 10$ (Gauss(10)), causes the reconstructed function to have dips (e.g., at $x = 40$). This is solved by a bigger $\sigma = 27$ that was fit through leave-one-out cross-validation. A global RBF, like a TPS, has no parameters but results in poor extrapolation (see $x < 0$).

3.3 Pose and View Interpolators

It is possible to synthesize a model, complete with deformations and appearance, simply by providing a set of modulation coefficients for each part. However, arbitrary coefficients will not typically give a plausible result. During rendering, the deformations and appearance need to be modeled as a function of the kinematic pose or the view direction. In other words, the influencing parameters used to synthesize the model need to be a subset of the pose and view direction. Sparse data interpolation is used to obtain a mapping from these influencing parameters to a corresponding coefficient vector, which in turn allows rendering. This process is simpler for geometric displacements as they are only a function of pose (§3.3.1), whereas appearance is a function of both pose and view (§3.3.2).

3.3.1 Synthesizing Geometry Displacements

Synthesizing a new geometry given a particular pose is similar to the methods used by the example-driven geometries from the graphics literature [148, 221, 203]. Assume that a part, i, is influenced by parameters $\mathbf{p}_i \subseteq \mathbf{\Theta}$. From the input video sequence, several correspondences between known parameters $\hat{\mathbf{p}}_{i,t}$ and their corresponding coefficient vectors $\hat{\mathbf{h}}^G_{i,t}$ are given. The objective is to find a mapping from any $\mathbf{p}_i$ to a $\mathbf{h}^G_i$ coefficient vector. The interpolation can be represented generically as

$$\mathbf{h}^G_i(\mathbf{p}_i) = g_i(\{(\hat{\mathbf{p}}^G_{i,t}, \hat{\mathbf{h}}^G_{i,t})\}^T_{t=1}, \mathbf{p}_i), \tag{3.17}$$

where $\hat{\mathbf{p}}^G_{i,t}$ are the corresponding influencing poses for the coefficient vector, $\hat{\mathbf{h}}^G_{i,t}$. For brevity, the first argument (the input for the interpolation) will be dropped when the context is clear.

In the simplest case, g_i, can be defined as a nearest neighbor interpolation:

$$g_i(\mathbf{p}_i) = \hat{\mathbf{h}}^G_{i,t}, \tag{3.18}$$

where $\hat{\mathbf{h}}^G_{i,t}$ corresponds to $\hat{\mathbf{p}}_{i,t} = \underset{\hat{\mathbf{p}}_{i,t'}}{\operatorname{argmin}}(\hat{\mathbf{p}}_{i,t'} - \mathbf{p})^2$.

However, a nearest-neighbor interpolation will not smoothly transition between poses. A common alternative in the example-based literature is to use radial basis functions:

$$g_i(\mathbf{p}_i) = \sum_{t=1}^{T} \lambda_{i,t} f_i(d(\mathbf{p}_i, \hat{\mathbf{p}}_{i,t}))\hat{\mathbf{h}}^G_{i,t}, \tag{3.19}$$

where $f_i(r)$ is a scalar RBF function. Common choices include a Gaussian kernel, $f_i(r) = \exp\left(-\frac{r^2}{2\sigma_i}\right)$, or a thin-plate spline (TPS) kernel, $f_i(r) = r^2 \log(r)$.

The function d computes the distance in parameter space. For a single angle, $d_\theta(a, b) = |a - b|$. For 3D rotations represented with quaternions, the distance can be taken as $d(\mathbf{a}, \mathbf{b}) = \cos^{-1}(\langle \mathbf{a}, \mathbf{b} \rangle)$, where $\langle ., . \rangle$ is the inner product. If $\mathbf{p}_i$ contains more than one joint, the distance can be defined as the sum of the pairwise distances between all of the joints.

The weighting terms, $\lambda_{i,t}$, are obtained by solving the system of equations:

$$\{\lambda_{i,t}\} = \underset{\{\lambda_{i,t'}\}}{\operatorname{argmin}} \sum_{t=1}^{T} |\hat{\mathbf{h}}^G_{i,t} - \sum_{t'=1}^{T} \lambda_{i,t'} f_i(d(\hat{\mathbf{p}}_{i,t}, \hat{\mathbf{p}}_{i,t'}))|^2. \tag{3.20}$$

For each mesh part, if the RBF has parameters (e.g., a Gaussian), they can be estimated from the spacing of the data ($\hat{\mathbf{h}}^G_{i,t}$) or by optimizing a leave-one-out cross-validation score.

The RBF is a general interpolation scheme for arbitrary dimension. However, the choice of interpolant (and parameters) is important and can cause problems in both interpolation and extrapolation (see Fig. 3.5). Linear interpolation works well in practice, but can be more complicated in higher dimensions due to the need to triangulate or tetrahedralize the high dimensional input. To overcome this issue, it is useful to perform a linear projection of the input interpolation parameters, $\mathbf{p}_i$. For example, although the joints are represented as quaternions, many joints (e.g., knee and elbow) are essentially hinges and can be projected into 1D through PCA. For the parts attached to these joints, the interpolation uses the projected $\mathbf{p}_i$ as input.

3.3.2 Synthesizing Appearance

View-dependent Appearance

The appearance is in general a function of both pose and viewpoint. For simplicity, first, lets recall simple view-dependent texture interpolation. One way to perform view-dependent interpolation for a given a view-direction, $\mathbf{l}_{\text{query}}$, would be to select and blend the coefficients from the closest n-views in the input. The contribution from each of these selected views could be weighted depending on the distance to $\mathbf{l}_{\text{query}}$. As desired, when $\mathbf{l}_{\text{query}}$ is close to an input view-direction, the texture would closely approximate the texture from the corresponding view.

For a dense sampling of views, the 3D view direction should first be parameterized with 2D spherical coordinates. In the case that the input views lie on a circular arrangement, the view interpolation is easier to perform by first mapping the 3D view direction to a 1D circle. The interpolation method can again be linear or based on RBFs in this mapped domain.

Pose and View-dependent Appearance

In general, the appearance interpolation has to take into account both pose and view. The parameters that influence the appearance are: $\mathbf{p}_i^A = [\mathbf{p}_i, \mathbf{l}_i]$, where $\mathbf{l}_i$ is the view direction relative to the part. RBFs can be used to interpolate based on these sample vectors in the same way as above. The distance,

$$d([\mathbf{a}, \mathbf{l}_a], [\mathbf{b}, \mathbf{l}_b]) = \alpha_{\text{pose}} d(\mathbf{a}, \mathbf{b}) + d(\mathbf{l}_a, \mathbf{l}_b), \tag{3.21}$$

uses the weight, α_{pose}, to control the influence that pose has on the appearance.

Prioritized Pose then View Interpolation

This weighting term, α_{pose} gives a way of prioritizing either pose or viewpoint in the appearance interpolation. However, when the cameras are arranged in either a circle or spherical pattern, it is easier to reason about the interpolation of pose and view separately. Ideally, to give priority to pose-based appearance, one intuitive approach would be to compute a set of coefficients based on closeness determined by pose. This would be done for each distinct input view (in space, not in time) independently. These coefficients could then be interpolated using a view-dependent weighting that uses the camera configuration (e.g., circular, linear, or dense) to choose the appropriate view interpolation. This sequential interpolation is represented as an interpolation of appearance coefficients for each view as a function of pose, followed by later interpolation by view:

$$\mathbf{h}_i^A(\mathbf{p}_i, \mathbf{l}_i) = g_i^A(\{(\hat{\mathbf{l}}_{i,j}(\mathbf{p}_i), \hat{g}_{i,j}^A(\mathbf{p}_i))\}_{j=1}^C, \mathbf{l}_i). \tag{3.22}$$

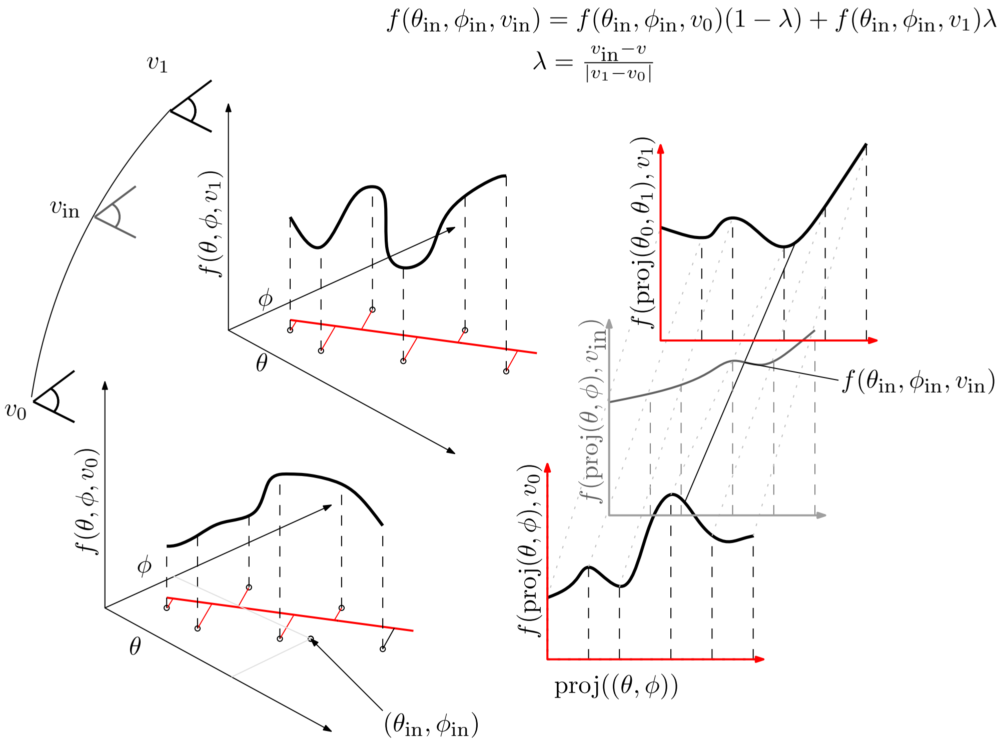
The notation above has been simplified. To be explicit, $\hat{g}_{i,j}^A(\mathbf{p}_i)$, interpolates all the texture coefficients for view j:

$$\hat{g}_{i,j}^A(\mathbf{p}_i) = \hat{g}_{i,j}^A(\{(\hat{\mathbf{p}}_{i,t}, \hat{\mathbf{h}}_{i,j,t}^A)\}_{t=1}^T, \mathbf{p}_i). \tag{3.23}$$

Also, the direction of part i to the view changes as a function of pose (as this view direction is relative from the part to the camera):

$$\hat{\mathbf{l}}_{i,j}(\mathbf{p}_i) = \hat{\mathbf{l}}_{i,j}(\{(\hat{\mathbf{p}}_{i,t}, \mathbf{l}_{i,j,t})\}_{t=1}^T, \mathbf{p}_i). \tag{3.24}$$

This sequential prioritized interpolation allows for pose and view interpolation to be done independently. For example, the interpolation over pose can be performed by projecting the influencing



$f(\theta_{\text{in}}, \phi_{\text{in}}, v_{\text{in}}) = f(\theta_{\text{in}}, \phi_{\text{in}}, v_0)(1-\lambda) + f(\theta_{\text{in}}, \phi_{\text{in}}, v_1)\lambda$
$\lambda = \frac{v_{\text{in}} - v}{|v_1 - v_0|}$
v_1
v_{in}
v_0
$f(\theta, \phi, v_1)$
ϕ
θ
$f(\theta, \phi, v_0)$
ϕ
θ
$(\theta_{\text{in}}, \phi_{\text{in}})$
$f(\text{proj}(\theta_0, \theta_1), v_1)$
$f(\text{proj}(\theta, \phi), v_{\text{in}})$
$f(\theta_{\text{in}}, \phi_{\text{in}}, v_{\text{in}})$
$f(\text{proj}(\theta, \phi), v_0)$
$\text{proj}((\theta, \phi))$

Figure 3.6: An example of pose and view interpolation of a function $f(\theta, \phi, v)$, which is a function of two pose angles (θ, ϕ) and one view direction. For each discrete viewpoint, v_0 and v_1, the function value is observed at several values of (θ, ϕ). As the sampled poses are approximately 1D, the pose interpolation can be performed in a 1D space by projecting the angles to a 1D value (e.g., $\text{proj}(\theta, \phi)$). In the 1D space, the interpolation is performed for each view independently (e.g., to approximate $f(\theta, \phi, v_0)$ and $f(\theta, \phi, v_1)$). These results are then interpolated by view direction to obtain an approximation to $f(\theta_{\text{in}}, \phi_{\text{in}}, v_{\text{in}})$.

pose into a lower dimensional subspace, and then by performing RBF interpolation. The view interpolation can then be performed using, e.g., linear interpolation on the spherical coordinates of the view (see Figure 3.6).

In practice, linear interpolation is sufficient and has one strong advantage over other interpolators: linear interpolation ensures the interpolated coefficient values lie in the convex hull of the input values. As the low-dimensional appearance basis can represent a wide range of effects outside of the appearance in the input, this constraint is important to keep appearance realistic (Figure 3.7).

In the case that the interpolation over both pose and view is linear, Eq. 3.22, reduces to a bilinear interpolation over the vector valued pose and viewpoint inputs.

3.4 Discussion

In this chapter, a compact model of pose-dependent geometry and pose- and view-dependent appearance was presented. The proposed model generalizes existing approaches that use linear bases to model object deformation. Specifically, a linear basis is used to model geometric variation (similar

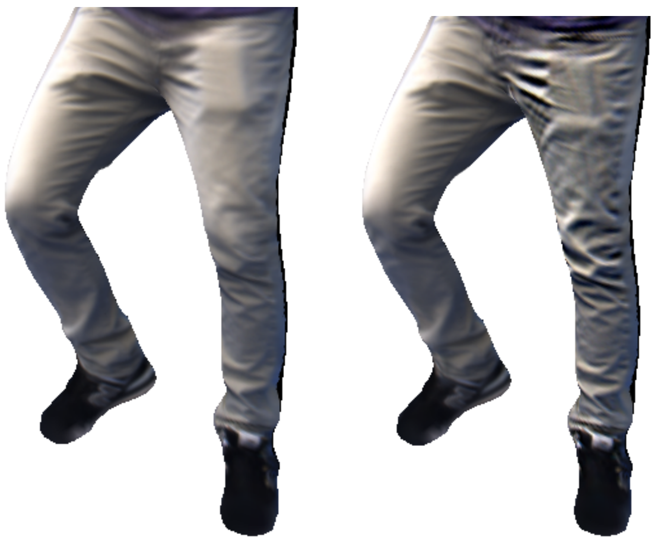


Figure 3.7: An example of the problems that can occur when the interpolation gives values outside the convex hull of the input values. Here the high frequency basis image is given too much weight, causing exaggerated wrinkles. Such problems are most likely with the global RBFs in areas of extrapolation.

to AAM [62] and graphics models [134, 34]), and a separate linear basis models appearance variation (used in AAM [62], dynamic textures [61], MMs [34]). These linear basis functions are layered on top of non-linear kinematic skeletal deformations (e.g., as in [148]). The geometric model is split into overlapping parts that roughly correspond to the kinematic links. Each mesh part has its own local basis (e.g., similar to [134, 182, 34]). The mesh parts allow interpolation parameters to have local effects, and in the case of view-dependent appearance it allows for different relative view directions on different parts.

In contrast to AAMs and MMs, both view- and pose-dependent variation are modeled in the appearance component. Variation on pose allows for effects, such as cloth wrinkling, to be modeled in texture. As the model is acquired from multi-view video, view-dependent variation is relied upon to compensate for inaccuracies in the geometry and also to model view-dependent light effects (e.g., specularities).

In order to render a novel animation from novel viewpoint, the pose and view parameters need to map to a deformed model with a corresponding appearance. This involves sparse data interpolation. To accomplish this task, ideas from the example-based geometric deformations, such as radial-basis function interpolation, are adopted and combined with the view-dependent interpolation methods used in IBR. In practice, it is helpful to simplify the interpolation by reducing the dimension (either linear for pose or spherical for view). These mappings make linear interpolation more practical. Extrapolation is known to be a problem for example-based geometries [130], and it can also be a problem when synthesizing appearance. Linear interpolation tends to have less artifacts due to extrapolation than global RBFs.

The sequential interpolation of appearance based on pose and then view allows a finer level of control as opposed to a combined interpolation. This sequential approach of interpolating is common when interpolating appearance based on illumination and view (e.g., [104]), but has not been used for pose then view.

The proposed model generalizes a wide class of image-based and graphics models, provided that the geometry has a fixed number of vertices and fixed topology. Examples include single static textured objects, single textured kinematic objects, view-dependent static objects, and AAM/MMs. For example, when there is only a single kinematic link to which all elements are attached, using a single mesh part with no pose interpolators gives a view-dependent static object (as in [61]). Free-viewpoint video can also be modeled by using a single part that is influenced by time (as an abstract parameter/hidden joint) and view direction. In all cases, the appearance and deformations are compactly represented through the linear basis (e.g., IBR).

There are some limitations in the model. First, the underlying surface motions must be well approximated by a linear basis. However, this assumption should be valid in a variety of cases as the same assumption is frequently used in facial modeling [183, 182, 34], and is even used to model the output of physical simulations on human skin [134]. Another related limitation is that the influencing parameters must be able to predict the true deformations. In other words, using joint angles alone to model the dynamic effects of loose clothing fluttering after limb motion has finished will likely fail. In such cases, it may be possible to introduce more influencing parameters (e.g., velocity of joint angle), but it may be better to model such phenomenon with physical simulation (e.g., [235]). For a range of *ordinary* clothing, our assumption that deformation is a function of pose is reasonable, but the training sequence needs to be devised such that dynamic effects are limited.

A few of the limitations are directly tied to the use of mesh parts. For example, the parts were used to allow different pieces to have different influencing parameters. One of the most important parameters is the view-direction (which varies over the kinematic pieces as they move). Using a single view direction over the entire part is just an approximation, as the view direction from a local point to a single camera varies as you move along the surface. The approximation of a single view direction for each camera to each part is best suited for cameras with a long focal length (e.g., they approach orthographic cameras), meaning the view-direction does not vary as much over the surface.

The blending of surface geometry and appearance over the overlapping regions of the parts may also lead to unexpected texture inconsistencies. The blending weights between parts help remove most of these problems. However, multi-band approaches of seamless blending between images (e.g., [46, 11]) can alleviate further inconsistencies, should they occur.

Finally, the above model does not explicitly take illumination effects into consideration. The downside is that rendered models will always have the same illumination as the input sequence. A simple approach to reduce the illumination effects is to acquire the object under diffuse illumination.

Another approach to overcome this limitation is to acquire the training sequence over various illumination conditions. A feasible way of achieving this may be to use time multiplexed light (e.g., [260]) with the subject performing slow motions. Light direction could be added as another influencing parameter, or a multi-linear model of the appearance component could be used (e.g., [254]). A similar approach, considered for free-viewpoint video [243], is to use photometric stereo to reconstruct dynamic surface normals and use the appearance component to model the dynamic surface normal and albedo instead of the lit texture.

4 Static Model Acquisition

For some time now the benefits of turntable-based vision acquisition systems for low cost 3D modeling have been recognized and exploited [28, 83]. Turntables boast the ability to quickly acquire images from all sides of an object that can quickly be calibrated and easily be foreground segmented for use in both silhouette and stereo reconstruction. In this chapter we argue that turntable acquisition is feasible for acquiring human scale geometry and a static appearance model, something that has only been exploited in few works [55, 57, 243]. The presented silhouette-based approach is inexpensive (requiring as few as two cameras), and compensates for unintended subject motion by interleaving tracking with geometric refinement.

4.1 Introduction & Motivation

As discussed in Chapter 2, convenient vision-based acquisition of static human geometry is useful, with example applications ranging from gaming to anthropometric studies. There exist full body laser range finders built exclusively for the task of recovering dense static human geometry, but this hardware is expensive (e.g., Cyberware ™'s Whole Body 3D Scanner $200K+). By comparison, a two camera system, such as the one presented in this chapter, costs on the order of hundreds or a few thousand dollars. In terms of applications in vision, a recent trend has seen many of the multi-view human tracking and deformation recovery methods being formulated around an initial laser scanned geometry [72, 71, 22]. Some of these methods go on to recover deformations over time from vision [71], but the dependence on a laser scanned geometry can be removed through the use of passive vision.

A common vision-based solution is to capture a human geometry using a large set of fixed, pre-calibrated cameras that observe a moving person [256, 186, 253, 271]. Such methods can reconstruct a time varying geometry using the visual hull [255] or multi-view stereo [186], which can then be related to each other either using differential constrains like scene flow [256], through feature point correspondences [253], or registered with marker-based motion capture data in the coordinate system of the joint [203]. Instead of this expensive approach of using a large number of cameras, we take a different approach and propose a method that acquires human geometry using a traditional turntable and only two cameras. A complete geometry at each time frame cannot be obtained with only two cameras, therefore our reconstruction utilizes turntable motion to integrate the temporal observations to get a static human model.

The primary hurdle in simply extending rigid-object turntable-based approaches to a human

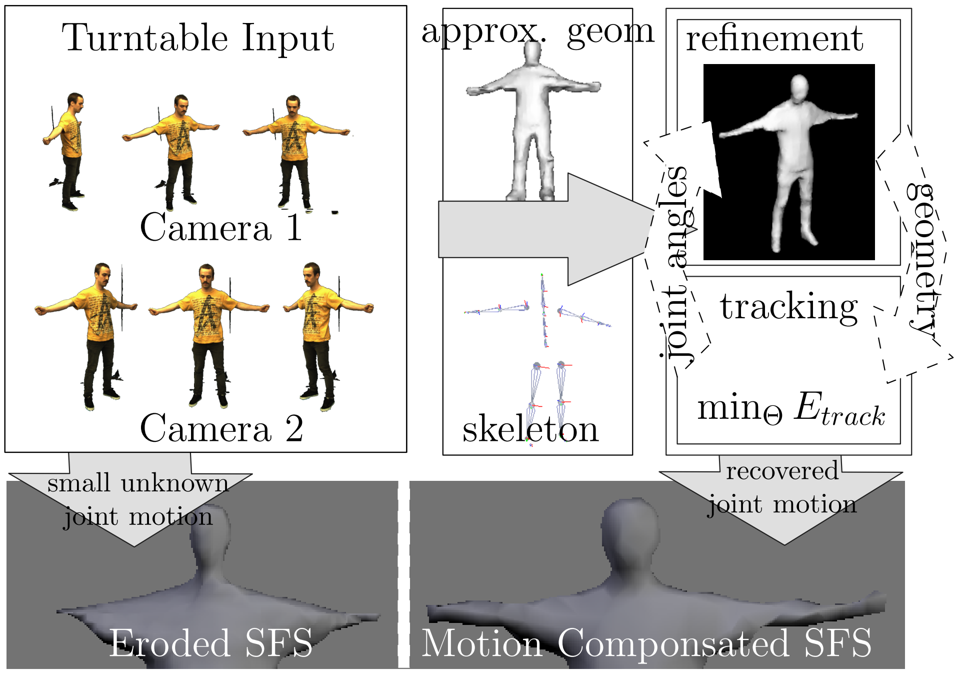


Figure 4.1: Overview of our solution. SFS with no motion compensation illustrates eroded body. Interleaving motion estimation with SFS gives more accurate results.

geometry is the fact that a rotating human is not rigid and will undoubtedly move slightly over time while rotating. Such motion causes methods like shape-from-silhouette to excessively carve the object (Fig. 4.1) and causes misalignment of any recovered appearance. Since the human is a kinematic chain containing a hierarchy of coordinate systems, this problem of registration can not be solved by a simple application of single rigid body calibration. Recovering the unintended motion of the subject is problematic with one view, which is why a wide-baseline pair of cameras is used.

Regardless of the cues that are used for reconstruction (either silhouettes over time or stereo between the camera views), the unintended motion of the subject relative to the turntable needs to be addressed. One possible approach is to further limit the subject motion with some sort of support system (see Figure 4.2), but such supports interfere with the geometric reconstruction and occlude portions of the texture data. Another alternative is to discard the turntable approach and use only a few images along with a template model (e.g., [225]). But this may limit the accuracy and even expressiveness of the human-like models that could reconstructed. Instead, we focus on recovering the relative motion of the subject to enable the reconstruction.

Classical feature-based correspondences or feature tracks, such as those used in standard structure from motion (SFM), offer one route to recover these joint motions. Articulated structure from motion factorization techniques decompose such feature tracks into rigid parts and the corresponding points, but are often based on restricted camera models [249, 278]. On the other hand, given that feature tracks are segmented into corresponding parts, the more recent applications of SFM that refine Euclidean camera parameters based on dense matches could also be used to recover the rigid

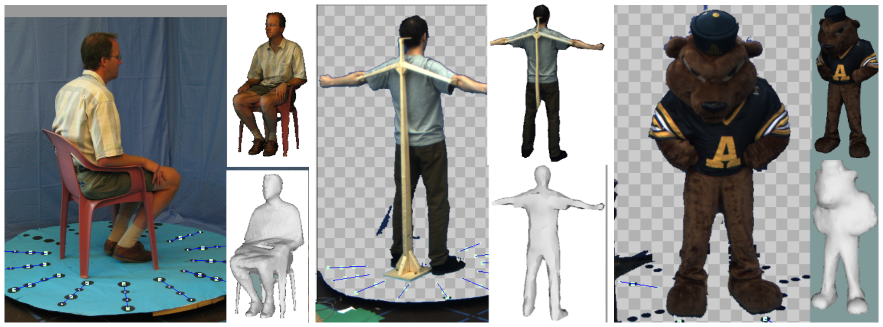


Figure 4.2: A single camera can be used when any subject wobbles are inhibited by a support system (either by a chair or vertical support). However, the support typically interferes with the geometry (left, chair) or the texture (vertical support). Using a human template geometry would restrict the acquisition to *normal* people-like geometries and would not allow such things as mascots or baggy clothing.

deformation of individual joints [91]. However, these feature-based methods may still be prone to failure in regions where few features are available, such as the arms which, in general, tend to be one of the more problematic body regions to capture.

As the geometry of these problematic regions is well classified by silhouettes, it is useful to consider silhouettes for the purpose of recovering the motion of the joints. It has been demonstrated that the relative position of cameras can be calibrated in a multi-view environment using dynamic silhouettes [220, 36], but in our case we use a calibration pattern to calibrate the relative poses of cameras and the turntable motion. Alternatively, similar cues such as epipolar tangents, frontier points, or silhouette consistency have been used to calibrate the position of cameras viewing a scene under restricted turntable motion [111, 92]. Silhouette cues have also been used to align a 3D geometry with images [144]. Unfortunately, it is not clear that these methods are directly applicable or extensible to the recovery of the kinematic motion of an unknown human subject.

One of the most relevant methods for recovering object motion for combining silhouettes over time utilizes both silhouette and image appearance cues. The shape-from-silhouette over time work of Cheung *et al.* [56] recovers the motion of a rigidly moving object observed by multiple image sequences by the use of colored surface points (e.g., frontier points with color values) and a silhouette constraint. The recovered transformation ensures surface points extracted at time t agree with the silhouettes extracted at time $t + 1$ (and vice-versa) and that the observed color values at these points is consistent. This method is also used to fuse images for recovery of human geometry under turntable motion and perform multi-view tracking [56, 57]. Unfortunately, the method relies on the colored surface points, which cannot be extracted at each time frame with only two cameras.

As discussed in Section 2.4.2, the vast assortment of multi-view human tracking methods attempt to solve the problem of recovering the motion of the kinematic links [18, 100, 128, 244]. These approaches rely on a known geometry and often combine multiple cues, such as stereo, flow and silhouettes, in order to recover the joint angles. A practical use of the silhouette is to minimize the

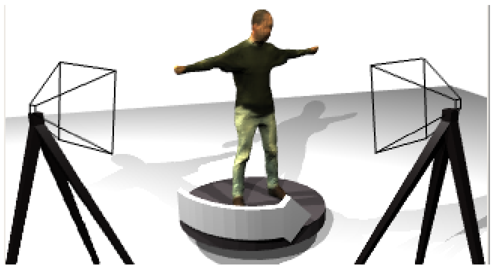


Figure 4.3: Capture setup illustrating the typical position of L and R cameras.

exclusive-or between input silhouette and model silhouette [244]; this cost function is closely related to silhouette-based camera calibration [36, 111].

Our approach leverages the strengths of the multi-view human tracking methods, which do not require salient features on all of the links and are robust to inaccuracies in the input geometry. The recovered motion is used to register all image observations over the turntable rotation in order to obtain a better static geometry. Many of the existing multi-view human tracking methods also try to refine geometries over time [112], deform temporal geometries between time-steps [253], interpolate SFS between time steps [2]. or ensure that the silhouette of the tracked model is consistent with input silhouettes (e.g., [259]). These dynamic geometries are often reconstructed each frame (e.g., often 6-8 or more views are available), meaning they rely mostly on the inter-camera correspondence between numerous fixed cameras for reconstructing geometry. Instead, in our two-view case where it is not practical to reconstruct a per-frame geometry, we exploit the intra-camera relationship for geometry reconstruction by recovering and compensating for the human motion that occurs on the turntable.

The contributions in this chapter are three-fold:

- Using as few as two cameras, the small kinematic human motion relative to a rotating turntable is tracked by utilizing silhouette consistency while enforcing kinematic constraints (§4.3.2).
- Recovered joint angles for a kinematic structure are used to re-compute a unified shape-from-silhouette model that is the union of the visual hull for each of the kinematic links (§4.3.3).
- Several approaches for reconstructing a single texture map on the human subject are discussed and evaluated (Section 4.4). These methods include weighted average (§4.4.2), super-resolution (§4.4.3), and multi-band blending (§4.4.4).

4.2 Problem Definition

As input we have two image streams $I_{L,t}, I_{R,t}$ and silhouette images $S_{L,t}, S_{R,t}$ at time $t \in \{1, \ldots, T\}$ for a left and right camera (denoted with subscripts L and R). The images observe a subject rotating on a turntable (see Figure 4.3). The projection matrices $\mathbf{P}_{L,t} = [\mathbf{K}_L|0]\mathbf{E}_t$ and $\mathbf{P}_{R,t} =$

$[\mathbf{K}_R|0]\mathbf{E}_{R,L}\mathbf{E}_t$ are also available. The relative pose between the cameras, $\mathbf{E}_{R,L}$, is fixed, and the motion of the cameras relative to the turntable is characterized only by the known transformation $\mathbf{E}_t$ (recovered using a pattern placed on the turntable - see Section 4.5).

We assume that the motion of the human rotating on the turntable is governed completely by the joint angles of her kinematic skeleton. The problem is then to recover both the geometry, G, and these joint angles, $\mathbf{\Theta}_{1:T}$, such that the geometry deformed by the joint angles is consistent with the two input image streams.[1]

4.3 Interleaved Tracking & Refinement

Based on the observation that multi-view silhouette-driven human tracking is often successful with an approximate geometric model, we propose to solve this problem by interleaving tracking and refinement. The entire procedure is summarized below:

1. *Initialize* a geometry, G, and align a kinematic structure.
    - Obtain a geometry, G, from SFS where silhouettes are grown to ensure the model has all appendages. Align and attach a kinematic structure to G.
2. *Iterate* tracking the joint angles and refining the model:
    - *Joint Tracking:* recover smoothly varying joint angles, $\mathbf{\Theta}_{1:T}$, that deform the geometry, G, to satisfy image cues, and keep the feet stationary.
    - *Refinement:* use $\mathbf{\Theta}_{1:T}$ to register the image observations in the coordinates of each joint, and then compute a piece of the SFS geometry, G_b for each joint, b. Take the union of the piece geometries to obtain a complete geometry, $G \leftarrow \bigcup_b G_b$, and attach this to the kinematic structure.

We first define our geometric and kinematic model and specify how it is attached to a posed kinematic structure (§4.3.1). This association of a skeleton with a geometry occurs in both the initialization when the pose is manually specified and after each refinement step when a new model has been computed. We then describe the tracking objective function used to obtain the joint angles for a given model (§4.3.2), followed by the geometric refinement procedure that utilizes these joint angles (§4.3.3).

4.3.1 Model Parameterization

During any phase of the tracking, we use the base model described in Chapter 3.1.1, which is the standard graphics model for human skeletons and consists of two parts: a mesh geometry and a kinematic skeleton (Fig. 4.4). The geometry, G, is used to *skin* the skeleton, meaning motion of

[1] Our approach easily generalizes to more images.

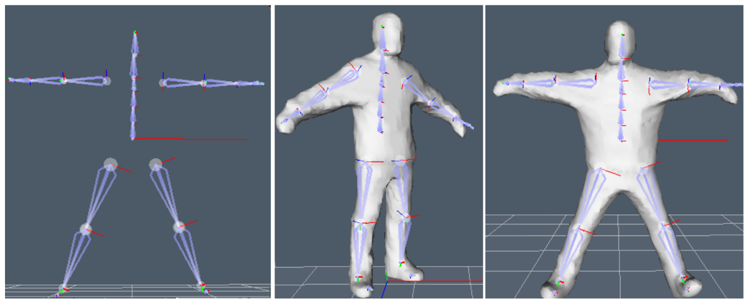


Figure 4.4: The skeleton in rest pose (left). Before being coupled to the skeleton, the input (and recovered) geometry are already posed along with the corresponding skeleton. Skinning weights are fit in the posed form to obtain the *rest* geometry, $\mathbf{v}_k(\mathbf{0})$, (right). The skinned geometry can now be animated back to the input (or other) poses.

Bone	Parent	Freedoms
Root	nil	R_x,R_y,R_z,Tx,Ty,Tz
Back	Root	$R_x \in [-20, 45]$ $R_y, R_z \in [-30, 30]$
Thorax	Back	$R_x \in [-20, 45]$ $R_y, R_z \in [-30, 30]$
Clavicle	Thorax	$R_y \in [-10, 20]$ $R_z \in [-20, 0]$
Humerus	Clavicle	$R_x \in [-60, 90]$ $R_z \in [-90, 90]$
Radius	Humerus	$R_x \in [0.01, 170]$
Femur	Root	$R_x \in [-160, 20] R_z \in [-70, 60]$
Tibia	Femur	$R_x \in [0.01, 170]$
Foot	Tibia	–

Table 4.1: A breakdown of the bone names, their freedoms, and their parents for a total of 34 freedoms.

the geometry is determined solely by the kinematic model–an assumption we will use during the tracking.

Kinematic Model

In the turntable acquisition, the subject is attempting to be motionless, so the kinematic motion is limited. We take this prior into account when choosing the kinematic model for our tracking. Specifically, we extract a default kinematic structure complete with joint angle limits from a subject in the CMU motion capture database [60] (Fig. 4.4). The restricted range of motion and corresponding degrees of freedom for each kinematic link are listed in Table 4.1. Before initialization the kinematic links are aligned to the subject (see §4.5).

Kinematic & Geometric Surface Coupling

After each stage of the tracking (or during initialization) we have a geometry that has not yet been coupled to the skeleton (and so cannot yet be used for tracking). Coupling of the geometry amounts to finding the skinning weights used in linear blend skinning (Eq. 3.2). This coupling of the geometry to the skeleton is specified when the geometry is in a rest pose. In our case, the input geometry is either the initial geometry or is the output of the geometric refinement phase. In either situation, the geometry is already given in the context of a posed kinematic structure (i.e., the vertices $\mathbf{v}_k(\mathbf{\Theta}_{\text{pose}})$

are already deformed with joint parameters $\mathbf{\Theta}_{\text{pose}}$). Therefore, the skinning weights are assigned to the geometry in this posed frame using the heat diffusion approach of Baran and Popović [25]. The rest geometry must then be obtained through the inverse of the transformation in Eq. 3.2, i.e., $(\sum_{i=1}^{|\mathbf{\Theta}|} w_{k,i}\mathbf{T}_i(\mathbf{\Theta}_{\text{pose}}))^{-1}$.

This process of coupling a reconstructed geometry to a kinematic skeleton means that our geometry is now articulated by a small set of joint angles, and can be used in the tracking phase.

4.3.2 Joint Tracking

Given a skinned geometric mesh parametrized only with joint angles, e.g., $G(\mathbf{\Theta}_t)$, we treat the recovery of all the joint angles, $\mathbf{\Theta}_{1:T} = \{\mathbf{\Theta}_1, \mathbf{\Theta}_2, ..., \mathbf{\Theta}_T\}$, as the optimization of a cost function that is a linear combination of several terms:

$$\min_{\mathbf{\Theta}_{1:T}} E = E_{sil} + \alpha_{\text{kin}} E_{\text{kin}} + \alpha_{\text{smooth}} E_{\text{smooth}}. \tag{4.1}$$

The silhouette term, E_{sil}, is based on an energy used in motion tracking [245] and measures agreement of the model with the input silhouettes. It is computed as a sum of XOR's over all input images:

$$E_{sil} = \sum_{t=1}^{T} \sum_{i \in \{L,R\}} \sum_{\mathbf{x}} \frac{S_{i,t}(\mathbf{x}) \otimes P_{i,t}(G(\mathbf{\Theta}_t), \mathbf{x})}{\text{width}(I_{i,t}) * \text{height}(I_{i,t})}. \tag{4.2}$$

where the shorthand $P_{i,t}(G(\mathbf{\Theta}_t))$ denotes the projected silhouette of the geometry by the camera matrix $\mathbf{P}_{i,t}$.

The smoothness term prefers no joint motion from one frame to the next:

$$E_{\text{smooth}} = \sum_{t=2}^{T} \|\mathbf{\Theta}_t - \mathbf{\Theta}_{t-1}\|^2. \tag{4.3}$$

Finally, due to the assumption of our input being a human rotating on a platform, the kinematic term, E_{kin}, enforces the constraint that the feet stay on the ground. As our kinematic skeleton is naturally rooted at the tailbone, we use the E_{kin} term to limit deviations of the feet position, $\mathbf{X}_{\text{foot}}$, in frames $t > 1$ from their position at time $t = 1$:

$$E_{\text{kin}} = \sum_{t=2}^{T} \sum_{\text{foot}} \|(\mathbf{X}_{\text{foot}}(\mathbf{\Theta}_t) - \mathbf{X}_{\text{foot}}(\mathbf{\Theta}_1))\|^2. \tag{4.4}$$

Due to the discrete nature of the silhouette XOR term, we use Powell's method to optimize the cost function [245]. The parameters are initialized with the pose from the first frame and all the parameters are optimized simultaneously.

4.3.3 Model Refinement

Tracking gives an updated estimate of coordinate transforms of each link at each time of the image sequence. These transformations are used to integrate all the silhouette observations and create a

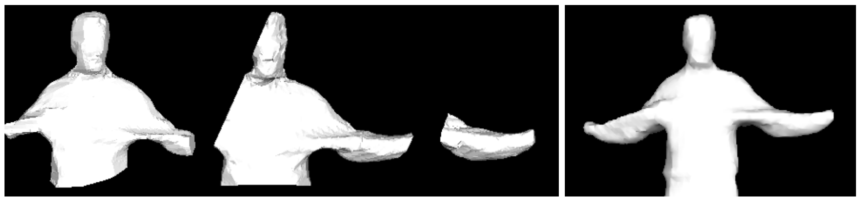


Figure 4.5: Overlapping pieces of geometry computed from SFS (corresponding to the head, shoulder, and elbow) are merged into a single manifold geometry.

new, refined geometry. The geometry is first computed as a number of partially overlapping pieces (one for each bone, b) that are subsequently merged (Figure 4.5). Each bone is assumed to be a rigid body, and the resulting joint to world transforms (from $\mathbf{\Theta}_{1:T}$) are concatenated with the world to camera transforms to get all observations in the coordinate frame of a bone. Recall from Eq. 3.1 that $\mathbf{M}_b(\mathbf{\Theta}_t)$ is the transformation from the bone to world. Therefore, the camera matrices relative to the bone are then

$$\mathbf{P}^b_{i,t} = \mathbf{P}_{i,t}\mathbf{M}_b(\mathbf{\Theta}_t). \tag{4.5}$$

The piece of geometry associated with the bone, $G_b = SFS(\{\mathbf{P}^b_{i,t}\}, \{S_{i,t}\})$, is computed from all observations (i.e., all time frames and both cameras) using SFS with the marching intersections (MI) data structure [240]. The SFS computation for bone b is bounded by the axis-aligned bounding box (AABB) of the set of vertices from the previous geometry whose skinning weights to link b are above a threshold:

$$\text{bounds}^b = AABB(\{\hat{\mathbf{M}}_b^{-1}\hat{\mathbf{v}}_k : w_{k,b} \geq \tau\}). \tag{4.6}$$

The threshold is low ($\tau = 0.05$) to allow for errors in the previous, approximate geometry. The result is then a piece of the mesh represented in the local coordinate system of bone b. This piece of the mesh, G_b, is then transformed to the turntable coordinate system at time $t = 1$ using $\mathbf{M}_b(\mathbf{\Theta}_1)$.

We now have pieces of geometries for each part that are originally disconnected and represented in the turntable coordinate system. A final manifold geometry is then obtained as the volumetric union of the pieces:

$$G \leftarrow \bigcup_b G_b \tag{4.7}$$

Subsequently, this posed geometry is attached to the skeleton by finding new skinning weights, $\{w_{k,b}\}$, and the corresponding rest vertices, $\hat{\mathbf{v}}_k$ (as in Section 4.3.1). This refined mesh is then used in the next iteration of the tracking (starting with the estimated motion parameters from the previous iteration).

4.4 Single Texture Reconstruction

A few interleaved iterations of tracking and refinement are sufficient to obtain a decent reconstruction. In this section the problem of estimating a single texture map for the reconstructed geometry

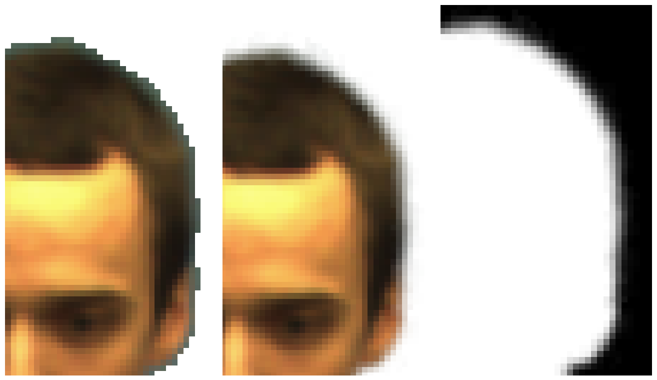


Figure 4.6: The input binary input images are matted to obtain a better estimate of the foreground color and alpha values. Input image (left), matted image (middle), and foreground alpha (right).

is considered. This process is divided into two phases: texture coordinate estimation (i.e., surface parameterization) and texture map estimation. In the first phase, texture coordinate estimation, the objective is to obtain a semi-consistent parameterization of the mesh that gives similar texture coordinates across different geometries (§4.4.1).

The second phase, estimation of the texture map, is concerned with recovering a color image $\mathcal{T}$ with size $tw \times th$ that best agrees with the input images (or produces the best renderings). Estimating $\mathcal{T}$ is challenging due to changes in illumination (as the object rotates relative to the light) and due to the approximate SFS geometry. Three approaches are investigated: a simple weighted average of warped texture images (§4.4.2), a super-resolution formulation (§4.4.3), and a multi-band blending approach (§4.4.4). In all cases, a single subscript over all available images from all cameras is used: $\{I_i\}_{i=1}^{TC}$. Input images are alpha matted to help remove the effects of the background color on the silhouette pixels before texture estimation [145] (Figure 4.6).

4.4.1 Texture Parameterization

Although the geometry will typically be genus 0, the mesh connectivity will be different across reconstructions. One approach to obtaining the 2D texture parameterization of the reconstructed mesh would be to align the mesh to a template mesh that has a fixed topology and predefined texture coordinates. An alternative template-free method, often used to parameterize arbitrary triangulated surfaces, consists of the following steps: split the object into disjoint partitions (i.e., charts), parameterize the individual charts, and pack the charts into a texture rectangle [147]. In order to maintain consistency between the texture parameterizations across reconstructions without requiring a template mesh, we adopt an approach between these two extremes: a set of sparsely defined salient points that have predefined texture coordinates are used. These salient points are used to split the human geometry into parts (front and back) as well as to define the boundary conditions for the least squares conformal mapping used to parameterize each part [147, 76].

The texture mapping uses 28 salient points that are distributed on the crotch, feet, armpits, arms,

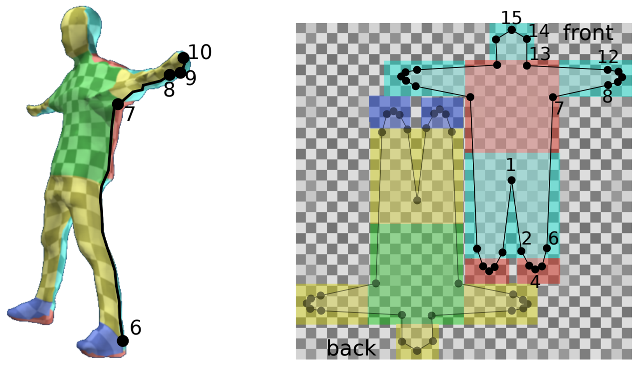


Figure 4.7: The 28 points on the boundary between the front and back have predefined texture coordinates. The front and back piece of the geometry are parameterized with this predefined mapping serving as boundary conditions. The points on the shortest path around the surface of the mesh are mapped onto a piecewise linear boundary connecting the 28 salient points. In the figure above, the 3D model (left) is texture mapped with the image (right) to illustrate the minimal distortion of the mapping.

and head. These points are automatically identified using the silhouette of the mesh in its rest pose. Each one of these points has a predefined corresponding texture point (see Fig. 4.7). Once correspondences have been identified between the salient points and the 3D mesh points, the shortest path between the mesh points is computed. This shortest path gives the discrete boundary, ∂B, separating the front and back pieces.

For each vertex, $\mathbf{x}_i = [x_i, y_i, z_i]$, in each piece, parameterizing the surface amounts to finding the corresponding 2D texture coordinates, $\mathbf{u}_i = [u_i, v_i]^\intercal$. For points on the boundary, $\mathbf{u}_i \in \partial B$, this is obtained by mapping the 3D boundary to the piecewise linear 2D boundary using an arc-length parameterization of the shortest path through the salient points on the 3D boundary. The remainder of the parameterization is then obtained by finding the texture coordinates that satisfy the linear conformal constraint for each vertex i subject to the boundary conditions [76]:

$$\sum_{j \in N(i)} (\cot(\alpha_{ij}) + \cot(\beta_{ij}))(\mathbf{u}_i - \mathbf{u}_j) = 0, \forall i \notin \partial B \tag{4.8}$$

Where $\alpha_{ij} = \angle_{kji}$ and $\beta_{ij} = \angle_{hij}$ for edge e_{ij} that separates $\triangle_{ikj}$ and $\triangle_{ijh}$. The result is a surface parameterization that minimizes angular distortion while mapping the boundary of the front and back to consistent locations in the texture (see Figure 4.7).

4.4.2 Weighted Average Texture Estimate

Now that the surface is parameterized, the color-valued texture map can be estimated. The simplest way to estimate a single texture map from the input images is to warp each input image to the texture space and take a weighted average:

$$\mathcal{T}_{\text{weighted}}(x, y) = \frac{\sum_{i=1}^{TC} w_i(x, y) I_i(\mathcal{W}_i(x, y))}{\sum_{i=1}^{TC} w_i(x, y)} \tag{4.9}$$

Where $\mathcal{W}_i(\mathbf{x}) : \mathbb{R}^2 \mapsto \mathbb{R}^2$ is the warp from a texel in the texture image to the corresponding pixel in image I_i. The warp is a piecewise linear function $\mathcal{W}_i$ that maps triangles from the 2D texture space to its projection in the image. The weighting functions, $w_i(\mathbf{x})$, assign a weight for image I_i to texel $\mathbf{x} = [x, y]^\intercal$. The weighting functions encode visibility, reliability, and the alpha value, $\alpha(x, y)$, of the image observation. A simple choice is to use the cosine of the angle between the surface normal, $\mathbf{n}$, and view-direction, $\mathbf{l}_i$, for camera i multiplied by the alpha matte:

$$w_i(\mathbf{x}) = \begin{cases} \alpha_i(x,y)\langle \mathbf{l}_i, \mathbf{n}\rangle & \text{if } (x,y) \text{visible in image } i \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

In practice, the warping and visibility calculation are easily and efficiently performed on the graphics card (see Appendix A on how the pinhole camera parameters of Eq. 2.5 can be transferred to the OpenGL projection matrix).

4.4.3 Super-resolution Texture Estimate

The weighted average estimate of a texture map minimizes variance in the warped texture space. Instead, the input image observations can be used directly if a different objective function is used. Specifically, the following formulation relates the unknown texture directly to the image observations:

$$E(\mathcal{T}) = \sum_{i=1}^{TC} \sum_{(x,y)\in S_i} (f_{\text{tex}}(\mathcal{T}, \mathcal{W}_i^{-1}(x,y)) - I_i(x,y))^2 + \tau \sum_{x,y} |\nabla \mathcal{T}|^2, \tag{4.11}$$

where $\mathcal{W}_i^{-1}(x, y) : \mathbb{R}^2 \mapsto \mathbb{R}^2$ is the warp from a pixel in image I_i to the corresponding 2D texel, which can be obtained by back-projection of the image point onto the surface. The f_{tex} function models the texture mapping function used during rendering.[2] Minimizing the first term ensures that the rendered image best agrees with the input images. Unfortunately, estimating the texture using only the first term is ill-posed if there are not enough constraints from the image pixels. This is possible when the texture resolution for a surface region is larger than the image resolution for the same region, which typically happens due to stretching of the surface in the parameterization. To make the problem well-posed, the second regularizing term is essential.

The formulation in Eq. 4.11 is similar to the typical continuous formulation of super-resolution [156, 101]. In super-resolution, the f_{tex} term models the image acquisition process (e.g., assuming that the image sensor performs a low-pass filter and downsampling of the true underlying high resolution image). However, in reconstruction from images it is desirable to obtain a solution that when rendered will give a result that best agrees with the input data. As it is unlikely that during rendering we will approximate the real camera sensor (e.g., by rendering a high resolution image, and then downsampling) our f_{tex} instead models the rendering process that occurs on the graphics card.

[2]The f_{tex} function is performing something similar $f_{\text{tex}} \simeq \mathcal{T}(\mathcal{W}_i^{-1}(x, y))$, but is written as a function of the location and the entire texture to allow it to model image sensor filtering as well as bilinear interpolation as is the case during rendering.

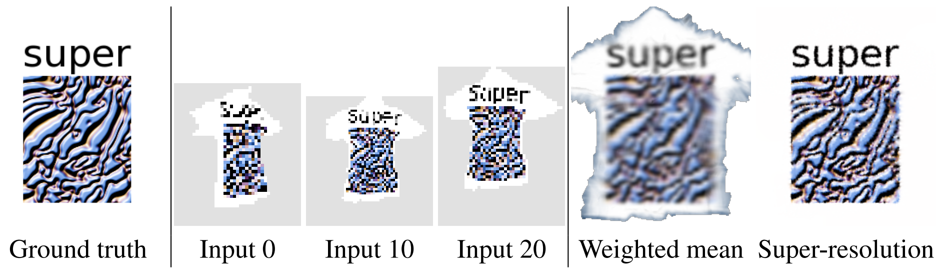


Figure 4.8: Ground truth texture, three of 25 low resolution input images of a textured shirt geometry, the weighted average reconstructed texture, and super-resolution texture for a synthetic sequence of a shirt. The super-resolution result maintains the detail of the ground truth image.

For bilinear texture filtering, f_{tex} can be accurately modeled as follows

$$\begin{aligned} f_{\text{tex}}(\mathcal{T}, x, y) =& (1-a)(1-b)\mathcal{T}(\lfloor x \rfloor, \lfloor y \rfloor) + a(1-b)\mathcal{T}(\lfloor x \rfloor + 1, \lfloor y \rfloor) \\ &+ ab\mathcal{T}(\lfloor x \rfloor + 1, \lfloor y \rfloor + 1) + (1-a)b\mathcal{T}(\lfloor x \rfloor, \lfloor y \rfloor + 1), \end{aligned}$$

where $a = x - \lfloor x \rfloor$ and $b = y - \lfloor y \rfloor$. The commonly used trilinear filtering could be modeled in the same manner.

With the bilinear model (or any f_{tex} that is linear in $\mathcal{T}$) Eq. 4.11 can be solved directly using a sparse least squares solver. Instead, the discrete equivalent of the Euler-Lagrange is solved by stacking all the bilinear constraints for all images into a sparse matrix A, giving the normal equations:

$$((A^{\mathsf{T}}A)\mathcal{T} - A^{\mathsf{T}}I) - \tau\nabla\cdot\nabla\mathcal{T} = 0 \tag{4.12}$$

Where notation is abused slightly by treating $\mathcal{T}$ as a vector in the first term and as a matrix in the second, and I is the concatenated vector of all image observations from all images.

Under ideal circumstances (no illumination changes, accurate geometry, and accurate calibration), the above formulation will exhibit super-resolution properties (see Figure 4.8). Unfortunately, these ideal conditions cannot be guaranteed during acquisition. Nevertheless, reconstruction of the texture in this manner produces a texture that minimizes a meaningful objective function.

4.4.4 Multi-band Blended Texture Estimate

When the geometry or calibration is approximate, the averaging of multiple texture images can cause ghosting and blurring [11, 101]. An alternative to just blending the input images directly is to decompose the image into several frequency bands, blend the bands independently with different weights, and recombine the blended images [11, 27]. These approaches are based the work of Burt & Adelson [46], who blended the levels of a Laplacian pyramid to achieve realistic 2D image mosaics.

Baumberg extended this mosaic approach to texture mapped image-based objects [27]. In his two-band approach, input images are separated into low- and high-frequency components. The high-frequency component of the texture is directly extracted from the high-frequency component of the input image with the best viewpoint. The low-frequency component is a weighted average of the

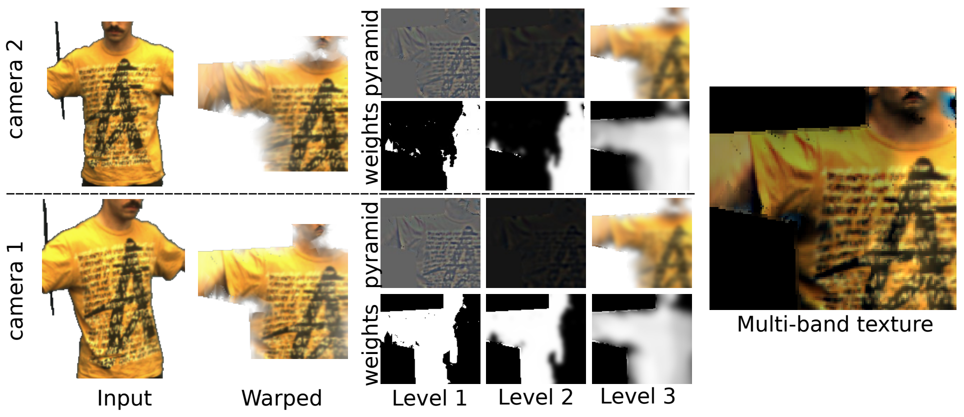


Figure 4.9: In the multi-band blending, the images are first warped to the texture space, and a Laplacian pyramid is built. The levels of the Laplacian pyramid are blended with per-level weight images. The final composed texture accumulates the blended results for each level. The above example illustrates the elements of the process for two input images.

low-frequency component from the images (e.g., using weighted blending § 4.4.2). This two-pass approach was later modified to a multi-band approach [11], which adheres closer to the spirit of the 2D mosaics [46].

Instead of performing a multi-band blend directly on the input images (e.g., as in [11]), the formulation below performs the multi-band blend on a Laplacian pyramid that is created on the warped images. The process is illustrated in Figure 4.9. First, each input image is warped to the texture space to obtain a set of textures $\{\mathcal{T}_i\}_{i=1}^{TC}$, with $\mathcal{T}_i(x, y) = I(\mathcal{W}_i(x, y))$. Then for each texture, a Gaussian pyramid of height p is defined by successive convolutions with a Gaussian filter g, giving $G_{i,1} = \mathcal{T}_{i,1}$ and $G_{i,n} = g * \mathcal{T}_{i,j-1}$ for $2 \leq j \leq p$. From this, the Laplacian pyramid is obtained as $L_{i,j} = G_{i,j} - G_{i,j-1}$, with the lowest level $L_{i,p} = G_{i,p}$. The reconstructed texture is obtained from the blended components:

$$\mathcal{T}(x, y) = \sum_{j=1}^{p} \underbrace{\left(\frac{\sum_{i=1}^{TC} w_{i,j}(x, y) L_{i,j}(x, y)}{\sum_{i=1}^{TC} w_{i,j}(x, y)} \right)}_{\text{Blended result for level j}} . \tag{4.13}$$

The weights for the levels of the pyramid, $\{w_{i,j}\}$, are defined as follows. First, $w_{i,1}(x, y)$ is 1 if and only if $w_i(x, y) \geq \max_j w_j(x, y)$. In other words, the weight is on only for the image that best views the surface. The lower levels are blurred versions of the upper levels: $w_{i,j} \approxeq g * w_{i,j-1}$, for $j > 1$. This ensures that the weight images have roughly the same scale as the corresponding frequency band they are averaging [11]. The blurring of the weight images is not done with a simple Gaussian, it is achieved with an isotropic diffusion that ensures the boundary conditions of $w_{i,j} = 0$ wherever w_i is zero [11]. In this way, occluded regions in the high-resolution texture do not contribute to the blending on any level of the pyramid.

The primary difference between the above formulation and the existing approach is that the

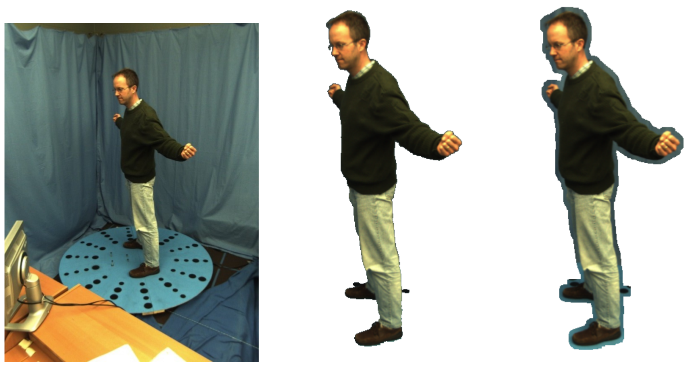


Figure 4.10: The turntable is a wooden platform with a calibration pattern painted on it. Acquisition takes place in a room with blue curtains, allowing chroma keying to extract the background. The silhouettes are extruded a few pixels to ensure recovered geometries have all appendages.

Laplacian pyramid and weight images are represented in texture space[3]. To deal with the weight images and the pyramid consistently, diffusion is performed in the texture space for all the Gaussian smoothing. During the diffusion the boundaries of charts in the texture space can be handled properly (similar as was done for super-resolution texture maps [101]). Before constructing the Laplacian pyramid, which requires blurring the input textures, any missing data due to occlusion is filled in with the weighted mean texture (reconstructed using Eq. 4.9):

$$\mathcal{T}_i(x,y) = \begin{cases} I_i(W_i(x,y)) & \text{if } w_i(x,y) > 0 \\ \mathcal{T}_{\text{weighted}}(x,y) & \text{otherwise} \end{cases} \tag{4.14}$$

4.5 Implementation

There are a number of components required to make a working system. In this section, the implementation choices for acquisition, camera calibration, background subtraction, model initialization, and parameter settings are discussed. The final tracking and refinement algorithm is implemented in C++ using OpenGL to render the model for computing the XOR score.

4.5.1 Turntable Acquisition

The turntable acquisition takes place in a small studio surrounded by at least two firewire video cameras. The turntable itself is a plywood base placed on top of a lazy susan intended to be used as a TV stand (can be purchased for roughly \$20). Once the subject is in place, a human operator uses a string wrapped around the TV stand to rotate the turntable. Acquisition takes roughly 10 seconds. The input video streams are then resampled to have a single turn containing about 30 images per camera.

[3]The existing multi-band implementation also has a more elaborate scheme to estimate the weights at the highest level by solving a discrete labeling problem.

4.5.2 Camera Calibration

As the cameras are static with respect to the world, the relative transformation between the camera, $\mathbf{E}_{R,L}$, can be obtained using methods commonly used to calibrate camera studios (e.g., wand-based methods [166]). We take a similar approach to obtain this relative calibration, and here focus only on the calibration of turntable motion. The discussion below assumes $\mathbf{E}_{R,L}$ is known in advance and that the cameras are fixed during the acquisition.

Like single camera turntable approaches, we determine the turntable motion over time, $\mathbf{E}_t$, through an easily identifiable pattern of dots that is placed on the turntable [28] (see Figure 4.10). The pattern is detected automatically, and gives correspondences between the 3D planar pattern points, $\mathbf{X}$, and their projections in the input images: $\mathbf{x}_{L,t,i} = \Pi(\mathbf{P}_{L,t}\mathbf{X}_i)$, and $\mathbf{x}_{R,t,j} = \Pi(\mathbf{P}_{R,t}\mathbf{X}_j)$. Using these correspondences, the pose of each camera relative to the turntable is estimated independently [287], which gives two rigid transforms $\mathbf{E}_{R,t}$ and $\mathbf{E}_{L,t}$ (e.g., $\mathbf{P}_{L,t} = [\mathbf{K}_L|\mathbf{0}]\mathbf{E}_{L,t}$). Ideally, as the cameras have not moved relative to one another, the transformations should be related by $\mathbf{E}_{R,t} = \mathbf{E}_{R,L}\mathbf{E}_t$. But since $\mathbf{E}_{R,t}$ and $\mathbf{E}_{L,t}$ are estimated independently, this equality may not hold. To obtain a single estimate of the turntable motion, these two transformations are averaged: $\hat{\mathbf{E}}_t = \text{average}(\mathbf{E}_{L,1}, \mathbf{E}_{R,L}^{-1}\mathbf{E}_{R,t})$.

This simple averaging is not optimal in terms of reprojection error. Therefore, this estimate is refined to obtain an $\mathbf{E}_t$ that minimizes the non-linear reprojection equations:

$$F_{\text{relcal}}(\mathbf{E}_t) = \sum_i (\Pi([\mathbf{K}_L|\mathbf{0}]\mathbf{E}_t\mathbf{X}_i) - \mathbf{x}_{L,t,i})^2 + \sum_j (\Pi([\mathbf{K}_R|\mathbf{0}]\mathbf{E}_{R,L}\mathbf{E}_t\mathbf{X}_j) - \mathbf{x}_{L,t,j})^2. \quad (4.15)$$

As is common in typical camera calibration, Levenberg-Marquardt is used for the minimization, starting with the averaged estimate $\hat{\mathbf{E}}_t$. The unknown transformation is parameterized with an axis-angle representation of the rotation and a 3×1 translation vector. This calibration approach generalizes for more than two cameras.

4.5.3 Silhouette Extraction

The background is static, making it an ideal candidate for background subtraction methods (e.g., [113]). Our studio background is a solid color, therefore an interactive chroma-keying method is used to extract a sample of the background color, and each pixel is segmented independently using the color distance to this background sample.

4.5.4 Model Initialization

In all of the experiments, the geometric model is initialized by dilating the input silhouettes and running traditional shape-from-silhouette. The kinematic skeleton is then aligned manually. First, the default skeleton is rigidly aligned to the initial geometry, and then the lengths of the bones (and positions of the joints) of the kinematic skeleton are optimized to fit a set of manually specified constraints on the positions of the joints. The constraints are specified as correspondences between a few joint positions and their 3D locations. The optimization refines the skeleton (bone lengths and pose) such that these positional constraints are met using inverse kinematics.

4.5.5 Parameter Settings

In the experiments, a few (4-5) iterations of the tracking/refinement algorithm were performed. Unless specified otherwise, the following weights were used to combine the terms in Eq. 4.1: $\alpha_{\text{kin}} \approx 0.4$ and $\alpha_{\text{smooth}} \approx 0.25$. These parameters were chosen heuristically. The kinematic weight, α_{kin}, only needs to be large enough to remove motion of the feet. As suggested in the following section, a larger smoothness may be required depending on the camera configuration, but in all the experiments $\alpha_{\text{smooth}} \leq 1$.

An iteration of the tracking takes roughly 16 minutes; the refinement and merging takes about 2 minutes.[4]

4.6 Experiments

To evaluate the accuracy the motion compensated reconstruction, several synthetically generated sequences with a ground truth reference model have been used. The strengths of the refinement algorithm are also demonstrated on several real-world sequences. The interleaved tracking and refinement gives improved geometries on these sequences compared to simply ignoring the human motion and using shape-from-silhouette.

4.6.1 Synthetic Sequences

The accuracy of our method is first demonstrated using synthetic image sequences generated under similar conditions as the real turntable setup. Here the effects of camera configurations and the smoothness parameter, α_{smooth}, are investigated. All of these experiments consider the same setup and motion. Thirty images (from each viewpoint) of a skinned figure rotating on a platform were taken; the subject is 6 feet tall and motion was mainly in the back, neck, and arms (see Figure 4.11). The tracking and refinement procedure was initialized with the ground truth skeleton. All comparisons consider the symmetric point-to-point errors between the reconstructed mesh and the ground truth mesh computed at time $t = 1$. The initial geometry was held constant between all of these experiments.

Camera placement is an important design decision for the acquisition process. To identify good camera configurations, the reconstruction is evaluated under several different camera placements. We consider placing cameras above, in front, and to the side of the subject (Figure 4.12). The baseline between pairs of cameras is also altered to get three different camera spacing configurations. As the cameras are all roughly the same distance from the subject, the sequences are labeled by the

[4]These timings are for an unoptimized implementation running on a single core of an Intel®Quad Core™2.4GHz machine.

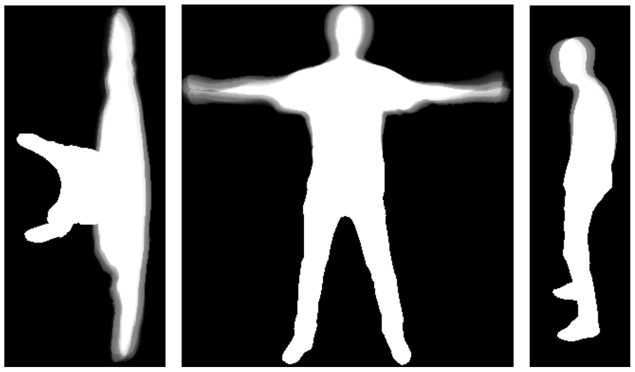


Figure 4.11: Composited renderings of cameras from the top, front, and side view of the subject used for synthetic experiments. Blurry silhouettes illustrate motion of the subject. Most of the motion was in the back, arms, and neck. The lower portion of the body (below the hips) had no motion.

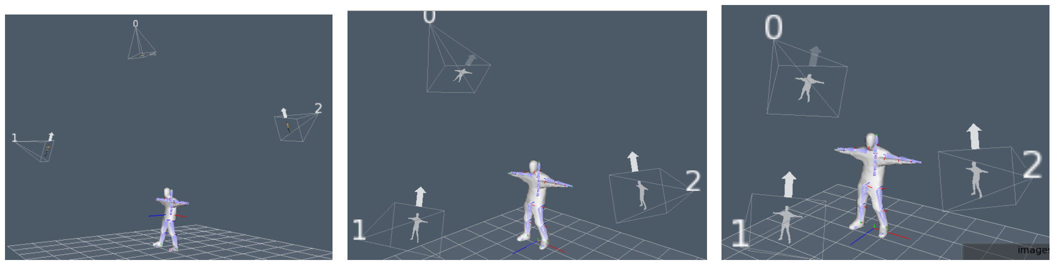


Figure 4.12: The camera configurations considered in the synthetic sequences consisted of pairs of cameras: either the (top, front), or (front, side) pairs. The images above label the cameras in the following order: top (0), front(1), side(2). Three camera spacing configurations were evaluated, roughly consisting of the cameras separated by 90° (Synthetic90), 45° (Synthetic45), and 22° (Synthetic22).

separation in degrees between the cameras: 90° (Synthetic90), by 45° (Synthetic45), and by 22° (Synthetic22).

Single Camera Reconstructions

First, the feasibility of using only one camera for the reconstruction is considered. The reconstruction is performed for the *top* and *front* cameras of the *Synthetic90* sequence independently. If using only one camera, the top view is a poor choice as its view direction is roughly parallel to the rotation axis of the turntable. This is illustrated by Figure 4.13, where the frontal rendering of the reconstruction from the *top* view is poor. The tracking/refinement allows the model to compensate for some of the subject motion, and thus gives even worse geometric results than if no compensation were used.

The situation is similar for the reconstruction using the *front* view. In this case, it is clear that the refinement has recovered some of the motion of the arms and back (given by the fuller geometry when observed from the front). However, as there is no constraint from the top, the motion compensated reconstruction is thicker than the ground truth object.

This experiment validates what one would expect: the reconstruction from only one view will

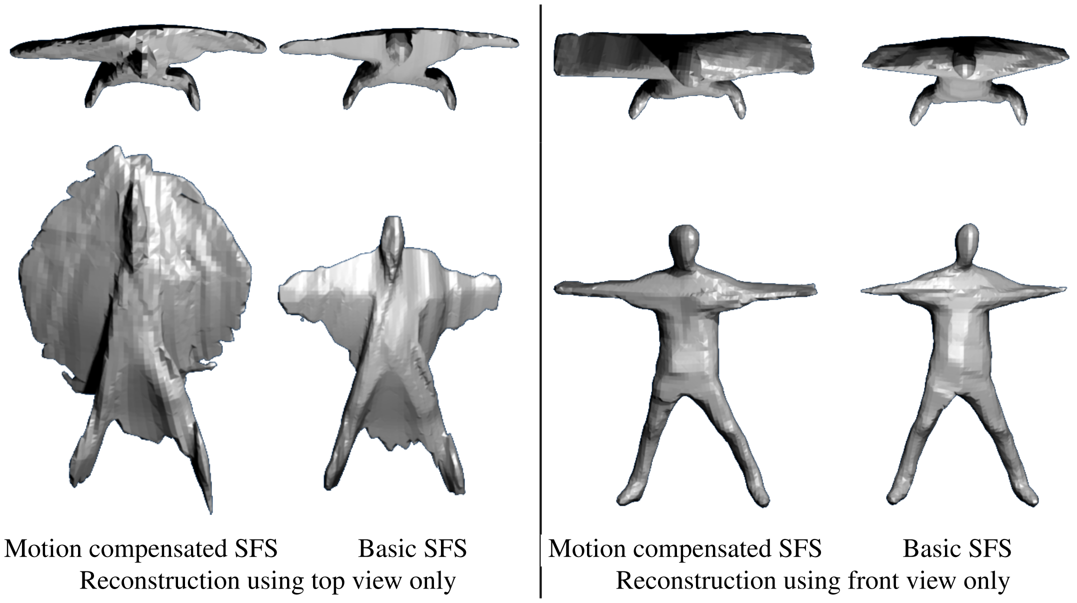

Motion compensated SFS
Basic SFS
Reconstruction using top view only
Motion compensated SFS
Basic SFS
Reconstruction using front view only

Figure 4.13: The results of using only one view for reconstruction. When using the top-view only, the motion compensation gives a worse looking geometry (with large wings), but the geometry looks accurate when viewed from the top. Reconstruction using the front view only gives a better approximation to the geometry in the front view (e.g., has bigger head, fuller arms); however, when viewed from the top, the reconstruction is too thick.

be able to reconstruct motion perpendicular to the view direction. As there are no constraints along the view direction (other than foreshortening), more than one view will typically be necessary for accurate reconstruction. Following intuition, the second view should be placed such that it has an orthogonal principle ray to the first view.

Choice of a Second View

The reconstruction from a single camera identified that the top view was a poor choice if only one camera was involved (e.g., the wings of Figure 4.13). However, the single camera experiments alluded that a good choice for the second camera would be one with an orthogonal principle ray to the first camera. As the object is rotating, the front and side cameras roughly see the same thing. In this experiment, we consider coupling the front view with either the top view or the side view for the reconstruction. Again, these views are approximately separated by 90°.

The geometric results for the refinement using the two pairs (front, top) and (front, side) both improve upon the non-motion compensated SFS (Table 4.2). Overall, the (front, top) view performs better on all of the statistics. When compared to the non-motion compensated SFS, the refinement reduced the mean, median, maximum error, and brought more of the surface closer (as illustrated by the 80%, 90%, and 95% statistics).

Figure 4.14 shows views of the recovered model without registration and with our registration reconstructed using the (front, top) pair of views. The unregistered model is missing portions of

Camera pair	Method	Average (std)	Median	80%	90%	95%	Maximum
(front, top)	Motion	1.20(0.47)	1.18	1.54	1.74	1.99	4.18
	SFS	1.74(1.65)	1.25	2.16	3.58	4.52	17.1
(front, side)	Motion	1.30(0.74)	1.19	1.59	1.92	2.55	7.48
	SFS	1.76(1.90)	1.27	2.04	3.14	4.50	19.8

Table 4.2: Point-to-point symmetric distances for the reconstruction results (in cm) compared to ground truth for the (front, top) and (front, side) camera pairs in the Synthetic90 sequence. The refinement improves all of the statistics, with the (front, top) pair capable of achieving slightly better overall performance. The percentile statistics indicate that $X\%$ of the surface was less than the given value.

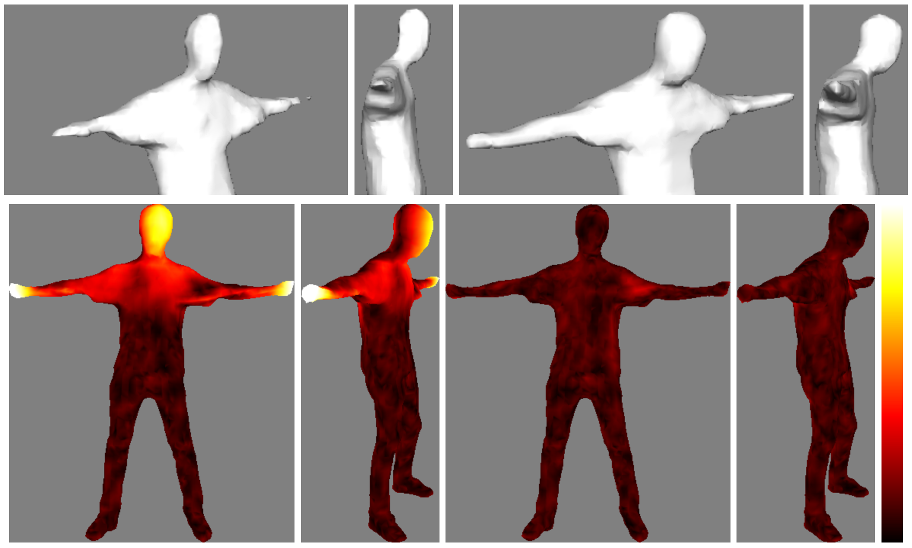


Figure 4.14: Visualization of the reconstruction using the (front, top) pair from the *Synthetic90* dataset. Top: recovered model without (left) and with motion compensation (right). Bottom: Ground-truth synthetic model colored with the distance to the recovered model without (left) and after refinement (right). Colormap: black=0cm, white=10cm.

the arm, and the head is carved due to motion in the back. Coloring the ground-truth model with the distance to the recovered model illustrates how our method improves the reconstruction in these regions (bottom right of Fig. 4.14, notice small error over the surface).

Baseline Between First and Second View

The previous experiments illustrated that with 90° separation, the reconstruction was successful. However, the single view configurations (which can be thought of as a degenerate pair) were incapable of improving the results. To further investigate the importance of the baseline between the camera pairs, we decreased the spacing between the two cameras and used the same camera pairs in the Synthetic45 and Synthetic22 sequences.

The reconstruction results for Synthetic45 and Synthetic22 sequences are in Table 4.3. There is still a benefit to doing the refinement in the Synthetic45 sequences, although the results are not as good as was obtained for the Synthetic90 configuration (refer back to Table 4.2). There is some

	Cameras	Method	Avg. (std)	Median	80%	90%	95%	Max
Synthetic45	(front, top)	Motion	1.38(0.67)	1.31	1.74	2.11	2.52	8.60
		SFS	1.75(1.63)	1.31	2.23	3.36	4.45	18.91
	(front, side)	Motion	1.58(1.09)	1.32	1.98	2.77	3.68	12.15
		SFS	1.79(1.79)	1.31	2.31	3.18	4.34	21.61
Synthetic22	(front, top)	Motion	1.58(1.10)	1.35	1.96	2.77	3.34	13.52
		SFS	1.75(1.66)	1.31	2.19	3.29	4.40	19.99
	(front, side)	Motion	1.77(1.23)	1.38	2.35	3.56	4.46	8.15
		SFS	1.81(1.84)	1.30	2.36	3.28	4.36	21.13

Table 4.3: Point-to-point symmetric distances for the reconstruction results (in cm) compared to ground truth for the Synthetic45 and Synthetic22 sequences. The refinement in both pairs for Synthetic45 is still improving the results. Only the (front, top) pair in Synthetic22 improves the results over no compensation, but the improvement is limited.

Sequence	α_{smooth}	Average (std)	Median	Maximum	Average XOR
Synthetic90	0.25	1.20(0.47)	1.18	4.18	4457
Synthetic45	$\frac{1}{64}$	1.57(1.45)	1.36	21.60	7469
	$\frac{1}{16}$	1.57(1.25)	1.33	16.20	7652
	0.25	1.38(0.67)	1.31	8.60	5912
	0.5	1.27(0.48)	1.25	4.31	5090
	1	1.33(0.52)	1.31	4.46	4680
Synthetic22	$\frac{1}{64}$	1.95(1.71)	1.45	21.50	8733
	$\frac{1}{16}$	1.92(1.79)	1.43	21.50	8503
	0.25	1.58(1.10)	1.35	13.52	6998
	0.5	1.39(0.67)	1.33	8.68	5667
	1	1.36(0.59)	1.31	6.20	4935

Table 4.4: The reconstruction results for the (front, top) camera pair on all sequences with varying smoothness used for the Synthetic45 and Synthetic22 sequences. The table also includes the *Average XOR* cost, which is a rescaled version of the data term. The narrower baseline on these sequences require a larger smoothness weight to attain good results. The results, however, are still not as good as the Synthetic90 sequence.

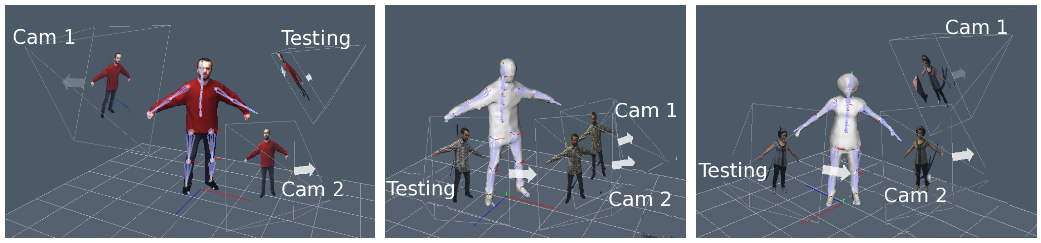
improvement for the Synthetic22 sequence when using the (front, top) pair; improvement in the (front, side) pair is inconclusive (e.g., the average is better, but the other stats are worse than simple SFS).

These results demonstrate that camera configuration is important. The results for the narrower baseline are worse than the 90 degree separation, with the improvement of using the tracking/refinement being negligible at the 22° setup.

Effect of Smoothness

As the principle viewing ray of the cameras start to become parallel, estimating subject motion along the view direction becomes ill-conditioned. However, the cameras are rotating relative to the subject, which means that at different times of the sequence the cameras are better at estimating different directions in the turntable coordinate system. For this reason, increasing the temporal smoothness weight should allow for a better reconstruction.

In this set of experiments, we used the (front, top) pair from the Synthetic45 and Synthetic22



Config1 Config2 Config3

Figure 4.15: Camera configurations used for the real experiments. The *Yellow Shirt*, *Green Sweater*, *Red Sweater* sequences all used Config1. The *Plaid* data set use Config2 and the *Girl* data set used Config3.

data sets, and performed the reconstruction with varying α_{smooth}. Up until this point all the reconstructions used the default value of $\alpha_{\text{smooth}} = 0.25$. Table 4.4 shows the results for various values of α_{smooth}. Here we see that small regularization leads to poor reconstructions. Both Synthetic45 and Synthetic22 achieve a better reconstruction when the value of α_{smooth} is increased. In fact, the *Average XOR* score decreases with larger regularization. As the regularization term is competing with the *XOR* term, this decrease in *XOR* with larger regularization suggests that refinement with small regularization is getting stuck in a local minimum. The reconstructions with larger α_{smooth}, although better than with the default α_{smooth}, are still not as good as what was obtained with the Synthetic90 sequence.

To summarize, narrower baselines lead to ill-posed reconstruction problems and will benefit from stronger regularization. With an increased value of α_{smooth}, a reconstruction with cameras as close as 22° becomes more reasonable.

4.6.2 Real Sequences

In our real experiments we have captured five data sets of human subjects rotating on a turntable (Fig. 4.16): *Yellow Shirt*, *Green Sweater*, *Red Sweater*, *Plaid*, and *Girl*. All of the data sets contain three video streams; two of the streams were used for reconstruction and the third was used for comparison. Figure 4.15 illustrates the three configurations used. The image sequences consist of 30 images (with the exception of the *Green Sweater* data set sequence which contains 22 images). The images are 800x600 color images captured from Point Grey grasshopper cameras.

In each case we bootstrapped our algorithm with a geometry that was obtained from all of the images in the data sets using SFS (without correcting for any human motion); the silhouette boundaries were extended (by roughly 5-6 pixels) to ensure that the extremities were present in the initial geometry. The *Plaid* data set contains significant motion causing the head and arms to be almost entirely eroded if the human motion is ignored (Fig. 4.17). From this example, we see that our method is capable of recovering an accurate geometry, even when the initial estimate is poor.

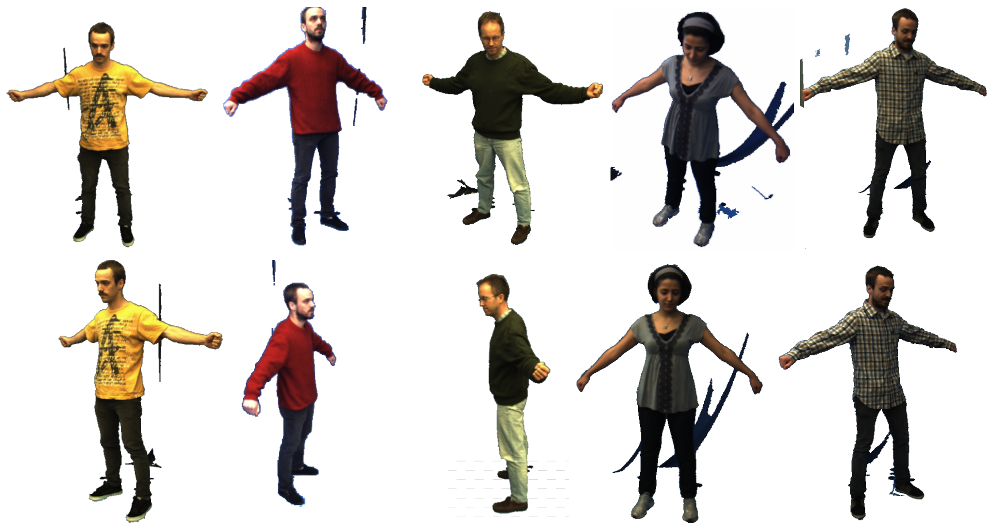


Figure 4.16: Sample input images for two of the views for the *Yellow Shirt*, *Red Sweater*, *Green Sweater*, *Girl*, *Plaid* data sets.

	Yellow Shirt	Red Sweater	Green Sweater	Girl	Plaid
Basic SFS	0.87	0.84	0.82	0.85	0.71
Motion Compensated	0.91	0.89	0.88	0.93	0.86

Table 4.5: Average Jaccard index in the extra view between silhouette and the uncompensated SFS model and the motion compensated refined model.

4.6.3 Qualitative Results

The final geometry and the SFS geometry without motion compensation for the remaining data sets are presented in Fig. 4.18. The texture mapped renderings are obtained with a single multi-band weighted texture (see Figure 4.19). In all cases the original SFS is again eroded, with parts of the arms missing and the bodies shaved too far in general. The motion compensation successfully recovers these parts of the geometry. Also note that the motion compensated models better agree with the input silhouettes.

Of these data sets the *Red Sweater* data set contained little motion, but the improvement is still evident when inspecting the silhouette agreement between the recovered models and the input images (Fig. 4.20).

4.6.4 Quantitative Results

As we do not have access to ground truth geometry for the real data, we use the extra video streams that were not used during reconstruction to perform a quantitative comparison. The average Jaccard index[5] between input silhouette and rendered silhouette of our motion compensated model is always higher (better) than that of the non-compensated model in this video stream (Table 4.5). This data suggests that the motion compensated model is a better match to the true visual hull.

[5]The Jaccard index between two sets A and B is $J = \frac{|A \bigcap B|}{|A \bigcup B|}$

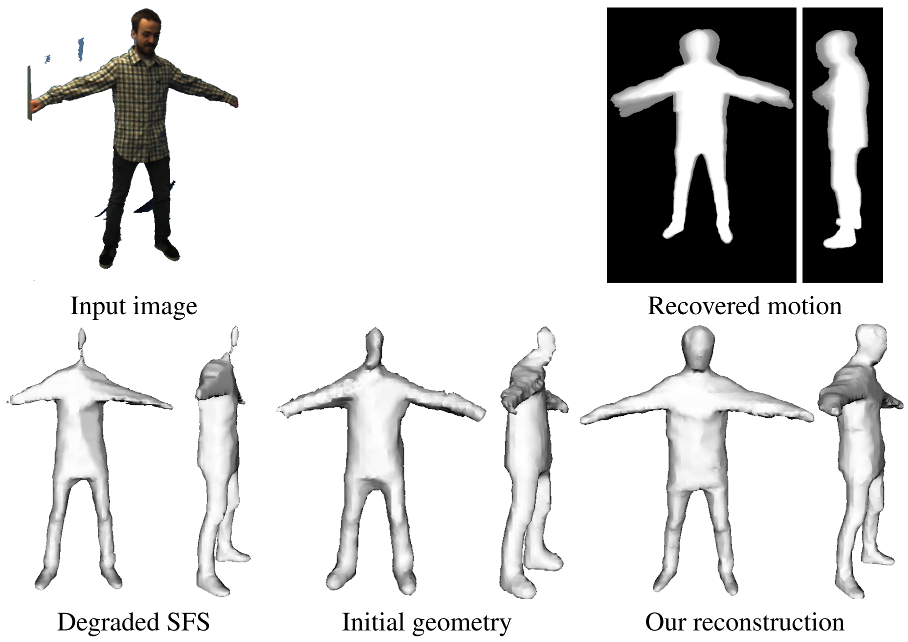


Figure 4.17: Top: an input image from the *Plaid* data set and a composite illustrating the recovered motion. Bottom: the degraded SFS model, the model used for initialization of the tracking, and the motion compensated reconstruction.

	Yellow Shirt	Red Sweater	Green Sweater
Two-view refinement	0.91	0.89	0.88
Three-view refinement	0.92	0.91	0.89

Table 4.6: We also considered using all three views for the reconstruction and again measured the Jaccard index in the test view for Config1. Using the three views only improved the Jaccard index slightly, suggesting that the model reconstructed with only two views is close to the true visual hull.

For the models captured with Config1, we also considered using all three cameras for the refinement. Table 4.6 shows the Jaccard index in the testing view for the two-camera and three-camera reconstructions. The improvement with the additional view is small, even considering that the error metric is computed in this third view. This again suggests that the two-view model is close to the true visual hull.

Refinement of the geometry only affects the E_{sil} term (Eq. 4.2) of the energy because it does not change joint angles. One may question the validity of using only SFS in the refinement, as SFS does not directly minimize the XOR silhouette energy, E_{sil}, meaning that on successive tracking/refinement iterations this energy could in fact go up. This is possible, and in fact the refinement step does sometimes increase the XOR score. However, after the tracking iteration the E_{sil} term often does go down, and only in the later iterations does the cost sometimes increase slightly. Figure 4.21 shows the XOR score for the data sets after subsequent refinement/tracking iterations.

4.6.5 Texture Analysis

Obtaining a single texture map of the object is challenging due to approximate geometry, errors in motion estimation, and changes in illumination due to subject rotation relative to the lights. Nevertheless, having a single texture map enables efficient renderings and could be useful for applications such as object tracking. We obtained single texture maps for the motion reconstructed models using the three methods presented in Section 4.4. To avoid any inconsistencies between cameras only one of the input cameras was used to create the texture images (with the exception of the *Girl* dataset, where we used both views to obtain full coverage).

Figure 4.22 shows the numerical comparisons over the input images used to compute the texture map. The comparison was obtained by rendering (in OpenGL) the tracked geometry over the input sequence using the reconstructed texture. The super-resolution directly minimizes this score, and thus always achieves the best score. However, qualitatively the multi-band approach produces superior results. Figure 4.23 illustrates detail views of the *yellow shirt* and *plaid* data sets. Here it is clear that the simple weighted averaging produces ghosted, blurry textures. Due to inaccuracies in the geometry and changes in relative illumination, the zoomed super-resolution model has some artifacts due to the sampling rate of the input image when adding linear constraints in the texture space.[6] The multi-band blending achieves the most realistic results, and is capable of producing a texture map that retains the high frequency detail with minimal ghosting artifacts.

4.7 Discussion

In this chapter, we presented an iterative method that uses as few as two widely separated camera streams to recover small human motion using silhouette cues. Silhouettes were used because texture features may not exist on all regions of the human. The recovered motion allows the registration of silhouettes to improve the geometry using SFS. A primary design objective was to keep the system low-cost, making it attainable to almost anyone with two cameras (and enough space for the turntable).

The presented reconstruction algorithm treats the tracking and refinement as independent but interleaved problems. Given a set of joint angles, the geometric refinement is obtained directly from shape-from-silhouette in the local coordinates of each bone (i.e., no direct optimization is necessary to reconstruct the geometry), meaning the geometry is a function of the joint angles. Although it is possible to include the simultaneous optimization of this implied shape directly in Eq. 4.1, such an optimization would be prohibitively expensive, which led to the iterative approach.

The experimental analysis has shown this iterative approach cannot reconstruct both the motion and shape with only one camera. Through the synthetic data sets, we have illustrated that a second view should be placed near 90° from the first view. Reconstruction with closer views is possible (as

[6]these artifacts could be reduced by further regularization, or by using a more elaborate point-spread model, or by using a trilinear texture model for f_{tex}.

suggested by the real experiments), but the smoothness parameter used on the joint angles may need to be increased. The emphasis has been on two camera configurations, but the presented approach easily generalizes to more than 2 cameras. In the case of many cameras, e.g., 8 evenly spaced cameras on a ring, it might be possible to even further limit subject motion by only requiring a rotation of 45° (e.g., this would provide full coverage of all views).

One limitation of our model is that it does not incorporate a texture consistency term, meaning the recovered motion may not respect texture motion cues. We tried adding a texture preserving term in the joint angle refinement (Eq. 4.1) but the initial geometry is too approximate. We also investigated the use of an optic-flow term, which could help ensure texture consistency without requiring an accurate model during the motion recovery. We found that it was hard to obtain reliable flow between regions of the arms which move dramatically between two views. Flow estimation in these parts posed problems even when taking into account turntable motion. Although our method works without the presence of reliable feature points, feature-based constraints should be exploited when available (e.g., as in typical structure and motion). These tracks would help ensure consistent texture motion while also aiding the reconstruction of concavities.

Another limitation is that our method needs to be bootstrapped with an initial geometry. Any appendage missing in the initial geometry will likely remain missing throughout the refinement. We currently based this geometry on an enveloping SFS geometry that is obtained by growing the silhouette boundaries. In practice, this was sufficient for our reconstructions, but poor inputs may require a more elaborate initialization (possibly based on a template).

For most of the sequences, shape-from-silhouette provides a good approximation of the human geometry. However, in some instances (especially near the front of the shoulders and near the armpits), the geometry is still approximate. Incorporating a stereo term into the refinement would help improve the reconstructions in these regions.

Even without an accurate geometry everywhere, it is still possible to obtain good renderings when the final geometry is texture mapped with a single texture. We found that the multi-band blending, while not necessarily minimizing a meaningful objective, provided the best qualitative results. In the future, we would like to explore the marriage of the super-resolution approach with the multi-band blending. For example, the super-resolution objective function could use constraints that try to match the band-pass renderings of the texture to the corresponding weighted band-pass images. This should fit in the same framework (simply by modifying the f_{tex} function) and would couple the multi-band blending to an objective function.

Instead of recovering a lit texture map, a more ideal acquisition setup may also consider using calibrated lights. With calibrated illumination, reconstructing the unlit surface albedo would be possible. Recovering such reflectance properties are essential to get highly accurate renderings under novel illumination. Furthermore, the use of calibrated illumination also offers benefits during the geometric reconstruction. For example, photometric stereo has been exploited for dynamic reconstruction in elaborate controlled sequences [260], but has not been used in a low-cost system for obtaining static human models.

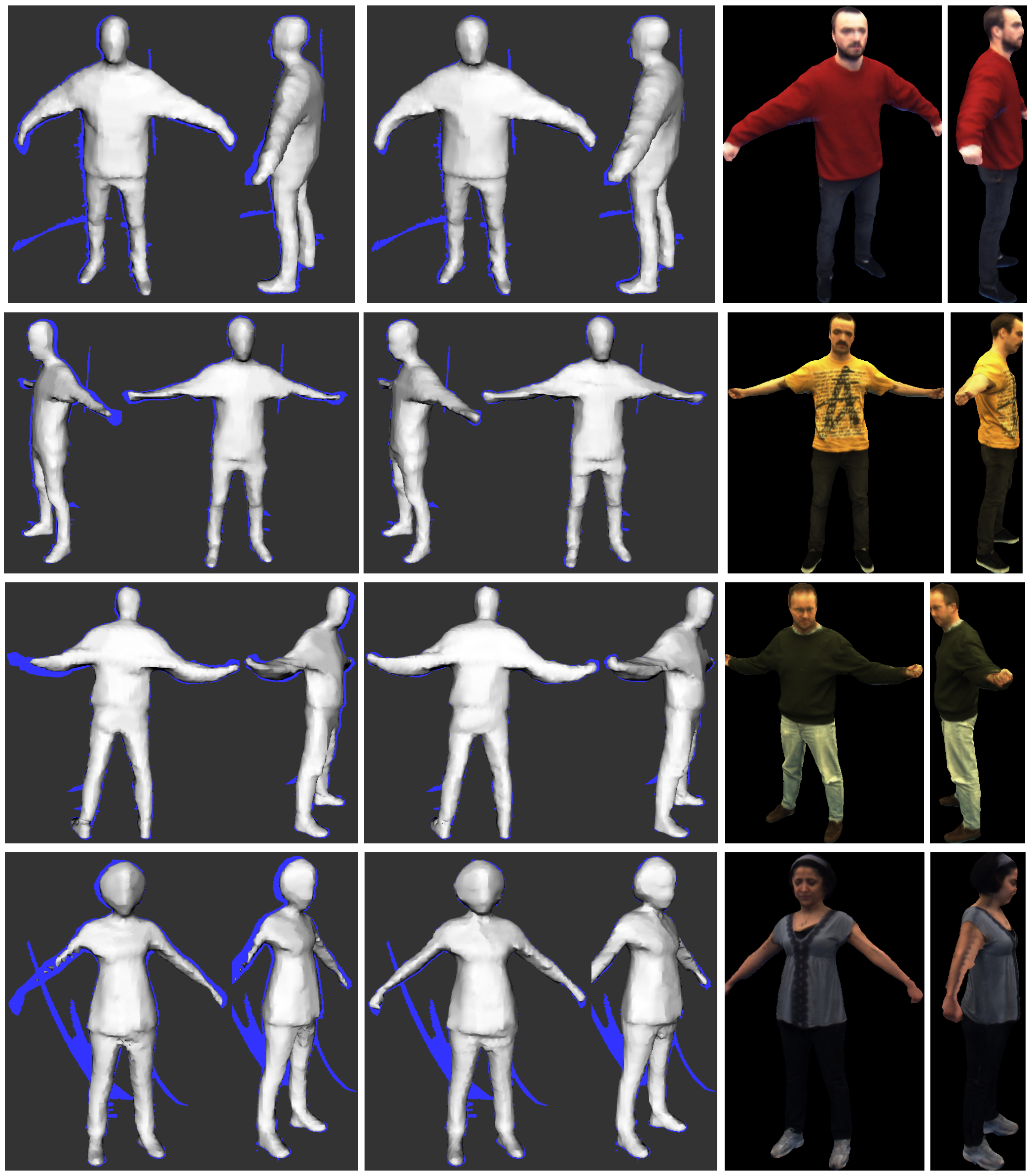


Figure 4.18: Uncompensated SFS, motion compensated SFS, and textured motion compensated views. The blue regions denote portions of the input silhouette not occupied by the reconstructed model. In all cases, the refined model covers more of the input silhouettes.

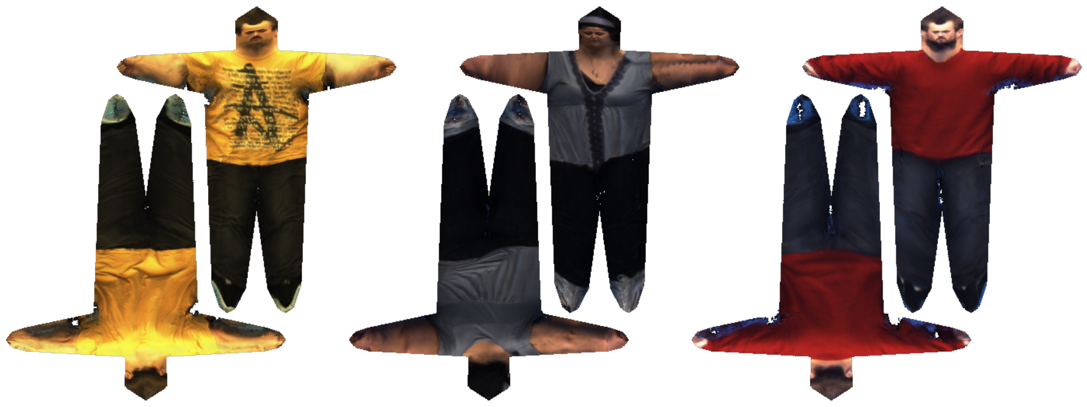


Figure 4.19: Texture images computed using multi-band weighting for three of the data sets.

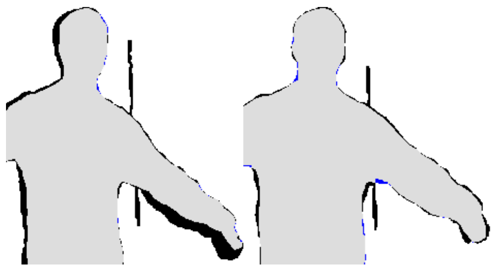
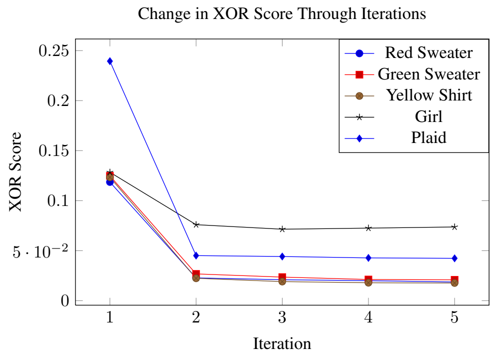
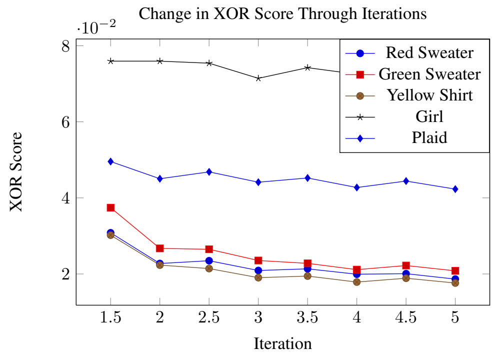


Figure 4.20: The *Red Sweater* data set had little motion, but SFS geometry (left) disagrees with the input silhouette (black indicates regions of input not covered by geometry). Motion recovered silhouette matches better (right).



(a) XOR measured after tracking



(b) XOR between tracking/refinement

Figure 4.21: The decrease of the XOR image score after interleaved tracking/refinement scores measured after tracking: (a) including the initial geometry (e.g., iterations 1-5), and (b) without the initial geometry (zoomed in on iterations 2-5) and measured after both tracking and refinement (half steps).

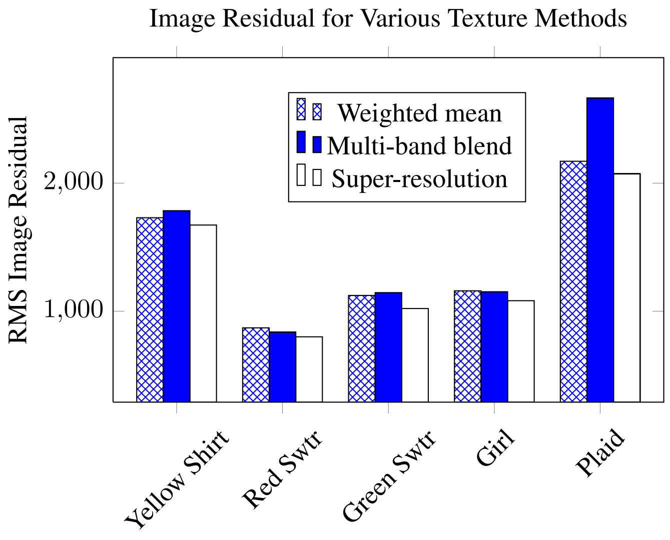


Figure 4.22: Root mean squared (RMS) image intensity score averaged over all images used to compute the texture. In all cases, the super-resolution model performs best, as this is the data term it minimizes. The weighted mean texture minimizes the variance in the warped texture space (often reflected with a low reprojection score). The multi-band blending doesn't directly minimize either the texture reprojection error or the warped image error, but sometimes performs better than the weighted mean (e.g., on *Red sweater* and *Girl*) on the above reprojection error. The multi-band blending has a larger residual on the *Plaid* sequence due to the high frequency texture and inaccuracies in the geometry.

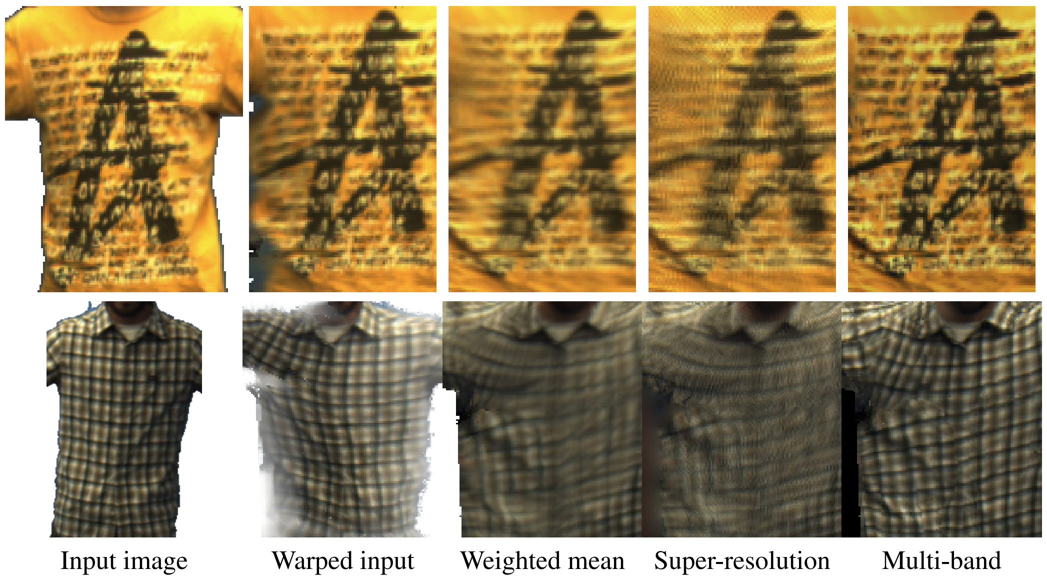


Figure 4.23: A cropped image, warped image, weighted mean (over all input), multi-band blend (over all input), and super-resolution (over all input) for the *Yellow shirt* and *Plaid* data set. The single warped input texture retains most of the detail in the regions viewed from the front. However, a weighted mean of all input images causes ghosting. Despite producing the best image SSD scores (Figure 4.22), the super-resolution model for the texture also gives artifacts due to approximate geometry and calibration. The multi-band method well approximates the high frequency data.

5 Coarse-scale Joint Tracking

The model proposed in Chapter 3 relies on a multi-view training sequence to synthesize realistic dynamic geometric and appearance effects. In order to register these geometric and appearance effects from the training sequence relative to the skeletal model, the motion of the underlying skeletal model must be recovered from the images. Such image-based motion capture also offers an avenue to obtain new human motions useful in reanimating the dynamic model. Therefore, the focus of this chapter is on model-based tracking in a controlled environment.

5.1 Introduction & Motivation

As discussed in Chapter 2, a great deal of work exists on model-based human tracking, with a variety of choices for the underlying model representation, such as truncated cones [75], super-quadrics [237, 100, 222], or meta-balls [184]. The use of linear-blend skinned meshes in tracking is becoming more popular [95, 22, 175], and they are used in this chapter also, for example, with the meshes reconstructed from Chapter 4. Like other methods that use multiple image cues (e.g., [22, 244]), we formulate the tracking problem through an energy function that combines silhouette and appearance cues with smoothness and pose prior terms. Two choices for silhouette energies common in the literature, a direct XOR [144, 244] and iterative-closest point [195, 238], are investigated in terms of accuracy and efficiency. In contrast, the appearance or image data term measures the sum of squared difference (SSD) between reference colors and input images, and it aids motion recovery when silhouettes are not available or when the pose cannot be resolved through silhouettes alone. This formulation of an image data energy extends the template-based intensity driven tracking of 2D patches [17] and morphable models [82] to a skinned mesh.

For efficiency, the energy function is minimized at each frame using local optimization. Local methods cannot compete with the robustness of slow global methods, but, in practice, local optimization is often sufficient as the motion between frames is small. Due to this small motion, the joint angles recovered from a previous frame serve as a good estimate to the joint angles in the current frame. Many existing methods attain good results for kinematic tracking through such local optimization, including approaches that use appearance-based cues [41], silhouettes [54], optic-flow and silhouettes [22, 244], or region-based and feature cues [94]. In terms of efficiency, the appearance-based methods, which use a template appearance, and region-based methods, which use color distributions to jointly segment and track, have an advantage: no image preprocessing needs to be performed. This leads to more efficient implementations. Methods utilizing these cues

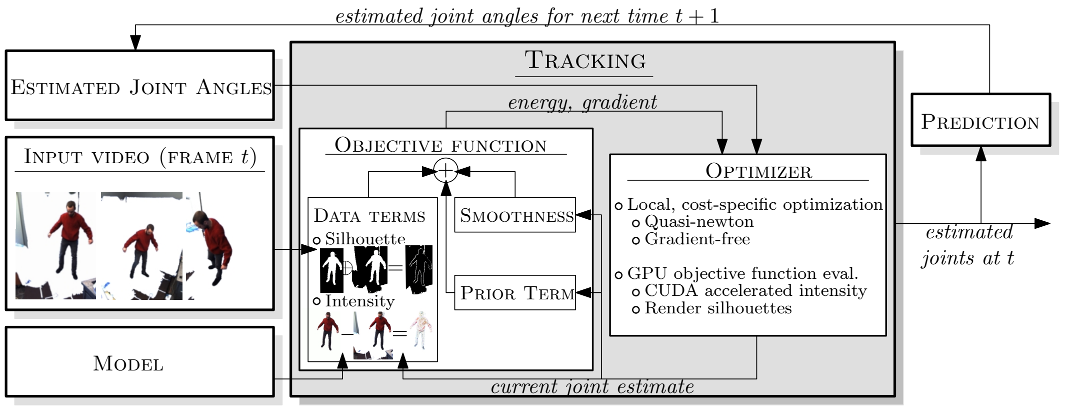


Figure 5.1: An overview of the energy-based tracking framework.

for the problem of rigid body tracking already give real-time results (e.g., region-based [189] and appearance-based [157, 215, 214]). Similarly, in this chapter, the intensity-based tracking leads to an efficient implementation.

As the local methods may fail to track through fast motions, the efficiency of the trackers is important if a manual operator needs to monitor the tracking progress and help recover from mistracks if necessary (e.g., [259]). To achieve an efficient implementation, our optimization uses the analytic Jacobian of the linear-blend skinned mesh, and the GPU is used to improve efficiency of evaluation and optimization of the energy functions. Specifically, through a CUDA implementation of intensity-based tracking, speed-ups of up to $10\times$ are possible, enabling real-time tracking of simple kinematic structures. Such real-time tracking also allows for interactive applications.

We demonstrate that these simple local methods work well and are capable of tracking a wide range of actions. Sensitivity of the trackers to bone placement is analyzed in synthetic sequences, and the trackers are compared on real-data captured from several multi-camera studios. The evaluation suggests that the XOR silhouette energy is more stable through noisy silhouettes, but is more costly to evaluate than the iterative closest point approach that works well on clean silhouette data. Also, the diversity of the linear-blend skinned mesh is illustrated in tracking non-rigid facial animations.

Figure 5.1 gives an overview of the components that make up the multi-view tracking described in this chapter. The energy function used in the tracking is composed of several terms. The details of the tracking energy and the energy terms are presented first in Section 5.2. Depending on the data terms used (and whether the derivatives are available), the optimization method may vary. The optimization methods for each of the data energies are discussed in detail in Section 5.3, followed by the details necessary to achieve an efficient implementation (Section 5.4).

5.2 Tracking Formulation

As input we have video streams, $\{I_{i,t}\}_{t=1}^{T}$, taken from $1 \leq i \leq C$ calibrated cameras with projection matrices, $\mathbf{P}_i$. Further, assume that a geometric model and an attached skeleton are available for tracking (see Chapter 3). The model is parameterized at each frame by a set of joint angles, $\mathbf{\Theta}_t$, and the objective of tracking is to recover the time-varying joint angles for the entire sequence, $\{\mathbf{\Theta}_t\}_{t=1}^{T}$.

Similar to the formulation used in static model recovery (Chapter 4), recovering the joint parameters is accomplished by minimizing an energy function. In contrast, as there may be a large number of frames, the objective function is defined and optimized per frame independently:

$$E_{\text{track}}(\mathbf{\Theta}_t) = E_{\text{data}}(\mathbf{\Theta}_t) + \lambda_{\text{prior}} E_{\text{prior}}(\mathbf{\Theta}_t) + \lambda_{\text{smooth}} E_{\text{smooth}}(\mathbf{\Theta}_t, \mathbf{\Theta}_{1:(t-1)}). \qquad (5.1)$$

The E_{data} term measures the agreement of the model posed with parameters to the input images, the E_{prior} term is used to penalize unlikely pose configurations, and E_{smooth} is used to ensure smoothness of the recovered joint angle trajectory. Below we discuss several options for these terms.

5.2.1 Data Energy

Two independent alternatives for data energies, E_{data}, that utilize silhouette information are considered. Furthermore, a second type of data energy that directly utilizes image intensity information is also considered. This intensity-based term can be combined with either of the independent silhouette energies.

Silhouette XOR Energy

As in Chapter 4, a straightforward way to measure the difference between a rendered model and the input silhouettes is to take the exclusive-or between input silhouettes, $S_{i,t}$, and rendered model silhouettes [245]:

$$E_{\text{sil}}(\mathbf{\Theta}_t) = \sum_{i=1}^{C} \sum_{\mathbf{x}} \frac{S_{i,t}(\mathbf{x}) \otimes P_{i,t}(G(\mathbf{\Theta}_t), \mathbf{x})}{\text{width}(I_{i,t}) * \text{height}(I_{i,t})}, \qquad (5.2)$$

where the shorthand $P_{i,t}(G(\mathbf{\Theta}_t))$ denotes the projected silhouette of the geometry by $\mathbf{P}_{i,t}$.

Ideally, the silhouette term would penalize portions of the model that lie far from the input silhouette more. To achieve this goal, instead of taking the XOR between rendered and model silhouettes, the difference between the signed distance map of the input silhouette and rendered silhouettes can be taken; alternatively, the silhouettes can be smoothed and the image difference can be used [144]. Unfortunately, these energies are costly to evaluate as either the signed distance map of the projected model must be computed or the model silhouette must be smoothed during optimization. Alternative asymmetric approaches only make use of the signed distance map of the input image [96], saving computation of the signed distance function of the projected model. We restrict our implementation to the XOR as it is more efficient than methods that require either smoothing or distance maps, and because it works well in practice.

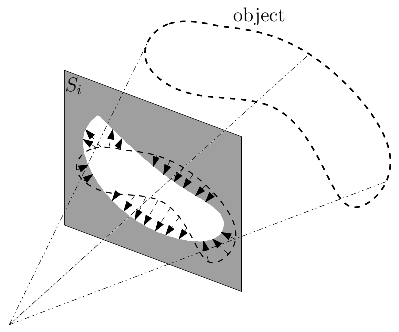


Figure 5.2: The silhouette ICP energy tries to minimize the distance between points on the projected model silhouette to the input silhouette. In this figure, the energy is equivalent to the lengths of the arrows.

Silhouette Closest Point Energy

An alternative to the above XOR approach is to minimize the distance between the projection of the occluding contour and the input silhouette. Such an energy assigns more penalty to the parts of the input model that are far away from the image silhouette:

$$E_{\text{icp}}(\mathbf{\Theta}_t) = \sum_{i=1}^{C} \oint_{\partial P_{i,t}(G(\mathbf{\Theta}_t))} |D_i(s)|^2 ds, \tag{5.3}$$

where $\partial P_{i,t}(G(\mathbf{\Theta}_t))$ is the projected contour of the model in image i posed by $\mathbf{\Theta}_t$, s is the arc length parameterization of the contour, and

$$D_i(\mathbf{x}) = \min_{\mathbf{x}' \in \partial S_{i,t}} \|\mathbf{x} - \mathbf{x}'\|, \tag{5.4}$$

is the distance to the image contour, $\partial S_{i,t}$. Intuitively, as illustrated in Figure 5.2, the energy at each point is proportional to the distance to the image silhouette.

As we will see in Section 5.3.2, this objective function is minimized by iteratively applying constraints to the occluding contour of the object such that it better approximates the image silhouette. This is similar to approaches that minimize the distance between points on the contour and backprojected Plücker lines [195, 238]. Due to the nature of the minimization, this energy is referred to as the image iterative closest point (ICP) energy [286].

Sum of Squared Difference (SSD) of Intensity Energy

The above silhouette terms require that silhouettes are available for the tracking. Although several region-based extensions can segment the image while tracking [207, 189, 195, 69], these silhouette and region-based methods provide no way to resolve poses that do not influence the silhouette of an object. As an example, roll of the forearm may not be distinguishable by silhouettes alone, but image texture information (e.g., on the hand) may help discriminate the pose. For this reason, it is useful to consider an energy that takes into account intensity or texture changes on the surface of the object. In the human tracking literature, image intensity differences are used by parameterizing optical flow through the underlying model [40, 244], and these flow-based terms have been shown to improve tracking when combined with silhouettes [244]. In contrast to these flow-like approaches [82], and more similar to the 2D template-based trackers [17, 30] and morphable model intensity-based trackers [82], we utilize a set of reference colors on the object vertices and seek to align these with the input images.

Let us assume that each model vertex $\mathbf{v}_j = [x_j, y_j, z_j, 1]^\intercal$, with $1 \leq j \leq \#_{\text{points}}$, is associated with an RGB color $R(\mathbf{v}_j)$. The reference colors can either be provided by an input texture map, or they can be extracted from the previous frame (in which case the method is similar to the model-based optic-flow methods, e.g. [40, 244]).

Recall from Chapter 3, in linear-blend skinning that the vertex position, $\mathbf{v}_j$, is a function of the joint angles, $\mathbf{\Theta}$. Let us represent this dependence explicitly as

$$f_j(\mathbf{v}_j, \mathbf{\Theta}_t) = \sum_{b \in B(j)} w_{jb} \mathbf{T}_b(\mathbf{\Theta}_t)\mathbf{v}_j, \tag{5.5}$$

where $\mathbf{T}_b(\mathbf{\Theta}_t) = \mathbf{M}_b(\mathbf{\Theta}_t)\mathbf{M}_b(\mathbf{0})^{-1}$ is the relative transformation for bone b and $B(j) = \{b | w_{jb} > 0\}$.

The image intensity term then follows the 2D template-based tracking approaches. Assuming brightness constancy holds, the following energy function relates input observations to the template colors of the vertices using a sum-of-squared differences (SSD):

$$E_{\text{tex}}(\mathbf{\Theta}_t) = \sum_{i=1}^{C} \sum_j \nu_{ij}^2 |I_{i,t}(\underbrace{\Pi(\mathbf{P}_i f_j(\mathbf{v}_j, \mathbf{\Theta}_t))}_{g_{ij}(\mathbf{v}_j, \mathbf{\Theta}_t)}) - R(\mathbf{v}_j)|^2, \tag{5.6}$$

where ν_{ij} are weighting terms that encode both visibility and reliability of the observation of point j in image i. The function, $g_{ij}(\mathbf{v}_j, \mathbf{\Theta}_t)$, is a shorthand for the deformation of a point, j, from world coordinates that is projected into image I_i.

5.2.2 Pose Prior

In practice, joint angle limits can be enforced during the optimization. However, many joint angle configurations, even within the joint angle limits, may be unlikely. Such unlikely joint angle configurations can be penalized through the use of a prior energy that favors plausible pose configurations. Following Gall *et al.* [96], the pose prior is modeled with a non-parametric density:

$$E_{\text{prior}}(\mathbf{\Theta}_t) = -\log(p(\mathbf{\Theta}_t)). \tag{5.7}$$

The upper and lower body are assumed to be independent:

$$p(\mathbf{\Theta}_t) = p_{\text{lleg}}(\mathbf{\Theta}_t) p_{\text{rleg}}(\mathbf{\Theta}_t) p_{\text{larm}}(\mathbf{\Theta}_t) p_{\text{rarm}}(\mathbf{\Theta}_t) p_{\text{back}}(\mathbf{\Theta}_t). \tag{5.8}$$

$p_{\text{lleg}}, p_{\text{rleg}}, p_{\text{larm}}, p_{\text{rarm}}, p_{\text{back}}$ depend only on the parameters of the part and are modeled through a Parzen-Rossenblatt density using an adaptive, anisotropic kernel:

$$p_*(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_i(\mathbf{x} - \mathbf{x}_i). \tag{5.9}$$

where for each part, $\{\mathbf{x}_i\}_{i=1}^n$, are a set of training samples of the concatenated joint angles for the part (e.g., using motions from the publicly available CMU motion capture database [60]). The use of an adaptive, anisotropic kernel allows the density estimator to better model the local density (e.g., by using a smoother kernel in regions with fewer samples). The adaptive kernel, K_i, is defined as follows [43]:

$$K_i(\mathbf{x}) = \frac{1}{\sqrt{|\mathbf{\Sigma}_i^{-1}|(2\pi)}} \exp\left(-\frac{1}{2}\mathbf{x}^\intercal \mathbf{\Sigma}_i^{-1}\mathbf{x}\right), \tag{5.10}$$

where

$$\mathbf{\Sigma}_i = \lambda\mathbb{I} + \sum_{i=1}^{n} K_h(|\mathbf{x}_i - \mathbf{x}_j|)(\mathbf{x}_i - \mathbf{x}_j)(\mathbf{x}_i - \mathbf{x}_j)^\intercal, \tag{5.11}$$

and K_h is a Gaussian kernel with bandwidth h:

$$K_h(r) = \frac{1}{\sqrt{(2\pi h^2)^d}} \exp\left(-\frac{r^2}{2h^2}\right). \tag{5.12}$$

The parameter h is heuristically estimated from the training data using the average distance to the fifth nearest neighbor, and $\lambda = \frac{h}{20}$.

5.2.3 Smoothness

There are a variety of alternatives for enforcing a reconstruction of smoothly varying joint angles. For example, a non-parametric density can be used to model the joint probability density of the poses over time [43], or Gaussian process regression can be used to obtain an autoregressive prediction used as a smoothness prior [96]. In contrast to these elaborate approaches, simple temporal smoothness based on an estimate of the velocity has been shown to work well [197]. We define the prediction, $f_{\text{pred}}(\mathbf{\Theta}_{1:(t-1)})$, based on a backward difference estimate of the velocity using the previous two time steps:

$$f_{\text{pred}}(\mathbf{\Theta}_{1:(t-1)}) = \underbrace{(\mathbf{\Theta}_{t-1} - \mathbf{\Theta}_{t-2})}_{\text{velocity}*\text{dt}} + \mathbf{\Theta}_{t-1}. \tag{5.13}$$

The smoothness term ensures that the recovered joint angles lie close to this prediction:

$$E_{\text{smooth}}(\mathbf{\Theta}_t, \mathbf{\Theta}_{1:(t-1)}) = |\mathbf{\Theta}_t - f_{\text{pred}}(\mathbf{\Theta}_{1:(t-1)})|^2 = |\mathbf{\Theta}_t - 2\mathbf{\Theta}_{t-1} + \mathbf{\Theta}_{t-2}|^2. \tag{5.14}$$

5.3 Optimization

Section 5.2 discussed the various components that make up our energy function used in tracking. All of the E_{data} terms are non-linear and non-convex as they project into and look up values from

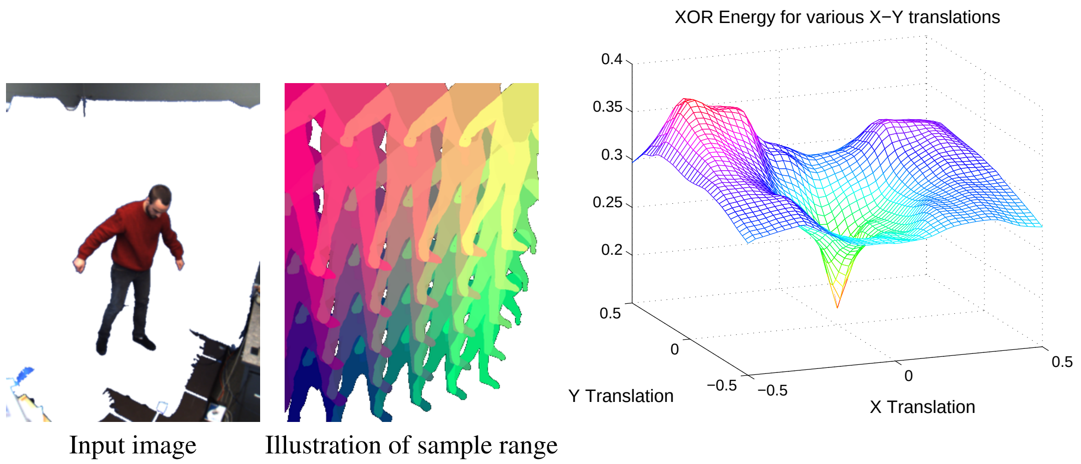


Figure 5.3: Illustration of the non-convex landscape of the silhouette XOR energy for one image. The 3D model was shifted in the world x-y plane and the cost was evaluated.

an underlying arbitrary image. As an example, the energy landscape for the XOR function by varying the two translation parameters for a 3D human model are plotted in Figure 5.3. The simple smoothness term is convex. The prior term uses a non-parametric density that is likely multimodal, implying that it is also non-convex. The resulting combined objective is a non-linear, non-convex function. In order to ensure a global minimum in all environments, global optimization methods, such as particle filters [96, 18], should be used. However, recent local-global approaches note that the global optimization is only typically needed when there is fast motion or when silhouettes are poor [97]. In this way, the slow global optimization can be invoked only when necessary.

In many cases, simply performing local optimization is sufficient, and reasonable results have been obtained on a variety of formulations (e.g., ICP [195, 110], XOR [245], visual-hull [175], combined feature-point and ICP [94], combined flow and ICP [22]). In this section, local optimization schemes for the various data energies are discussed.

5.3.1 Silhouette XOR

The XOR energy accumulates binary values over a finite set of pixels, meaning it is a discrete valued function of continuous variables. For this reason, the energy is often minimized without requiring the derivatives, e.g., by using a coordinate descent or through Powell's method [245]. These methods iteratively perform a line search along a set of chosen descent directions. To reduce the high dimensional search space, it is common to perform a hierarchical optimization [245], where the parameters of the trunk of the body are first optimized (holding all others fixed), and subsequently each kinematic chain (the arms, and legs) are refined (holding the other parameters fixed).

An advantage of using gradient free minimization is that the prior, smoothness terms, and other data energy terms are easily incorporated (as derivatives are not required). However, optimization in this manner may require many energy function evaluations. Even when an efficient evaluation of the objective functions is used (§5.4.1), the function evaluations are still costly and can cause such methods to be slow.

5.3.2 Silhouette ICP

The energy function given in Eq. 5.3 can be approximately minimized by successively applying a set of constraints to the model that pull its boundary closer to the input silhouette. For a given approximation of the model parameters, $\mathbf{\Theta}_t^1 = f_{\text{pred}}(\mathbf{\Theta}_{1:(t-1)})$, let $\{\mathbf{v}_j^i\}_{j=1}^{M_i}$ be the set of 3D model points lying on the discrete projected contour for an image i.

For each of the model points on the projected contour, the closest corresponding point on the input silhouette contour, $S_{i,t}$ is found:

$$\tilde{\mathbf{u}}_j^i = \underset{\mathbf{u} \in \partial S_{i,t}}{\operatorname{argmin}} |\mathbf{u} - \Pi(\mathbf{P}_i \mathbf{v}_j^i)|^2. \tag{5.15}$$

A better estimate of the joint angles is then obtained by minimizing the following equation using the current correspondences:

$$\mathbf{\Theta}_t^{k+1} = \underset{\mathbf{\Theta}}{\operatorname{argmin}} \sum_i \sum_{j=1}^{M_i} |\tilde{\mathbf{u}}_j^i - \Pi(\mathbf{P}_i \mathbf{v}_j^i)|^2. \tag{5.16}$$

The resulting joint angles approximately minimize the original objective function, Eq. 5.3.

In order to solve Eq. 5.16, the points on the projected model contour are represented as a barycentric combination of the vertices that make up the triangle containing the point:

$$\mathbf{v}_j^i = \sum_{k=1}^{3} \lambda_{jk}^i f_{[A(j)]_k}(\mathbf{v}_{[A(j)]_k}, \mathbf{\Theta}_t^k), \tag{5.17}$$

where $A(j)$ is the triangle containing point j, $[A(j)]_k$ is the k^{th} vertex in the triangle, and λ_{jk}^i are the barycentric coordinates for point j on the contour of image i.

Equation 5.16 can be solved using a non-linear minimizer or it can be seen as a non-linear least squares problem, for which Levenberg-Marquardt can be used. The sequence of constraining the current model points to lie on the silhouette can be iterated using the previous solution, which gives improved estimates of $\mathbf{\Theta}_t^k$. The steps are summarized for a maximum number of iterations, $\#_{\text{iters}}$, in Algorithm 1.

Algorithm 1: ICPTrackFrame

```
$\mathbf{\Theta}_t^1 = f_{\text{pred}}(\mathbf{\Theta}_{1:(t-1)})$ ;
while not converged and $k \leq \#_{iters}$ do
    Project current model in input images using $\mathbf{\Theta}_t^k$ ;
    Find model contour points, $\{\mathbf{v}_j^i\}_{j=1}^{M_i}$, on projected model ;
    Find corresponding closest points, $\{\mathbf{u}_j^i\}$ ;
    Solve Eq. 5.16 for $\mathbf{\Theta}_t^{k+1}$ ;
```

The analytical derivatives of the projected model point are readily obtained as:

$$\underbrace{\frac{\partial}{\partial \boldsymbol{\Theta}} \Pi(\mathbf{P}_i \mathbf{v}_j^i)}_{2\times\#\text{params}} = \begin{bmatrix} \frac{1}{\hat{z}} & 0 & \frac{\hat{x}}{\hat{z}^2} \\ 0 & \frac{1}{\hat{z}} & \frac{\hat{x}}{\hat{z}^2} \end{bmatrix} \mathbf{P}_i \underbrace{\frac{\partial \mathbf{v}_j^i}{\partial \boldsymbol{\Theta}}}_{4\times\#\text{params}} . \tag{5.18}$$

With

$$[\hat{x}, \hat{y}, \hat{z}]^\intercal = \mathbf{P}_i \mathbf{v}_j^i, \tag{5.19}$$

$$\frac{\partial \mathbf{v}_j^i}{\partial \theta_k} = \sum_{k=1}^{3} \lambda_{jk}^i \frac{\partial}{\partial \theta_k} f_{[A(j)]_k}(\mathbf{v}_{[A(j)]_k}, \boldsymbol{\Theta}_t). \tag{5.20}$$

Where the mesh Jacobian, $\frac{\partial}{\partial \theta_k} f_j(\mathbf{v}_j, \boldsymbol{\Theta}_t)$, is

$$\frac{\partial}{\partial \theta_k} f_j(\mathbf{v}_j, \boldsymbol{\Theta}_t) = \sum_{b \in B(j)} w_{jb} \frac{\partial \mathbf{T}_b}{\partial \theta_k} \mathbf{v}_j. \tag{5.21}$$

$\frac{\partial \mathbf{T}_b}{\partial \theta_k}$ is non-zero for all θ_k that are in the ancestor chain of parameter k. $\frac{\partial f_j}{\partial \theta_k}$ is zero for those parameters that are not in the ancestor chain for some bone in $B(j)$.

Combining the ICP Term with Smoothness and Prior Terms

The smoothness energy can easily be incorporated into the minimization in Eq. 5.16. The smoothness can also be represented as a non-linear equation system, therefore, a Levenberg-Marquardt solver can be used [165]. The prior cannot be expressed as a non-linear system of equations, so a non-linear minimizer must be used when solving with a prior. The LBFGS-B package is used as an optimizer when the problem cannot be cast as a non-linear least squares formulation [288].

5.3.3 Image SSD Minimization

Similar to the silhouette ICP, the SSD score can also be minimized by the use of quasi-Newton methods, such as Gauss-Newton or Levenberg-Marquardt, to solve the non-linear least squares problem in Eq. 5.6. Quasi-Newton methods solve non-linear systems by iteratively solving a linear approximation to the non-linear system. To make the structure of the optimization clear, here we derive an optimization method directly by linearizing the data term. This formulation is an extension of intensity-based tracking of 2D templates [17], rigid objects [157, 215, 214], and morphable models [82] to a linear-blend skinned mesh.

Given an approximation to the joint angles, $\hat{\boldsymbol{\Theta}}_t^k$ (e.g., $\hat{\boldsymbol{\Theta}}_t^1 = f_{\text{pred}}(\boldsymbol{\Theta}_{0:(t-1)})$), the motion at time t can be estimated using a fixed-point iteration: $\hat{\boldsymbol{\Theta}}_t^{k+1} \leftarrow \hat{\boldsymbol{\Theta}}_t^k + \delta\boldsymbol{\Theta}_t^k$. Linearizing the image observation at $\hat{\boldsymbol{\Theta}}_t^k$ gives the following approximation:

$$I_{i,t}(g_{ij}(\mathbf{x}_j, \hat{\boldsymbol{\Theta}}_t^k + \delta\boldsymbol{\Theta}^k)) \approx I_{i,t}(g_{ij}(\mathbf{x}_j, \hat{\boldsymbol{\Theta}}_t^k)) + \underbrace{\nabla I_{i,t} \frac{\partial g_{ij}}{\partial \boldsymbol{\Theta}}}_{G_{i,j}} \delta\boldsymbol{\Theta}^k \tag{5.22}$$

Substituting Eq. 5.22 into Eq. 5.6 yields:

$$\sum_{i=1}^{C}\sum_{j=1}^{\#\text{points}} \nu_{ij}^2 |I_{i,t} - R(\mathbf{v}_j) + \underbrace{\left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right]}_{1\times\#\text{params}} \delta\mathbf{\Theta}^k|^2, \tag{5.23}$$

where, for brevity, the indexing of $I_{i,t}$ and $\nabla I_{i,t}$ with $g_{ij}(\mathbf{x}_j, \mathbf{\Theta}_t^k)$ has been dropped. Differentiating Eq. 5.23 with respect to $\delta\mathbf{\Theta}^k$ and setting to zero, gives a $\#_{\text{params}} \times \#_{\text{params}}$ linear equation system in $\delta\mathbf{\Theta}^k$:

$$\begin{gathered}\sum_{i=1}^{C}\sum_{j=1}^{\#\text{points}} \nu_{ij}^2 \left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right]^{\mathsf{T}} (I_{i,t} - R(\mathbf{v}_j)) + \\ \sum_{i=1}^{C}\sum_{j=1}^{\#\text{points}} \nu_{ij}^2 \left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right]^{\mathsf{T}} \left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right] \delta\mathbf{\Theta}^k = \mathbf{0}\end{gathered} \tag{5.24}$$

The solution to Eq. 5.24 (and minimizer of Eq. 5.23) is then:

$$\delta\mathbf{\Theta}^k = \mathbf{H}^{-1} \underbrace{\sum_{i=1}^{C}\sum_{j=1}^{\#\text{points}} \nu_{ij}^2 \left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right]^{\mathsf{T}} \underbrace{(R(\mathbf{v}_j) - I_{i,t})}_{e_{i,j}}}_{Gtb}. \tag{5.25}$$

with

$$\mathbf{H} = \lambda\mathbb{I} + \sum_{i=1}^{C}\sum_{j=1}^{\#\text{points}} \nu_{ij}^2 \left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right]^{\mathsf{T}} \left[\nabla I_{i,t}\frac{\partial g_{ij}}{\partial \mathbf{\Theta}}\right] \tag{5.26}$$

$\mathbf{H}$ is usually regarded as the Gauss-Newton approximation to the Hessian of the original non-linear equations (Eq. 5.6). Throughout this work, $\mathbf{H}$ will be referred to as the Hessian matrix. The $\lambda\mathbb{I}$ term is added to ensure $\mathbf{H}$ is non-singular; this makes the behavior of our iterative solution similar to Levenberg-Marquardt.

The basic steps of the tracking algorithm to track a single frame using the E_{tex} energy as a data cost are outlined in Algorithm 2. Visibility weights are calculated as

$$\nu_{ij} = \begin{cases} 0 & \text{if } \mathbf{x}_j \text{ is not visible in camera i} \\ \langle \mathbf{n}_j, \mathbf{d}_{ij} \rangle & \text{otherwise} \end{cases}$$

where $\mathbf{d}_{ij}$ is the unit length vector pointing from $\mathbf{x}_j$ to the camera center, and $\mathbf{n}_j$ is the surface normal at $\mathbf{x}_j$.

Algorithm 2: SSDTrackFrame

```
Obtain input images, $I_{i,t}$, and compute image gradients, $\nabla I_{i,t}$ ;
Update the visibility and point-wise weights $\nu_{ij}$ using the z-buffer ;
Use the previous parameters as an initial estimate: $\mathbf{\Theta}_t^1 = f_{\text{pred}}(\mathbf{\Theta}_{0:(t-1)})$ ;
while not converged and $k \le \#_{iters}$ do
    Compute image differences, $e_{i,j}$, and $\mathbf{H}$ ;
    Update $\mathbf{\Theta}_t^{k+1} = \mathbf{\Theta}_t^k + \delta\mathbf{\Theta}_t$ using Eq.5.25 ;
```

	XOR	ICP	SSD	Prior	Smooth
XOR	Co. Descent	Co. Descent	Co. Descent	Co. Descent	Co. Descent
ICP	Co. Descent	Non-lin. Lst Sqr	Non-lin. Lst Sqr	Non-lin. Min.	Non-lin. Lst Sqr
SSD	Co. Descent	Non-lin. Lst Sqr	Non-lin. Lst Sqr	Non-lin. Min.	Non-lin. Lst Sqr

Table 5.1: Combining data terms with other terms. When combining terms, the least general minimizer possible is preferred. Gradient-free minimization is necessary whenever the XOR term is used. Non-linear minimization is required whenever the prior term is used. All the other combinations could use non-linear minimization or gradient-free minimization, but they are actually non-linear least squares problems.

Combining the SSD Term with Smoothness and Prior Terms

Similar to the ICP data energy, the smoothness energy can easily be incorporated into the SSD minimization. Again, to make use of the prior, a more general non-linear minimizer must be used.

.

5.3.4 Combining Data Terms

So far, the details for optimizing individual data terms were discussed. The SSD and ICP energies, are best formulated as non-linear least squares problems and can make use of appropriate solvers. These data energies are easily combined with the prior and smoothness terms, but the prior term cannot be expressed as a non-linear least squares problem implying that a more general non-linear minimizer must be used (Table 5.1).

A similar argument holds for combining data terms. The SSD term can easily be combined with the XOR term to obtain a new data term, e.g., $E_{\text{data}} = \lambda_{\text{ssd}} E_{\text{ssd}} + E_{\text{XOR}}$ (referred as XOR+SSD). However, as the XOR must be minimized with a coordinate descent method, the combined energy must also use the coordinate descent method.

The ICP silhouette and SSD data energies can also easily be combined (ICP+SSD). Although possible to couple these using a nonlinear least squares solver, in our implementation this is done through a quasi-newton minimizer (LBFGS-B).

5.4 Implementation Details

The previous section described the optimization methods used to minimize the various energy functions. The efficiency of the optimization is directly tied to the efficiency of the objective function evaluation. In this section, the implementation details of the data terms are discussed.

5.4.1 Silhouette XOR Implementation

In order to efficiently evaluate the XOR energy, the input silhouettes are packed into the bit planes of a single image. The GPU is then used to render the model silhouette from each view into individual bit planes of the framebuffer. The model silhouettes are read back from the framebuffer and the

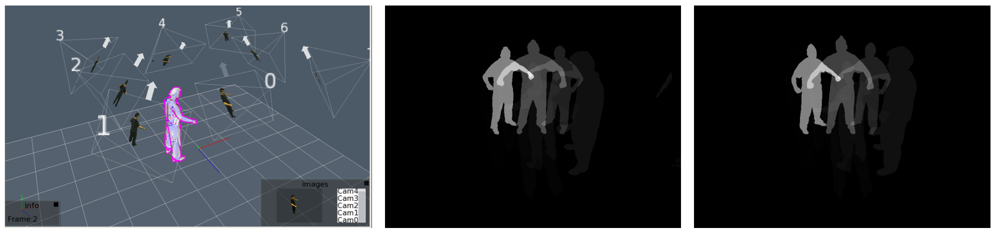


Figure 5.4: The silhouettes of the input images and rendered silhouettes are packed into bit planes of a single image. The XOR term is evaluated per-pixel by performing a bit-wise XOR and counting the number of bits that are set.

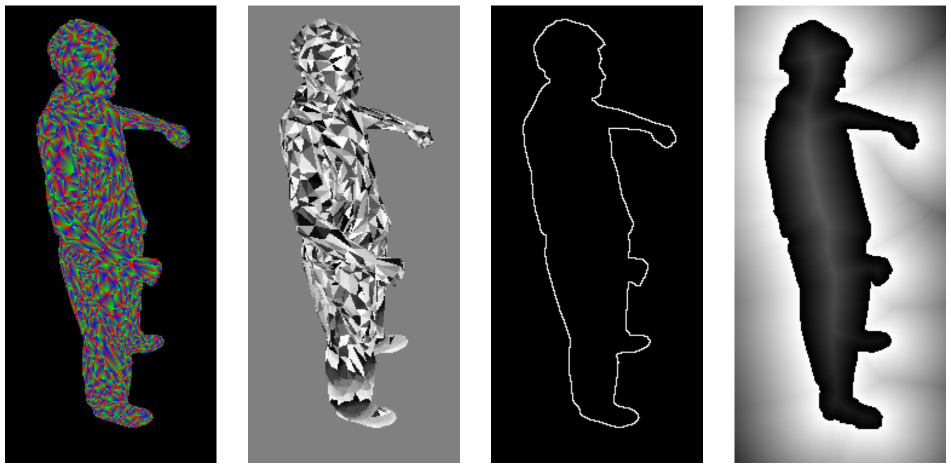


Figure 5.5: The points on the model that lie on the occluding contour are easily found by rendering the model triangles with barycentric colors (e.g., vertex 1 uses red, vertex 2 green, vertex 3 blue). The triangle indices are also rendered; the points on the projected model contour can then directly index the point on the model. The *Input distance map* is used to quickly find the corresponding image contour point.

XOR function is efficiently computed per-pixel through a binary XOR (Figure 5.4). The number of bits set are counted using a lookup table. Implementation in this manner limits the amount of data transferred back from the GPU and enables fast aggregation of the XOR cost.

The above functionality can be further sped up by either distributing the computation amongst many computers or by limiting the image region where the XOR is computed [242].

5.4.2 Silhouette ICP Implementation

The ICP optimization discussed in Section 5.3.2 has two time consuming components: 1) find the model points on the contour of a projected model, and 2) find the closest points on the input silhouette for each of the points on the model contour.

Graphics hardware is used to efficiently find the model points on the contour of the projected model. For each view, the model is rendered twice: once to create a triangle label image and a second time to create the barycentric weight image. The barycentric weight image is obtained by setting the colors of the vertices in a triangle according to their order in the triangle. Figure 5.5 illustrates these two images. Model points on the projected silhouette are then found on the boundary of the triangle image; the rendered triangle indices give the corresponding model triangle, and the rendered barycentric weights identify the position within the triangle.

The closest corresponding point on the image contour can be efficiently computed through the use of a precomputed distance map. Danielsson's approximate distance map algorithm is used to precompute the distance map for each of the input images [70]. This precomputation is useful as the distance maps can be reused on all iterations of ICP for the frame. The distance map algorithm precomputes the distance to the contour and also the offset to the closest point.

For efficiency, we can make some approximations with the ICP approach. Solving the constraints at each iteration is a non-linear problem (Eq. 5.16), but as the association of closest points change on each iteration, the equation system need not be solved until convergence. Therefore, a first efficiency benefit comes from only applying a few iterations of Levenberg-Marquardt (or a non-linear solver) when solving the constraints at each iteration.

The speed of the constraint solver is proportional to the number of constraints. There is one constraint for each model contour pixel in the each image. To improve efficiency, only a random subset of these constraints (e.g., 50%) need to be used.

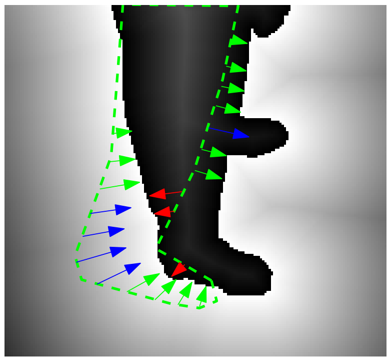


Figure 5.6: Illustration of accepted (green), length limited (blue), and rejected (red) constraints.

A final improvement comes through filtering the constraints (Figure 5.6). First, the distance to the corresponding image silhouette point from the model contour is clamped to a maximum of 10 pixels. This helps avoid problems with noisy silhouettes. Second, simply looking up the closet pixel in the distance map does not guarantee that the constraint will move the surface in the correct direction. To further reject bad matches, the model surface normal is used. If the model point is outside the object, the constraint is only accepted if it moves the surface in a direction opposite the surface normal. If the surface point is currently inside, the constraint is only accepted if it moves in the direction of the surface normal. This filtering helps the tracker converge from a wider range. The red arrows in Figure 5.6 illustrate constraints that were rejected due to this normal constraint. This surface normal filtering is similar to the 3D ICP methods that choose the closest point based on a combination of geometric distance and normal distance [175].

5.4.3 SSD Implementation

Efficiency of registration-based tracking has been addressed thoroughly in the context of 2D SSD tracking. For example, the inverse compositional algorithm (IC) avoids recomputing the Hessian at each iteration by computing it on the template once [17]. This technique has been applied to 3D tracking but only for restricted camera models. Furthermore, the Hessian/Jacobian are still view-dependent and must be recomputed occasionally, for example, when the viewpoint changes roughly 20 degrees [194, 277].

More recently, factorization-based approaches have been extended to 3D MM's [82]. The Jacobian is factored into static components pertaining to the structure of the object (e.g., surface normals and vertex positions) and dynamic components based on the current motion parameters. This pre-computation speeds up a tracking iteration (roughly 5 times for rigid objects but only 2 times for MM's with more parameters) but requires some extra caching of this partial data. Although these techniques were applied to 3D morphable models, it is not clear that they can be extended to a skinned mesh.

```
Function culteration
  Data: $w_{ij}$: a full matrix of skinning weights
        $v_{ij}$: the cost weights (including visibility)
        $\mathbf{\Theta}$: the current estimate of parameters
  // Transform points and compute jacobian;
  $\mathbf{X}$ = transformPoints (bones, $\mathbf{\Theta}$, $\{\mathbf{x}_i\}$);
  $J_{mesh}$ = cuGetMeshJacobian (bones, $\mathbf{\Theta}$, $w_{ij}$);
  H = zeroMatrix($\#_{parms}$, $\#_{parms}$);
  // Process visible points in each view;
  for $i \in 1$ to $C$ do
      // Extract visible points and transfer to GPU $d_i = [j|\nu_{ij} > 0]$;
      $\nu_i = [\nu_{ik}|k \in d_i]$;
      // Extract visible colors and points;
      $R_i$ = cuIndexVector ($R$, $d_i$);
      $\mathbf{X}_i$ = cuIndexVector ($\mathbf{X}$, $d_i$);
      for $k \in 1$ to $\#_{parms}$ do
          $J_i[k]$ = cuIndexVector ($J_{mesh}[k]$, $d_i$);
      $[\mathfrak{C}, p, dr, dg, db]$ = cuSample ($\mathbf{P}_i$, $\mathbf{X}_i$, $\nu_i$);
      for $k \in 1$ to $\#_{parms}$ do
          $G_i[k]$ = cuGrad ($\mathbf{P}_i$, $p$, $dr$, $dg$, $db$, $J_i[k]$);
      $\Delta\mathfrak{C}$ = cuColorDiff ($\mathfrak{C}$, $R_i$, $\nu_i$);
      accumGradDiff ($G_i$, $\Delta\mathfrak{C}$, H, Gtb);
  return $\mathbf{\Theta}$ + $H^{-1}$ Gtb; // Compute parameter update
```

For these reasons, we opt for a GPU accelerated implementation. The majority of a tracking iteration naturally maps to a CUDA implementation. The function *cuIteration* illustrates this idea. As an initial step, the skinning weights w_{ij} and image textures are transferred to GPU memory. The vertices are transformed through f_j and transferred to the GPU. Then the mesh Jacobian, J_{mesh} (e.g., column i is $\frac{\partial f_j}{\partial \theta_i}$ stacked for all j), which is common to all views, is computed on the GPU (*cuGetMeshJacobian*). For each view, a list of the visible points (e.g., d_i contains indices of points with $\nu_{ij} > 0$) and their corresponding weights are compacted into ν_i. The index vector, d_i, is used as a texture function to extract the associated visible points, $\mathbf{X}_i$, their reference colors, R_i, and the corresponding rows of the mesh Jacobian, J_i.

```
Function cuGrad($\mathbf{P}_i$, p, dr, dg, db, J)
    parallel cuda kernel over k;
    float du, dv;
    float3 $d_p$ = $\mathbf{P}_i$ J[k];
    du = (p[k].z*$d_p$.x - p[k].x*$d_p$.z)/(p[k].z*p[k].z);
    dv = (p[k].z*$d_p$.y - p[k].y*$d_p$.z)/(p[k].z*p[k].z);
    gout[k].x = (dr[k].x*du + dr[k].y*dv);
    gout[k].y = (dg[k].x*du + dg[k].y*dv);
    gout[k].z = (db[k].x*du + db[k].y*dv);
```

```
Function accumGradDiff($G_i$, $\Delta\mathfrak{C}$, H, Gtb)
    // Read back Jacobian and colors;
    Vector<float3> $\Delta\mathfrak{C}^H$ = $\Delta\mathfrak{C}$;
    Vector<float3> $G_i^H$[$\#_{\text{params}}$];
    for $k \in 1$ to $\#_{params}$ do
        $G_i^H[k]$ = $G_i[k]$
    for $ii \in 1$ to $\#_{params}$ do
        for $vi \in 1$ to $\#_{points}$ do
            Gtb[$ii$] += $\langle G_i^H[ii][vi], \Delta\mathfrak{C}^H[vi] \rangle$
        for $j \in 1$ to $\#_{params}$ do
            for $v \in 1$ to $\#_{params}$ do
                H[$ii, j$] += $\langle G_i^H[ii][v], G_i^H[j][v] \rangle$
```

The subset of transformed points, $\mathbf{X}_i$, are projected into the input image through *cuSample*, giving projected points p, the corresponding colors, $\mathfrak{C}$, and the derivatives of the image colors at these points (in dr, dg, db). These derivatives are premultiplied by ν_i. The combined Jacobian, $G_i[k]$, is then computed for each parameter (*cuGrad*), followed by the color difference computation in *cuColorDiff*, which again multiplies the differences with the weights in ν_i.

The final step accumulates the Jacobian into the Hessian, H, and accumulates the product of the Jacobian with the residuals into *Gtb*. The code listing here performs this accumulation by reading back the results (*accumGradDiff*). We have also experimented with CUDA-based reduction algorithms and found reading back to be a faster solution (for our examples and hardware).

This implementation ensures that almost all of the computation is done on the GPU. In the above pseudo-code, all of the functions prefixed with *cu* are evaluated on the GPU. The only data transferred to the GPU per iteration are the transformed points and the per-camera visibility/weights list. This implementation of *accumGradDiff* requires that the columns of the Jacobian and the color differences are transferred from the GPU to the CPU once per iteration.

5.4.4 Parameter Settings

The tracking energy in Eq. 5.1 allows one to combine different data energies with motion priors or smoothness. In general all terms are required to obtain successful tracking results, but in many cases the use of a data energy alone is often sufficient to provide reasonable tracking results. For this reason, we favour this simple case of using only the data energy, meaning there is no need to balance parameters. More specifically, the prior term is typically disabled, $\lambda_{\text{prior}} = 0$. For the synthetic data sets and the experiments with E_{ssd} we used $\lambda_{\text{smooth}} = 0$. Also, for the E_{sil} data term we found that the prediction sometimes caused poor results (possibly due to the nature of the gradient free optimizer), and achieved better results by disabling prediction and using a default setting of $\lambda_{\text{smooth}} = 0$. However, the prediction and smoothness were beneficial for the E_{icp} data energy, so a relatively small $\lambda_{\text{smooth}} = 100$ was used.

Optimization of E_{ssd} and E_{icp} energies require linearization and iteration. A larger number of iterations will be required to handle larger inter-frame motion, while fewer iterations give a faster running time. These iteration bounds were chosen heuristically such that they allow good tracking in the tested cases. For the synthetic experiments with E_{ssd}, the maximum iterations was $\#_{\text{iters}} = 10$, but this was reduced for efficiency in the real-time qualitative experiments to $3 \leq \#_{\text{iters}} \leq 6$. The E_{icp} used $10 \leq \#_{\text{iters}} \leq 14$ when motion prediction was enabled, but a larger $\#_{\text{iters}} = 20$ was used in the experiments when motion prediction was disabled. For the E_{xor} energy, the implementation of Powell's method was limited to 10 sweeps through the coordinate descent directions.

5.5 Experiments

In this section the local tracking methods are evaluated on both synthetic and real sequences. The synthetic sequences enable a ground truth comparison. With these sequences the ground truth geometry, skeleton, and reference colors are used as input and only joint angle estimation is considered. We start with synthetic examples of the simplest case (a single rigid body, § 5.5.1), and progress into objects with more articulation (§ 5.5.2). Using synthetic data, we also test the robustness of the tracking methods with respect to errors in both geometry and bone placement (§ 5.5.3). Finally, we present qualitative and quantitative results on real data acquired from various multi-camera studios (§ 5.5.4). In the real sequences, the geometry was acquired in advance using stereo, through shape-from-silhouette, or through a laser scan; in these cases a skeleton was manually aligned.

5.5.1 Rigid Object, Synthetic Sequence

First, the accuracy of the local methods is compared on a simple rigid object. For this experiment, only the independent data terms SSD, XOR, and ICP were used (e.g., there was no temporal prediction or smoothing). This simple object gives insight into the accuracy attainable by each of the methods. The sequence contains two frontal views of a moving house composed of ˜$20k$ triangles

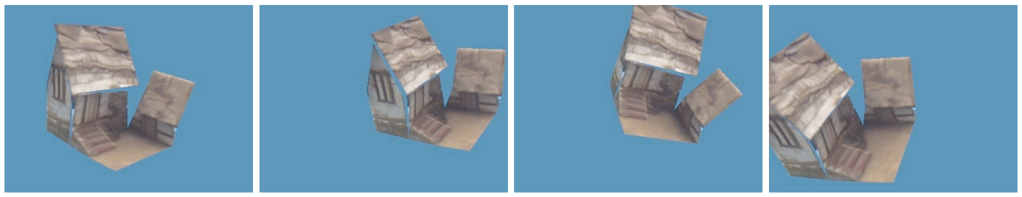


Figure 5.7: Frames 0, 30, 100, and 130 of 200 from one view of the fast moving synthetic house sequence. The object has sufficient texture (for the SSD method) and the silhouettes are perfect.

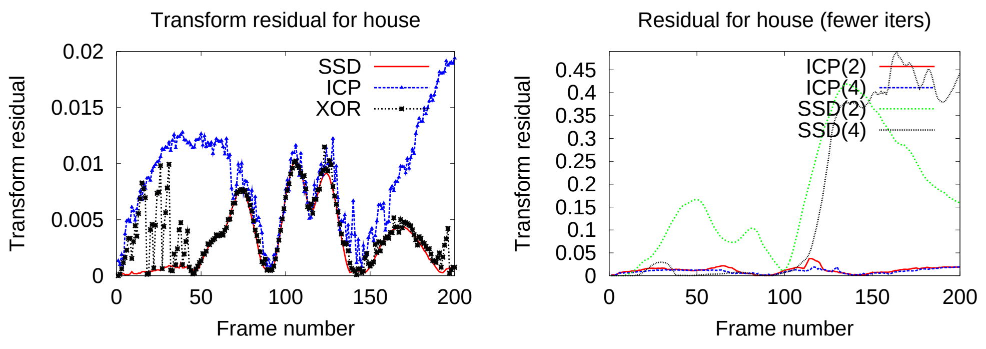


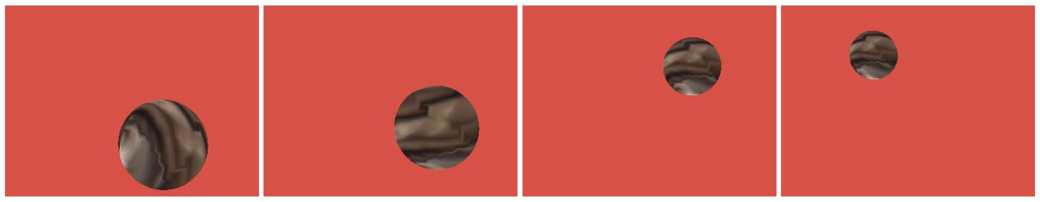
Figure 5.8: Left: the transformation residual over the synthetic house sequence for the various methods. Right: limiting the number of iterations can be used to gain efficiency, but it comes at a penalty of poorer tracks. Results are shown for 2 and 4 iterations for the ICP and SSD methods. The ICP is more tolerant to using fewer iterations.

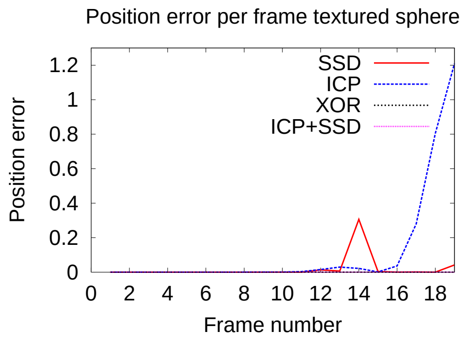
(the motion of one view is shown in Figure 5.7). The residual of the reconstructed transformation was measured for each of the tracking methods. The distance between two Euclidean transforms was taken to be $|\mathbf{t}_1 - \mathbf{t}_2| + \arccos(\langle \mathbf{q}_1, \mathbf{q}_2 \rangle)$, where the transformations are represented with a translation, $\mathbf{t}_i$, and quaternion, $\mathbf{q}_i$.[1] The SSD and ICP all used a maximum of 10 iterations, whereas the XOR tracker was run until convergence. The results are illustrated in the left hand side of Figure 5.8. In this case, each of the trackers perform well, with the ICP tracker behaving slightly worse close to the end of the sequence, possibly because of too few iterations to capture the fast motion.

Increasing the number of iterations and running to convergence is important to obtain high accuracy. However, for interactive or real-time tracking, the number of iterations needs to be limited. As both the SSD & ICP achieved similar results with 10 iterations, we have reduced the number of iterations to see how the methods behave (right hand side of Figure 5.8). Using fewer iterations has less of an effect on the ICP than on the SSD tracker. This is due to the approximate problem being solved by each method at every iteration: ICP uses constraints proportional to the distance to silhouettes, whereas the SSD linearization uses only image intensities and image gradients.

For the house sequence, either silhouette or SSD scores work well, and there would be little advantage in combining the two terms. However, it is easy to imagine cases where either one of the

[1]Adding angular and positional residuals in this way can be problematic if the scale of the spatial coordinates is large. In the synthetic experiments, the object size is always around 1 unit, meaning the scale of the positional residual and angular residuals in radians are typically close.





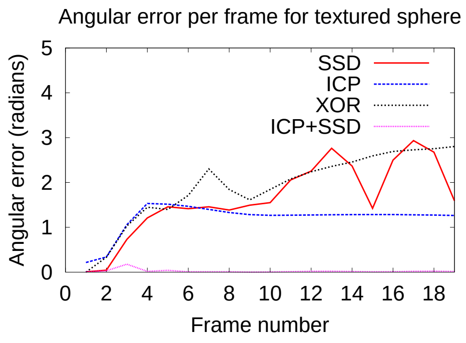


Figure 5.9: This sequence of a moving textured sphere illustrates the limitations of either SSD or silhouette-based approaches. This 20 frame animation of a ball that rotates, translates to the right, translates fast to the left and back to the right. The silhouette methods can accurately track the position but fail on the orientation. The SSD tracker succeeds in tracking the orientation only in the first few frames and fails when the motion is too large. Combining the ICP with SSD (ICP+SSD) gives a tracker that is able to track orientation and through faster motion.

methods would fail: SSD fails when there is too much motion or not enough texture, and silhouettes fail when object rotation is not encoded in the silhouette. These cases are illustrated on a simple example of a rotating textured sphere (Fig. 5.9), where it is shown that combining the data terms gives the best tracking results.

Method	House	Sphere
SSD	0.481	0.185
ICP	0.415	0.285
XOR	2.98	0.612
ICP+SSD	N/A	0.87975

Table 5.2: Seconds per frame for the rigid tracking sequences.

Table 5.2 shows the timing results for the rigid trackers on these two sequences. The SSD and ICP methods are the most efficient. The silhouette XOR is slower than ICP, primarily due to the optimization method. Combining ICP+SSD uses a general Levenberg-Marquardt implementation, and is more costly than the sum of the two SSD and ICP methods.

5.5.2 Highly Articulated Object, Synthetic Sequence

The previous experiment demonstrated that all of the methods were capable of achieving accurate tracks for a simple rigid object. In the next experiment, the accuracy of the trackers are compared on a synthetic sequence of a highly articulated giraffe (Fig. 5.10). The giraffe is roughly 1.7 meters high, contains 4868 vertices, and has 12 kinematic links each having 3 DOF (plus 6 DOF for the root), giving a total of 39 parameters. The sequence contains 220 time frames from 4 views that surround the object. The SSD method is run on a non-segmented background (e.g., the grassy

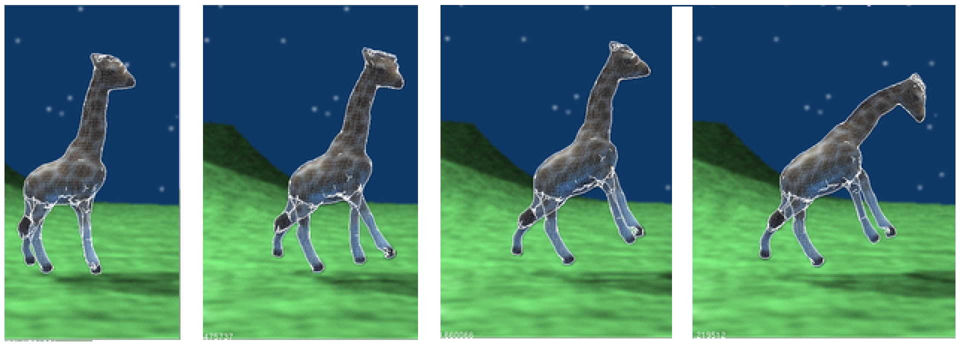


Figure 5.10: Several frames from a view of the giraffe sequence with projected tracked model from the SSD methods overlayed (white silhouette).

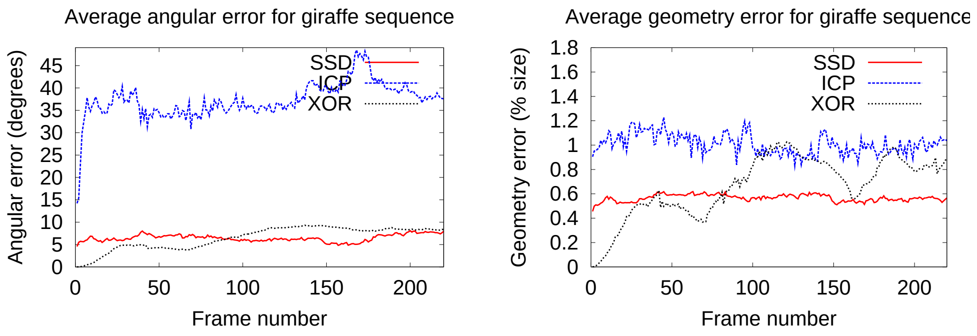


Figure 5.11: Comparison of angular and geometric residuals on the *giraffe* sequence for the various methods. The ICP method tends to cause internal rotations that negatively affect the angular results but have no effect on the deformed geometry. In fact, in spite of this, the ICP still has a low geometric error (around 1%).

surface), and the silhouette methods again use the ground truth silhouette.

The difference between recovered angles and the ground truth animation for each of the methods is illustrated in the left of Fig. 5.11. The SSD tracker overcomes the texture in the background and thin legs and achieves an average angular error of roughly 6 degrees over the entire sequence. On the other hand, even though the silhouette methods minimize related objectives, the silhouette XOR clearly outperforms the other silhouette method. Unfortunately, the poor performance of the ICP method is due to it hallucinating internal bone rotations that cancel out, making the angular residual high.

As another measurement of accuracy, the distance between corresponding vertices of the tracked mesh and the ground truth animated mesh were compared. When considering this geometric error (Fig. 5.11, right), all methods demonstrate good performance, with the error being less than 1% of the object size.

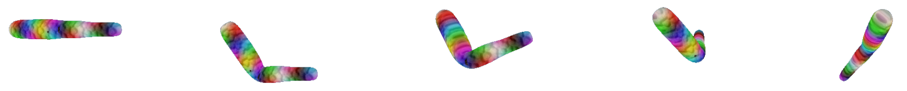


Figure 5.12: Frames 0, 30, 60, 100, and 130 from one view of the arm sequence.

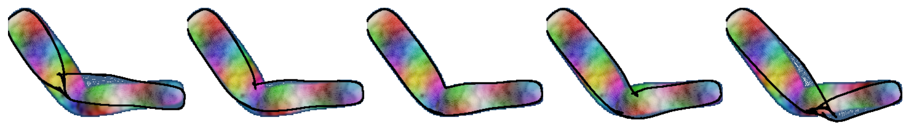
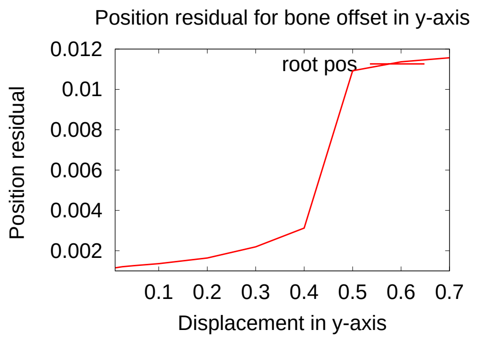


Figure 5.13: Offsetting the elbow in the direction of the bone (y-axis in 3D) causes the tracker to try and compensate. Tracked mesh silhouette is outlined in black. From left, offsets of: -1.3, -0.5, 0, 0.5, 1.3.

5.5.3 Tracker Sensitivity to Model and Skeleton Accuracy

Under real circumstances the ground truth geometry and the location of the bones are unknown. To test the sensitivity of both the geometry and the bone locations, we have used another sequence that consists of a two link armature. The sequence contains 3 views of a textured arm, which is roughly 10 units long with a diameter of about 2 units. We generated and rendered synthetic motion of this arm (Fig. 5.12) and then tested the results of the tracking with a skinned geometry that is attached to a perturbed skeleton. To separate the effects of an approximate geometry, we used the ground truth mesh. The mesh is automatically attached to the perturbed skeleton by generating new skinning weights using the technique of Baran and Popović [25]. This mimics the skeletal alignment errors that would occur in a real-world example.

The motion of this sequence has positions with the arm straightened, which means that tracking by silhouette alone is not adequate (the silhouette-based trackers sometimes infer unobserved roll in the arm when it is straight). For this reason we compare only the SSD methods.



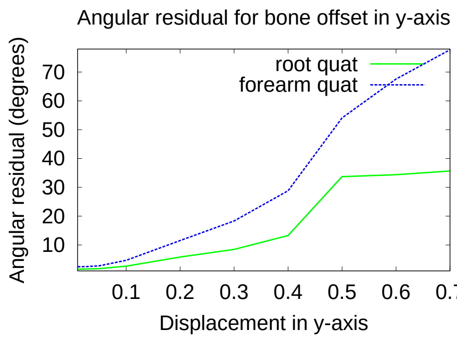


Figure 5.14: Plot of forearm bone offset versus transformation residual on the two-link arm sequence. The *root pos* gives the positional error, *root quat* is the error in the root orientation, and *forearm quat* is the error in the forearm orientation. The error varies smoothly as a function of the errors in the skeleton. A displacement of 0.7 is about 15% of the length of a bone.

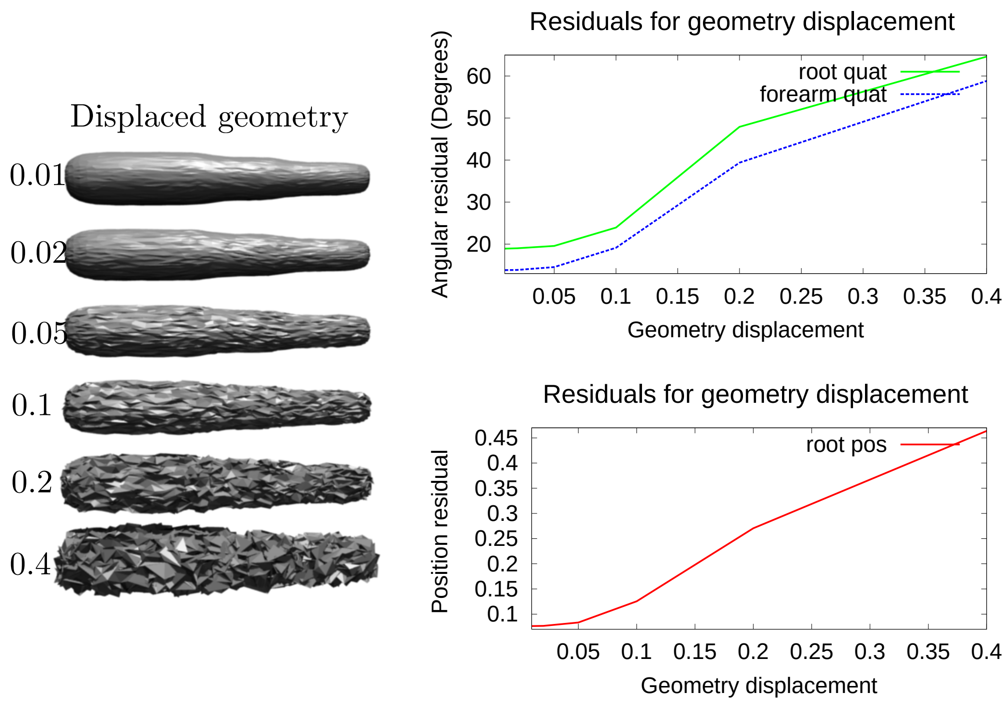


Figure 5.15: To test sensitivity of the SSD tracking with respect to geometric error, the two link-arm geometry was perturbed by various amounts (left). The transformation residual is plotted against random geometry perturbation for the SSD method. on the two-link arm sequence (right). The *root pos* gives the positional error, *root quat* is the error in the root orientation, and *forearm quat* is the error in the forearm orientation.

Perturbed Skeleton

We have perturbed the location of the forearm in the x-, y-, and z-axis independently and then tracked the entire sequence using these perturbed kinematic structures. Figure 5.14 compares these offsets to the average transformation residual over the entire sequence for the $y-$axis. Displacement of each coordinate axis directly affects the residual of the forearm quaternion, and less so the quaternion of the root link. Displacement in $x-$axis also adversely affected the recovered position of the root. Degradation with increasingly poor bones can be expected; however, in all of the cases the tracking was reasonably good, with some of the larger offsets giving rise to jiggles in the tracking. A displacement of 0.7 units almost puts the bone outside of the geometry for the x- and z-axis, and in the y-axis it corresponds to a displacement of roughly 15% of the length of a bone. This is a large tolerance to error justifying that manual insertion of an approximate skeleton is sufficient. Also, the displacement in y, which is along the length of the bone, appears to have the most adverse effect. But even in this case, a residual of 30 degrees at the displacement of 0.7 is acceptable for such a poor initialization of the bones. Figure 5.13 illustrates how the tracker will compensate for such errors in the model, noting that even the offset of 1.3 units still gives rise to a more or less successful tracking.

Perturbed Geometry

To complement the above experiment, we have kept the skeletal structure constant (using the ground truth mesh and reference colors) and modified the mesh by varying random amounts. The offsets for each displacement level were chosen randomly for each coordinate to be in the range of $[-disp, disp]$. The distance from the recovered transformations to the ground truth were measured and averaged over the entire sequence and also averaged over 11 trials with different random perturbations. Again, the tracking is successful with most values of the displacement (Figure 5.15), although the larger displacements give less smooth trajectories. Tracking is still successful with geometric displacement of 0.4 units, which is almost half the radius of the arm.

These experiments suggest significant resilience to errors in the geometry as well as placement of the skeletal structure (which in turn affects the skinning weights). It is important that the tracking is accurate in spite of inaccurate geometry and skeletons, which are inevitable in real cases.

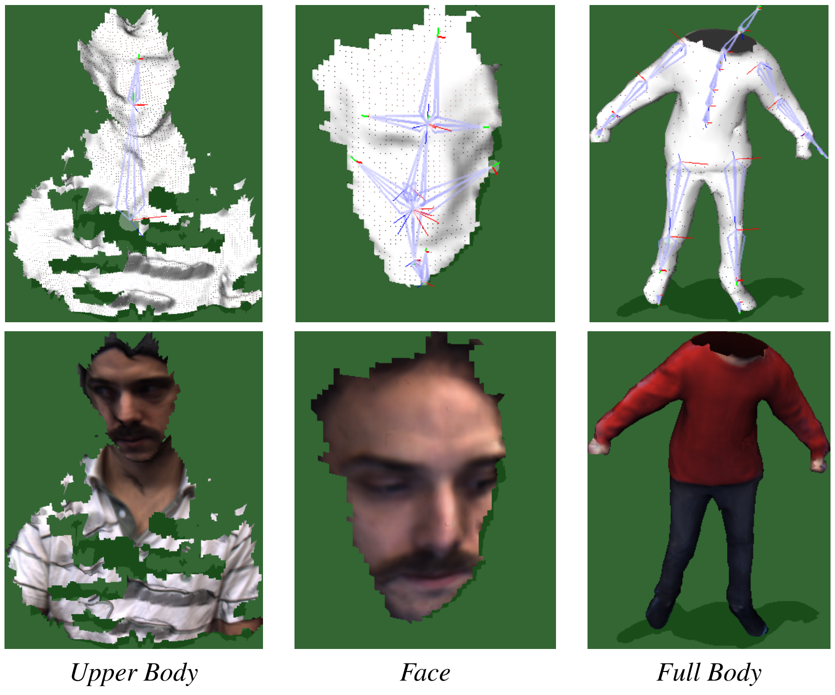


Figure 5.16: Input geometries and underlying skeletons used for the real SSD tracking sequences.

5.5.4 Real Sequences with SSD Energy

The SSD score is best suited to applications that have sufficient texture and cannot be tracked by silhouettes. For this reason, unlike silhouette methods, the intensity-based method is more suited to representing non-rigid animations not typically encoded with a kinematic hierarchy (e.g., facial animation). The SSD method can work with a single view, and can be made fast, allowing for real-time applications. In this section, we perform the real experiments for the SSD-based tracking and postpone the silhouette-based methods until Section 5.5.5.

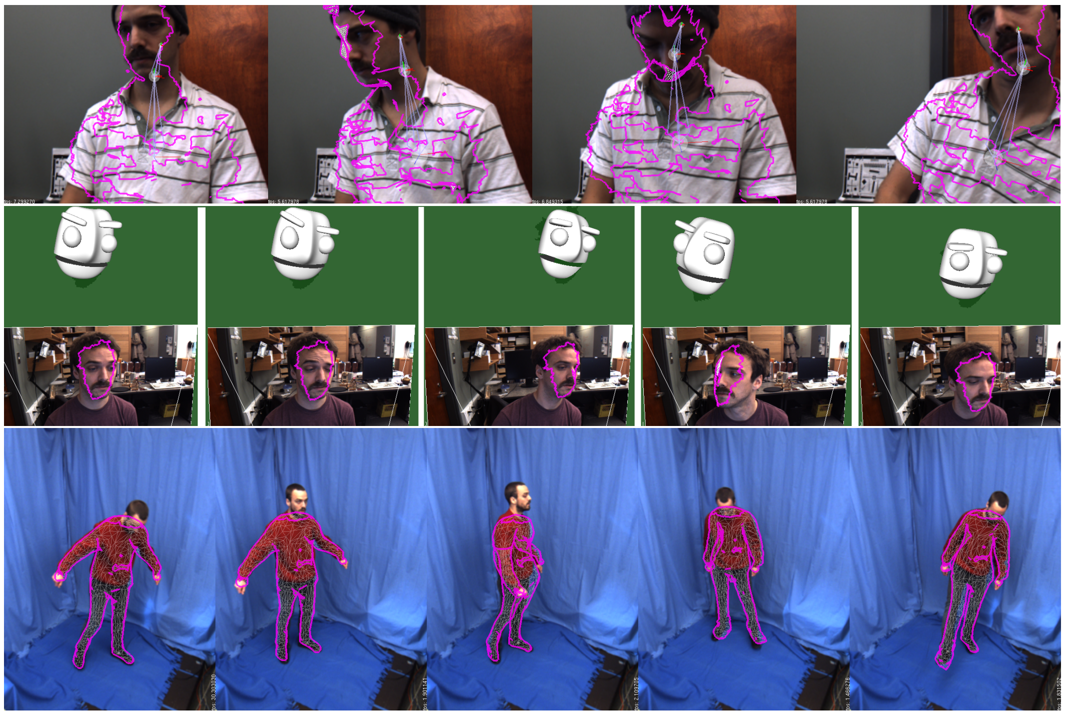


Figure 5.17: The results of SSD tracking on real sequences with tracked mesh overlayed. From top to bottom, the *Upper Body*, the *Face*, and *Full Body* sequences. The rows show one view of the sequence over time. The projected silhouette of the mesh is overlayed to illustrate the tracking results.

We have applied the SSD articulated tracking to a variety of data sets. The input geometry and underlying skeleton are illustrated in Figure 5.16 and the tracked results are in Figure 5.17. The *Upper Body* data set consists of a human with two kinematic links: one 6 DOF link for the body and one 3 DOF rotational link for the head. The input geometry was obtained from stereo in the first frame and contains roughly 7000 vertices. The input images are 400×300 and comes from two views. Silhouettes are not available. Through intensity differences the tracker still correctly follows the subject through motions of the body and the head.

Our second real sequence is of an animating face, *Face*, where the root of the head is 6 DOF and the eyes and mouth each have an additional DOF in the bones. The input geometry is again obtained from stereo and the skeleton is manually aligned. When the automatic skinning weights are generated, this configuration allows for facial expressions to be tracked. For the face sequence we have mapped the animation parameters for the mouth and eyes to a virtual character that is displayed in real-time alongside the results (Figure 5.17). These motions of the eyebrows and mouth are unlikely to be trackable by silhouettes alone.

We have also tested our method on a fully articulated model of a human performing a standing motion observed by three cameras (bottom of Figure 5.17). For this case we manually aligned a previously computed visual hull geometry with the first frame and extracted the reference colors, R,

Sequence	C	T	$\#_{\text{points}}$	$\#_{\text{iters}}$	$\#_{\text{params}}$	CPU	GPU	FPS	Speedup	Speedup†
Giraffe	4	250	4868	3	39	392s	39s	5.64	10	12.4
Face(1 view)	1	220	1467	6	9	9.2s	7.6s	28.9	1.2	1.33
Face(3 view)	3	220	1467	6	9	30.5s	16.2s	13.6	1.88	2.51
Upper Body	2	220	6998	6	9	76s	18.7s	11.8	4.06	5.46
Face(dense)	1	220	8705	6	9	49s	16.1s	13.6	3.04	3.74
Full Body	3	350	1400	4	32	127s	78s	4.5	1.63	2.31

Table 5.3: Timing results for SSD tracker: the GPU-based implementation gives better speedup when there are more cameras (e.g., comparing Face (1 view) to Face (3 view)) or a large number of freedoms (e.g., Giraffe) or a large number of vertices (e.g., Upper Body).

from the first image in the sequence (each input view used a different set of input colors). This is a situation that is often handled by more specific human tracking routines and is better suited to the silhouette costs. Using a larger λ (for all other cases it was 0.1) and a pre-blur filter on the image, this geometry was tracked fairly well, with some minor mistracks near the end of the sequence. This sequence demonstrates the versatility and ability of the SSD method to track fully articulated geometries in the presence of inexact geometry and skeleton.

The SSD tracker is capable of tracking a variety of different objects without requiring extraction of the silhouette. The above examples were tracked successfully, but the SSD method only works when there is sufficient texture and relatively slow motion. Unlike the silhouette-based methods, intensity differences (and image gradients) drive the tracker, meaning that the SSD is less likely to be able to recover from severe mistracks.

However, one benefit of the SSD tracker is that it easily allows for an efficient GPU implementation (§5.4.3). The timing results for the data sequences used are given in Table 5.3. The *Face (1 view)* and *Face (dense)* use the same input as the *Face* sequence but use only one camera, and *Face(dense)* uses a denser sampled geometry. The timing results were obtained from pre-recorded sequences so the total time takes into account the load times as well. The tests were performed on an AMD Phenom™II X4 920 with a GeForce 9800 GT running Linux.

In all cases the total time to track using the standard CPU and CUDA implementation is given. These timings include the amount of time required to load the frames from disk. The *Speedup*† column reports the speedup achieved when not taking into account the loading time. In this case, load times for the CUDA implementation take slightly longer due to transfer of textures. For an even comparison of the gains in the tracking (without taking into account load times), the *Speedup*† is computed as

$$\text{Speedup}^{\dagger} = T_{\text{Track}}/(T_{\text{Cuda Track}} + T_{\text{Cuda Load}} - T_{\text{Load}}).$$

Where T_* are, in order of appearance, the average s/w tracking time, average CUDA track time, and average load times for the two implementations. In all cases the GPU implementation is faster; the benefits are most dramatic when there are many freedoms (e.g., a speedup of 10 times on the *Giraffe*), or many vertices (e.g., *Upper Body* gives a speedup of 4 times).

5.5.5 Real Sequences with Silhouette-based Energy

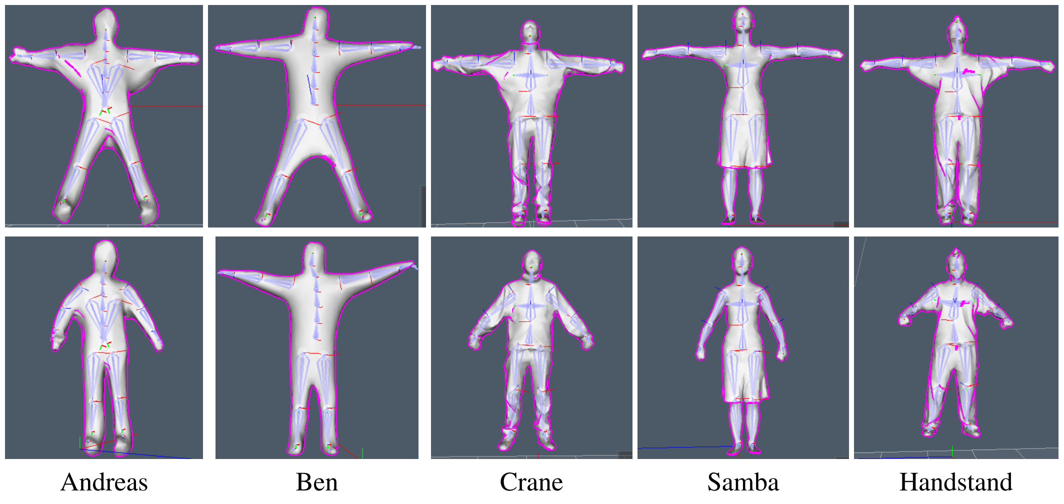


Figure 5.18: The geometry and skeleton used for the real sequences. The top row shows the geometry in a rest pose whereas the bottom shows the initialization pose. *Andreas* and *Ben* were obtained using shape from silhouette, and the remaining geometries are laser scanned.

In this section, we present qualitative and quantitative results for the various silhouette methods on several different input sequences. The length of the sequences, number of vertices in the geometry, and number of parameters in the skeleton are given in Table 5.4.

Qualitative Results

Sequence	N	T	$\#_{\text{points}}$	$\#_{\text{params}}$
Andreas	5	400	1545	61
Ben	6	128	868	61
Neil	4	318	1557	32
Crane	8	170	2441	36
Samba	8	175	1995	38
Handstand	8	175	2402	42

Table 5.4: Statistics on the real sequences. The lower three sequences are used in the quantitative analysis.

For qualitative analysis, three data sets are used: the *Andreas* sequence from the IXMAS data set [272], the *Ben* from the GrImage platform [2], and *Neil* captured in our own multi-camera setup. The *Andreas* sequence is extracted from IXMAS data starting from frame 409 until frame 808. The shape-from-silhouette geometry (see Figure 5.18) is extracted from frame 409 and manually smoothed. The subject in this sequence is wearing uniformly colored baggy clothing, and the motion consists of the following: walk in a small circle, then walk in a larger circle, followed by a hand-wave and a punching motion. The ICP silhouette tracker loses track of the arms almost immediately, which is likely due to the lack of free space between the subjects arms and body. The silhouette XOR tracker with no temporal prediction/smoothness and no shape prior tracks fairly well (Fig.5.19). Due to under-segmented regions near the floor, the tracker confuses the orientation of the feet in some frames. The tracker eventually loses track of the arms near the end of the sequence during the fast punching motion (frame 375 of 400).

[2]http://grimage.inrialpes.fr/

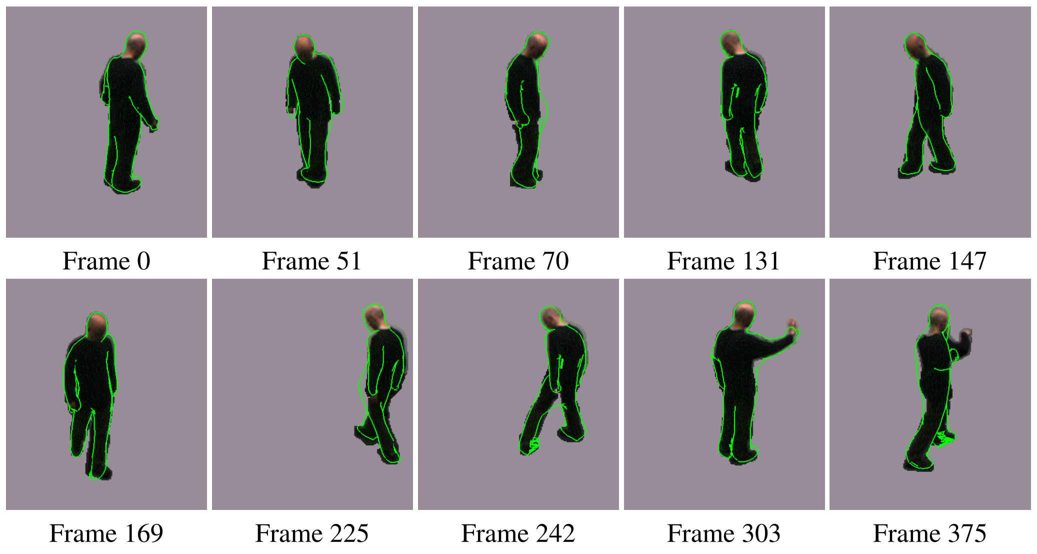


Figure 5.19: The *Andreas* sequence observes a subject walking in a small circular motion, then a bigger circular motion, and finally stopping to wave and punch. The silhouette XOR tracking was the only successful silhouette method, with small minor mistracks on the feet during the small circle (e.g., frame 51). The tracker completely loses track of the arms during the fast punch action (frame 375).

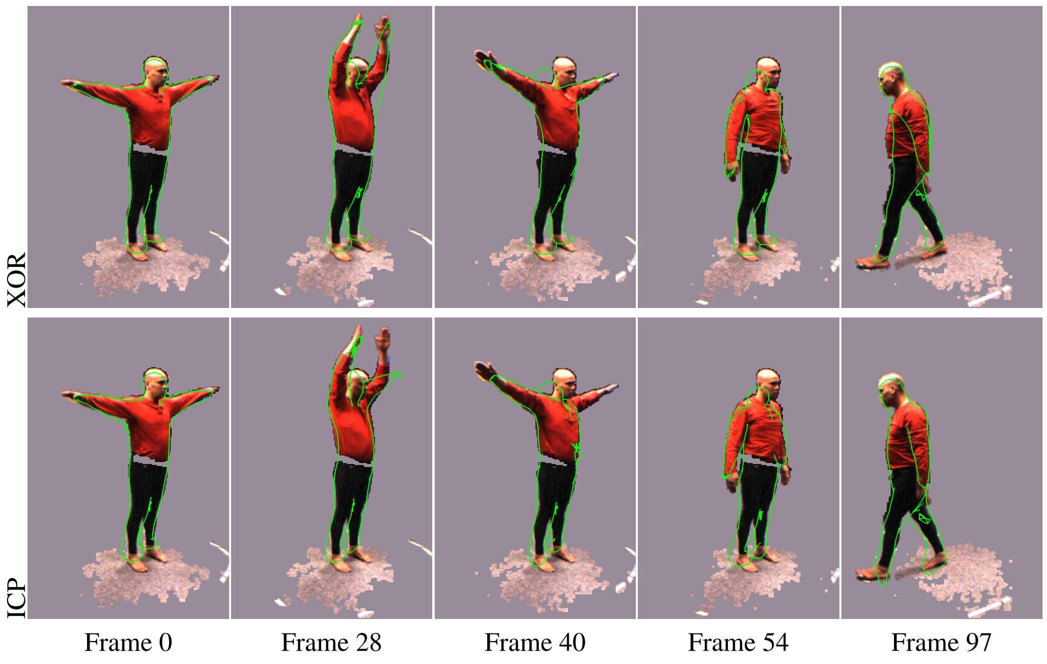


Figure 5.20: The *Ben* sequence observes a subject clapping his hands above his head and then walking off the stage. The silhouettes are poor, with under-segmentation near the floor, and some images missing the head (causing both trackers to mistrack the head. See frame 40). The XOR tracker tracks well except for some regions around the feet, whereas the ICP tracker loses track of the arm during the clap, and recovers the arm at frame 80. The poor silhouettes near the ground cause the ICP tracker to give poor estimates of the feet.

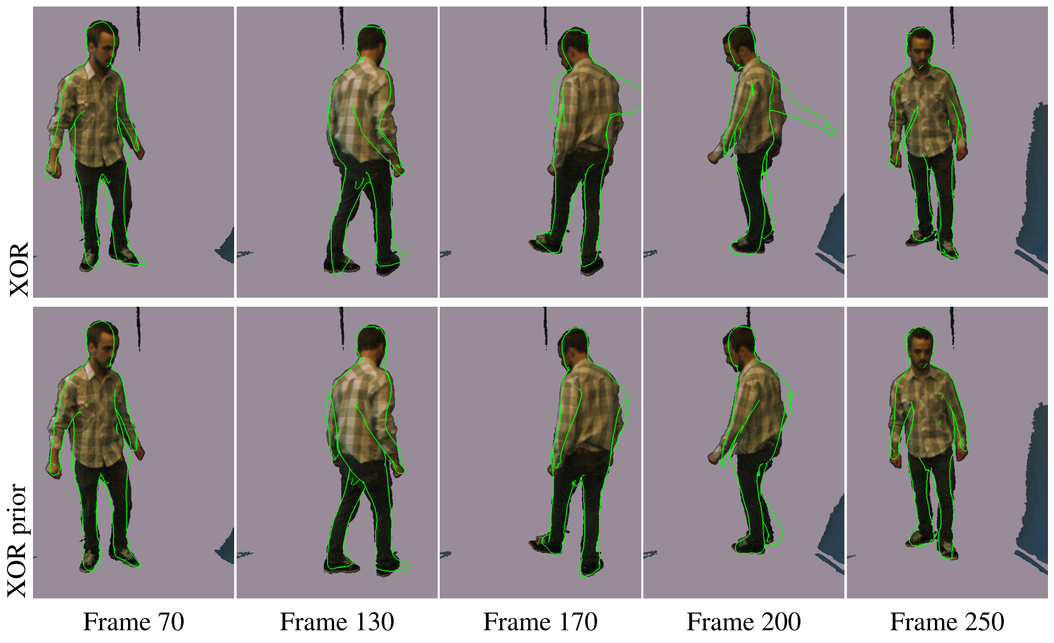


Figure 5.21: The *Neil* sequence observes a subject walking in a circle. The silhouettes are poor, and the images are mostly on one side of the subject. The ICP trackers lose track quickly, and the XOR tracker alone loses track of one arm (see frames 170 and 200). Adding the prior term helps recover accurate motion using the XOR data term (bottom row).

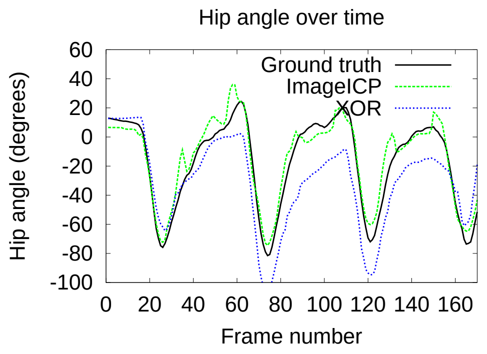
Using the *Ben* sequence, we illustrate that the local trackers are robust to silhouette noise. The subject in the *Ben* sequence performs a hand clap, and then walks off the stage. The sequence has several poorly segmented frames, mostly near shadows on the floor. Additionally, the body region is sometimes over-segmented, and the head is completely missing in one view for several frames. Figure 5.20 illustrates the tracked results for the ICP tracker and the XOR silhouette tracker. In both cases, the general motion of the subject is recovered; however, both trackers have mistracks in the head due to the poor segmentation. Both trackers successfully recover. The ICP has more trouble dealing with the under-segmentation near the feet, causing hallucinated motion in the feet.

In the *Neil* sequence, the benefit of the pose prior is demonstrated. This sequence consists of a subject walking in a circle (Fig. 5.21). Without the pose prior, the ICP is incapable of dealing with the poor silhouettes, and the XOR tracker loses track of the arms at frame 130. When utilizing a pose prior (Section 5.2.2) created from a subject in the CMU motion capture database[3] (subject #13, Trial #02, which consists of a subject walking and sitting), the silhouette XOR tracker successfully tracks the motion in the sequence.

Quantitative Results

The previous sequences had both poor silhouettes and approximate geometries, but the simple local XOR tracker was able to track the complete motion fairly well. The remaining three sequences

[3]http://mocap.cs.cmu.edu/



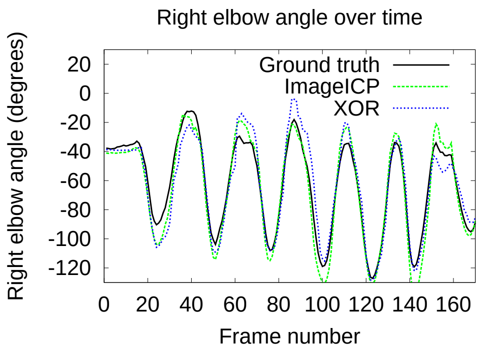


Figure 5.22: Comparison of angular trajectories for the right hip and right elbow for the *Crane* sequence.

from the publicly available MIT data have high quality silhouettes and an accurate laser scanned geometry [259][4]. For these three sequences, we use the existing tracking results from MIT as ground truth [259]. This allows for quantitative comparison for the silhouette-based methods. The three sequences have the following motion: the *Crane* sequence consists of a march-like walk with bird-like motion of the arms (Figure 5.23), the *Samba* consists of a woman in a dress performing a samba dance (Figure 5.24), and *Handstand* records a male performing a handstand (Figure 5.25).

Sequence	XOR	ICP	ICP-LBFGS
Crane	3.18	2.73	2.35
Samba	3.46	2.17	1.93
Handstand	9.27	4.41	3.79

Table 5.5: Distance between joints (cm), averaged over joints and frames for the real sequences.

For the quantitative results we computed the distance between the recovered joints to the ground truth tracked joints. The tracking residuals averaged over the sequences for all joints are in Table 5.5. Unlike the sequences with poor silhouettes, the ICP method achieves a much better alignment and outperforms the XOR method in all cases. This table also includes results for an ICP tracker that uses LBFGS optimization. The ICP-LBFGS consistently outperforms the basic ICP tracker because the stopping condition for the optimization is tighter[5]. Although not used in this experiment, the pose prior cannot be represented as a non-linear system of equations and must use such a non-linear minimizer (e.g., LBFGS).

The ICP tracks the *Crane* sequence accurately, but the XOR tracker has some mistracks (see Figure 5.23). Figure 5.22 shows the tracked angles versus the ground truth for the hip and elbow. The XOR tracker drifts on the hip angle after frame 70, whereas the ICP trajectory is close to the ground truth.

The dress in the *Samba* sequence contains a large amount of non-rigid motion that is not explainable by the linear skinned model. However, as illustrated in Table 5.5 and Figure 5.24, this sequence is tracked accurately with the ICP method in spite of the non-rigid motion of the dress. The XOR tracker performs well, but has minor mistracks in the legs.

[4]Data can be obtained through http://people.csail.mit.edu/drdaniel/mesh_animation/index.html
[5]This is due to the use of separate libraries for LBFGS and Levenberg-Marquardt, which use different stopping criteria.

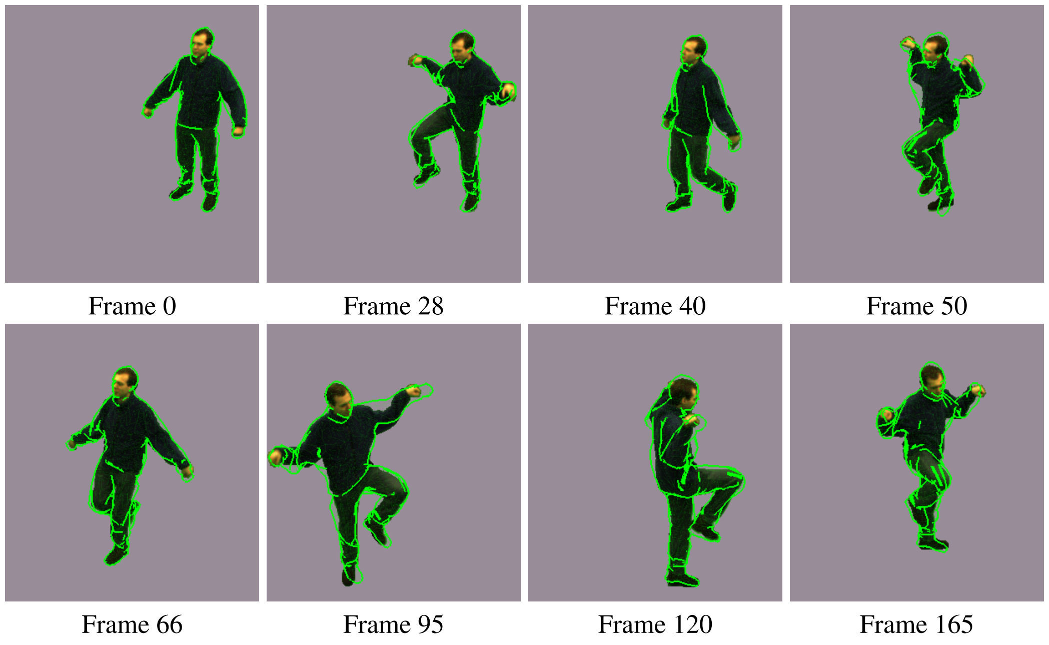


Figure 5.23: Tracking results for ICP on the *Crane* sequence overlayed onto the input from a single view. The fast motion is tracked well.

The handstand sequence contains a challenging inverted motion with fast motion of the legs (Fig. 5.25). The ICP method again outperforms the XOR method (which has some mistracks). This sequence demonstrates that the local methods can still perform well on fast complicated motions.

	LBFGS	No motion	ICP	XOR
Crane	4.3	1.76	1.68	19.9
Samba	5.84	3.36	2.22	15.6
Handstand	7.71	3.64	2.38	20.9

Table 5.6: Seconds per frame (spf) for the various tracking methods on the real sequences used for quantitative analysis. The *No motion* column gives the tracking times for the ICP method when not performing motion prediction.

The tracking methods discussed in this chapter all use local optimization. Therefore, there will be cases when they fail to track accurately. In such cases, either global methods can be applied (e.g., [95]) or a user can manually adjust the tracking (e.g., [259]). In the latter case, it is important for the local tracking methods to be efficient; this allows a user to monitor the tracking progress and adjust when necessary. The timing results for the sequences used in the quantitative analysis are given in Table 5.6. In all cases, the ICP method is the most efficient, taking about 2 seconds per frame. The use of motion prediction gives an improvement in speed as it provides a better initialization for each frame, which reduces the number of iterations required for convergence. For the ICP tracker with motion prediction/smoothness, a couple seconds of footage at 30HZ will only take a few minutes for an operator to review and correct. On the other hand, the XOR tracker is roughly 10 times slower, making it less practical for an interactive correction setting. Furthermore, the ICP speed can be increased by reducing the number of iterations at a potential larger chance of mistrack.

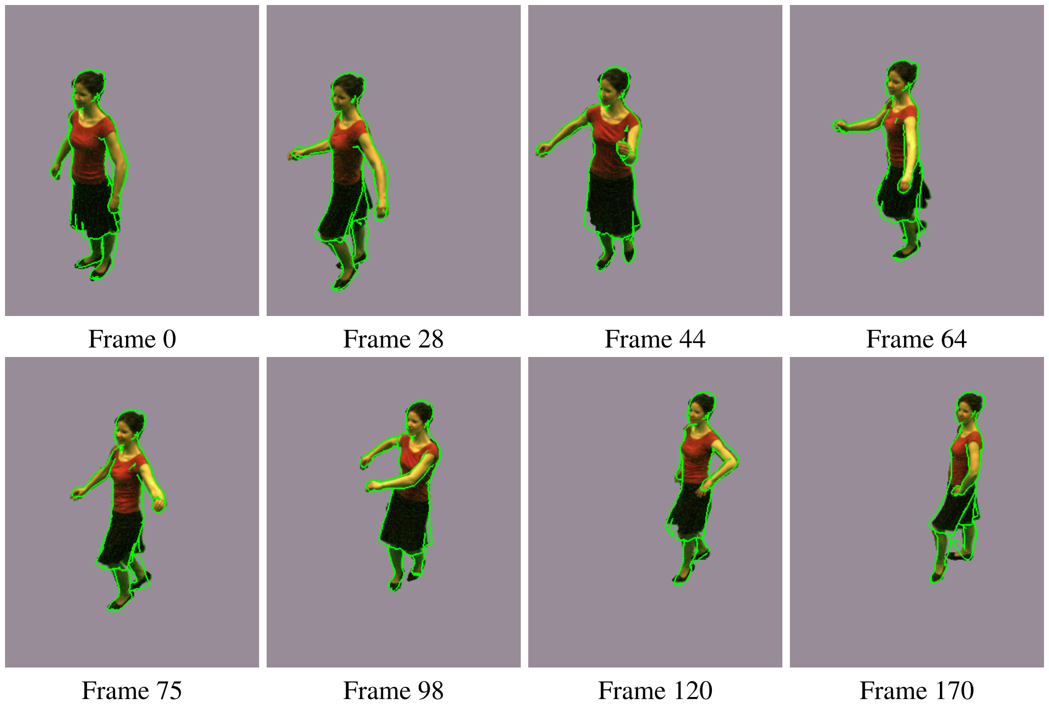


Figure 5.24: Tracking results for ICP on the *Samba* sequence overlayed onto the input from a single view. Despite the skinned geometry only being an approximation to the dress geometry, the tracking works well.

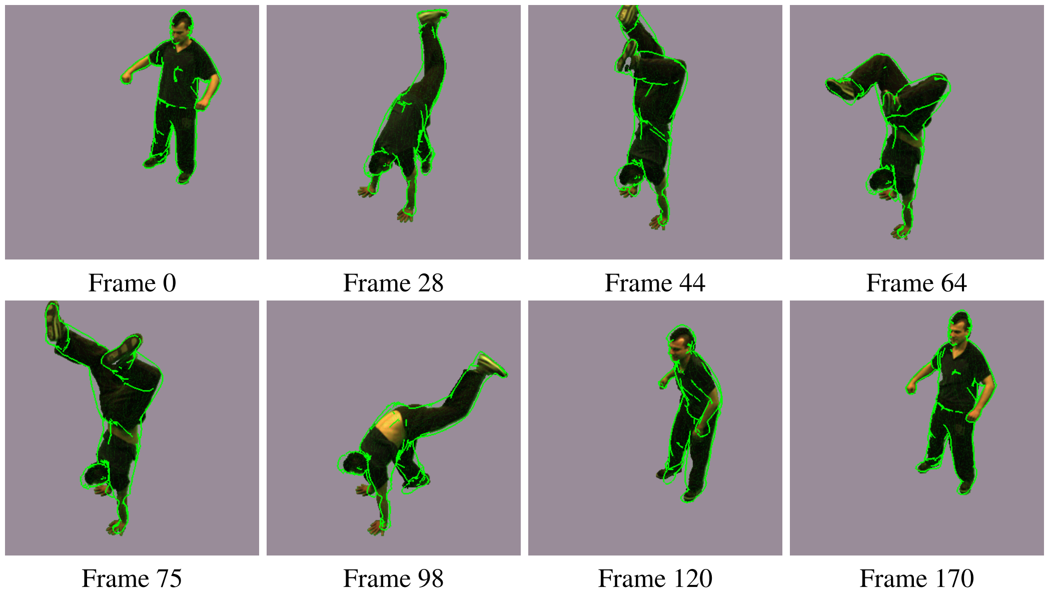


Figure 5.25: Tracking results for ICP on the *Handstand* sequence overlayed onto the input from a single view. The tracking has minor mistracks near the hands (e.g., frame 98).

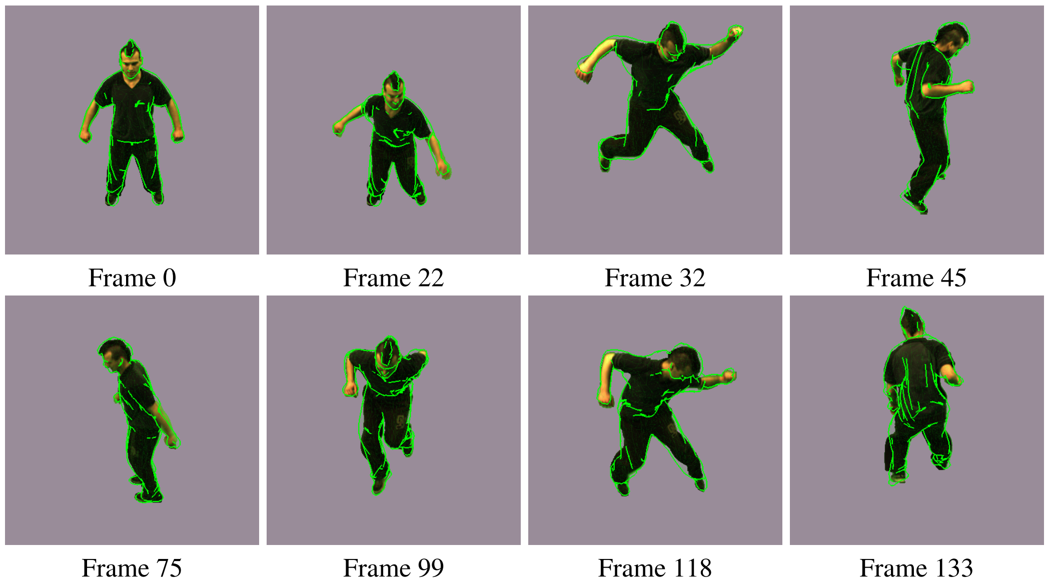


Figure 5.26: The *Bouncing* sequence has many fast motions of the arms, which cause the local methods to lose track of the arms. With minor editing of the arms in 10 out of 175 frames, decent results are obtained in a short amount of time.

			Method	
Sequence	Info	XOR	ICP	XOR+Prior
Andreas	Okay silhouettes, baggy clothes	Minor mistracks	Fail	N/A
Ben	Poor silhouettes	Mistracks (head)	Mistracks (head, arm)	N/A
Neil	Poor silhouettes, few cameras	Large mistracks (arm)	Fail	Success
Crane	Great silhouettes	Minor mistracks	Success	N/A
Samba	Non-rigid motion	Minor mistracks	Success	N/A
Handstand	Complex inverted motion	Minor mistracks	Success	N/A
Crane	Fast motion	Fail	Fail (success with edits)	N/A

Table 5.7: A summary of the results for the silhouette tracking on the various sequences. Although there were some minor mistracks, the XOR method appears to work better on the sequences with poor silhouettes. In contrast, the ICP method works well on the sequences with more viewpoints and good silhouettes.

For example, the *Bouncing* data set from MIT [259], contains very fast motions (Figure 5.26). Even with a larger number of iterations per frame (25), the ICP tracker gets lost after about 1s of video. However, with some edits of the arms on 10 frames, a reasonable tracking result is obtained. The manual monitoring and editing of the failed regions takes about 20 minutes for the 175 frame sequence.

Summary of Silhouette-based Experiments

Two separate types of silhouette data terms, XOR and ICP, were evaluated on a variety of data sets. The data sets included sequences with poor silhouettes, subjects with baggy clothing, subjects with non-rigid clothing (e.g., dresses), and complex motions (e.g., the handstand). A summary of the sequences and the tracking results are illustrated in Table 5.7.

In practice, the XOR behaves better in the presence of noisy silhouettes, which may completely confuse the ICP tracker. The tracking also works well without the prior term; however, the prior term is useful to constrain the poses when silhouettes are poor. In contrast, ICP delivers better accuracy and higher performance when silhouettes and the underlying model are accurate. This reinforces the results of existing local-global approaches that achieve good results by using ICP-based terms for the local refinement [97]. The requirement on good silhouettes is easy to obtain in controlled environments, but the ICP-based approach can be coupled with the segmentation module to improve the silhouettes while tracking even in outdoor environments [110].

5.6 Discussion

In this chapter an energy formulation has been used to perform model-based human motion tracking. The energy function, composed of image and silhouette data terms as well as smoothness and pose priors, is locally optimized. The formulation can easily be augmented with other constraints, such as joints must be above floor, or poses must avoid collisions. However, simply optimizing the data term alone often gives good results.

One of the primary differences between the formulation in this chapter and existing ICP-based approaches is the use of a skinned mesh and image ICP constraints. This ICP approach can successfully track through some fast motions and can track in the presence of loose clothing, such as the dress in the samba sequence. During minimization, the analytic Jacobian of the skinned mesh is used, allowing for more accurate motion models around joint angles. The minimization of the constraints does not require them to first be linearized (as in [195, 238]); they are linearized when necessary by the non-linear equation solver or the non-linear minimizer. Furthermore, our use of image ICP constraints is in contrast to other approaches that either use 3D motion constraints or constrain points on the mesh to lie on backprojected Plücker lines. The ICP implementation is incapable of dealing with large amounts of noise. This is due to inaccuracies in correspondence. A possible direction to overcome this limitation would be the use of a probabilistic or fuzzy correspondence method.[6]

The image intensity data objective uses an SSD score that is similar to the objective functions that are used in 2D template-based trackers [17] and morphable models [82]. However, in our case, we make use of several cameras and need to take vertex visibility into account. By using a linear-blend skinned mesh, this data objective can be used to track rigid objects, kinematic objects, or even other non-rigid objects that can be represented with skinning (such as human faces). A key limitation of the SSD tracker is that it does not work well if there is no texture. However, this data objective is easily combined with the silhouette cues in the same energy framework (e.g., ICP+SSD).

As the tracking methods are locally optimized, there may be mistracks. In order to allow interactive manual correction, a secondary goal has been to make the tracking implementations efficient. To this end, GPUs have been used to accelerate computation of the objective functions for the XOR-,

[6]Such fuzzy assignments have been used in the similar problem of non-rigid registration of 2D point sets[59].

ICP-, and SSD-based data terms. The use of more general purpose programming via CUDA, brings the SSD-based approach to near real-time: from 5 fps for highly articulated models in multi-view sequences (e.g., giraffe) to 30 fps for monocular sequences of faces. This allows real-time interactive applications when using the SSD tracker, for example, controlling virtual avatars.

Although our GPU SSD implementation does show a speed improvement over a standard implementation, we would like to further investigate whether it is feasible to design an approach that keeps the Hessian fixed for several iterations. Additional improvements could include adapting the techniques used to improve robustness against light changes from the template-based tracking or 3D tracking approaches. Another improvement would be to integrate the cost function over the surface of the triangles instead of only computing it at vertex locations.

Other future work includes evaluating the improvements in results when the geometric model has a dynamic component. For example, after acquiring the deformation model (Chapter 6), the XOR-based tracker could be used to track through new sequences. Through the use of a more accurate dynamic model, there should be a corresponding improvement in the reconstruction accuracy.

6 Fine-scale Correspondence

The joint angle tracking of Chapter 5 provides coarse correspondence of the base geometry, meaning the deformed geometry roughly agrees with the silhouettes and input images. However, the coarse tracking does not account for the fine-scale displacements on the geometry required to make the surface photo-consistent nor does it account for tangential motion, such as cloth shifting, of the surface. In this chapter, algorithms to recover these fine-scale displacements and correspondences between each vertex of the initial geometry and its location in a multi-view training sequence are discussed. Specifically, the focus is on how to obtain the deformations and correspondences when input views are sparse or widely separated. This is accomplished by making use of the known coarse motion and enforcing a temporal constraint on the trajectory of surface points.

6.1 Introduction & Motivation

For morphable models (which are similar to the model presented in Chapter 3), the correspondences have been obtained manually [183] or through optic flow registration of cylindrical laser-scans [34]. Manual correspondences are impractical in our case, but a strictly vision-based alternative to laser scans would be to reconstruct the geometry independently at each time frame and then register the surfaces in a similar manner.

For example, many of the existing methods discussed in §2.4.2 assume that there are enough input views to create a photo-consistent geometry at each time frame (e.g., [112, 229, 37]). Geometric features, or image feature points, can be used to bring these geometries into correspondence [37, 233, 253, 72, 264, 47]. However, such alignment is mostly performing a model-free tracking (i.e., gross alignment), and can only guarantee accurate correspondence at locations of features. Instead, it is possible to refine the model after the tracking has been performed, e.g., by using silhouettes to carve geometry [203], displacing accurate geometries to fit the silhouettes [95, 259, 238], or by using both geometric and silhouette cues to deform the tracked geometry [245].

It is also possible to make use of dense temporal matching to compute the correspondence over time. Color-coded patterns make this correspondence finding easy [209, 273], but are not appropriate when trying to capture appearance. Scene flow methods also do a dense matching in view and time but often only give short range correspondence [256, 114, 270]. For longer range dense correspondences, the photo-consistency and flow-consistency can be optimized over several views over time with meshes [64]. However, these methods do not make use of the already tracked joint angle motion and typically require a large number of frames. Only a few methods (e.g., [250]) explicitly use intra-camera observations to improve the quality of the reconstruction.

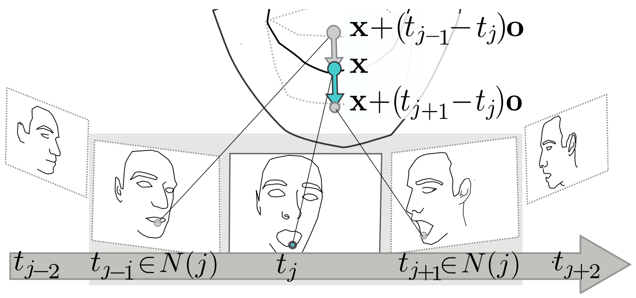
The above formulations often assume many cameras are available. Instead, our formulation in this chapter exploits constraints on the temporal motion to enable and improve reconstructions when there are few views or when camera views have little overlap. Similar to the image-based scene flow methods (e.g., [114, 270]), we use a variational formulation to directly reconstruct 3D scene flow and structure without requiring point correspondences. In contrast, we use an image-based disparity-map and displacement-from-surface representation while enforcing constraints on the temporal surface motion. These temporal constraints allow longer range correspondences than typical scene-flow formulations. Our formulation also tries to best take into account known motion of the coarsely tracked geometry. The method is presented in two sections.

First, the problem of reconstructing structure and flow when there are not enough views is addressed. In particular, we restrict the problem to monocular sequences and demonstrate that a simple assumption on the flow, namely that the flow has a constant velocity over a small sequence of images, allows the reconstruction of both depth and the scene flow. Here we assume that the camera is moving and that the camera motion is known. This is equivalent to knowing the coarse motion of a kinematic link, and focusing on the reconstruction of that link only. The contributions in this section are as follows:

- When correspondences are known, we derive a straightforward linear algorithm to reconstruct both surface structure and 3D flow of points undergoing constant velocity in a short temporal window (Section 6.2.1).
- As an alternative to requiring known correspondences, the constant velocity constraint is augmented with the brightness constancy assumption. These constraints are integrated into a variational algorithm that recovers both the flow and geometry directly from images without requiring a-priori correspondences (Section 6.2.2). In contrast to existing variational approaches, our representation, although image-based, does not require rectified stereo pairs and can easily integrate information from several views.

In the second part, these temporal constraints are generalized to a multi-view application of scene flow on a moving base surface (Section 6.3). Here scene flow estimation is posed directly on a base, or *proxy*, surface. Utilizing available proxy surface motion, such as coarse tracking of a human subject, allows the reconstruction process to exploit intra-camera observations not only for motion but also for 3D structure. To obtain longer range correspondences, the scene motion over time is represented with a low-dimensional set of temporal basis functions. As recently demonstrated with 2D optic flow [99], the advantages of a low dimensional representation of motion are that it uses fewer variables, provides temporal smoothness, and is less sensitive to noise. In this 3D formulation of basis constrained scene flow, we make the following contributions:

- A unified variational formulation of scene flow and stereo on a proxy surface where the un-



$\mathbf{x}+(t_{j-1}-t_j)\mathbf{o}$
$\mathbf{x}$
$\mathbf{x}+(t_{j+1}-t_j)\mathbf{o}$
t_{j-2}
$t_{j-1} \in N(j)$
t_j
$t_{j+1} \in N(j)$
t_{j+2}

Figure 6.1: Reconstruction in reference view j using neighboring images, $\mathcal{N}(j)$; constant velocity is more likely to hold over a small temporal window.

derlying proxy surface may be undergoing some known motion (e.g., from coarse tracking) is proposed. This formulation naturally links many views (as opposed to two-frame, two-time methods), and directly recovers structure & flow from image intensities.

- The scene flow formulation directly utilizes a temporal linear basis, which allows integration of many time steps using fewer parameters. The basis constrains and improves geometric reconstruction when the proxy surface undergoes some known motion.
- With the appropriate proxy and basis choices, our formulation generalizes and provides unified treatment of optic flow (e.g., proxy surface is the image domain), basis-constrained optical flow, two-frame stereo (e.g., no flow terms), and two-frame scene flow.
- We demonstrate results for depth and flow reconstructions on static proxy surfaces, rigidly moving proxy surfaces, and linear blend skinned proxy surfaces.

Finally, as not all image sequences lend themselves to reconstruction using intensity-based methods, we discuss an approach to utilize silhouette information in order to improve per-frame geometries (Section 6.4). Following the recent successes of silhouette information in rendering of performance-based animation [95, 259, 203, 238], the input silhouettes are used to deform the approximate tracked geometry. Silhouettes work well when there is little camera overlap, but the silhouette deformation does not guarantee accurate temporal surface correspondences or tangential flow. However, silhouettes constraints ensure that the displaced geometry provides a good proxy for rendering at each time instant.

6.2 Monocular Scene Flow

Let us first restrict the problem of recovering non-rigid point trajectories in a simple single view setting, where the motion of the camera is known. For now, we treat the recovery of non-rigid motion of points from a single moving camera as an independent problem, but note the connection to recovering non-rigid correspondences in a multi-view setting of a human subject: for example, if we restrict our attention to a single kinematic link on the subject, since the coarse motion of the

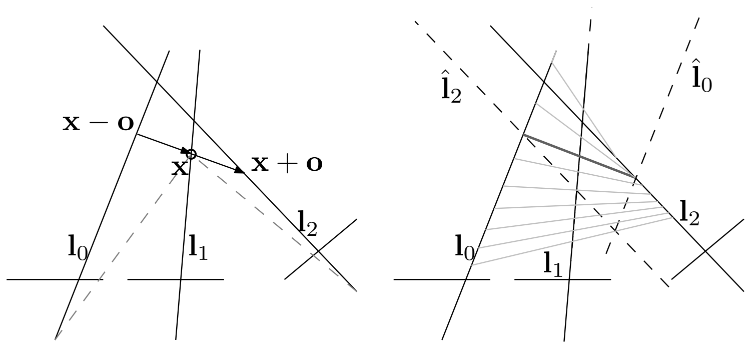


Figure 6.2: Three view setup with ambiguous reconstruction. Geometrically, the solutions can be obtained by rotating $\mathbf{l}_0$ and $\mathbf{l}_2$ about $\mathbf{l}_1$ (giving $\hat{\mathbf{l}}_0$ and $\hat{\mathbf{l}}_1$) for each rotation center (e.g., depth) on $\mathbf{l}_1$

subject has been tracked, the motion of any kinematic link relative to any static camera is known. The abstraction into this simple setting of a known moving camera lets us analyze the problem of recovering motion and flow of a non-rigid scene without complicating the discussion with details about kinematic structure and tracked joint parameters.

Assume we are given a time sequence of T images I_i with calibration $\mathbf{P}_i = \mathbf{K}_i[\mathbf{R}_i\mathbf{t}_i]$ taken at times $\{t_i\}$ observing a non-rigid scene. The objective is to reconstruct a dense geometry and 3D velocity field at each time frame, j. As there is only one moving view, a small temporal neighborhood of images, $\mathcal{N}(j)$, with $|\mathcal{N}(j)| \geq 2$, will be used for the reconstruction. The situation is illustrated in Figure 6.1.

Below, the reconstruction problem is formulated using a simple temporal constant velocity constraint, which is first used to obtain a reconstruction in a single reference frame when corresponding image projections are known (Section 6.2.1). These constraints are then used to derive a variational energy for reconstructing both geometry and flow directly from images (Section 6.2.2). Finally, we develop constraints to ensure longer sequences will be temporally consistent (Section 6.2.3).

6.2.1 Known Correspondences

Given correspondences, $\mathbf{a}_{ki} = [a_{ki}, b_{ki}]^\intercal$, of a 3D point k in the images, the objective is to reconstruct the shape, $\mathbf{x}_k = [x_k, y_k, z_k, 1]^\intercal$, and velocity, $\mathbf{o}_k = [u_k, v_k, w_k, 0]^\intercal$, of the point in a reference time frame.

In a two view case where camera motion is known and the object is slowly moving (or almost rigid), most of the variation between the two images will be due to depth variation along the known epipolar geometry. Two-dimensional flow components orthogonal to the epipolar lines are surely due to 3D flow, but 3D flow can also cause variation along the epipolar line. In such cases it is impossible to separate depth from flow; even when depth is known, the flow is ambiguous. Constraints on the flow offset, such as the flow vector should be minimal or the flow vector is parallel to the image plane, will give a unique solution, but neither of these are plausible constraints.

An unconstrained three time frame setup leads to a similar ill-posed configuration. However, with the extra frame it is possible to derive more physically plausible constraints. Assume that the point has a *constant velocity* through time $t_i \in \{0, 1, 2\}$ and is observed by a moving camera. In such cases it is often possible to reconstruct a unique solution. Unfortunately, some degenerate camera configurations exist: no camera motion implies infinitely many solutions, and coplanar backprojected rays give a one parameter family of solutions (see Figure 6.2 for a 2D top down example). Note, in the latter degenerate case, unlike the two-view example, knowing the depth allows the recovery of correct 3D flow. Reconstructing the structure and velocity at t_1 gives three non-linear constraints on the 3D point, $\mathbf{x}_k$, and its offset, $\mathbf{o}_k$:

$$\Pi(\mathbf{P}_0(\mathbf{x}_k + (t_0 - t_1)\mathbf{o}_k)) = \Pi(\mathbf{P}_0(\mathbf{x}_k - \mathbf{o}_k)) = \mathbf{a}_{j0} \tag{6.1}$$

$$\Pi(\mathbf{P}_1(\mathbf{x}_k)) = \mathbf{a}_{j1} \tag{6.2}$$

$$\Pi(\mathbf{P}_2(\mathbf{x}_k + (t_2 - t_1)\mathbf{o}_k) = \Pi(\mathbf{P}_2(\mathbf{x}_k + \mathbf{o}_k)) = \mathbf{a}_{j2} \tag{6.3}$$

where Π is the projective division operator.

Much like triangulation in multi-view stereo [107], this non-linear problem can be transformed into a linear one. Expanding the perspective projection operator gives

$$\frac{[\mathbf{P}_i]_1(\mathbf{x}_k + t_{i1}\mathbf{o}_k)}{[\mathbf{P}_i]_3(\mathbf{x}_k + t_{i1}\mathbf{o}_k)} = a_{ki}, \tag{6.4}$$

$$\frac{[\mathbf{P}_i]_2(\mathbf{x}_k + t_{i1}\mathbf{o}_k)}{[\mathbf{P}_i]_3(\mathbf{x}_k + t_{i1}\mathbf{o}_k)} = b_{ki}, \tag{6.5}$$

where $t_{i1} = t_i - t_1$ (e.g., $t_{01} = -1$, $t_{11} = 0$, and $t_{21} = 1$), and $[\mathbf{P}_j]_m$ is row m of matrix $\mathbf{P}_j$. Multiplying by the denominator gives

$$[\mathbf{P}_i]_1(\mathbf{x}_k + t_{i1}\mathbf{o}_k) - [\mathbf{P}_i]_3(\mathbf{x}_k + t_{i1}\mathbf{o})a_{ki} = 0, \tag{6.6}$$

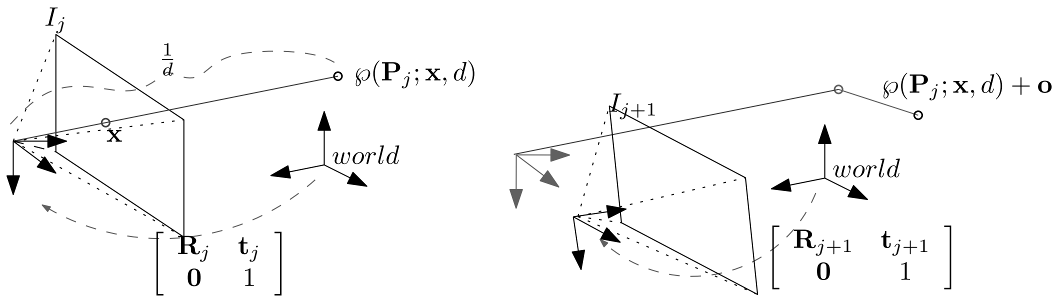
$$[\mathbf{P}_i]_2(\mathbf{x}_k + t_{i1}\mathbf{o}_k) - [\mathbf{P}_i]_3(\mathbf{x}_k + t_{i1}\mathbf{o})b_{ki} = 0, \tag{6.7}$$

for a total of 6 equations in the 6 unknowns. The system of equations (Eq. 6.6 and Eq. 6.7) is rank deficient whenever the backprojected rays from the correspondences lie in the same plane.

The above argument suggests that $|\mathcal{N}(j)| = 2$ is sufficient for reconstruction of geometry and flow, but this minimal view formulation is sensitive to noise. Non-degenerate configurations give a unique solution that satisfies the non-linear system exactly. An obvious way to circumvent noise is to use more image frames (e.g., increase $|\mathcal{N}(j)|$). Unfortunately, given that all the frames are taken at fixed time intervals, using more frames from the sequence means that constant velocity must hold over a longer time period, which may become less likely in a real setting. An alternative to using extra image frames is to regularize the reconstruction; this approach is investigated with a variational formulation in the following section.

6.2.2 Monocular Depth and Flow Reconstruction from Images

The previous discussion demonstrated that it is possible to simultaneously reconstruct both scene flow and depth from a single moving camera with known motion. The reconstruction required



I_j
$\frac{1}{d}$
$\wp(\mathbf{P}_j;\mathbf{x},d)$
$\mathbf{x}$
$world$
$\begin{bmatrix}\mathbf{R}_j & \mathbf{t}_j\\ \mathbf{0} & 1\end{bmatrix}$
I_{j+1}
$\wp(\mathbf{P}_j;\mathbf{x},d)+\mathbf{o}$
$world$
$\begin{bmatrix}\mathbf{R}_{j+1} & \mathbf{t}_{j+1}\\ \mathbf{0} & 1\end{bmatrix}$

Figure 6.3: The scene is represented as an inverse depth map from reference frame j. In the next frame, the camera has moved relative to the scene and the point has moved with constant velocity, $\mathbf{o}$.

correspondences between points be known in advance. Instead of obtaining correspondences from some procedure that is unaware of the constant velocity assumption (e.g., through optic flow), in this section, we derive a direct estimation process using a variational formulation that encodes the constant velocity assumption.

When the correspondences are unknown, the objective can be restated as the recovery of a dense per-pixel mapping of structure and flow at a reference frame (e.g., frame j). The shape in this frame can be represented by a per-pixel map, the disparity (or inverse depth) map. The 3D point corresponding to a 2D point $\mathbf{x} = [x, y]^\intercal$ with disparity d is the following (see Figure 6.3):

$$\wp(\mathbf{P}_j;\mathbf{x},d) = \mathbf{R}_j^\intercal(\mathbf{K}_j^{-1}[x/d, y/d, 1/d]^\intercal - \mathbf{t}_j). \tag{6.8}$$

The constant velocity assumption can now be stated in conjunction with the brightness constancy constraint (e.g., the constraint typically used in optic flow), giving the *constant velocity and constant brightness constraint*:

$$\sum_{i\in\mathcal{N}(j)} |I_i(\Pi(\mathbf{P}_i(\wp(\mathbf{P}_j;\mathbf{x},d) + (t_i - t_j)\mathbf{o})) - I_j(\mathbf{x})|^2, \tag{6.9}$$

which states the intensities of the flowed point in nearby temporally adjacent frames, $\mathcal{N}(j)$, should minimize the difference to the brightness/color in frame j. We use the notation, $\underset{j\to i}{W}$, as the shorthand representation for the backprojection of the 2D point, which is then displaced, and projected into image i:

$$\begin{aligned}\underset{j\to i}{W}(\mathbf{x},d,\mathbf{o}) = \underset{j\to i}{W}(\mathbf{x},d,u,v,w) = \left[\frac{\hat{x}}{\hat{z}},\ \frac{\hat{y}}{\hat{z}}\right]^\intercal &= \\ \Pi(\mathbf{P}_i(\wp(\mathbf{P}_j;\mathbf{x},d) + (t_i - t_j)\mathbf{o})) &= \\ \Pi(\mathbf{K}_i\mathbf{t}_i + \mathbf{K}_i\mathbf{R}_i(\wp(\mathbf{P}_j;\mathbf{x},d) + t_{ij}\,[u\ v\ w]^\intercal)).&\end{aligned}$$

The dense structure and flow, $\mathbf{d} = [d, u, v, w]^\intercal$, for a reference frame j can then be recovered as the minimum of the following energy functional:

$$\begin{aligned} E_j(\mathbf{d}) = &\sum_{i\in\mathcal{N}(j)} \int \Psi_d((I_i(\underset{j\to i}{W}(\mathbf{x}, d, u, v, w)) - I_j(\mathbf{x}))^2)d\mathbf{x} \\ &+ \alpha \int \Psi_s(|\nabla d|^2)d\mathbf{x} \\ &+ \beta \int \Psi_u(|\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2)d\mathbf{x}, \end{aligned} \tag{6.10}$$

where $\Psi_*(x^2)$ are robust functions. This functional is an extension of the variational optic flow methods to disparity and monocular scene flow [45]. Similar to these methods, we use $\Psi_d(x^2) = \Psi_s(x^2) = \Psi_u(x^2) = \sqrt{x^2+\epsilon^2}$ for some small ϵ. The first term, the data term, measures the consistency using the *constant velocity and brightness constraint*; the other terms regularize the disparity and flow separately.

The minimum of Eq. 6.10 is an image-based representation of structure and flow for a single reference frame j that utilizes the constant velocity assumption over a temporal window $\mathcal{N}(j)$. The temporal window $\mathcal{N}(j)$ should contain a subset of the images where this assumption is likely to hold.

Defining $|\nabla\mathbf{o}|^2 = |\nabla u|^2 + |\nabla v|^2 + |\nabla w|^2$, the Euler-Lagrange equations of Eq. 6.10 are

$$\sum_{i\in\mathcal{N}(j)} \Psi'_d(I_{ij}^2)I_{ij}I_{id} - \alpha\nabla\cdot(\Psi'_s(|\nabla d|^2)\nabla d) = 0 \tag{6.11}$$

$$\sum_{i\in\mathcal{N}(j)} \Psi'_d(I_{ij}^2)I_{ij}I_{iu} - \beta\nabla\cdot(\Psi'_u(|\nabla\mathbf{o}|^2)\nabla u) = 0 \tag{6.12}$$

$$\sum_{i\in\mathcal{N}(j)} \Psi'_d(I_{ij}^2)I_{ij}I_{iv} - \beta\nabla\cdot(\Psi'_u(|\nabla\mathbf{o}|^2)\nabla v) = 0 \tag{6.13}$$

$$\sum_{i\in\mathcal{N}(j)} \Psi'_d(I_{ij}^2)I_{ij}I_{iw} - \beta\nabla\cdot(\Psi'_u(|\nabla\mathbf{o}|^2)\nabla w) = 0. \tag{6.14}$$

Where the following abbreviations have been used:

$$I_{ij} = I_i(\underset{j\to i}{W}(\mathbf{x}, d, u, v, w)) - I_j(\mathbf{x}), \tag{6.15}$$

$$I_{id} = \frac{\partial}{\partial d} I_i(\underset{j\to i}{W}(\mathbf{x}, d, \mathbf{o})) = \nabla I_i \frac{\partial}{\partial d} \underset{j\to i}{W}(\mathbf{x}, d, \mathbf{o}), \tag{6.16}$$

$$\frac{\partial}{\partial d}\underset{j\to i}{W} = \begin{bmatrix} \frac{\hat{z}\frac{\partial\hat{x}}{\partial d} - \hat{x}\frac{\partial\hat{z}}{\partial d}}{\hat{z}^2} \\ \frac{\hat{z}\frac{\partial\hat{y}}{\partial d} - \hat{y}\frac{\partial\hat{z}}{\partial d}}{\hat{z}^2} \end{bmatrix} \tag{6.17}$$

$$\frac{\partial\mathbf{p}}{\partial d} = \mathbf{K}_i\mathbf{R}_i\mathbf{R}_j^\intercal\mathbf{K}_j^{-1}\,[1\ 1\ 1]^\intercal\,\frac{1}{d} \tag{6.18}$$

$$\frac{\partial\mathbf{p}}{\partial u} = \mathbf{K}_i\mathbf{R}_i\,[(t_i - t_j)\ 0\ 0]^\intercal\,, \tag{6.19}$$

with $\mathbf{p} = [\hat{x}, \hat{y}, \hat{z}]^\intercal$. $\frac{\partial\mathbf{p}}{\partial v}$ and $\frac{\partial\mathbf{p}}{\partial w}$ are similar to $\frac{\partial\mathbf{p}}{\partial u}$. The abbreviations, I_{iu}, I_{iv}, I_{iw}, are defined similar to I_{id}.

The Euler-Lagrange equations are solved in a similar manner as the optic flow counterpart [45]. First, a fixed point iteration is defined and the data term is linearized (see Appendix B). This linearized equation is solved for several iterations over several scales of an image pyramid (we perform several linearizations at each scale) with a geometric multi-grid method that uses a point coupled Gauss-Seidel method as the basic solver for pre- and post-smoothing. The dimensions of levels of our image pyramid differ by a factor of 2 (see Algorithm 3).

Initialization: One approach to compute a suitable initialization would be to compute optic flow between the reference image and its neighbors and then use techniques in Section 6.2.1 to extract an initial disparity and flow. However, we found it sufficient to obtain a coarse initial disparity from stereo on lower resolution images, which implicitly assumes zero flow. The lower resolution images help account for some of the flow. Disparity is first estimated in a discrete framework (e.g., similar to Zhang *et al.* [282]). These disparities are refined using a variational disparity estimation (e.g., Eq. 6.10 with no flow components). As the coupled flow and disparity refinement is initialized on a low resolution, we found it sufficient to initialize the 3D flow to zero.

Algorithm 3: disp_flow($I_j, \mathcal{N}(j)$)

```
Data: $\mathbf{d}_0 = \{d_0, v_0, u_0, w_0\}$ (e.g., $d_0$ from stereo, $u_0, v_0, w_0$ zero). Reference image $I_j$ and
      neighbors $\mathcal{N}(j)$, $\alpha$, $\beta$
Result: The disparity and flow for $I_j$, $\mathbf{d} = \{d, u, v, w\}$
$\mathbf{d} = \mathbf{d}_0$ ;
// Solve on image pyramid;
for $l \leftarrow l_{max}$ to $0$ do
    $\mathbf{d}$ = resample_solution_for_level($l$, $\mathbf{d}$);
    // Linearize and solve several times;
    for $h \leftarrow 1$ to $n_{lin}$ do
        // Solve linearized problem using multi-grid;
        $\mathbf{d}$ = solve_Ej($\mathbf{d}, \alpha, \beta$);
        // update $\mathbf{d}$ in place, e.g., k = k + 1;
```

6.2.3 Longer Temporal Constraints

It is unlikely that the constant velocity assumption will hold over a long range of frames, so it is desirable to have a formulation that will allow for changes in velocity. A possible image-based method is to solve the variational problem for each image using a close set of neighboring images and then to enforce constraints between these independent solutions.

First, consider how a depth consistency constraint could be encoded in the variational approach. If the problem is to only recover depth from each view (e.g., flow is regularized to be zero), it is straightforward to incorporate a constraint that links the shape in each image. Motivated by the geometric consistency used by Zhang *et al.* [282], which uses neighboring disparity maps to penalize the matching score, we can define an image-based penalizer on disparity.

For example, given the geometry of the cameras, a disparity d for a point $\mathbf{x}$ in image 0 will map the point to some location in image 1. The disparity in image 1 is used to map the point back to image 0, giving a point $\hat{\mathbf{x}}$. Inconsistent disparity maps can be penalized by minimizing $|\hat{\mathbf{x}} - \mathbf{x}|^2$. The situation is illustrated in Figure 6.4.

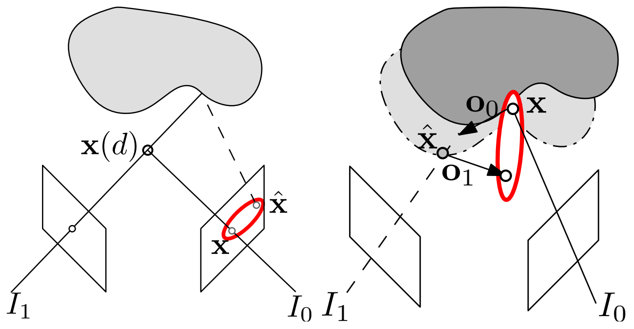


Figure 6.4: Left: a simple disparity map consistency constraint where the disparity for point $\mathbf{x}$ is penalized by the distance to the point $\hat{\mathbf{x}}$. Right: a flow constraint penalizing the distance in 3D.

Similar constraints can be defined when each image contains independent estimates of both structure and flow. Instead of defining these constraints in the image plane, the constraints are defined in 3D (see Fig. 6.4). Using image j as a reference, the constraints can be incorporated into the variational problem as the following:

$$E_{ji}^{\text{cons}}(\mathbf{d}) = \int\int \kappa \Psi_d(|h_i(\underset{j\to i}{W}(\mathbf{x},\mathbf{d})) - \wp_j(\mathbf{x},\mathbf{d}))|^2)\mathbf{dx}, \tag{6.20}$$

where $\wp_j(.,.)$ is the backprojection function for image j. The function h_i is the backproject-and-offset function and uses the current estimates of structure and flow from image i. Using this formulation, the structure and flow for all images can be extracted in a manner that ensures consistency and allows deviations from constant velocity.

The combined problem of estimating the disparity and flow from all frames with constraints can be thought of as finding the minimum to the following expression:

$$E_{\text{total}}(\{\mathbf{d}_i\}) = \sum_{j=1}^{T}(E_j(\mathbf{d}_j) + \sum_{i\in\mathcal{N}(j)} E_{ji}^{\text{cons}}(\mathbf{d}_j)). \tag{6.21}$$

We obtain an approximate solution to this equation by first solving for the disparity and flow at each image independently (e.g., minimize E_j), and then in a second pass introduce the constraints, E_{ji}^{cons} and solve again for E_j holding all the other disparity and flow constant (Algorithm 4).

6.2.4 Parameter Settings

The algorithm has two parameters to control the regularization of the disparity and flow, α and β, and an additional weight, κ, used for longer sequences. Algorithm 3 also has a parameter to control the number of linearizations performed, n_{lin}.

In practice, the defaults of $\alpha = 0.5$ and $\beta = 0.5$ are manually tuned by looking at the warped images overlayed on the reference image. If the warped images do not align well with the reference image, then it is likely that the regularization is too strong and both weights α and β are reduced. On the other hand, if the reconstructed depth or flow is too rough, then the regularization for that component is increased.

```
Algorithm 4: disp_flow_many($\{I_i\}$)
Result: $\{\mathbf{d}_j\}$, disparity and flow for all images.
// Solve for each view independently;
for $j \leftarrow 1$ to $T$ do
    $d$ = basic_stereo($j, \mathcal{N}(j)$);
    $\mathbf{d}_j$ = $\{d, \mathbf{0}, \mathbf{0}, \mathbf{0}\}$;
    $\mathbf{d}_j$ = disp_flow($j, \mathcal{N}(j)$, $\mathbf{d}_j$);
// Solve again with constraints;
for $l \leftarrow l_{max}$ to $0$ do
    for $j \leftarrow 1$ to $T$ do
        $\mathbf{d}_j$ = resample_solution_for_level($l$, $\mathbf{d}_j$);
        for $h \leftarrow 1$ to $n_{iters}$ do
            $\mathbf{D} = \{\mathbf{d}_k | k \in \mathcal{N}(j)\}$;
            // Linearize and minimize Eq. 6.21 w.r.t $\mathbf{d}_j$;
            $\mathbf{d}_j$ = solve_cons($j, \mathcal{N}(j)$, $\mathbf{d}_j$, $\mathbf{D}$);
```

If reducing the smoothness cannot give accurate warped images (or the resulting displacements or offsets are too irregular), then the number of linearizations, n_{lin}, is increased. This is often necessary when the disparity or offsets in the scene are large.

The parameter, κ, need only be non-zero for longer sequences where it is desired to have consistency between the reconstructions of neighboring images.

6.2.5 Monocular Scene Flow Experiments

In this section, the validity of the monocular flow using the constant velocity assumption is evaluated. As it is hard to get quantitative results for real sequences, synthetic sequences are first used to evaluate the performance of the variational flow recovery under varying camera baseline. Qualitative results are illustrated for real sequences containing both rigid and non-rigid motion.

Sensitivity to Varying Baseline

We first consider a synthetic example where the motion obeys our constant velocity assumption. The setup consists of three views roughly 4 units from a planar object that deforms into a bumpy surface with some shifting (Fig. 6.5). The object is 2 units wide and has flows of 10% of this size.

To test the sensitivity of our estimate to perturbations in camera baseline, we have run our algorithm on varying proportions of the full camera baseline (which is roughly 1.3 and 1.9 units for left and right cameras respectively). The weights of $\alpha = \beta = 0.83$ were chosen from a set of 6 options $(6.67, 4, 2, 1.33, 1, 0.833)$ as these settings gave the best results compared to ground truth.

Figure 6.6 shows the distance to ground truth for the initial disparity estimation and for the refined disparity and flow estimations (these are computed as average difference in depth for the disparity values and average vector difference for the flows). The figure shows that our method outperforms the initial disparity, which is ignorant of the flow. Also, as expected the estimation

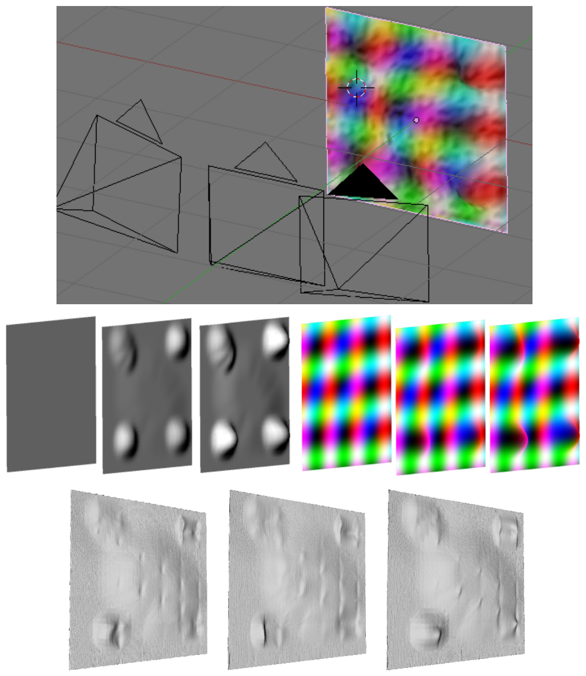


Figure 6.5: Top: the moving camera setup (full baseline depicted here). Middle: shaded and textured (side-views) of the deforming object. Bottom: the initial disparity estimate (e.g, zero flow), which averages the deformations.

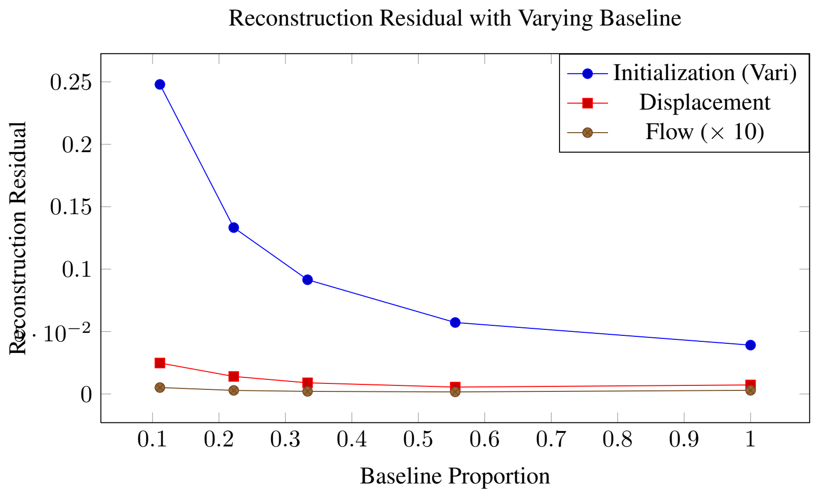


Figure 6.6: Distance to the ground truth depth for the initial depth estimate and recovered depth (measured as the average absolute distance). Also, the distance between the recovered flow and the ground truth flow (measured as the average distance between the flow vectors).

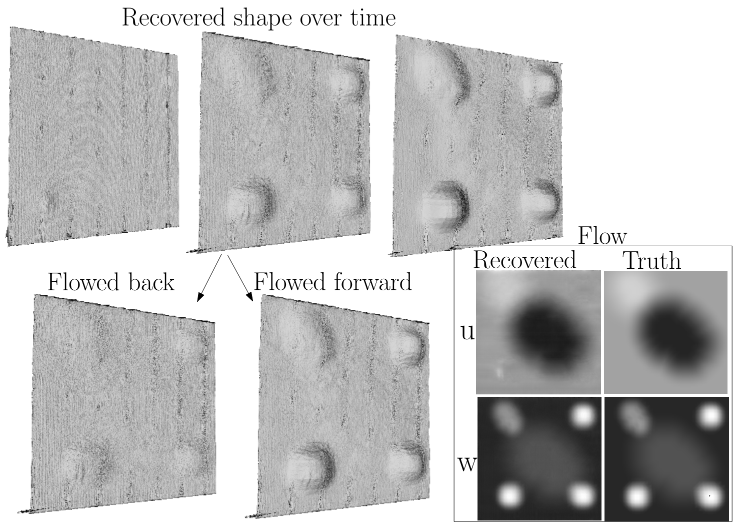


Figure 6.7: Top: the recovered geometry for each time step (and the flowed geometry from time 1) are similar to the ground truth (Fig. 6.5). Bottom right: recovered u and w components beside the ground truth ($t = 1$, $\alpha = 0.83$).

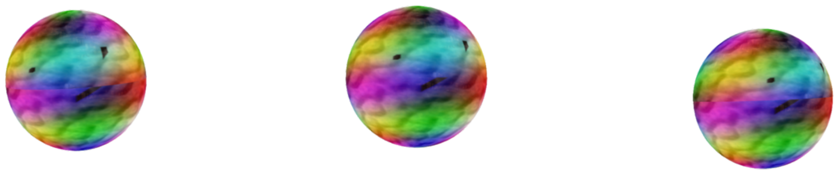


Figure 6.8: The three input images for the rotating sphere.

degrades with too small baseline, and the reconstructed flow degrades with too large of a baseline (possibly due to increased occlusions).

Figure 6.7 illustrates our results for the full baseline case. Notice that this geometry looks like the recovered (and ground truth) geometry in the neighboring frames. This similarity is evident in the image representations of the u and w components compared to the ground truth. The average depth error is 0.007 ($\sim 0.35\%$ of the object size) and average flow error of 0.0029 ($\sim 0.15\%$ of the flow size).

Scene with Motion that Does not Have Constant Velocity

In a second synthetic experiment we used a similar camera motion as the previous case, but the scene is a sphere with two hemispheres rotating in opposite directions (similar to Huguet & Devernay [114]). One hemisphere rotates 5 degrees and the other 10 degrees (Figure 6.8). In this case the motion does not obey our constant velocity assumption.

In Figure 6.9 we see that standard disparity estimation fails in this case, but even with this poor initialization our algorithm produces something much more sphere-like (parameters $\alpha = \beta = 2$).

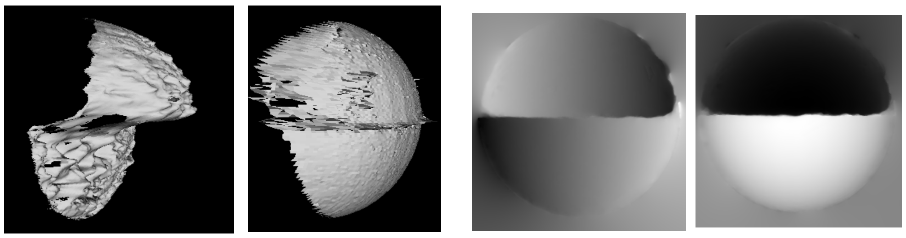


Figure 6.9: Rotating sphere results, from left: side view of stereo geometry, refined geometry, recovered u and w flows.

Sequence	T	size	α	β	κ	l_{max}	n_{iters}
Two house	3	800x600	8	4.8	0.02	4	10
Mouth	3	800x600	6.25	5.63	0.18	4	20

Table 6.1: Information and parameter settings for the real sequences used in the monocular flow experiments.

The recovered flow maps approximate the non-linear rotation motion with a constant velocity.

6.2.6 Real Sequences

The first sequence has three views of two houses: the left house is rotating counter clockwise (from top), the right house translates to the right and is draped with a shirt that moves non-rigidly, and the background is stationary (parameters are in Table 6.1). Camera motion is extracted relative to the background using a calibration pattern, and motion is manually introduced into the scene. Figure 6.10 shows the input views and the reconstructed geometry at each time from a novel viewpoint; Figure 6.11 shows the 3D flow and another viewpoint. The geometry is consistent through the 3 frames, and the flow (Fig. 6.11) represents the rigid house motion, the non-rigid cloth motion, and the stationary ground.

The second monocular real data set was three views of a person opening his mouth. Camera motion was again extracted with a calibration pattern and is relative to the subjects skull. Figure 6.12 shows the input views and the resulting geometric reconstructions. In this case there is some motion in the z-coordinate not present in the sequence. Notice that the motion of the camera is significant enough to make this sequence challenging for strict optic flow; the addition of depth and flow allows successful recovery. See Figure 6.13 for the images warped to time frame 1.

6.3 Basis Constrained Scene Flow on a Proxy

The previous section illustrated that with a simple temporal constraint it is possible to reconstruct structure and non-rigid motion from a single moving camera. It was assumed that the camera motion was known; in a human setting this motion could come from the coarse tracking for each each kinematic link. Although the above formulation was useful to demonstrate the feasibility of simple temporal constraints for monocular reconstruction, it took place in a simple setting and had sev-

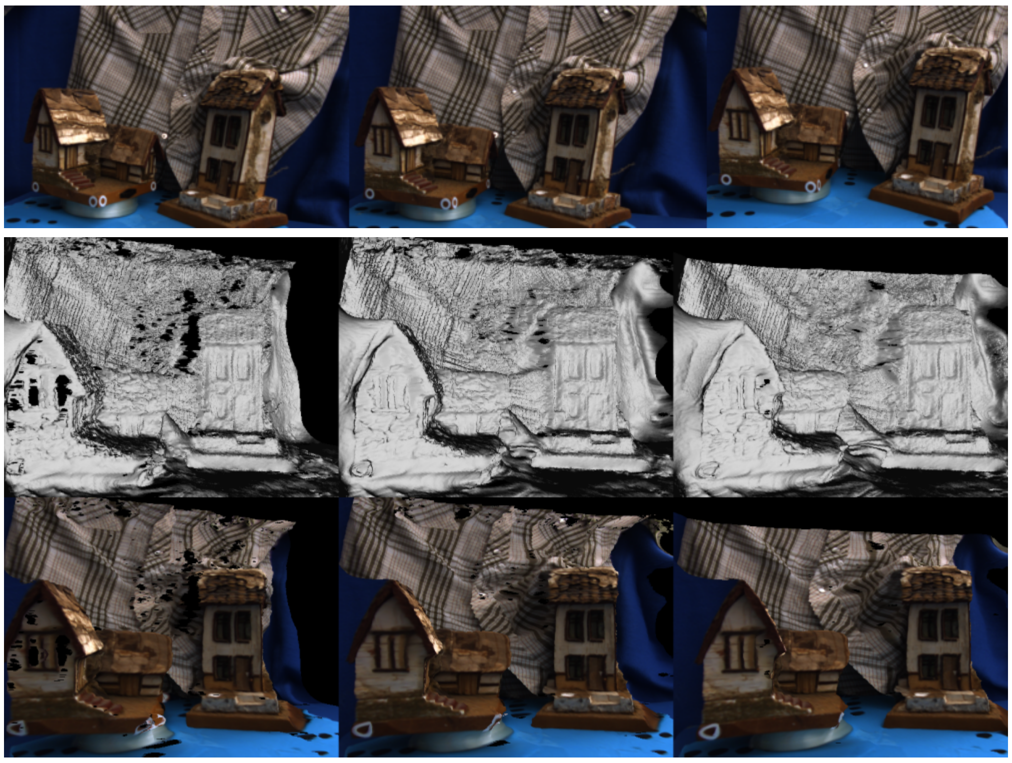


Figure 6.10: Top: Cropped input views. Bottom: This novel viewpoint illustrates that the geometry was accurately reconstructed for each of the time sequences (notice the rightward motion of the right house and the rotation of the left house).

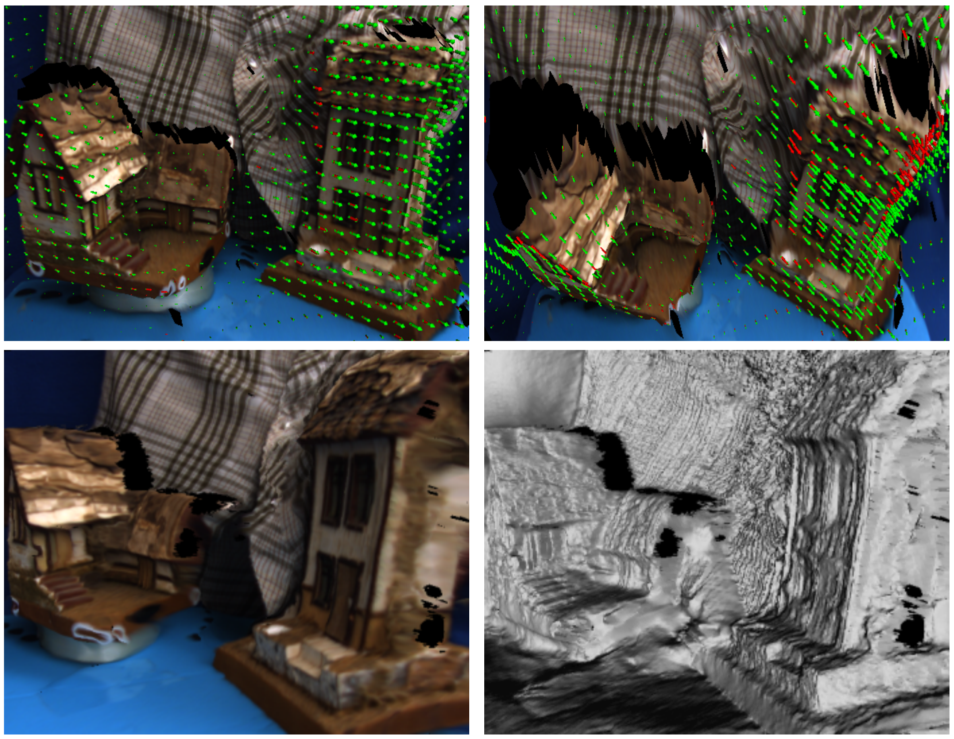


Figure 6.11: Novel views and flow visualization for the house sequence. Top: 3D views of the overlaid flow (red is in front of object, green is behind). Bottom: novel viewpoint of the geometry at time 1.

eral limitations in the context of acquiring surface deformations of a kinematically tracked human: working with an image-based representation of shape for multiple different moving kinematic links is clumsy, long-range constraints were awkwardly linked through constraints on multiple image-based reconstructions, extra cameras were not accounted for, and the constant velocity limitation is too restrictive.

In this section, these issues are addressed by formulating the problem directly on the coarse tracked geometry. The use of the coarse moving geometry means that other views can more naturally be taken into account (should they overlap). Furthermore, more general constraints on the motion through the use of temporal basis functions are enforced, which allows for longer-range correspondences while still enabling monocular improvement of geometry. We refer to the coarsely tracked geometry as a proxy and reconstruct displacements and scene flow relative to this proxy.

Before describing the basis constrained scene flow on a proxy, an intuitive example is used to illustrate the concept of using a basis to constrain temporal point motion when some coarse scene motion is known. Assuming known correspondences (similar to Section 6.2.1), this intuitive example highlights some of the benefits of using a temporal basis to represent surface flow, including the ability to overcome noise and obtain reconstructions with missing data.

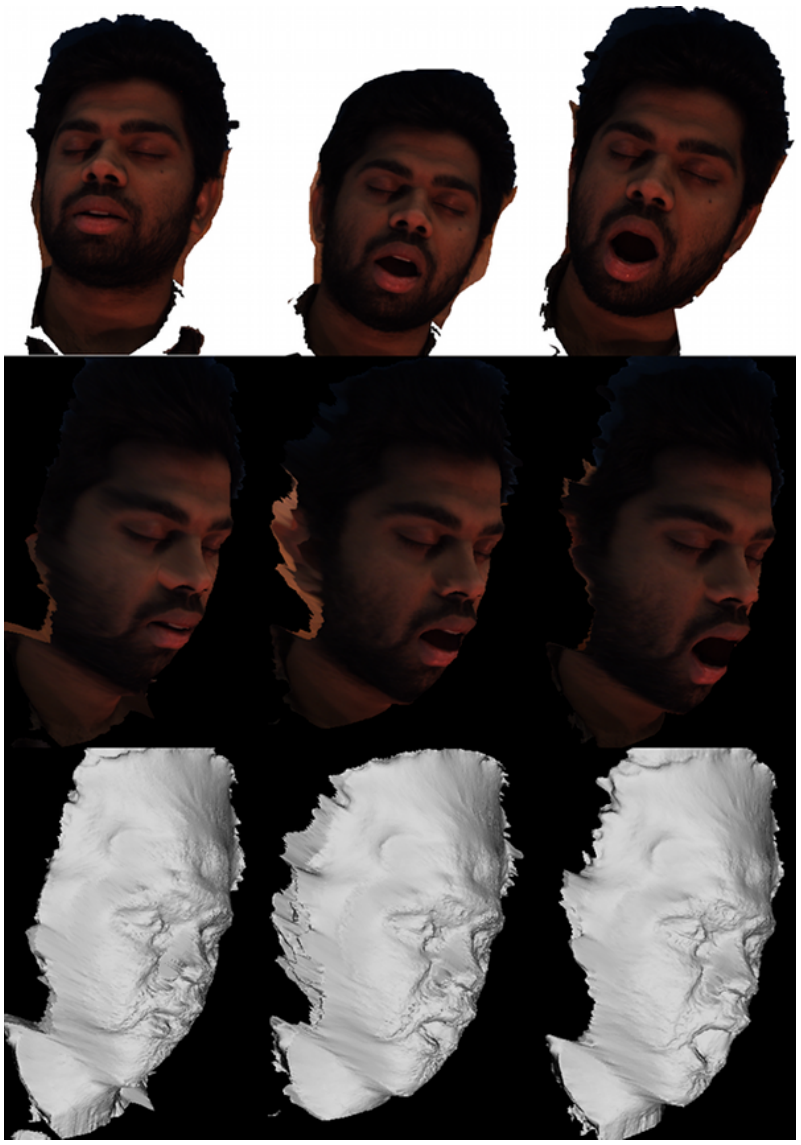


Figure 6.12: Top: monocular input views of the facial motion sequence. Middle & Bottom: textured and shaded results from a novel viewpoint.

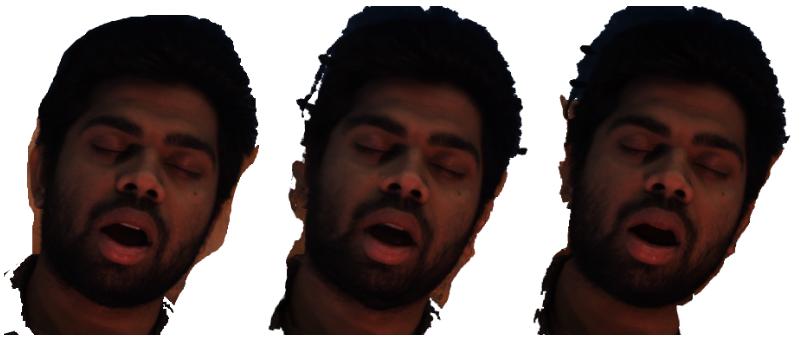


Figure 6.13: Warped images for the monocular facial motion sequences: images 0 (left) and 2 (right) warped to image 1 (middle) using the recovered depth and flow.

6.3.1 Known Correspondences

Let us consider the simple case (as in Section 6.2.1), where a single moving camera with 3×4 projection matrix, $\mathbf{P}_1$, and camera motion $\{\mathbf{E}_t\}_{t=1}^{T}$ observes a dynamic scene[1]. If correspondences of a moving point, $\mathbf{u}_t = [u_t, v_t]^\intercal$, are known, one can formulate an estimation process for the unknown moving 3D point, $\mathbf{x}(t)$, like triangulation:

$$\min \sum_{t=1}^{T} |\Pi(\mathbf{P}_1 \mathbf{E}_t \mathbf{x}(t)) - \mathbf{u}_t|^2. \tag{6.22}$$

As there are 2 constraints per image, modeling the motion as independent 3D points over time (e.g., $\mathbf{x}(t) = [x_t, y_t, z_t, 1]^\intercal$) gives $3T$ unknowns, leading to an ill-posed problem. Section 6.2.1 discussed constraining the motion by splitting the trajectory into a mean point, $\hat{\mathbf{x}}$, with a constant velocity, $\bar{o} = [o_1, o_2, o_3]$. This gives fewer variables (e.g., 6):

$$\mathbf{x}(t) = \hat{\mathbf{x}} + \mathbf{o}(t) = [\hat{x}, \hat{y}, \hat{z}, 1]^\intercal + (t - t_0)[o_1, o_2, o_3, 0]^\intercal. \tag{6.23}$$

Again, when camera motion is sufficient, this representation allows for a unique reconstruction if $T \geq 3$. The restrictive constant velocity assumption can be generalized by using a temporal basis to encode the time-varying displacements:

$$\mathbf{o}(t) = \sum_{k=1}^{H} \lambda_k \mathcal{B}_k(t), \tag{6.24}$$

where $\{\mathcal{B}_k(t)\}_{k=1}^{H}$ are temporal basis functions ($H < T$).

This same framework holds if there are several views, $\{\mathbf{P}_i\}_{i=1}^{C}$, with correspondences $\mathbf{u}_{it}$. In this case, one can think of $\mathbf{E}_t$ as the coarse rigid motion of the scene (if known), and Eq. 6.22 would be accumulated over each view:

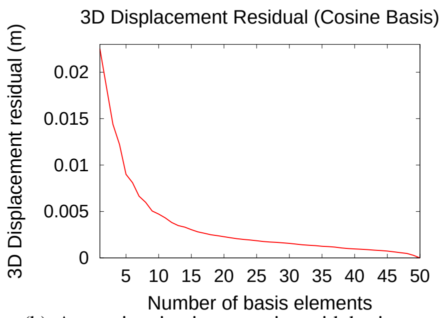
$$\min \sum_{i=1}^{C} \sum_{t=1}^{T} |\Pi(\mathbf{P}_i \mathbf{E}_t \mathbf{x}(t)) - \mathbf{u}_{it}|^2. \tag{6.25}$$

When more than two cameras observe the point in each frame, the reconstruction is no longer ill-posed. However, using a basis to represent $\mathbf{o}(t)$ has several benefits: temporal smoothness enforced by the appropriate basis helps overcome noise, there are fewer parameters, and reconstruction is possible with missing observations.

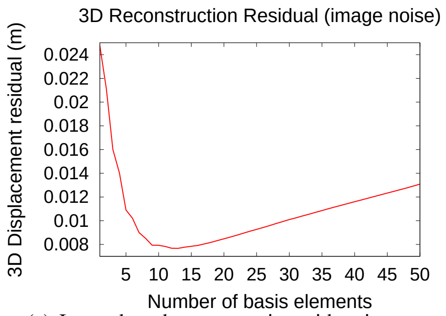
In the case of a multi-view studio filming an actor, the coarse geometry can be tracked, which gives the motion of each bone over the sequence. However, the real surface exhibits residual non-rigid motion in addition to the skeleton (e.g., clothing). For a point on the surface, the coarse motion of the scene, $\mathbf{E}_t$, is given by the corresponding bone motion; this motion will be different for points attached to different bones. A linear algebraic solution for the displacement trajectory (i.e., the λ_k

[1]Here the motion of the camera is parameterized in a separate Euclidean transform but it could be absorbed into a varying projection matrix. This formulation makes it more apparent that we can utilize known scene motion as well.

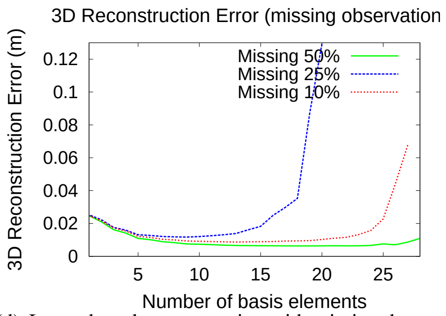
(a) Input sequence



(b) Approximating input motion with basis



(c) Image-based reconstruction with noise



(d) Image-based reconstruction with missing data

Figure 6.14: a) Images 0, 20, 40, 50 from camera 1. b) the *displacement residual* when approximating the 3D displacements using varying basis elements. c) two-view image-based reconstruction from noisy correspondences using Eq. 6.26 achieves better results when using roughly 13 basis elements. d) using 10-20 basis elements enables reconstruction with missing data from the second view.

coefficients) can be obtained by solving the space-time triangulation problem; the linear version of the problem can again be obtained by multiplying by the denominator (as was done in Section 6.2.1):

$$\begin{aligned}\min_{\lambda_k} &\sum_{i,t}([\mathbf{P}_i\mathbf{E}_t]_1 - u_{it}[\mathbf{P}_i\mathbf{E}_t]_3)(\mathbf{x}_t + \sum_k \lambda_k \mathcal{B}_k(t))^2 \\ &+ \sum_{i,t}([\mathbf{P}_i\mathbf{E}_t]_2 - v_{it}[\mathbf{P}_i\mathbf{E}_t]_3)(\mathbf{x}_t + \sum_k \lambda_k \mathcal{B}_k(t))^2,\end{aligned} \tag{6.26}$$

where $[\mathbf{P}]_i$ is the i^{th} row of $\mathbf{P}$.

To illustrate this formulation, we test the ability of a low-dimensional basis to approximate and reconstruct non-rigid trajectories layered on a human skeleton. The motion capture, deformed meshes, and surface displacements are from the MIT *crane* data set [259]. We approximated the displacements relative to the bone in the first 50 frames with the discrete cosine basis. Figure 6.14 illustrates 3D residuals for an increasing number of basis functions. At about 15 basis functions a good approximation of the surface motion is obtained; this requires only $\frac{1}{3}$ the parameters of the unconstrained motion. Furthermore, when reconstructing displacements from two views with noisy image observations[2] using Eq. 6.26, fewer variables (e.g,. 12 basis functions) gives better results than more variables (Fig. 6.14 c), implying the lower dimensional basis helps in the presence of noise.

The ability to reconstruct in the presence of missing data is also illustrated in Figure 6.14 d), where 10%, 25%, and 50% of the observations in the second camera were unknown. In these cases, solving for unconstrained motion is ill-posed, and the best reconstruction comes from restricting the motion to a lower (e.g., 10-20) dimensional basis.

6.3.2 Variational Basis Constrained Scene Flow on a Proxy

In the previous subsection, a temporal basis was used to recover flow when coarse scene motion ($\mathbf{E}_t$) was known and correspondences were given. In this section, the idea of coarse motion (e.g., motion of bones) is replaced with the formalism of a moving approximate proxy surface. The formulation below assumes any known proxy surface (although in practice it will be the skinned geometry). Instead of assuming known correspondences, the low-dimensional basis constraint is used to directly estimate the scene flow and scene structure relative to the proxy surface. The problem is posed below, followed by a general definition of a proxy surface, and scene flow estimation is formulated relative to this proxy surface.

Problem Definition

Given input images, $I_{i,t}$, taken from $1 \leq i \leq C$ cameras at stationary viewpoints over time $t \in \{1, \cdots, T\}$, the objective is to reconstruct the dense structure of the surface and the 3D surface flow with respect to a known approximate proxy-surface. The camera calibration is given: $\mathbf{P}_i = [\mathbf{K}_i|\mathbf{0}][\mathbf{R}_i\mathbf{t}_i]$, with internals $\mathbf{K}_i$, external rotation, $\mathbf{R}_i$, and translation $\mathbf{t}_i$. As a shorthand, $\Pi_i(\mathbf{x}) = \Pi(\mathbf{P}_i\mathbf{x})$. If the proxy is moving (as is the case for human geometry), it is assumed that this motion is approximately known through an external tracking process.

Proxy Surface

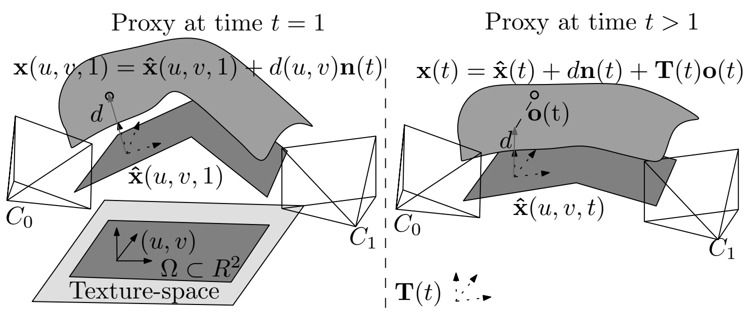
As input, a known 3D proxy surface $\hat{\mathbf{x}}(u, v, t) : (\Omega \subset \mathbb{R}^2) \times \mathbb{Z}^+ \mapsto \mathbb{R}^3$, embedded in 2D at discrete time steps t is provided as input. The surface normal is defined as

$$\mathbf{n}(u, v, t) = \frac{\hat{\mathbf{x}}_u(u, v, t) \times \hat{\mathbf{x}}_v(u, v, t)}{|\hat{\mathbf{x}}_u(u, v, t) \times \hat{\mathbf{x}}_v(u, v, t)|}, \tag{6.27}$$

with $\mathbf{x}_u$ and $\mathbf{x}_v$ being the surface tangent vectors. The tangent frame matrix is defined as

$$\mathbf{T}(u, v, t) = \left[\frac{\hat{\mathbf{x}}_u(u, v, t)}{|\hat{\mathbf{x}}_u(u, v, t)|}, \frac{\hat{\mathbf{x}}_v(u, v, t)}{|\hat{\mathbf{x}}_v(u, v, t)|}, \mathbf{n}(u, v, t) \right]. \tag{6.28}$$

[2]Uniform noise with a maximum of 1.5 pixels was added to the x, y coordinates independently



Proxy at time $t = 1$
Proxy at time $t > 1$
$\mathbf{x}(u,v,1) = \hat{\mathbf{x}}(u,v,1) + d(u,v)\mathbf{n}(t)$
$\mathbf{x}(t) = \hat{\mathbf{x}}(t) + d\mathbf{n}(t) + \mathbf{T}(t)\mathbf{o}(t)$
d
$\mathbf{o}(t)$
d
$\hat{\mathbf{x}}(u,v,1)$
$\hat{\mathbf{x}}(u,v,t)$
C_0
C_1
C_0
C_1
(u,v)
$\Omega \subset R^2$
Texture-space
$\mathbf{T}(t)$

Figure 6.15: The flow and structure is represented relative to a possibly moving proxy surface observed by several cameras. At a reference frame (e.g., $t = 1$), the true surface is simply a displacement d from the proxy along the normal. In subsequent frames, the surface is represented as the displaced point with an additive flow component.

Recovering Displacements and Flow

Scene motion is modeled relative to the moving scene proxy with a time-dependent 3D temporal offset, $\mathbf{o}(u, v, t)$, and a displacement along the normal $d(u, v)$:

$$\mathbf{x}(u,v,t) = \hat{\mathbf{x}} + \mathbf{T}(u,v,t)\mathbf{o}(u,v,t) + d(u,v)\mathbf{n}(u,v,t). \tag{6.29}$$

The displacement, d, is with respect to a reference time, e.g., $t = 1$. The third component of the offset vector $\mathbf{o}(u,v,t) = [o_1(u,v,t), o_2(u,v,t), o_3(u,v,t)]^\intercal$, accounts for temporal changes in normal motion.[3] Using, $\mathbf{T}$, the tangent frame of the surface, implies that the offsets are represented in the coordinate frame of the surface (see Figure 6.15). Setting $\mathbf{T}$ to the identity means the surface motion would be represented in the global world coordinate frame.

Unlike the previous section, which assumed known correspondences, here 3D surface displacements, d, and long range flow, $\mathbf{o}$, are recovered directly from image intensities. The desired displacements and flow should be both photo-consistent and flow-consistent. Therefore, the displacement and flow can be recovered by minimizing the following functional, which uses brightness constancy to measure flow- and photo-consistency:

$$F(d,\mathbf{o}) = \underbrace{\sum_{t=1}^{T} F_s(d,\mathbf{o},t)}_{\text{Stereo}} + \alpha \underbrace{\sum_{t=2}^{T} F_f(d,\mathbf{o},t)}_{\text{Flow}} + \underbrace{\sum_{t=1}^{T} F_r(d,\mathbf{o},t)}_{\text{Regularization}} \tag{6.30}$$

$$F_s(d,\mathbf{o},t) = \int_{\Omega} \sum_{j \neq i} w_{ijt} \Psi(|I_{i,t}(\Pi_i(\mathbf{x}(t))) - I_{j,t}(\Pi_j(\mathbf{x}(t)))|^2) dA \tag{6.31}$$

$$F_f(d,\mathbf{o},t) = \int_{\Omega} w_{it1} \Psi(|I_{i,t}(\Pi_i(\mathbf{x}(t))) - I_{i,1}(\Pi_i(\mathbf{x}(1)))|^2) dA \tag{6.32}$$

$$F_r(d,\mathbf{o},t) = \beta_2 \int_{\Omega} \Psi(|\nabla \mathbf{o}(u,v,t)|^2) dA + \beta_1 \int_{\Omega} \Psi(|\nabla d(u,v)|^2) dA \tag{6.33}$$

[3]Due to o_3, the displacement component, d, can be dropped. However, we find it advantageous to regularize the displacement component, d, separately and choose basis functions with zero displacement at the reference time.

Problem	Time	Proxy (static/dynamic)	An example basis
Optic flow [45]	$t=2$	Image plane (static)	$\mathcal{B}_1(t)=[1,0,0]$, $\mathcal{B}_2(t)=[0,1,0]$
Optic flow with basis [99]	$t>1$	Image plane (static)	E.g., $\mathcal{B}_1(t)=[t,0,0]$, $\mathcal{B}_2(t)=[0,t,0]$
Stereo with basis for depth	$t\geq 1$	Image plane (static)	E.g., $\mathcal{B}_1=[0,0,t]$, cosine basis, ...
Displaced proxy [262]	$t\geq 1$	Approx geom. (dynamic)	E.g., $\mathcal{B}_1=[0,0,t]$, cosine basis, ...
Scene flow [114]	$t=2$	Image plane (static)	$\mathcal{B}_1=[1,0,0], \mathcal{B}_2=[0,1,0], \mathcal{B}_3=[0,0,1]$
Scene flow on proxy	$t>1$	Approx geom. (dynamic)	$\mathcal{B}_1=[t,0,0], \mathcal{B}_2=[0,t,0], \mathcal{B}_3=[0,0,t]$

Table 6.2: Our basis-constrained scene flow on a proxy generalizes a number of approaches. In optic flow, simple stereo, or scene flow, the proxy would be the image plane. The basis functions limit the reconstruction to the 2D offsets (for optic flow) or the depth (for stereo), and can easily extend to multi-frame estimates. Our application is the most general: scene flow on top of a dynamic proxy.

$$|\nabla\mathbf{o}|^2 = \sum_{j=1}^{3} |\nabla o_j|^2 \tag{6.34}$$

where $\mathbf{x}(t)$ is a shorthand for $\mathbf{x}(u,v,t)$ in Eq. 6.29, $dA = dudv$, and $\Psi(x^2) = \sqrt{x^2+\epsilon^2}$ is a smooth L1 norm [45]. The weighting term $w_{ijt}(u,v)$ (resp. w_{it1}) is used to encode visibility or reliability of the image observations for stereo pairs (resp. flow pairs). See Section 6.3.3 for implementation details.

The stereo term, F_s, ensures the displaced surface is photo-consistent at each time; the flow term, F_f, like optic flow, ensures the intensity at the flowed points is in agreement with the reference frame (e.g., $t = 1$). The regularization acts both on the displacements d and the offsets $\mathbf{o}$ and prefers to have smoothly varying 3D flow at each time t.

Temporal Basis-constrained Flow

The functional in Eq. 6.30 is dependent on the shape displacement $d(u,v)$, and the surface offsets $\mathbf{o}(u,v,t)$, at each time $2 \leq t \leq T$. Allowing arbitrary displacements would require a large number of variables and would need an extra term for temporal smoothness. Instead, as in Eq. 6.24, the motion is modeled with a low-dimensional basis, but here the coefficients of motion, $\lambda_k : \Omega \mapsto \mathbb{R}$, are now spatially varying scalar fields. The 3D displacement field is then

$$\mathbf{o}(u,v,t) = \sum_{k=1}^{H} \mathcal{B}_k(t)\lambda_k(u,v), \tag{6.35}$$

where again $\{\mathcal{B}_k(t)\}_{k=1}^{H}$ are a set of basis functions. For example, a constant velocity basis using time 1 as a reference would use $\mathcal{B}_1(t) = [t-1,0,0]^\mathsf{T}, \mathcal{B}_2(t) = [0,t-1,0]^\mathsf{T}$ and $\mathcal{B}_3(t) = [0,0,t-1]^\mathsf{T}$. The total number of unknown scalar fields reduces from $3(T-1)+1$ to $H+1$.

The basis elements can be more elaborate. As in the 2D flow (e.g., [99]), the motion basis functions can be factored from a sparse set of representative tracks. We utilize the discrete cosine basis, which has been used to constrain trajectories of moving points observed from moving cameras [178]. As a consequence, when proxy motion is known (and sufficient), it is possible to reconstruct the 3D flow trajectory when the surface is only visible by one camera.

An advantage of this framework is that it naturally generalizes a number of variational approaches involving dense correspondence, including optic flow, stereo, and scene flow. Table 6.2 lists the corresponding proxy and basis functions for a number of these applications.

Euler-Lagrange Equations

Scalar fields d and $\{\lambda_k\}_{k=1}^{H}$ that minimize Eq. 6.30 must satisfy the Euler-Lagrange equations:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{j\neq i} & w_{ijt}\Psi'(I_{ijt}^2)I_{ijt}\frac{\partial}{\partial d}I_{ijt}+ \\ \alpha\sum_{t=2}^{T} & w_{it1}\Psi'(I_{it1}^2)I_{it1}\frac{\partial}{\partial d}I_{it1}- \\ & \beta_1\nabla\cdot(\Psi'(|\nabla d|^2)\nabla d)=0,\end{aligned} \tag{6.36}$$

and for all $1 \leq k \leq H$:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{j\neq i} & w_{ijt}\Psi'(I_{ijt}^2)I_{ijt}\frac{\partial}{\partial \lambda_k}I_{ijt}+ \\ \alpha\sum_{t=2}^{T} & w_{it1}\Psi'(I_{it1}^2)I_{it1}\frac{\partial}{\partial \lambda_k}I_{it1}- \\ & \beta_2\nabla\cdot(\Psi'(|\nabla \mathbf{o}|^2)\sum_{c=1}^{3}\nabla o_c[\mathcal{B}_k(t)]_c)=0.\end{aligned} \tag{6.37}$$

Here, the shorthand $I_{ijt} = I_{i,t}(\Pi_i(\mathbf{x}(t))) - I_{j,t}(\Pi_j(\mathbf{x}(t)))$, $I_{it1} = I_{i,t}(\Pi_i(\mathbf{x}(t))) - I_{i,1}(\Pi_i(\mathbf{x}(1)))$. The partial derivatives of these terms w.r.t. d and λ_k are

$$\frac{\partial}{\partial d}I_{ijt} = \nabla I_{i,t}\frac{\partial}{\partial d}\Pi_i(\mathbf{x}(t)) - \nabla I_{j,t}\frac{\partial}{\partial d}\Pi_j(\mathbf{x}(t)), \tag{6.38}$$

$$\frac{\partial}{\partial d}\Pi_i(\mathbf{x}(t)) = \underbrace{\Pi'(\mathbf{K}_i(\mathbf{R}_i\mathbf{x}(t)+\mathbf{t}_i))}_{2\times 3}\underbrace{\mathbf{K}_i\mathbf{R}_i\mathbf{n}(u,v,t)}_{3\times 1}, \tag{6.39}$$

$$\frac{\partial}{\partial \lambda_k}I_{ijt} = \nabla I_{i,t}\frac{\partial}{\partial \lambda_k}\Pi_i(\mathbf{x}(t)) - \nabla I_{j,t}\frac{\partial}{\partial \lambda_k}\Pi_j(\mathbf{x}(t)), \tag{6.40}$$

$$\frac{\partial}{\partial \lambda_k}\Pi_i(\mathbf{x}(t)) = \underbrace{\Pi'(\mathbf{K}_i(\mathbf{R}_i\mathbf{x}(t)+\mathbf{t}_i))}_{2\times 3}\underbrace{\mathbf{K}_i\mathbf{R}_i\mathbf{T}(u,v,t)}_{3\times 1}\mathcal{B}_k(t), \tag{6.41}$$

$$\Pi'((x\ y\ z)^\intercal) = \begin{bmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} \\ 0 & \frac{1}{z} & -\frac{y}{z^2}\end{bmatrix}. \tag{6.42}$$

$\frac{\partial}{\partial d}I_{it1}$ and $\frac{\partial}{\partial \lambda_k}I_{it1}$ are similar.

The above Euler-Lagrange equations are linearized and solved using geometric multi-grid methods (see Appendix C).

6.3.3 Implementation

The formulation in the previous section operated on a generic, possibly moving proxy surface. In this subsection, we specify implementation details for our specific choice of proxy surface: a linear-blend skinned triangulated mesh. Details are also given for the embedding of the proxy surface and how the tangent space is defined.This choice of base mesh representation easily models static, rigidly moving, or even kinematic proxy surfaces. This section also has further details on the initialization and optimization.

Embedding & tangent space

Each triangle s_i has a corresponding set of 2D texture coordinates, $(\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \mathbf{u}_{i,3})$. The piecewise linear mapping from 2D to 3D for a point $(u_i, v_i)^\intercal \in \triangle(\mathbf{u}_{i,1}, \mathbf{u}_{i,2}, \mathbf{u}_{i,3})$ with barycentric coordinates $(u_i, v_i)^\intercal = \sum_{j=1}^{3} a_j \mathbf{u}_{i,j}$, is

$$\hat{\mathbf{x}}(u_i, v_i, t) = \sum_{j=1}^{3} a_j \mathbf{v}_{s_{i,j}}. \tag{6.43}$$

The tangent frame matrix is also interpolated across triangles in the 2D domain using these same barycentric weights, a_j. The surface normal at each vertex is the average of unnormalized triangle normal (e.g., $(\mathbf{v}_{s_{i,2}} - \mathbf{v}_{s_{i,1}}) \times (\mathbf{v}_{s_{i,3}} - \mathbf{v}_{s_{i,1}})$).

Tangent directions are estimated as follows. The edges of a triangle in 3D are linearly related to the edges of the 2D parameterization through the tangent directions, $\mathbf{x}_x(\triangle_i)$ and $\mathbf{x}_y(\triangle_i)$:

$$(\mathbf{v}_{i,2} - \mathbf{v}_{i,1}) = \mathbf{x}_x(\triangle_i)[\mathbf{u}_{i,2} - \mathbf{u}_{i,1}]_x + \mathbf{x}_y(\triangle_i)[\mathbf{u}_{i,2} - \mathbf{u}_{i,1}]_y \tag{6.44}$$

$$(\mathbf{v}_{i,3} - \mathbf{v}_{i,1}) = \mathbf{x}_x(\triangle_i)[\mathbf{u}_{i,3} - \mathbf{u}_{i,1}]_x + \mathbf{x}_y(\triangle_i)[\mathbf{u}_{i,3} - \mathbf{u}_{i,1}]_y. \tag{6.45}$$

The per-triangle tangents are easily estimated from these equations:

$$\mathbf{x}_x(\triangle_i) = \frac{[\mathbf{u}_{i,3} - \mathbf{u}_{i,1}]_y(\mathbf{v}_{i,2} - \mathbf{v}_{i,1}) - [\mathbf{u}_{i,2} - \mathbf{u}_{i,1}]_y(\mathbf{v}_{i,3} - \mathbf{v}_{i,1})}{c} \tag{6.46}$$

$$\mathbf{x}_y(\triangle_i) = \frac{[\mathbf{u}_{i,2} - \mathbf{u}_{i,1}]_x(\mathbf{v}_{i,3} - \mathbf{v}_{i,1}) - [\mathbf{u}_{i,3} - \mathbf{u}_{i,1}]_x(\mathbf{v}_{i,2} - \mathbf{v}_{i,1})}{c} \tag{6.47}$$

$$c = [\mathbf{u}_{i,2} - \mathbf{u}_{i,1}]_x[\mathbf{u}_{i,3} - \mathbf{u}_{i,1}]_y - [\mathbf{u}_{i,3} - \mathbf{u}_{i,1}]_x[\mathbf{u}_{i,2} - \mathbf{u}_{i,1}]_y. \tag{6.48}$$

As in the case for normals, the tangents for each vertex are averaged over the triangles that contain it.

Initialization and Optimization

In general, the proxy mesh can either come from a sparse set of stereo points or a manually initialized selection of points. In our application, the proxy mesh is the coarse skinned geometry. The uv-parameterization can be obtained by projection of vertices into a single-view, through automatic methods, or it can manually assigned by a user in modeling software.

In the case of no approximate proxy motion, all motion will be relative to the original proxy surface. When a kinematic skeleton is available, the surface of the skeleton is tracked using the silhouettes or image-based scores presented in Chapter 5.

The image weights, w_{ijt}, in Eq. 6.30 are set as $w_{ijt}(u, v) = w_{it}(u, v)w_{jt}(u, v)$, where

$$w_{it}(u, v) = \begin{cases} \langle \mathbf{n}(u, v, t), \mathbf{l}_i \rangle & \text{if } \mathbf{x}(u, v, t) \text{ visible in } I_i \\ 0 & \text{otherwise,} \end{cases}$$

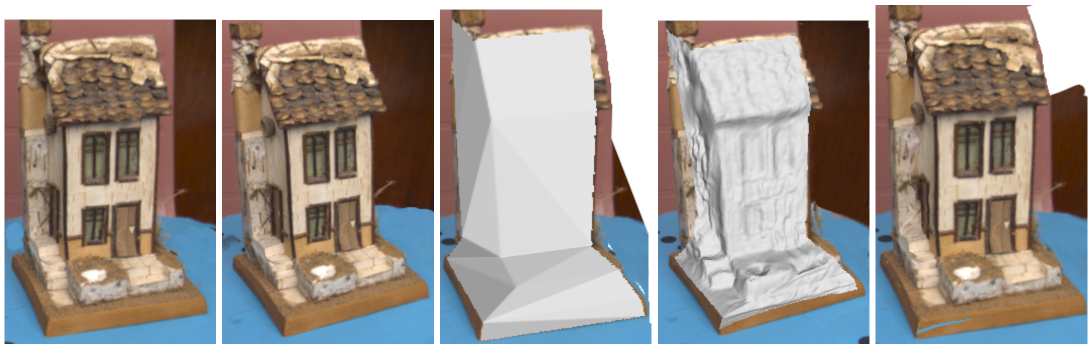


Figure 6.16: From left: two of four input views, the initial proxy, the recovered displaced mesh, and the textured displaced mesh.

with $\mathbf{l}_i$ being the ray to camera i. The flow weights, w_{it1}, are defined similarly.

The Euler-Lagrange equations (Eq. 6.36 & 6.37) are solved on multiple resolutions of the uv-space and the input images. Solutions from lower resolutions are propagated to the higher resolutions by simple image resampling.

Parameter Settings

If the flow in the scene varies from constant velocity, then we approximate the motion with the discrete cosine basis. The number of basis functions used depends on the motion of the scene, but we typically use about $\frac{1}{3}T$ basis functions.

The relative weights of the terms in Eq. 6.30 are determined in a similar heuristic manner as the weights for the monocular flow in Section 6.2.4. In the multi-view case, it is helpful to first estimate the regularization parameters for the displacement, β_1, using the images from a single time instant. If the recovered displacement is too noisy then β_1 needs to be increased. Alternatively, if warping images from one view to the other using the reconstructed depth is inaccurate, then the regularization is likely too strong and needs to be reduced.

The flow component, α, can then be introduced. This term is increased if the warped images from $t > 1$ do not agree with the reference time frame. It can be helpful to tune this parameter using only a restricted version of the problem where T is limited to two. The relative weight of β_2 is then adjusted to ensure a sufficiently smooth flow field.

6.3.4 Basis Constrained Scene Flow Experiments

In this section, the results of the basis-constrained flow implementation are demonstrated on several synthetic and real data sets. Although the representation was designed for a moving skinned geometry, in this section the generality of the basis-constrained flow is demonstrated on a number examples applications from Table 6.2, ranging from simple depth estimation, to flow estimation on a rigidly moving object, to estimation of flow on a kinematic skeleton.

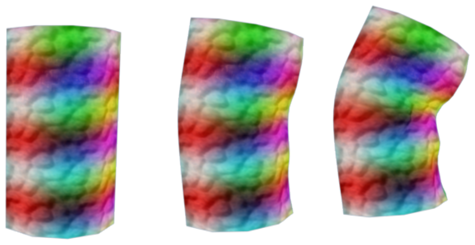


Figure 6.17: The three input images (from camera 1) for the synthetic skinned cylinder illustrate the motion and texture on the object.

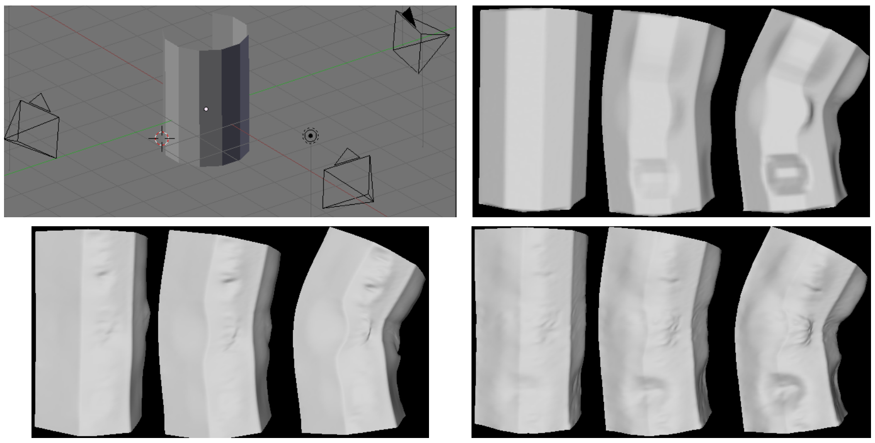


Figure 6.18: Top left: the non-overlapping camera configuration used in the cylinder example. Top right: ground truth geometry from first viewpoint. Bottom left: inaccurate reconstruction results with no constraint on the motion. Bottom right: more accurate reconstruction using a constant velocity constraint.

Static Proxy, Depth Only

Figure 6.16 illustrates one of the applications that our framework generalizes: depth from a static proxy. A static proxy geometry was manually created for the house object, which is observed by four calibrated cameras. The depth is refined with no temporal basis (as there is only one time instant). The resulting geometry accurately portrays the details of the house, and the texture mapped mesh looks indistinguishable from the input images.

Skinned Object, no Camera Overlap

In this synthetic data set we illustrate the helpful effect of using the basis constraints in the case of kinematic motion when there is no overlap between the camera views. In this experiment, constraining the motion of the surface with a basis is shown to improve the surface reconstruction. The sequence contains 3 views and 3 time instances of a cylindrical object (roughly 3 units high by 1.5 units wide). The two bones of the object undergoing a small bend and small translation motion (see Fig. 6.17 for input and Fig. 6.18 for camera configuration). The base motion of the skeleton is assumed to be known, and the ground truth skinning weights are used. The underlying motion over

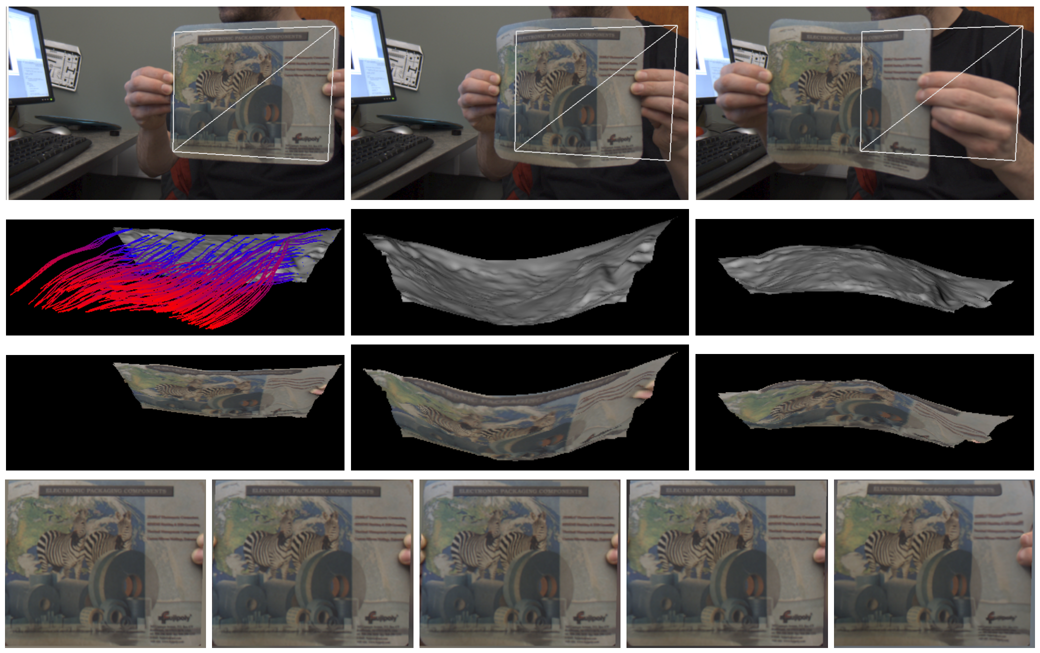


Figure 6.19: Top: input frames 0, 7, and 15 from the mousepad sequence (white rectangle shows static proxy). Middle: shaded and textured reconstructions (several trajectories plotted in $t = 0$). Bottom: rectified surface texture over several frames.

the three frames is linear.

The surface motion was reconstructed both with a constant velocity basis and with an unconstrained flow basis. As was the case for monocular sequences (Section 6.2.5) the use of a constant velocity basis allows more accurate reconstruction of the depth (Fig. 6.18). Although the unconstrained basis does recover some of the tangential flow, the depth estimates are still inaccurate. The average of the median position error at each time step was 0.0209 and 0.0135 for the unconstrained and constrained reconstructions respectively. The base surface error, with no displacements, is 0.0722.

Static Proxy, Multiple Views

The proxy motion need not be known in advance (e.g., the proxy can be static). In this example, data was collected from a static stereo setup that captured a translating deforming mousepad (Fig. 6.19). The base geometry consisted of 2 triangles specified manually in the first frame. A cosine basis with 8 basis elements (for each flow component) is used to model the motion over 16 frames. The surface motion is successfully modeled (illustrated by the reprojection of texture) and the flowed geometric surface exhibits the deformations of the mousepad.

Rigid Moving Proxy, Single View

This example illustrates that taking motion of the proxy into account aids the reconstruction. A single view from the previous experiment was used, and the flow results from the multi-view case

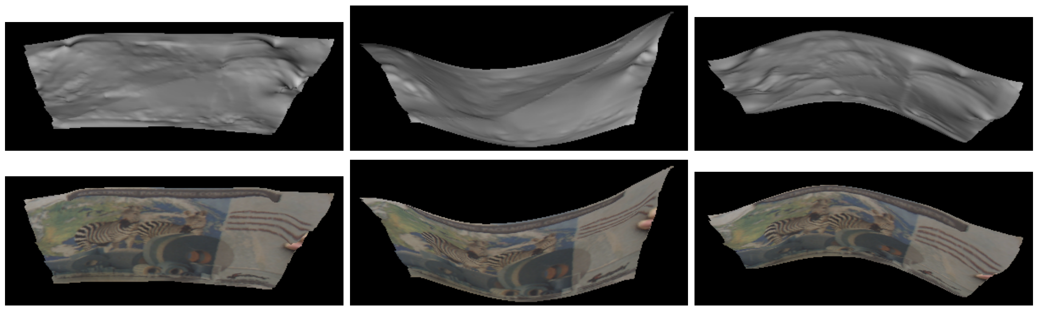


Figure 6.20: Shaded and texture reconstructions for the single-view time-varying depth only reconstruction of the mousepad on frames 0, 7, and 15.

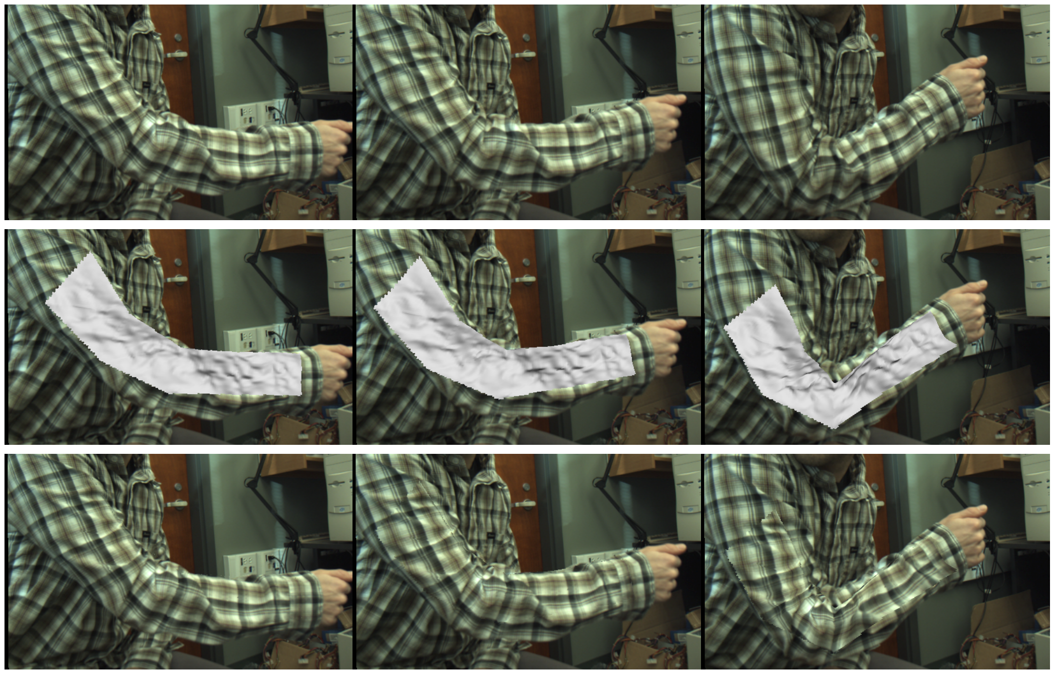


Figure 6.21: Top: input images from time 0, 5, 14. Middle: reconstructed geometry. Bottom: deformed textured with $I_{1,0}$.

were used to find the rigid motion of the proxy. The time-varying depth on the surface was then recovered over the sequence using the first 3 cosine basis functions to model the displacement along the normal (flow components were not modeled). As illustrated in Figure 6.20, the time-varying depth was successfully recovered from the single-view sequence.

Two Link Arm, Skinned Proxy

The full model is demonstrated in this final example, which contains a moving skinned proxy observed by multiple cameras. The proxy object is a moving arm (Fig. 6.21). Two views were used to track the geometry and to obtain an initial shape (a skeleton was manually inserted and skinning weights were obtained automatically [25]). The texture resolution was 128x128 and a cosine basis with 7 elements for each coordinate were used to model the motion over 16 frames. Notice that the geometry starts to recover the wrinkles.

6.4 Silhouette Displacements

The previous two sections dealt with reconstructing geometry and flow from image brightness constraints, possibly when the input cameras had little overlap. However, these methods did not utilize silhouette information. In practice it is desirable to recover deformations that ensure the model silhouettes are similar to those in the input views. In order to bring silhouettes of a model into alignment with input silhouettes, we use an approach that applies constraints to the occluding contour of the object [259]. In order to extrapolate these sparse displacements over the mesh while retaining the overall shape of the reference mesh, a differential representation in terms of the Laplacian coordinates is used.

The Laplacian coordinates provide an intrinsic representation of the mesh that is commonly used for mesh editing applications. For example, a user can edit a mesh by applying constraints to a few vertices while having the edits affect the entire surface. The Laplacian coordinates provide a way to regularize such edits while preserving the local geometric features of the original mesh [172]. The Laplacian coordinates are defined as follows:

$$\delta\mathbf{v}_i = \left(\sum_{j \in \mathcal{N}_i} c_{ij}\mathbf{v}_j\right) - \mathbf{v}_i$$

where $\mathcal{N}_i$ are the neighbors of vertex i. The simplest weights, c_{ij}, are uniform (e.g., $c_{ij} = \frac{1}{\mathcal{N}_i}$), but cotangent weights are reported to give better results [7].

Let $\mathbf{X} = [x_1, x_2, \cdots]^\intercal$, $\mathbf{Y} = [y_1, y_2, \cdots]^\intercal$, and $\mathbf{Z} = [z_1, z_2, \cdots]^\intercal$ be vectors containing the x, y, and z components of the mesh vertices. Defining the matrix, $\mathbf{V} = [\mathbf{XYZ}]$, the Laplacian coordinates can be expressed in a matrix form as:

$$\mathbf{LV} = (\delta\mathbf{v}_1, \delta\mathbf{v}_2, ...)^\intercal = \delta\mathbf{V}$$

The Laplacian matrix $\mathbf{L}$ is sparse with $\mathbf{L}_{ij} = c_{ij}$ if vertex j is a neighbor of i, and $\mathbf{L}_{ii} = -1$.

The Laplacian coordinates encode intrinsic detail and provide a natural way to regularize ill-posed problems involving the mesh. Specifically, aligning the mesh to the silhouettes involves forming a sparse set of positional constraints that pull occluding contours towards the silhouette boundary. The positional constraints are simply the desired position of some location on the mesh. Each constraint target point, $\mathbf{b}_j = [b_{j,x}, b_{j,y}, b_{j,z}]^T$, can be expressed as a barycentric combination of 3 mesh vertices, given by triangle $\Delta_j \in \mathbb{Z}^3$ and barycentric weights β_{jk}:

$$(\sum_{k=1}^{3} \beta_{jk}\mathbf{v}'_{\Delta_{jk}}) = \mathbf{b}_j. \tag{6.49}$$

As the constraints are sparse, in order to find displacements for all vertices, the constraints are included in an objective that constrains the recovered vertices, $\mathbf{V}'$, to have similar Laplacian coordi-

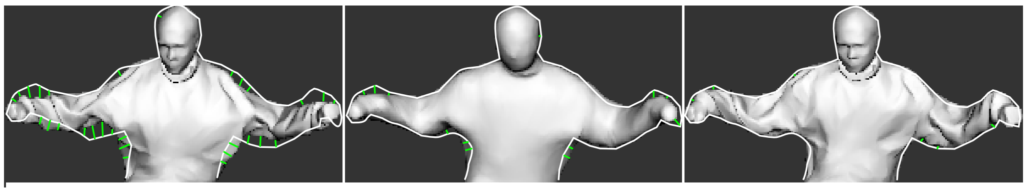


Figure 6.22: Left: the posed mesh doesn't agree with the silhouette (white line). Constraints are added (green lines) and Eq. 6.51 is solved with a low-level of detail to give the results (middle). The process is repeated with increasing detail to give the final result (right). The final result retains detail of the original model, but better fills the silhouette.

Algorithm 5: SilhouetteConstraints

```
foreach detail level, $l \in \{0.0, 0.2, 0.4, ..., 1\}$ do
    foreach time $t$ do
        Find constraints, $\mathbf{B}$, from all input views ;
        Compute transformed Laplacian coordinates using $\mathbf{\Theta}_t$ (Eq. 6.51) ;
        Solve Eq. 6.50 (using scaled coordinates, $l\delta\mathbf{V}$) to get displaced vertices $\mathbf{V}_t$ ;
    Temporally smooth displacements.
```

nates as the rest mesh,

$$\min_{\mathbf{V}'} \|\mathbf{L}\mathbf{V}' - \delta\mathbf{V}\|^2 + \lambda\|\mathbf{C}\mathbf{V}' - \mathbf{B}\|^2, \tag{6.50}$$

where matrix $\mathbf{C}$ is a sparse constraint matrix: $\mathbf{C}_{ij}$ is non-zero if constraint i constraints vertex j, and row j of $\mathbf{B}$ is $\mathbf{b}_j$. The weight, λ, controls the relative strength of positional constraints versus the regularization. Equation 6.50 is easily solved by a direct sparse linear least squares solver.

As the Laplacian coordinates are rotation variant, motion of the underlying skeleton needs to be accounted for. Instead of using $\delta\mathbf{V}$ directly, the Laplacian coordinates are first transformed by the skinning equation using the recovered joint angles [259]:

$$[\delta\hat{x}_i, \delta\hat{y}_i, \delta\hat{z}_i, 0]^\intercal = \sum_{i=1}^{|\mathbf{\Theta}|} w_{j,i}\mathbf{T}_i(\mathbf{\Theta})[\delta x_i, \delta y_i, \delta z_i, 0]^\intercal. \tag{6.51}$$

The constraints are then imposed on vertices that are likely to be on the silhouette. In other words, the input silhouettes are discretized and a constraint is added at regular intervals along the silhouette curve (Figure 6.22).[4] If the backprojected ray does not intersect the surface, a constraint pulling the closest point on the surface geometry towards the ray is added. On the other hand, if the ray intersects the geometry, the closest surface point to the deepest intersection on the ray is pulled towards the ray. In either case, vertices with normals that differ too much from the normal of the silhouette curve are penalized when finding the closest surface point.

Finding and enforcing constraints is done in several iterations using the previous deformed mesh to locate the new constraints. Additionally, the Laplacian coordinates are scaled down, which re-

[4]This is in contrast to the ICP correspondence method used in the tracking of Chapter 5 that established correspondences between each point on the projection of the mesh to the closest image silhouette point. By establishing correspondence at evenly spaced samples on the input silhouette, it is more likely that the deformed model will completely fill the silhouettes. However, this requires the input silhouettes to be accurate and free of noise.

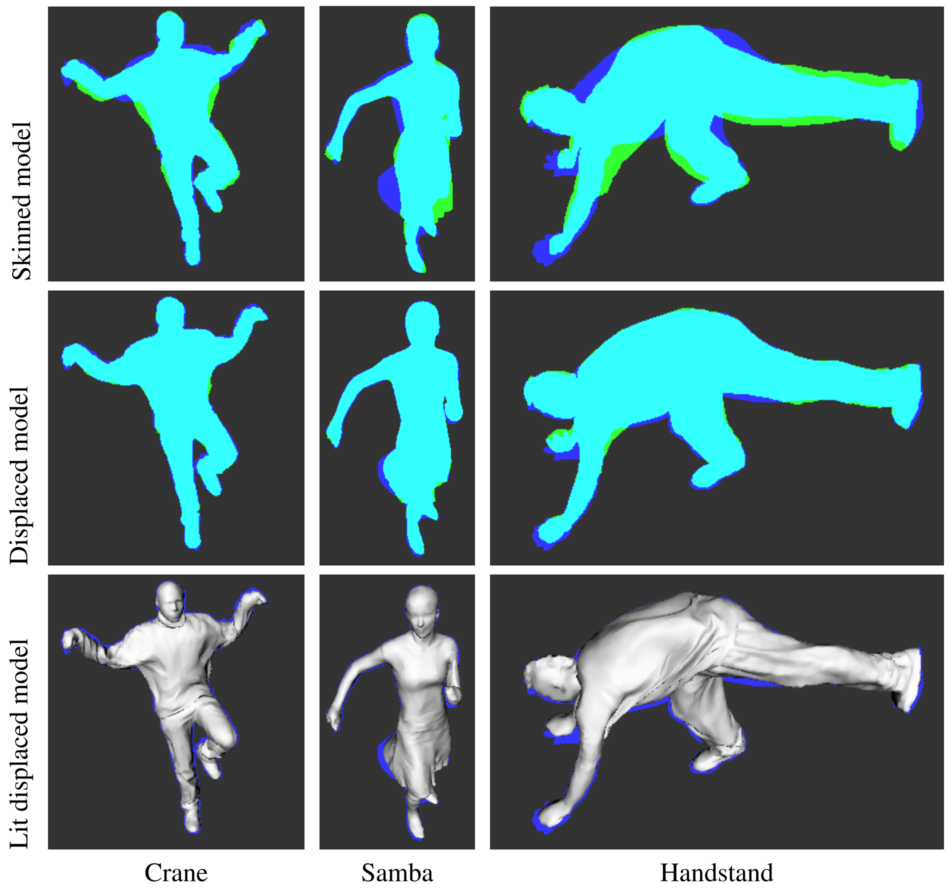


Figure 6.23: Skinned silhouette overlap (top row), displaced silhouette overlap (middle row), and displaced geometry (bottom row). Notice that the displaced geometry has a better silhouette overlap and that the geometry retains detail information from the laser scan geometry.

moves high frequency features. Solving the system of equations extrapolates the sparse constraints over the entire mesh giving a dense displacement for all of the vertices. Temporal consistency is improved by Gaussian smoothing the displacements over time. The algorithm is outlined in Algorithm 5.

Data set	Skinned	Displaced
Crane	0.876	0.940
Samba	0.861	0.936
Handstand	0.840	0.942

Table 6.3: The silhouette overlap (Jaccard index) for the skinned and displaced mesh.

As described above, this method is completely oblivious of appearance, and hence has no way to ensure flow- or photo-consistency. On the other hand, it is effective for models that lack sufficient texture, and could be combined with other positional constraints based on flow or stereo. To illustrate the effect of using the silhouette consistency force, Figure 6.23 plots the silhouette overlap of the skinned mesh and the displaced mesh for a single frame (input sequences and tracking results were from the MIT data sets [259]). The silhouette consistency increases when using the silhouette displacement framework. Table 6.3 contains quantitative measurements of silhouette overlap on these three sequences. Figure 6.24 shows

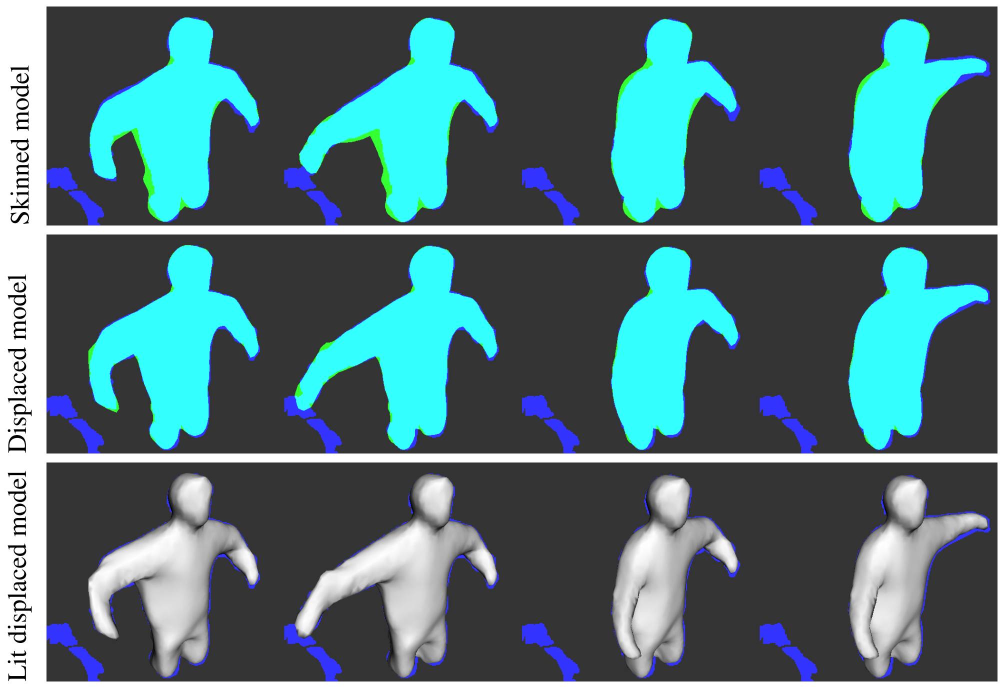


Figure 6.24: In this 4 camera sequence (one view shown here), due to noisy silhouettes the displaced model has some errors (near right arm). However, overall the displaced model improves silhouette overlap.

the results of the silhouette displacements on a sequence with noisier input silhouettes. In this case, noisy silhouettes in one of the other viewpoints causes a small bump to appear near the right elbow. Due to this, the regularization has to be increased, but the silhouette displaced model is still a better fit to the input images.

6.5 Discussion

In this chapter the problem of reconstructing dense geometric deformations and temporal flow correspondences of a non-rigid surface were discussed. This task is possible even when there is little overlap between cameras provided temporal constraints are imposed on the flow.

In the case of a moving monocular camera, a variational method was proposed to estimate scene flow and geometry. Assuming a constant velocity over a small temporal window in such circumstances allows for reconstruction of both geometry and structure; this assumption is embedded in the variational formulation and does not require separate estimation (§ 6.2.2). Constraints similar to constant velocity have been used in the literature. Avidan and Shashua show that points undergoing linear motion can be recovered with as few as five views from a moving camera [16]. Constant velocity is a subset of linear motion, allowing a reconstruction with as few as three views. Similarly, some have considered restricting motion to a scaled piecewise affine model over a larger time window [285]. As in our case, the constraint over a time window allows more accurate estimation of the parameters (e.g., in their case the affine model parameters, and in our case the velocity). The constant velocity assumption over a temporal window has been alluded to in earlier scene flow works (e.g., [49]). In fact all multi-camera methods that compute scene flow using image sets from two adjacent time frames are essentially assuming constant velocity for the two frames, but this assumption has not been exploited to extract structure and flow for more than two consecutive images from a monocular sequence as it is in this chapter.

A specific limitation of this formulation is that the flow should be well approximated by constant velocity over a short time frame. A worst case scenario is when the surface motion reverses directions. Clearly, surface motion in an infinitely small time window can be well approximated as constant velocity. But we rely on the motion of the camera relative to the object in order to reconstruct the depth, so there is a trade-off between being able to reconstruct depth (e.g., big enough baseline) and the approximation of constant velocity holding.

This monocular formulation (and representation) was completely image-based, and required linking of several reconstructions over longer sequences (e.g., using constraints on the flow). Furthermore, the monocular formulation required the camera motion to be known and sufficient. In the case of recovering flow on a human geometry, it is assumed that this relative camera motion would come from the human's articulated links (extracted through multi-camera tracking). In the second component of this chapter (§ 6.3), this image-based limitation was lifted and a more general formulation of flow on a moving proxy was proposed. The formulation on a moving proxy is better suited to multi-view acquisition, and easily allows both photo-consistency and flow-consistency terms in the energy functional. The formulation utilized an abstract notion of a proxy surface, which in our implementation uses a linear-blend skinned mesh. Motion of the proxy, through joint angle tracking, is utilized during flow reconstruction, implying that intra-camera observations can influence the shape (as was the case in the monocular framework).

The formulation on the moving proxy utilized a more general temporal constraint, namely, that the temporal 3D motion of the surface points were restricted to lie in a lower-dimensional temporal basis. This basis formulation generalizes the constant velocity assumption, and allows for longer range flow to be recovered with fewer unknowns than if arbitrary motion were allowed. Furthermore, the use of a basis to represent 3D flow on a proxy generalizes existing approaches for optical flow, multi-view stereo, and scene flow.

Although the proxy representation is natural for representing structure and flow, there are some limitations. Most importantly is that the proxy must be appropriate for the scene. Specifically, the underling unknown scene must be a function of depth from the proxy. This is likely the case for a human geometry, but it may not be the case for more general scenes. As only brightness constraints are used, the surface flow can be confused when there are illumination changes. Such illumination changes are likely due to relative surface motion with respect to the lights. This problem can be addressed by using a data term less sensitive to illumination (e.g., using the gradient of the image, or changing the aggregation to use normalized cross-correlation).

The implementation of the basis flow algorithms also have some practical limitations. The linearization in Appendix C uses $(H+1) \times (H+1)$ tensors for each data term (and each pixel). These tensors limit the number of basis functions that can be used for long range sequences. Removing the robust norms on data terms may allow for a more efficient memory implementation. Also, our implementation currently does not handle discontinuities in the uv-parameterization, although it is possible to take this into account in the discretization (e.g., as in [101]).

Recovering dense deformations directly from intensities is complicated by self-occlusions, cloth-folding, and appearance changes. Therefore, as an alternative to the intensity driven scene flow methods, the silhouette-based deformation method of Vlasic *et al*. [259] is used to ensure the geometric deformations fit the input silhouettes. Such silhouette-based deformation is useful to produce a geometric proxy for image-based rendering. The Laplacian deformation framework used to apply the silhouette constraints also allow for the integration of stereo cues.

7 Animating and Rendering the Compact Model

The static model geometry (Chapter 4), coarse motion via tracked joint angles (Chapter 5), and dense surface deformations (Chapter 6) are necessary components to build the pose- and view-dependent model from image sequences. In this chapter, we combine these components and validate the use of the pose- and view-dependent model described in Chapter 3.

A foundational assumption of this model is that the appearance of an object can be represented as a function of pose (and viewpoint). This assumption is qualitatively examined in Section 7.2. As the quality of the final model is influenced by errors in any one of the three acquisition stages, initial experiments are performed on combinations of the various components in increasingly challenging situations (see Table 7.1). The simplest instance is one with a single geometric part and no object motion. On synthetic imagery, this simple instance is used to justify the impact of combining a geometric component with an appearance component as opposed to using a strictly appearance-based model (Section 7.3.1). Increasing complexity, the model is analyzed on real images using a single kinematic link with a single input view (Section 7.4), a single kinematic link viewed from multiple viewpoints (Section 7.5), and a multiple part interpolation space example of a human face animation with no kinematic model (Section 7.6). The full model is demonstrated on multiple kinematic links from several viewpoints for an upper human body (Section 7.7), and the lower body is re-animated under novel pose and viewpoints (Section 7.8)

The compact model was designed to encode pose- and view-dependent appearance and geometry of human subjects for the purpose of re-rendering. This same model can also be used in an applications such as compression of free-viewpoint performance captures (Section 7.9.2) and the acquisition and rendering of a non-rigid accordion (Section 7.9.1).

7.1 Model-building Considerations

Each experiment has several considerations regarding acquisition and modeling. Acquisition choices include how the base model is obtained, how coarse joint angle tracking was recovered, and how fine-scale deformations were obtained. The base model was created with one of the following alternatives:

- *truth:* the ground truth base geometry was used (e.g., for some synthetic experiments).
- *manual:* the base geometry was manually edited using input photos as reference.

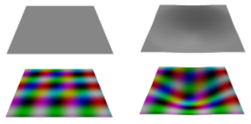
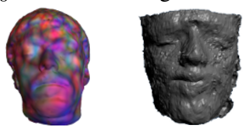
§ 7.2 Is appearance pose-dependent? C 1 Synth: No Kin: N/A #parts: N/A #interp: N/A	§ 7.3.1 Are geometry deforms necessary?  C 4 Synth: Yes Kin: No #parts: 1 #interp: 1
§ 7.3.2 Blending of multiple parts  C 2 Synth: Yes Kin: No #parts: 3 #interp: 1	§ 7.4 Silhouette/appearance in real setting C 1 Synth: No Kin: Yes #parts: 2 #interp: 1
§ 7.5 Different pose and view sampling C Many Synth: No Kin: Yes #parts: 2 #interp: 1	§ 7.6 Qualitative: face interpolation C 4 Synth: No Kin: No #parts: 3 #interp: 1
§ 7.7 Qualitative: upper body model C 4 Synth: No Kin: Yes #parts: 4 #interp: 2	§ 7.8 Qualitative: novel leg animation C 6 Synth: No Kin: Yes #parts: 4 #interp: 2
§ 7.9.1 Non-rigid accordion model C Many Synth: No Kin: Yes #parts: 1 #interp: 1	§ 7.9.2 Free-viewpoint compression C 8 Synth: No Kin: Yes #parts: 3 #interp: 1

Table 7.1: An overview of the statistics (e.g., number of views (C), if synthetic data was used, if a kinematic model was used, number of parts, maximum number of joints used as interpolators) of the models used in each section as well as the purpose of the experiment.

- *stereo:* the base geometry was obtained from multi-view stereo.
- *shape-from-silhouette (SFS):* the base geometry was obtained using shape-from-silhouette.
- *laser-scanned (scanned):* the base geometry obtained from a laser scanner.

The coarse joint angle tracking is typically provided by one of the tracking methods in Chapter 5. Some of the experiments had different acquisitions (e.g., using a robot) and allowed for different methods to acquire the joint angles. The complete set of alternatives for joint angle tracking are the following:

- *none:* when the object has no skeleton motion, tracking is not necessary.
- *robotic:* joint angles were extracted from the robot.
- *silhouette (XOR):* XOR tracking from Chapter 5.
- *silhouette (ICP):* ICP tracking from Chapter 5.
- *manual:* point correspondences between the models were used to obtain joint angles (only used for one example with a sparse sampling in pose).

In practice, fine-scale correspondences can be hard to obtain when texture or silhouettes are not available. In this chapter, the following methods were used to obtain fine-scale correspondences:

- *none:* no fine-scale displacements were used.
- *silhouette:* silhouette-consistent displacements from Chapter 6 were used.
- *backprojected intra-view 2D optic flow:* optic flow within viewpoints was used to define the 3D motion constraints and backprojected onto the base mesh to get the fine-scale correspondence (see Appendix D for more details).
- *basis-constrained flow:* flow reconstructed using a basis-constrained flow (§ 6.3).
- *uv-parameterized flow:* flow reconstructed in the same manner as the basis-constrained flow (§ 6.3), although with no constraint on the motion.
- *diffused:* a sparse set of manual correspondences that were used to recover the tracking were diffused over the surface using the Laplacian deformation framework in Eq. 6.50.[1]

Rendering from the pose- and view-dependent model also involves some important considerations regarding the interpolation method. For geometry interpolation, the options are as follows:

- *RBF:* basic RBF interpolation using distance between quaternions (§ 3.3.1)
- *1D linear:* the influencing joint angles are projected onto the largest principle direction and interpolation is linear in 1D.
- *2D linear:* the influencing joint angles are projected onto the largest principle direction and interpolation is linear in 2D.

And for appearance interpolation:

- *combined RBF:* basic RBF interpolation combining pose and view (§ 3.3.2).
- *sequential:* sequential interpolation uses different interpolation for pose then view (§ 3.3.2). The pose methods are the same as above. View interpolation is one of two alternatives:
  - *circular*: cameras are assumed to lie in circular arrangement and this circular angle is used as the domain for linear interpolation.
  - *spherical*: cameras are assumed to lie in circular arrangement and the spherical 2D angles are used as the domain for linear interpolation.

For brevity, the decisions used in acquisition and modeling will be in an outlined box at the beginning of each section.

7.2 Repeatability and Acquisition Preliminaries

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
1	N/A	N/A	N/A	N/A	N/A	N/A

Before continuing with actual and simulated model acquisitions, it is useful to consider whether the appearance and geometry of deforming cloth can be represented in a pose dependent manner.

[1]The *diffused* method only used for one example with a sparse sampling in pose.

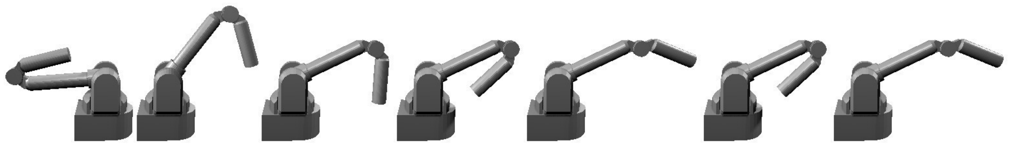


Figure 7.1: The WAM motion used in the repeatability experiments. The WAM moves from its rest pose into a downward pose where the elbow is moved several times in a similar pattern. The appearance of the cloth on the WAM is then compared across these similar poses after motion has occurred.

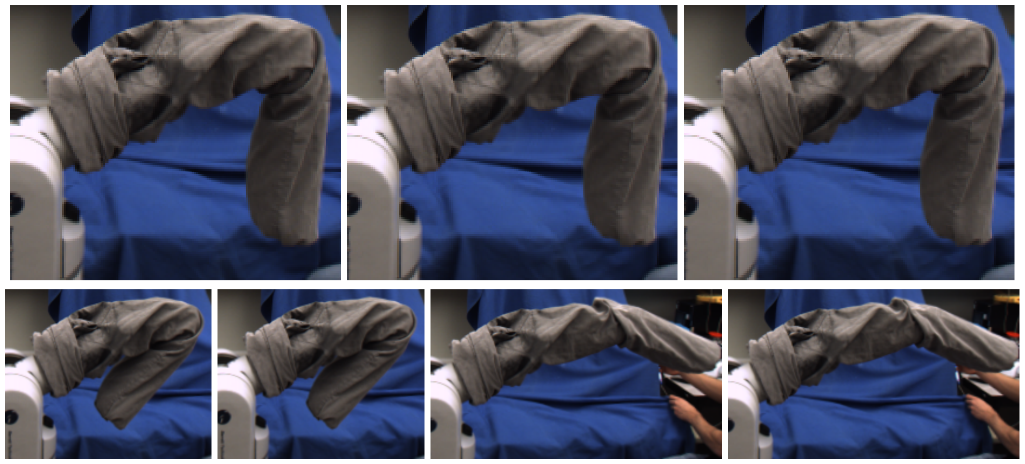


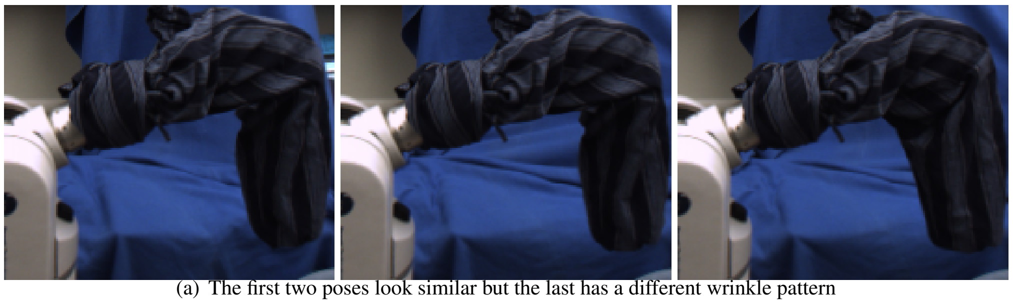
Figure 7.2: For the corduroy pant, although the robot has moved through different poses of the elbow, when it returns to the same pose the wrinkle pattern looks similar. In other words, the fabric appears to change as a function of pose, suggesting it is appropriate to model this fabric as pose dependent.

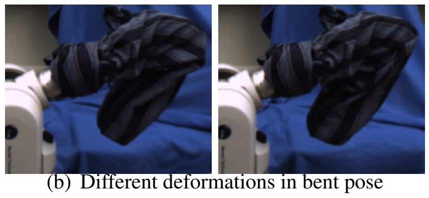
This is a central assumption in our pose- and view-dependent model. Clearly, a given cloth will not always wrinkle the same way after having went through large arbitrary motion. In this first experiment, the objective is simply to determine whether it is feasible to represent cloth as pose dependent when moving through a restricted set of arm articulations.

In order to qualitatively evaluate the repeatability of surface appearance as a function of pose, a robotic Whole Arm Manipulator (WAM) was covered with two types of clothing: a corduroy pant and a cloth pajama pant. As illustrated in Figure 7.1, the robotic arm was moved through various poses, including multiple observations of similar elbow poses. The input images at similar poses are qualitatively compared directly without any modeling.

As the pose changes, the wrinkles on the corduroy pant change and deform. However, when returning to the same pose, the cloth exhibits similar deformations (Figure 7.2). Although the wrinkles move slightly, the similarity of the appearance when viewed in the same poses indicates that the fabric can easily be modeled as pose dependent.

The softer and baggier cloth pajama is more likely to appear different than the thicker corduroy pant when positioned in the same pose. Figure 7.3 shows that the cloth pajama mostly exhibited similar appearance in the extended pose, but the wrinkle pattern was different when the elbow was


(a) The first two poses look similar but the last has a different wrinkle pattern


(b) Different deformations in bent pose

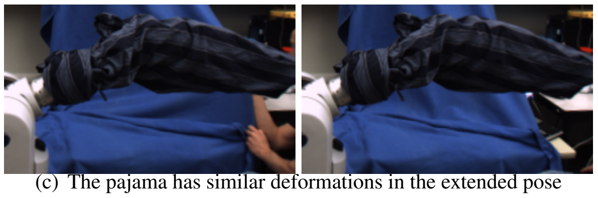

(c) The pajama has similar deformations in the extended pose

Figure 7.3: The baggier cloth pajama exhibits similar deformations in the extended pose, but the appearance is different for multiple observations in the bent and middle postures.

bent. In the neutral pose the cloth wrinkles and appearance were similar in two of the three frames. Even though the appearance is different in the same poses, the cloth pajama pant did exhibit similar deformations for some of the poses. For this reason it is still feasible to represent such a surface as pose-dependent.

However, the baggier cloth illustrates that care must be taken when designing the training sequence used to acquire a pose-dependent model. Specifically, as the same pose may give rise to different appearance or geometry, the training sequence should avoid observing the object in the same pose multiple times.

7.3 Synthetic Data

7.3.1 Simulated Deforming Bump: Multiple Views, One Interpolation Part

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
4	1	truth	none	truth/uv-param.	RBF	combined RBF

In the simplest case, the geometry is affected by only one parameter. In this example, the geometry is like a cushion being pushed in the center. Specifically, the deformation is simulated as a simple quadratic bump deformation (see Figure 7.4. The deformation was rendered using four views, each having 26 frames, and there was no kinematic component. This configuration is similar to having a single kinematic link to which all the geometry is attached. Two top down views not used to obtain the model were used for quantitative analysis. The purpose of this experiment is to test the effect of geometrically modeling deformations as opposed to a completely appearance-based approach.

For the uniform textured sequence, the ground truth displacements were used and the model

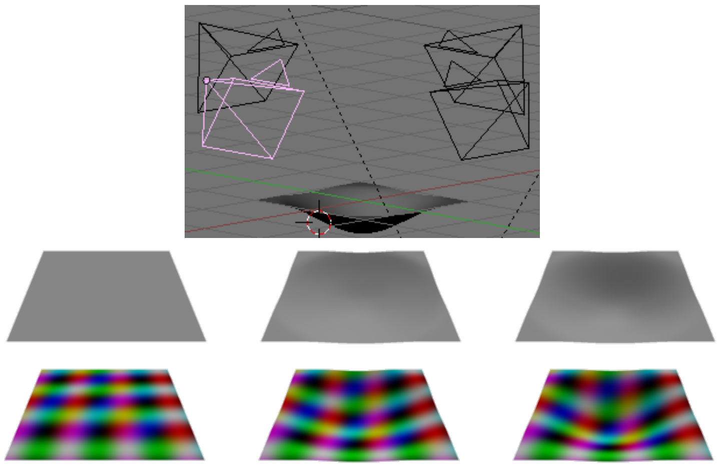


Figure 7.4: Left: the bump scene setup (and the 4 cameras). Right: 3 images from a single view of the uniform and textured sequences.

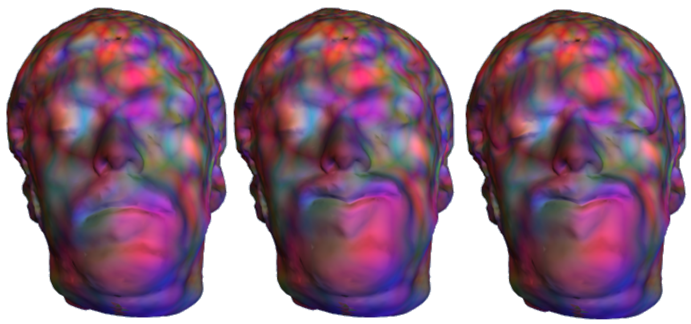
was built with 15 basis images for the textures and 1 for the geometry. A combined pose and view interpolation scheme was used, where a weight of $\alpha_{pose} = 100$ was given for the appearance component in Eq. 3.21. For comparison, another model was built with zero displacements and 16 appearance basis images. The input sequences were compared to images rendered by the model as the deformation was played back. As expected, in both cases, the image residuals over the 4 input views was low (on the order of 0.5%). However, the benefits of the displacements came in the comparison of the novel views, where the intensity error was roughly 0.8% versus the 1.6% for the non-displaced model. Even for this uniform textured model the use of displacements in the model aids rendering from novel views. Furthermore, the displaced model is more accurately able to reconstruct the silhouettes.

For the textured sequence, the flow was reconstructed using the uv-parameterized variational scene-flow algorithm (Section 6.3). The image residual on the two unused views was 0.8% for ground truth displacements and 13% for no displacements. The recovered displacements had a residual of 1% in the unused views. This example illustrates that when the surface is highly textured it is possible to reconstruct the deformations from images such that they rival the ground truth model. Notice the poor performance of ignoring displacements is exacerbated when there is texture. Specifically, the textured sequence has a much larger residual in novel viewpoints when displacements are ignored.

7.3.2 Synthetic Facial Animation from Multiple Views

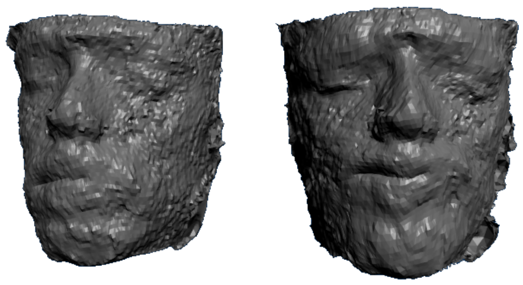
C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
2	3	stereo	none	uv-param.	RBF	combined RBF

In this example, a human face is modeled with 3 deformation parts, one for each eye and one

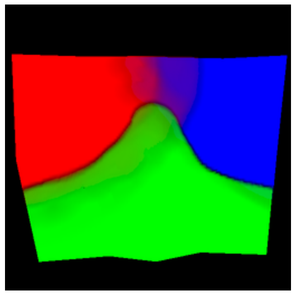


(a) Three frames of the input sequence from a single view

(b) A texture



(c) Reconstructed geometry for the first and last frames



(d) Mesh parts in uv-space

Figure 7.5: (a) Selected input frames, (b) a texture in uv-space, (c) reconstructed geometries, and (d) the hand selected geometric parts for the synthetic face example. The input facial animation has motion in the mouth and eyes.

for the mouth region. Two views of the face, each with 20 frames, were used as image input to the reconstruction (Figure 7.5). Another in-between view and top frontal view were used for validation in the quantitative experiments. Again, the kinematic skeleton was ignored (although it could compensate for some of the deformations). The purpose of this example is to test the blending of deformations and appearance between the distinct geometric parts.

The training sequence consists of a smile followed by each of the eyes blinking. The deformations were again recovered using the uv-parameterized flow. A model was built both with and without deformation modeling, and 8 appearance basis vectors and 8 displacement basis vectors were kept. The average RMS intensity error for the displaced model in the input views was 2.2% and 2.1% for displaced and non-displaced models. Both models gave similar reprojection error on the extra middle view (2.3% and 3.1% respectively); however, in the frontal top down view (which is far different from the input views) the displaced model has a similar low residual (2.9%) whereas the non-displaced model residual is twice as high at 6.0%. The low residual from the two original side views suggests that the reproduced deformations appear correctly. The larger error in the frontal top down view demonstrates that modeling both displacements and appearance is most effective, especially in modeling view variation farther from the input views.

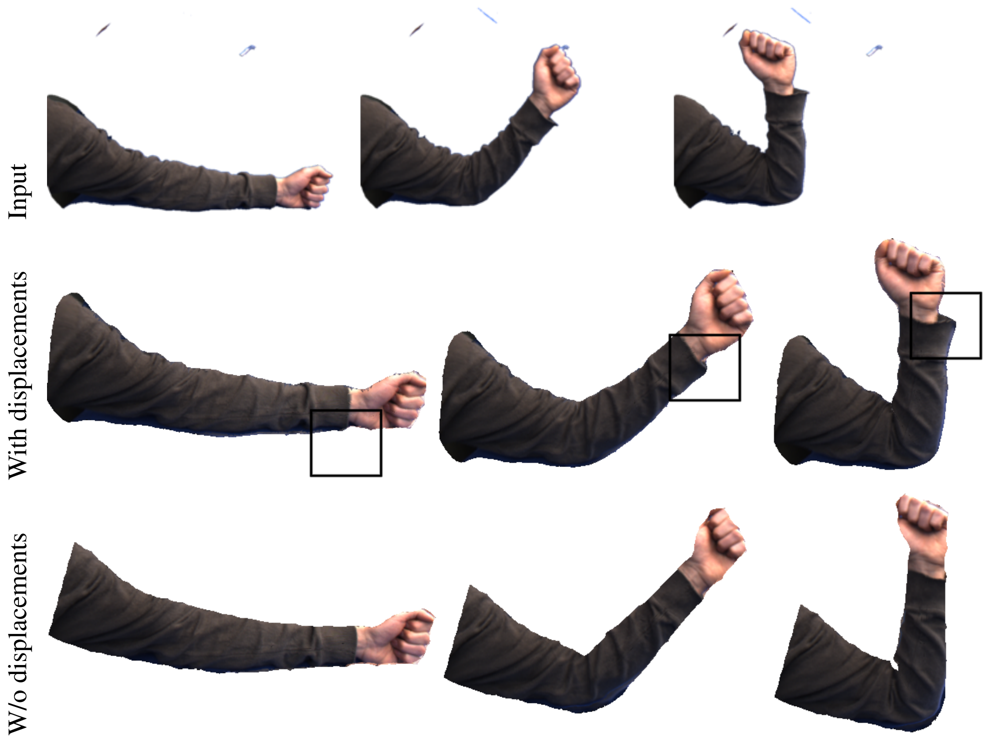


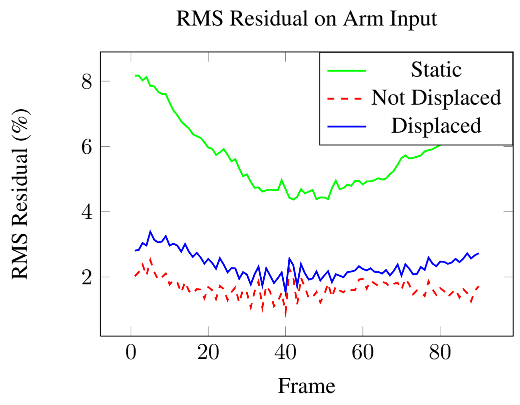
Figure 7.6: Top: input images of the arm deformation. Middle and bottom: renderings of the reconstructed model with displacements and without displacements (e.g., the latter is a skinned model with appearance modeling). Both respect the appearance, but silhouettes are only accurate in the former model.

7.4 Modeling Elbow Motion of a Human Arm from a Single View

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
1	2	SFS/manual	XOR	silhouette	1D linear	1D linear/1D circular

This example tests the kinematic structure and displacements in a simple setting of a single view and single deformation parameter. The input is 90 images of a bending arm exhibiting cloth wrinkles/shifting (Fig. 7.6).

The deformations were recovered using the basis constrained flow with silhouette constraints (§6.3). We found that the coarse joint angle tracking was adversely affected by the accuracy of the initial model. Such errors in the joint tracking cause misregistration of the textures, which will be reproduced by the appearance basis during rendering, causing a shifting of the texture on the object. To account for this, after recovering smooth temporal silhouette displacements, we recomputed the gross tracking using the more detailed deforming model and then again computed the displacements. The model was built with 16 basis vectors for appearance and 3 basis vectors for geometry. Every third image from the input sequence was used (i.e., 30 input images), and the other 60 images were



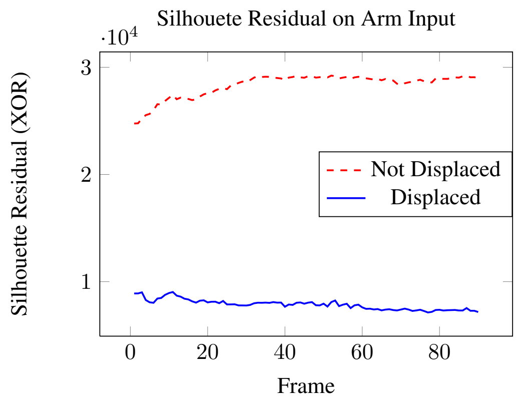


Figure 7.7: RMS intensity and silhouette residuals for the arm sequence. The RMS intensity residual is slightly lower for the non-displaced model, but the silhouette error is much lower for the displaced model. As visual fidelity is a function of both appearance (RMS) and silhouette quality, the displaced model is the most realistic.

used for testing. Even on this simple example where the deformation is in the plane and the model is essentially 2D, the usage of silhouettes in the displacement recovery improves the realism of the model (Fig. 7.6). Figure 7.7 illustrates the image residuals of the two models is very close (every 3rd image in the sequence gives a lower residual as it was used as input). However, the displaced model better agrees with the silhouettes. Also note that displacements alone with an average texture cannot achieve a low image residual, implying that both geometry and appearance components are required.

7.5 Varying View and Pose Sampling with a Single Pose Parameter

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
Many	2	SFS	truth	silhouette	1D linear	1D linear & 1D circular

In this experiment, the effects of sampling rate, in terms of input view and input pose density, are studied on a one degree of freedom kinematic object using different cloths. Typically, a capture scenario uses a limited number of fixed cameras, and, through the training sequence, obtains a dense temporal sampling of pose from this sparse set of views. Instead, in this example, a large number of views are acquired of various poses through the use of a robotic arm and one camera.

The appearance and deformation of three types of clothing (a *Plaid* shirt, *Corduroy* pant, and a cloth *Pajama* (*PJ*)) were obtained by placing the clothing on the robotic manipulator. The manipulator was positioned in 14 poses of the elbow joint (every 10 degrees), for which 33 views at 5 degree intervals (covering a total of 160 degrees) of motion were obtained by rotating the robot around the shoulder joint (Figure 7.8).

The base geometry for the pose- and view-dependent models were obtained from a shape-from-

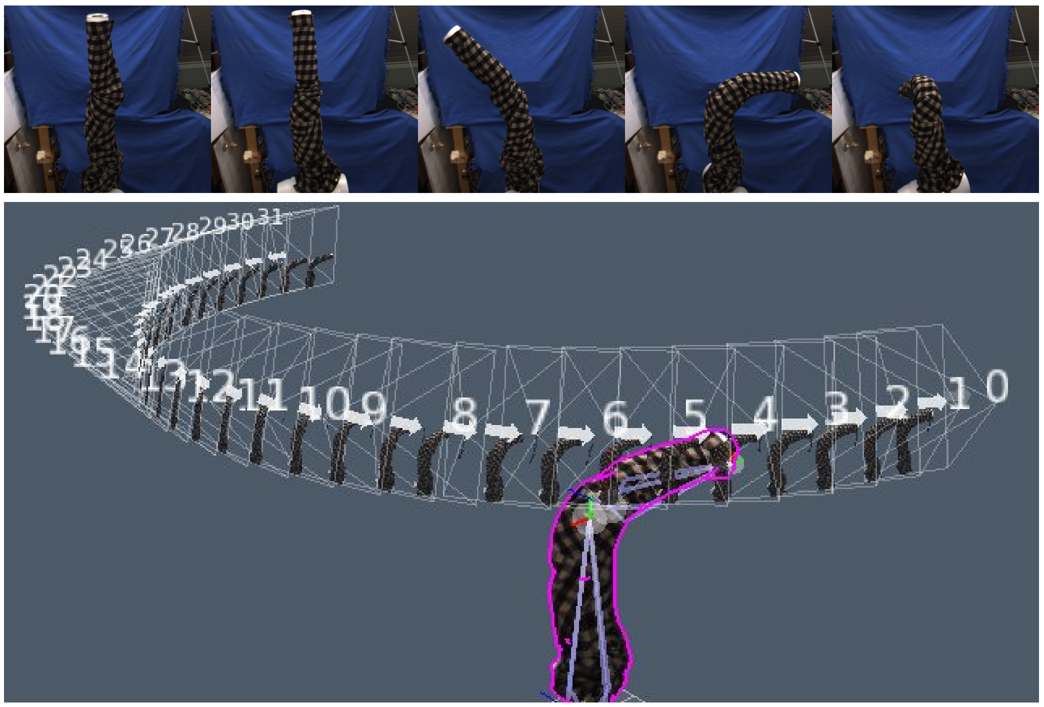


Figure 7.8: Input images of the *Plaid* cloth taken by a single camera (top). The arm was captured in 14 elbow poses differing by 10 degrees. For each pose, the base was rotated over 180 degrees giving 33 views for each pose (bottom)

silhouette model with the arm in the straightened pose. The kinematic structure and tracked joint angles were obtained from the ground truth robot model, which was calibrated relative to the input sequence using feature points. Geometric deformations were obtained using only the silhouette displacements from Chapter 6.

From these inputs and the 572 input images, a combined pose- and view-dependent model was built for each cloth type using 48 basis textures and 6 geometric modes. To study the effect of sparser sampling in both pose and view, four other models were created: a sparsely sampled view model that uses every second input view (SparseView(2)), a sparsely sampled view model that uses every fourth input view (SparseView(4)), a sparsely sampled pose model that uses every second pose (SparsePose(2)), and a sparsely sampled pose and view model that uses every second view and every second pose (SparseView(2)Pose(2)).

Figure 7.9 shows the RMS image error computed over all of the images in the sequence. The sparsely sampled viewpoint accurately represents all of the views (e.g., residual curve is close to the full model), suggesting that for these models a sampling less than every 10 degrees (i.e., SparseView(2)) is sufficient to produce accurate renderings. In many cases, an even sparser sampling of every 20 degrees (i.e., SparseView(4)) also gives reasonable results. In practice, the required density of input views will depend on the non-Lambertian reflectance properties of the object as well as the accuracy of the geometric reconstruction. On these representative clothing examples, the residual is

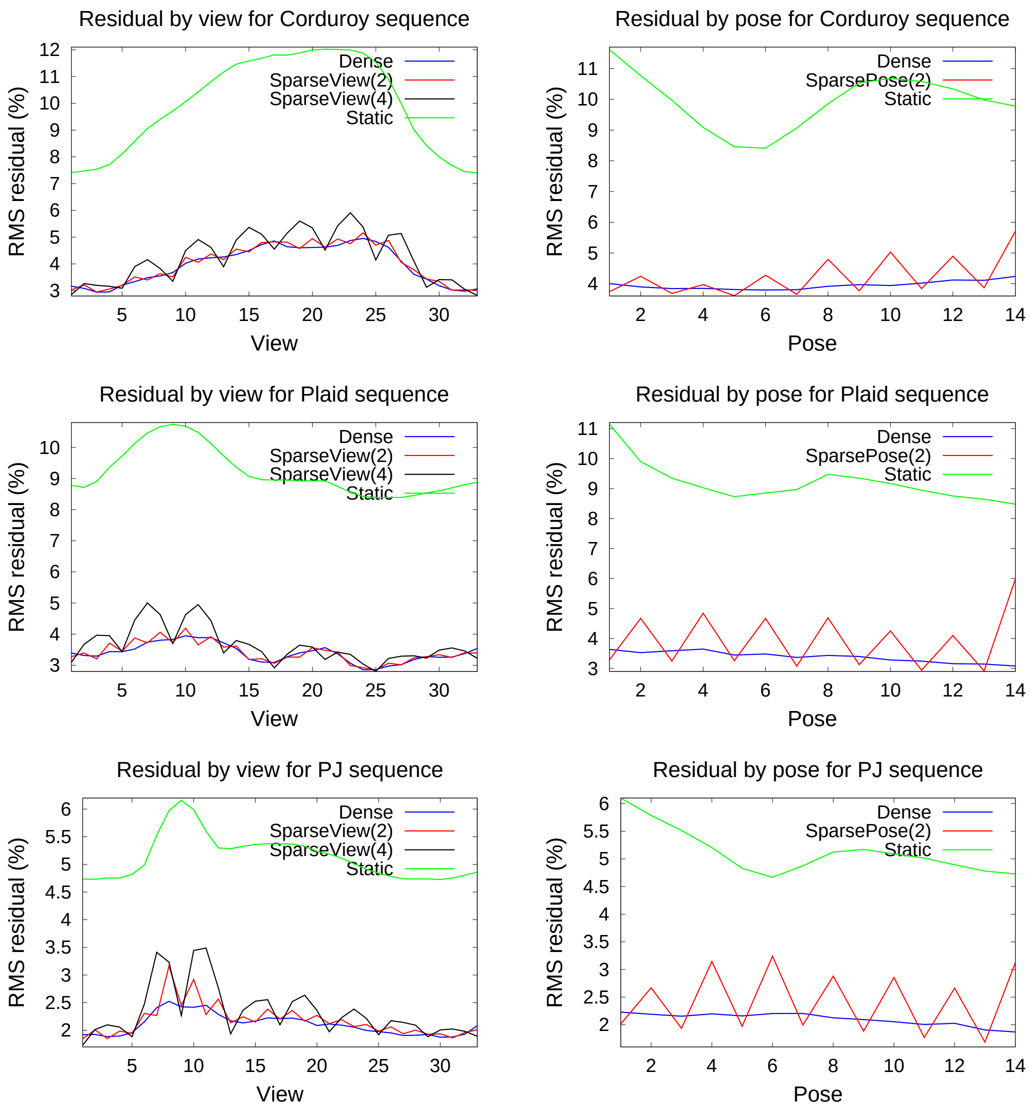


Figure 7.9: RMS image residuals for the 3 sequences captured from the robot arm. The left column plots the residuals averaged over poses for models created with different samplings of input views. The coarser sampling gives rise to spikes in the RMS residual for unseen views. The right column plots residuals for each pose (averaged over views) for a dense and sparser sampling of poses. The sparse pose sampling gives rise to large spikes in the RMS residual.

due to both geometric inaccuracies (as models were only created using silhouettes), and illumination changes (as the object moves relative to the light source).

When considering pose sample density, reducing the sampling to 20 degrees (in SparsePose(2)) causes a detrimental effect on the RMS image residual that is similar in magnitude to the reduced view sampling of 20 degrees. This is apparent for all clothing types as the residual spikes for every second pose (i.e., the poses not used to create the model). Although the sparser sampling in pose causes large spikes in the RMS residual, for some of the objects this sampling is still sufficient to create decent interpolated renderings at novel poses. Figure 7.10 illustrates rendered results for a single viewpoint (view 2) using the densely sampled model as well as the sparse view and time model. The renderings are illustrated for views and poses that were not used to build the SparseView(2)Pose(2) model. In this case, the effects of using a 10 degree sampling in viewpoint is negligible. On the other hand, the shortcomings of sparsely sampled poses is more apparent on the *PJ* sequence: in pose 4 and 6 there is significant ghosting around the elbow due too much cloth shifting (Figure 7.11). However, in the later poses (e.g., poses 10 and 14) there is less motion, and the sparsely sampled pose model allows for reasonable renderings.

The sparse sampling of poses seems to have less of an effect on the *Corduroy* sequence. This is likely due to the *Corduroy* cloth being stiffer and not having the sharp texture discontinuity that the *PJ* cloth has.

This experiment has demonstrated that sparse sampling of pose and view have similar effects on rendering residuals. Ideally the pose and view freedoms could both be sampled as densely as desired (e.g., in this robotic case we could easily sample both freedoms more densely). However, in practice one is often limited by the physical number of cameras, which typically affects the number of view samples. For some objects, a denser sampling of view may be obtained through the use of a turntable-based approach (see § 7.9.1). As a turntable-based approach is not always feasible, the above experiment suggests that a sparser sampling of view is sufficient when the recovered geometry is accurate and the appearance is Lambertian.

The ghosting effects of a sparser sampling in pose will diminish if the surface has limited deformations (e.g., is well explained by bones) or if the surface deformations can be accurately recovered through surface texture information. In practice, it is often easier to obtain a dense sampling of pose than of viewpoints (as the subject motion is captured at the framerate of the camera). In such conditions (with few viewpoints), improving the geometric accuracy is important to help improve the rendering using only a sparse sampling in viewpoints.

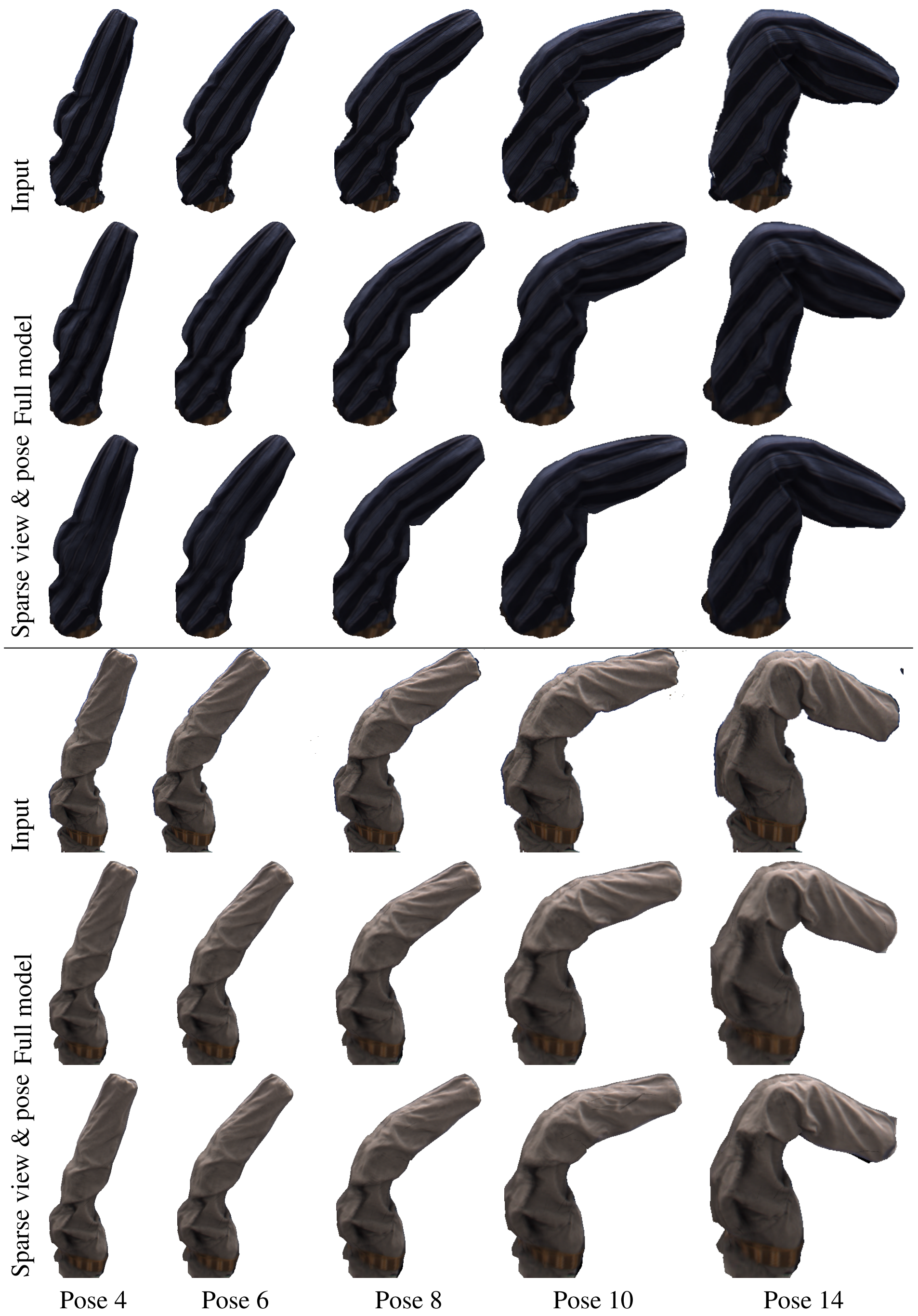


Figure 7.10: Input images from viewpoint 2, reconstructed results using all images, and reconstruction using sparse view and sparse pose for the *PJ* and *Corduroy* models. The model reconstructed using all images accurately portrays the changes in silhouette and texture (as these images are used as input). This view and the given poses are not used as input for the sparse model. It is clear by the ghosting of the elbow of the *PJ* (pose 4 and pose 6) that the pose sampling is too sparse (see Fig. 7.11). The sparse sampling gives reasonable renderings for the later poses (and for most of the *Corduroy* sequence).

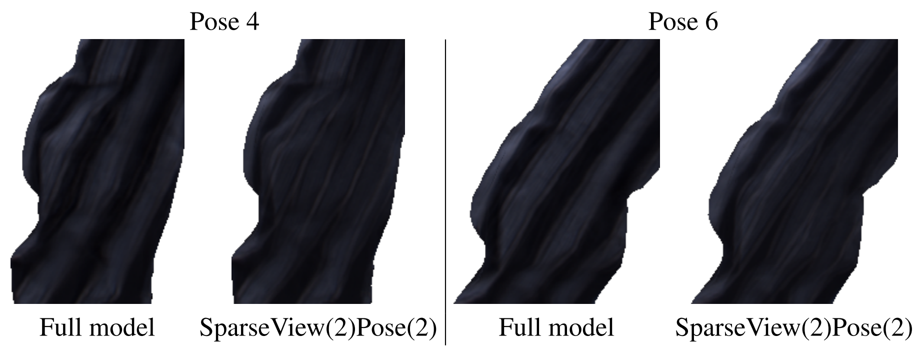


Figure 7.11: Detail view of poses 4 and 6 for the *PJ* models illustrate that the pose sampling is too sparse for the SparseView(2)Pose(2) to interpolate the texture. The resulting textures have ghosting.

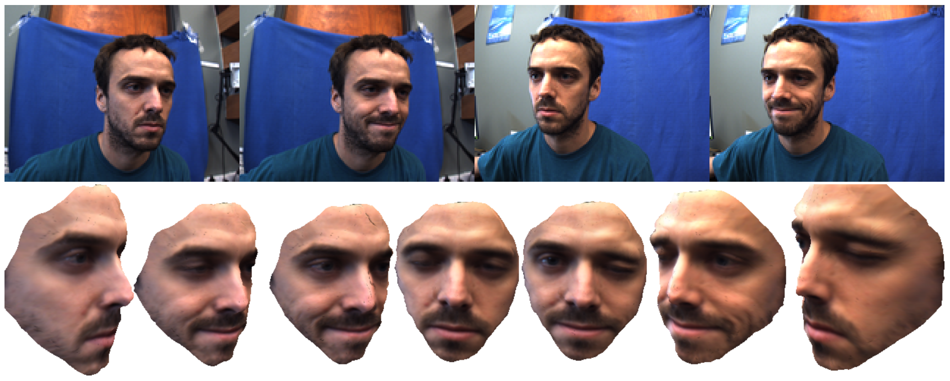


Figure 7.12: Top: two frames from two of the input sequences. Bottom: renderings from the real face model from several views with a variety of different facial and eye expressions.

7.6 Modeling Facial Expressions From Several Views using Multiple Parts

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
4	3	stereo	none	backprojected flow	RBF	combined RBF

This example is analogous to the synthetic face (§ 7.3.2), except that three cameras observed a face performing similar motion of smiling and blinking. There were 117 frames per sequence, and the face structure was acquired using a multi-view stereo method (based on [186]), and the surface displacements were obtained through the backprojected intra-view 2D optic flow.

The constructed model used 16 appearance vectors and 8 shape vectors; every third input image was used to build the model. The state of the eyes and mouth were manually annotated, and subsets of the input sequence were used to build the interpolation spaces for each part. To avoid duplicate poses for the eyes, about 20 frames of the eyelid closing were used. And for the mouth, only the 60 frames having a transition from neutral to smiling were used. The average RMS intensity error over the entire sequence and all of the cameras is 3.1%.

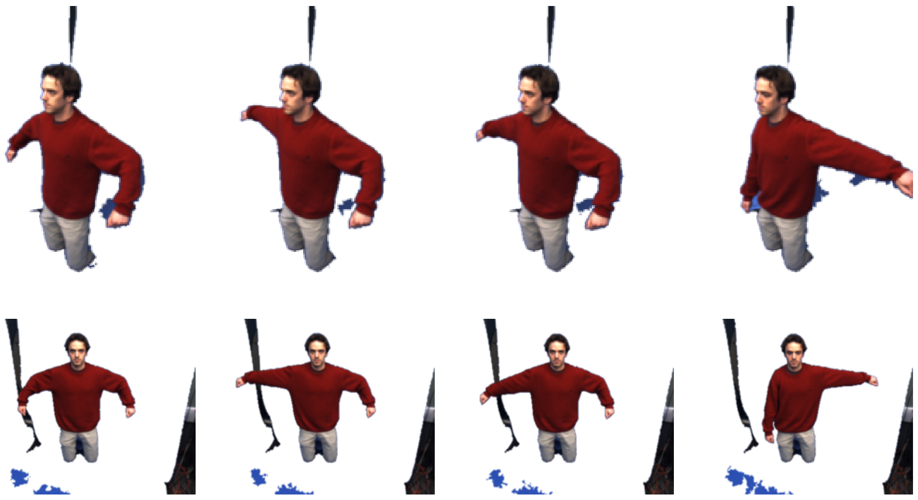


Figure 7.13: Input images from 2 of the 4 views of the upper body sequence.

The three input views were mostly frontal, but the object easily plays back motion of smiling and blinking from a variety of views (Fig. 7.12). Even though the input geometry is approximate, IBR allows for a realistic rendering Unfortunately, the cameras were not color calibrated; hence switching viewpoints sometimes causes color artifacts.

7.7 Upper Body: Several Views, Kinematic Structure, Multiple Parts

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
4	4	SFS	XOR	silhouette	RBF	combined RBF

This data set simultaneously tests all the components of the model: multiple parts, different influencing parameters, and several viewpoints. The input is four views with 320 frames of the upper body of a human (Fig. 7.13). The approximate structure was estimated using shape-from-silhouette in an initial view (the viewpoints were widely separated making improvements by stereo difficult). The skeleton was manually aligned and the skinning weights were obtained automatically [25]. The four parts were extracted from the skinning weights (Fig. 7.14).

The displacements were extracted using the silhouettes only; this helps especially when the arms are in the down position. Figure 7.15 compares renderings under novel poses with an average texture rendering. The single texture averages all of the wrinkles, whereas the wrinkles move with the pose in our renderings.

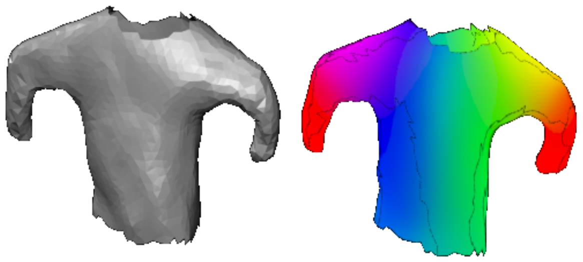


Figure 7.14: Recovered input geometry and corresponding parts for the upper body. For illustration, red is used for both the forearms.

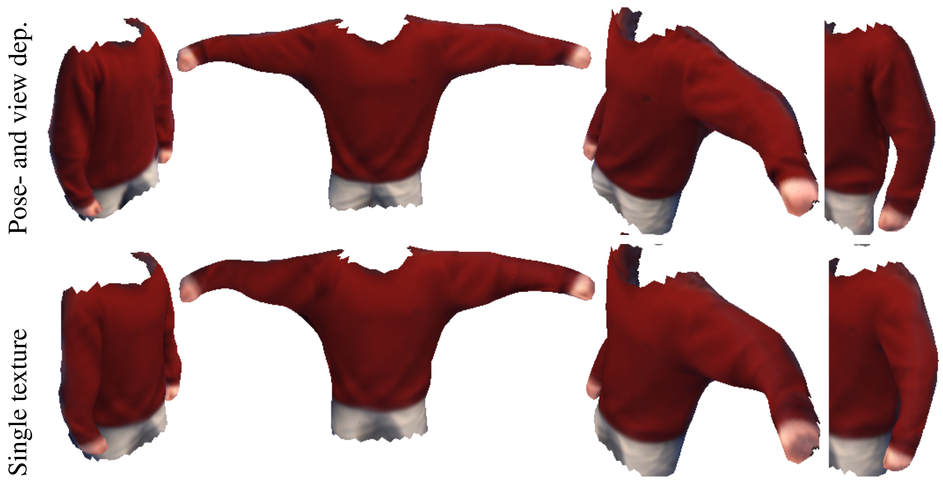


Figure 7.15: Pose- and view-dependent rendered results (top) and an average texture (bottom). The average texture blurs out the wrinkles, whereas with our model they vary appropriately with pose (and view).

7.8 Lower Body

In this section, two acquisitions of the lower body are obtained. The acquisition of each model is targeted to a different reanimation purpose: climbing up stairs or walking.

7.8.1 Stairs

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
6	4	SFS/manual	ICP	silhouette/none	1D linear	1D linear & circular

For the first example of modeling legs, cloth motion in the knees is captured. Figure 7.16 illustrates the camera configuration. The input motion consisted of moving the knees up and down in succession. An initial SFS geometry was created, and due to the few viewpoints it was manually cleaned. A kinematic skeleton of the legs was manually aligned to this initial geometry.

To avoid duplicate poses in the interpolation space, only the motion of the knees in the upward direction were used. The pose-dependent appearance of the shins are influenced only by the

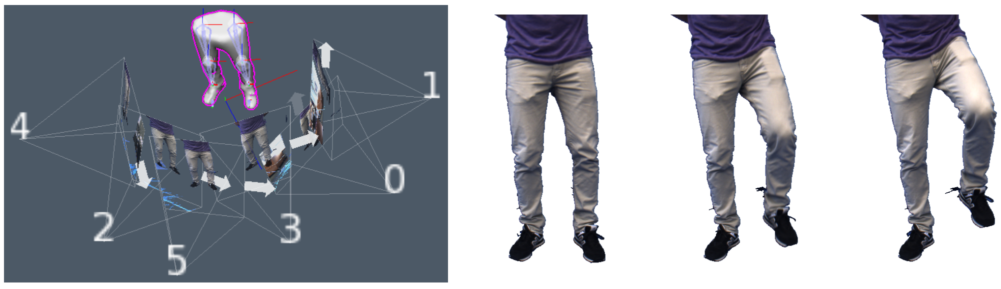


Figure 7.16: The camera configuration (left), and the input sequence of the right leg the khaki pants.

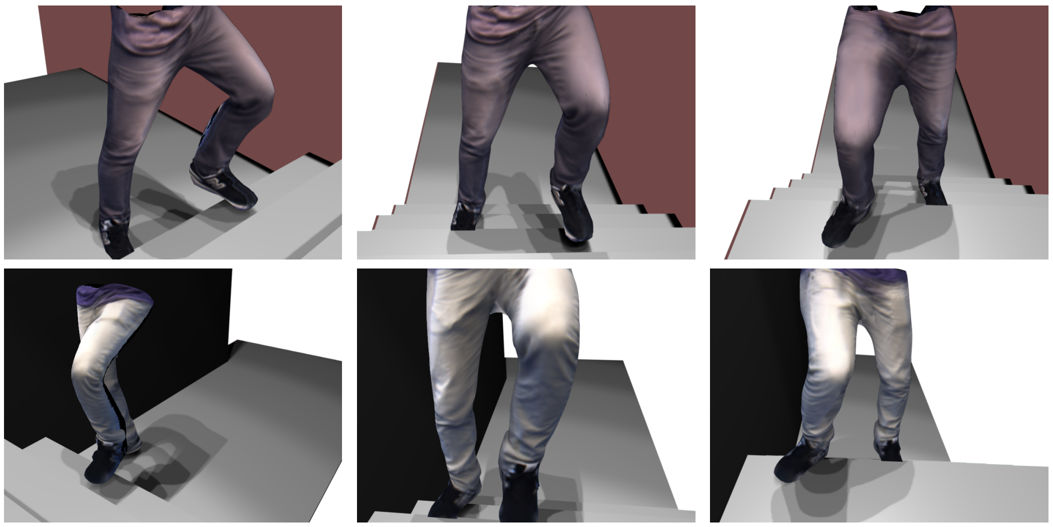


Figure 7.17: Novel stair walking animation of the pose- and view-dependent models acquired using two different jeans types.

.

respective knee angle, whereas the thighs are influenced by both the hip and knee angles.

The input motions for each mesh part (e.g., thighs and shins) are modeled with 1D linear interpolation (e.g., by projecting the joint angles into the principle direction as in Section 3.3). The view component was modeled with a circular camera projection using a 1D linear interpolant.

The recovered models were then rendered under novel animation of walking up stairs (Figure 7.17). For each type of trousers (gray and khaki), the novel animation accurately portrays the change in the illumination on the upper thigh and the wrinkles around the knee.

7.8.2 Walk Cycle

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
6	4	SFS/manual	ICP	none	1D linear	2D linear & circular

In this second lower body example, re-animation on a significantly different animation sequence is demonstrated. The goal is to acquire a model capable of being animated through a walk cycle.

The same camera setup as Section 7.8.1 was used to acquire input. As the multi-camera rig is facing downwards, it is hard to acquire the texture of the trailing leg through a walk cycle. For this reason, a motion sequence designed to go through similar poses as a walk cycle while going through a more forward facing motion was used. The subject lifted each leg in sequence, by moving the knee up, straightening the knee, and then lowering the leg (see Fig. 7.18).

The input poses are different than what would be in a typical walk cycle. Synthesizing texture for the novel animation means that the texture will only vary near poses in the input sequence, such as when the knee is upwards. However, the difference in poses from the synthetic animation to the input lies mostly in a constant offset of the hip angle. Dynamic pose-dependent effects over the entire novel sequence can be achieved through the use of a slight edit.

The edit is a simple transformation of the synthetic animation poses to lie nearer to the input animation. For this input sequence, the pose-dependent interpolation and editing can naturally happen independently for each part (e.g, thigh, shin) in a 2D space. Each of these parts are synthesized independently and are modeled as a function of both the pose of the knee and the hip angle. The orientation of these two joints over the input sequence can accurately be represented by a linear projection into a 2D space. The projection is obtained by PCA over the concatenated axis-angle representation of the hip and knee joints.

Figure 7.19 illustrates the projection of the pose of a single leg into 2D for both the input and synthesizing animation. An affine transformation (consisting mostly of translation) can be used to bring the synthetic animation into closer alignment with the input sequence for the purpose of interpolation. As the transformation brings the synthetic animation closer to the input sequence, it better allows the transfer of the pose-dependent texture effects from the input sequence.

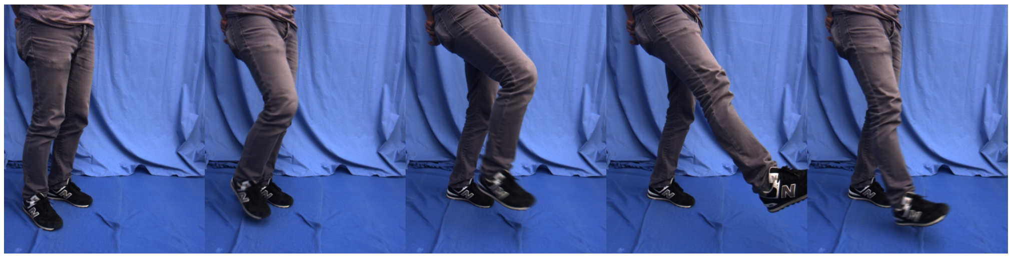


Figure 7.18: The input sequence from one of the views for the walk cycle consisted of movements of the hip and knee. The movement of the knee up, straightened knee, and leg down, causes the cloth to wrinkle in the knee and crotch. This wrinkle pattern is similar to what would be observed in a walk cycle.

Figure 7.20 illustrates the rendered reconstructed model for three different walk cycles. The walk cycles were generated using a physical simulation of a passive walker that parameterizes the gait using a stride length and desired speed [44]. In each of the three walk cycles, the wrinkles on the knees appear when the leg is bent, and re-appear when the leg is straightened (as in the input). The illumination and wrinkles of the upper thigh (as it is brought upwards) are also represented correctly.

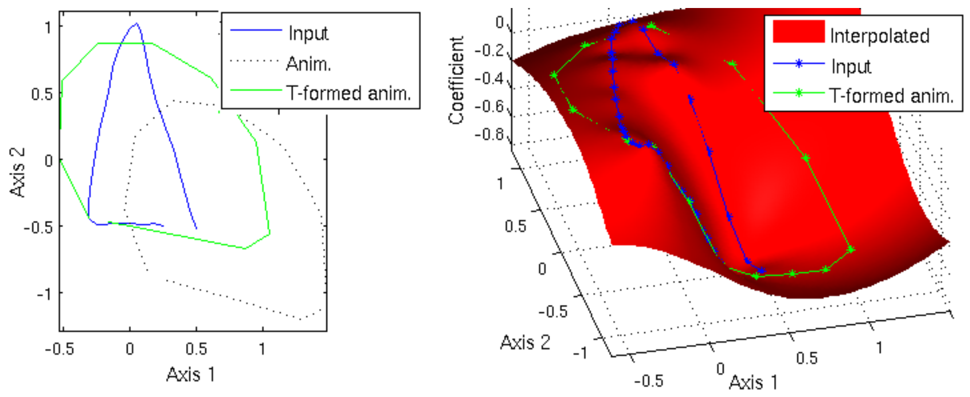


Figure 7.19: The re-animation sequence is significantly different from the captured walk sequence, meaning that re-animated poses lie far from input poses in the interpolation space. A transformation in the interpolation space brings the re-animation sequence into closer alignment. After transformation, the re-animated coefficients lie closer to input points, making the interpolation more plausible

.

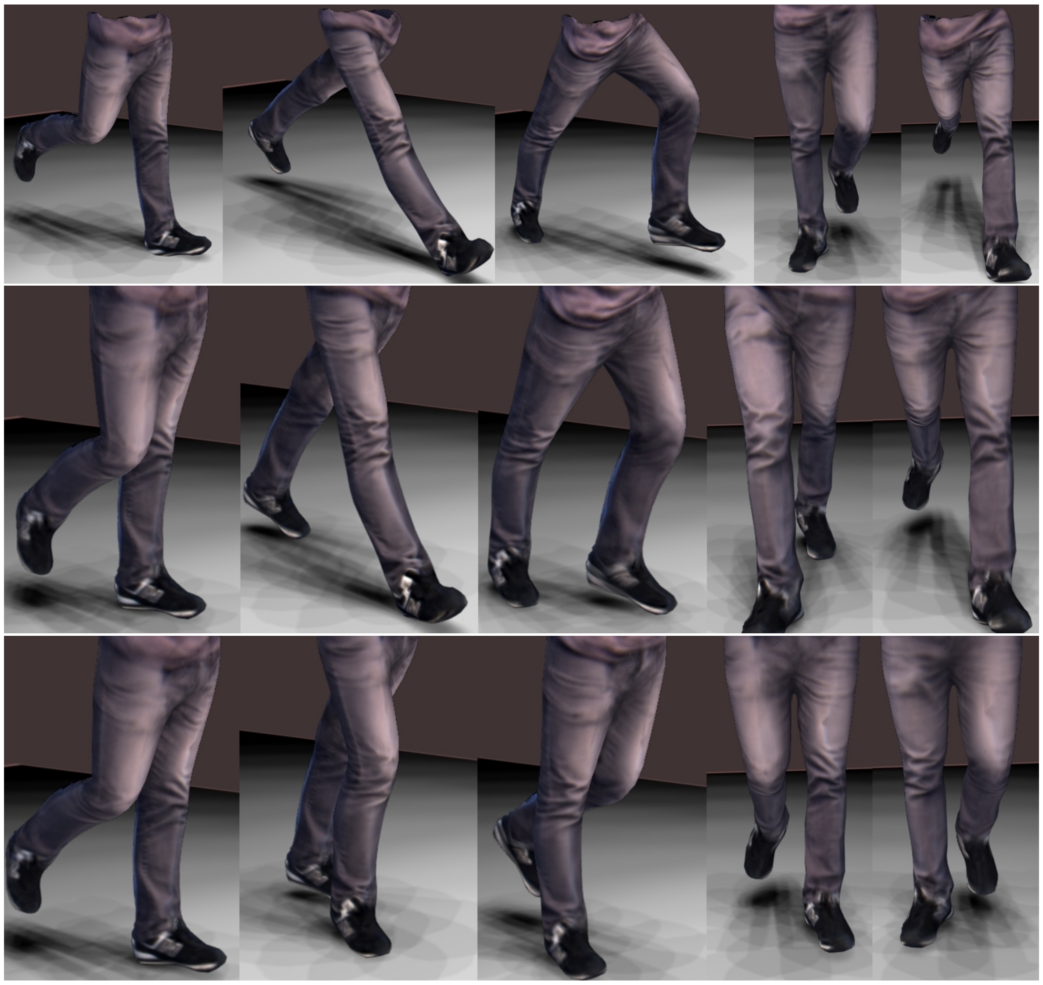


Figure 7.20: Animated walk cycle results for three different walk cycles. Wrinkles on the knees disappear when the knees are bent, and the shading on the upper thigh changes as the hip is moved.

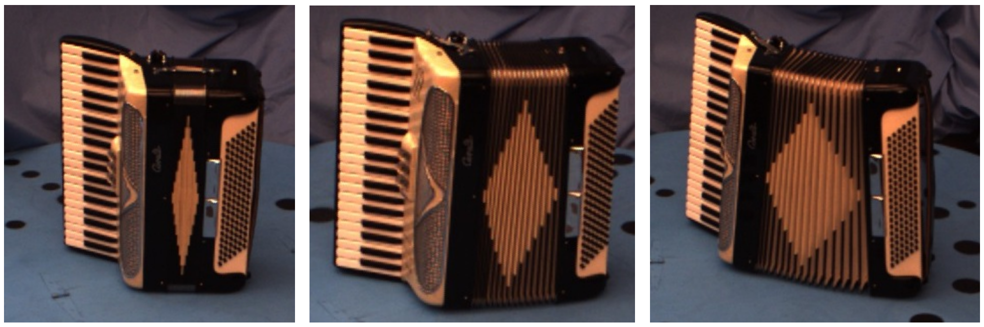


Figure 7.21: The accordion was captured in 3 different states using a single camera and a turntable.

7.9 Applications

7.9.1 Accordion

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
many	3	manual	manual	diffused	1D linear	1D linear & circular

In this example, we illustrate the application of the pose- and view-dependent appearance model to an accordion.The bellows of the accordion deform non-rigidly when the two rigid end-pieces are moved. For this particular accordion, the outside surface of the bellows are also painted with a black and white diamond pattern, while the inside surfaces of the bellows are white. When closed the diamond is most apparent, and when open the inside white surfaces become exposed (Figure 7.21). Furthermore, when viewed obliquely, the inside (white) of the bellows become less visible. These pose- and view-dependent appearance changes could be modeled with a dense geometry with accurate surface deformations as a function of the pose. However, the appearance changes can also be modeled through a more elaborate appearance model.

To acquire the appearance models, the accordion was placed on a turntable and three distinct acquisitions were obtained, one when the accordion was closed, slightly open, and wide open (Figure 7.21). A simple hand modeled proxy with two bones was aligned with the images in the slightly open state, and the skinning weights were initialized to vary linearly across the bellows. The kinematic pose of the bones was then aligned to the other two acquisitions through 30 manual correspondences selected through the input images. The appearance model over the non-rigid portion of the bellows was then created using all three states, while the rigid portions were modeled only from the slightly open state.

Three other view-dependent models were created (one for each input state). Figure 7.22 illustrates that the view- and pose-dependent model is necessary. If a model is created solely from the closed state, there is no way for the deformed model to exhibit the exposed white portions of the bellows when the bellows are open. Similarly, if a view-dependent model is created only when the

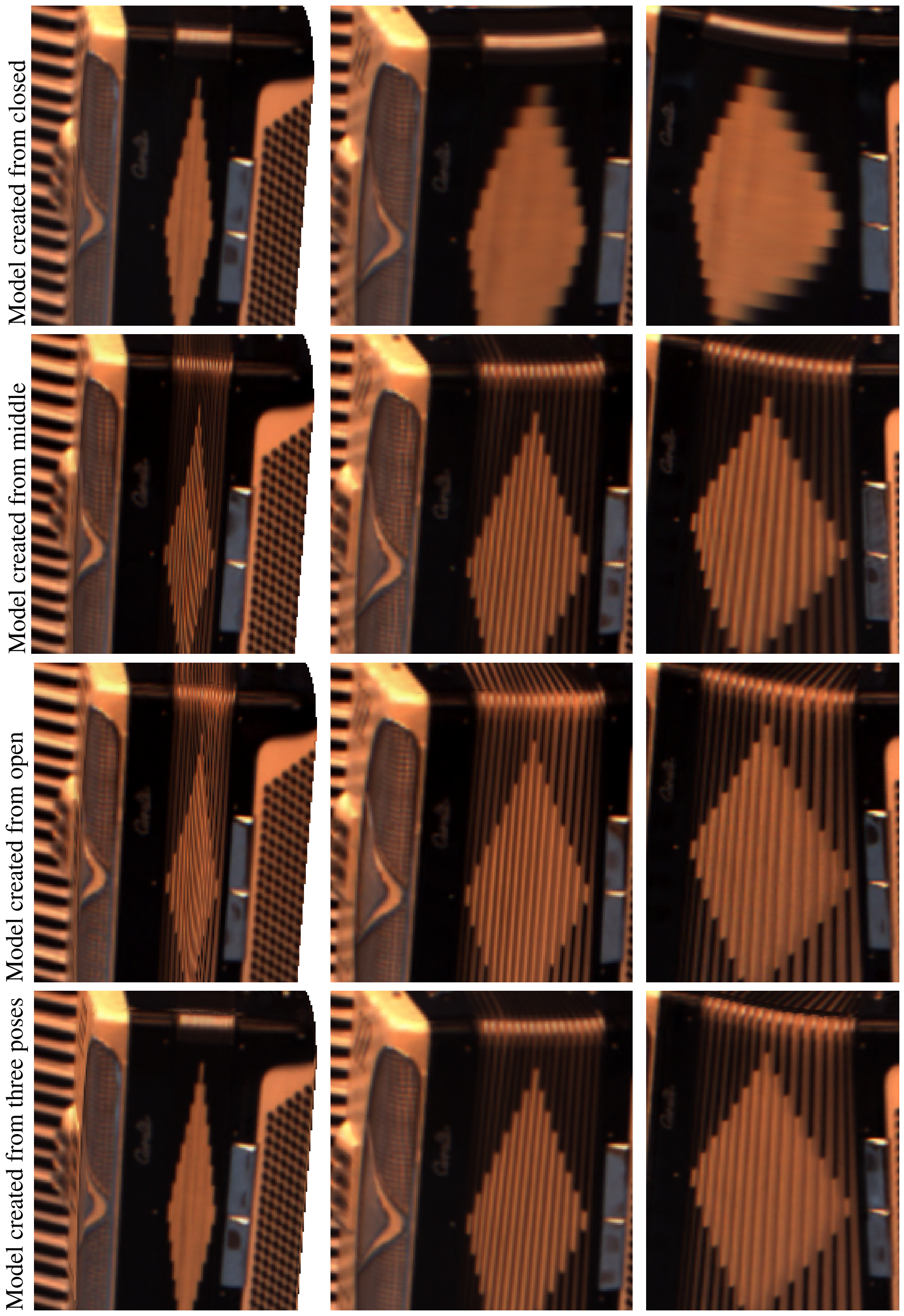

Model created from closed
Model created from middle
Model created from open
Model created from three poses

Figure 7.22: A view-dependent model using a single acquisition of the accordion (either in closed, slightly open, or open) is not able to represent the appearance of the bellows through the deformation. Acquisition of the model in the closed state accurately models the closed bellows, but the texture lacks the detail of the open bellows. Acquisition of the open bellows cannot accurately represent the appearance when the bellows are closed. A pose- and view-dependent model (bottom) created with images of the 3 below states can accurately represent the appearance of the bellows.

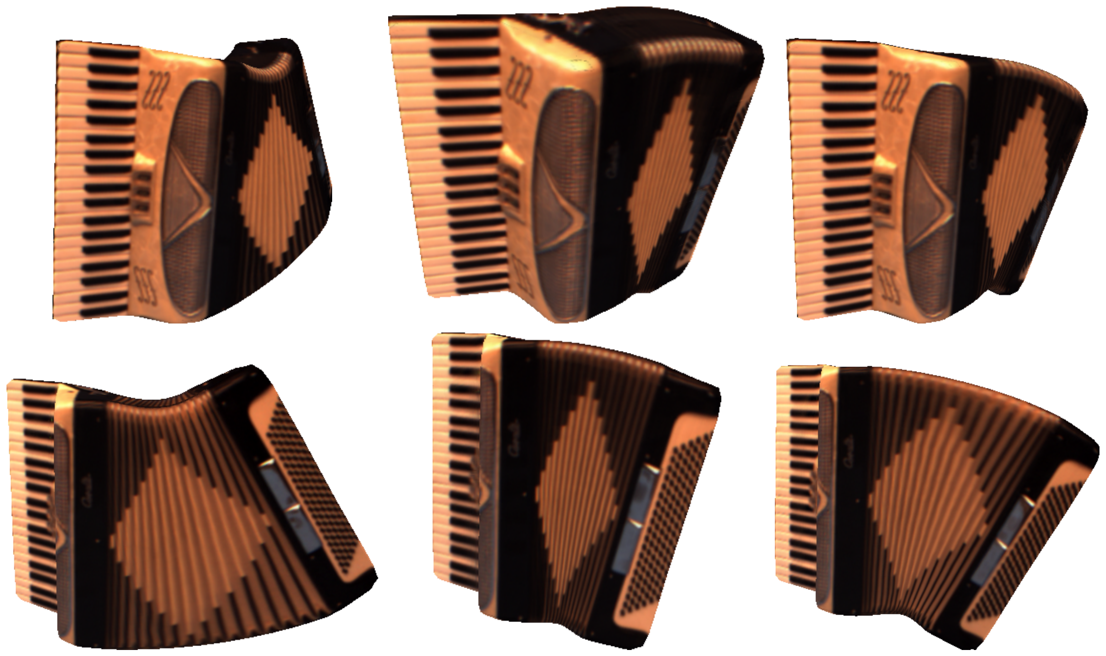


Figure 7.23: The pose- and view-dependent model can be rendered under novel pose and viewpoints. Notice that the bellows have white stripes when viewed head on and appear mostly black when viewed from the front.

bellows are in the open state, the model is unable to exhibit the high contrast diamond in the closed state. Instead, when the bellows of these models are closed, the white stripes are still visible. In contrast, the pose- and view-dependent model can accurately portray the bellows in the closed, slightly open, and open states. Here the texture rendering was achieved using a bi-linear interpolation over both pose and viewpoint.

The resulting pose- and view-dependent model can then be rendered from novel pose and viewpoints. Figure 7.23 illustrates the bellows in other similar deformations viewed from different camera positions. Notice that the renderings exhibit the rich below detail across viewpoint and pose.

7.9.2 Free-viewpoint Compression

C	Num. parts	Base model	Tracking	Fine-scale	Pose interp.	Appear. interp.
8	1	laser	truth	truth	1D linear	1D linear & circular

The compact model of appearance and geometric deformation was designed to reanimate geometric and appearance variation, and the same model can be used for free-viewpoint rendering of performance animations. Using the model for this purpose requires only slight modifications and retains some useful advantages: 1) the same rendering framework can be used for free-viewpoint video, and 2) the compact representation provides a method for lossy compression. In fact, the use of a kinematic deformation model along with PCA dimensionality reduction are techniques are often

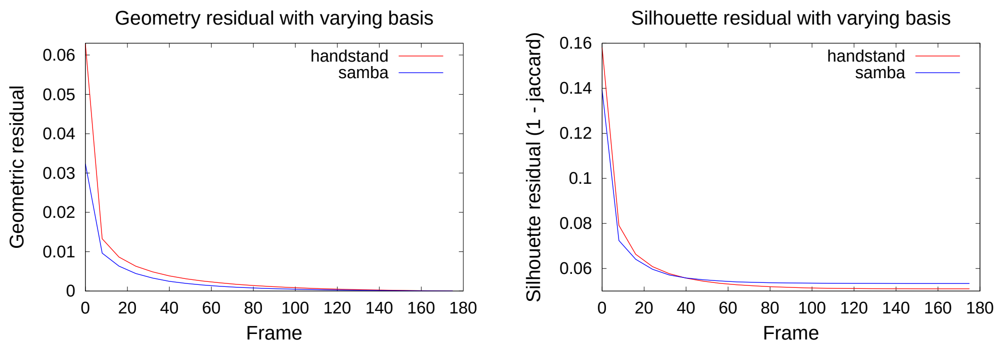


Figure 7.24: The geometric deformations of the 175 frame sequences are accurately modeled with roughly 40 basis vectors (left). The corresponding silhouette error, measured as $s_{\text{accuracy}} = 1 - Jaccard$, closely corresponds to the geometric accuracy, where again 40 basis vectors are suitable.

used for compressing time-varying geometries (e.g., [142] [5]). The only modifications necessary are to define an appropriate interpolation mode for the geometry and appearance. The geometric deformations are dependent on time instead of pose. A *time bone* can be used to mimic the effect of time-dependence. No vertices are directly attached to this artificial bone, but the pose of this bone is directly correlated with the input frame index, allowing the geometric and appearance interpolation to be specified as pose-dependent (e.g., influenced only by this time bone). Unless doing interpolation between time frames, the appearance interpolation method should only blend between PCA coefficients at each given time instant.

Using the above modifications, the combined model of appearance and deformation is applied to free-viewpoint video using data from the MIT data sets[259][2]. Each data set contains 8 viewpoints of 800×600 images[3], 175 time frames, and a deforming geometry with 10000 vertices. First, the effectiveness of compressing the geometry is evaluated followed by texture compression.

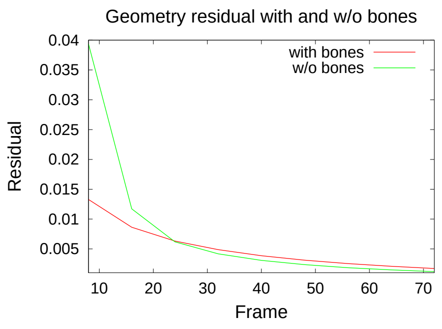
In order to evaluate the effectiveness of the geometric component, the combined model was built using all of the geometric input from the data set. The geometric and silhouette residual as a function of the number of geometric basis elements are plotted in Figure 7.24. As expected, the first few basis elements account for most of the geometric variability; after 40 basis elements, there is little improvement in including more geometric modes. For these sequences, the geometric model contains 10000 vertices, meaning the storage requirements (in number of floating-point values) would reduce from $175 \times \text{numvert} \times 3$ to

$$\underbrace{40 \times \text{numvert} \times 3}_{\text{basis}} + \underbrace{175 \times 40}_{\text{coefficients}}, \tag{7.1}$$

for a compression ratio of 0.23.

For the *Handstand* sequence, we also evaluate the impact of representing the geometric deformations relative to the skeletal animation as opposed to simply representing geometric deformation

[2]Data can be obtained through http://people.csail.mit.edu/drdaniel/mesh_animation/index.html
[3]The images were downsampled from the original 1600×1200



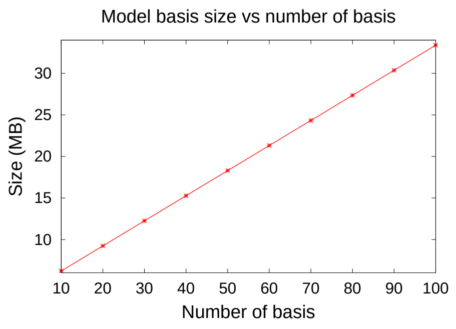
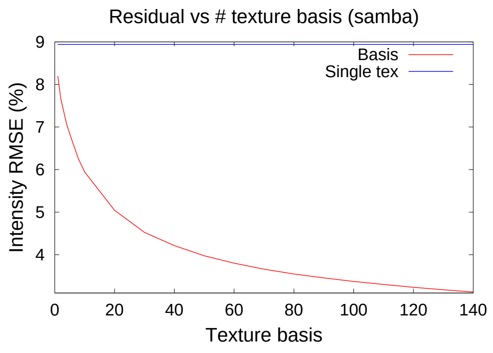


Figure 7.25: Representing geometric modes relative to the skeleton (with bones) reduces geometric variation allowing the first basis vectors to better model geometric variation as opposed to without bones (w/o bones).

Figure 7.26: The uncompressed model size increases linearly with the number of texture basis elements. The model sizes listed here include a 32 basis element geometry and the necessary data for the interpolation.



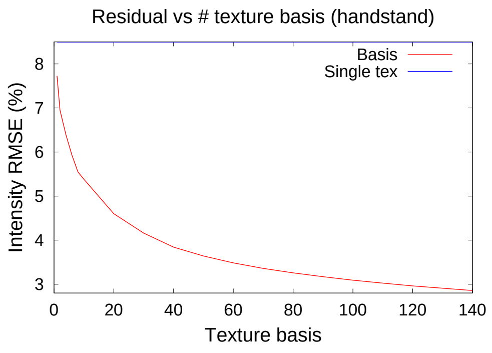


Figure 7.27: RMS image intensity error for the *Samba* and *Handstand* datasets averaged over all 8 cameras and all times for a varying number of texture basis images. The first 40-60 basis images cause a significant reduction in RMSE. Afterwards, increasing the number of basis elements gives diminishing returns meaning the basis can be truncated to achieve compression.

relative to a mean shape. The use of a skeletal model and the first few modes of variation are better able to reconstruct the geometry than if the skeleton were ignored (Figure 7.25). As skeletal animation parameters don't require much storage (e.g., 100s of bytes per frame), the representation of geometric deformations relative to the skeleton is still effective in applications of free-viewpoint compression.

For a quantitative evaluation of render quality on the *Samba* and *Handstand* sequences, models were built with varying numbers of basis elements using every second time frame as input. The texture size can be seen as another parameter that influences quality, with higher resolutions leading to better renderings and larger file sizes. In these experiments, this texture size was held constant at 512x512, which may cause some downsampling of input images. Figures 7.27 illustrates the RMSE for varying basis elements averaged over all time frames and all cameras (see Figure 7.28 for temporal image residuals averaged over all cameras). The majority of the variation is encoded in the first few dozen basis elements, meaning compression can be obtained by truncating the number

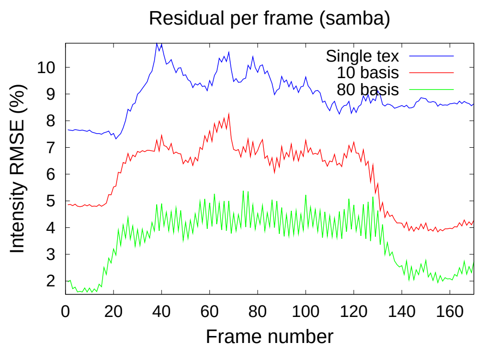


Figure 7.28: Intensity RMSE averaged over each camera for all time frames of the *Samba* sequence. The residual is not uniformly distributed over the sequence because the beginning and end have similar frames with slower motions.

of basis elements (e.g., see Figure 7.26). As the model size is linearly proportional to the number of basis elements, the decision to truncate basis elements can be made based on desired model size or on desired RMSE. The uncompressed input images occupy nearly 2GB ($800 \times 600 \times 3 \times 175 \times 8$), whereas the uncompressed truncated texture basis occupies anywhere from 5MB (resp. 35MB) for 10 (resp. 100) basis vectors.

Figure 7.29 shows the rendered results for the *Samba* sequence. Increasing the number of basis functions improves the detail across views. The *Handstand* sequence has a large amount of texture change due to large shirt motion that exposes the subjects stomach and back (Figure 7.30). For such a situation, a single texture or even a few basis functions are not capable of representing the texture shift. The use of 50 or more texture basis components is required to accurately portray this texture variation.

Despite the improved rendered quality with an increasing number of basis functions, it is possible for artifacts to appear. The largest cause of these artifacts is inaccuracies in the geometry, specifically around depth discontinuities. For example, if the reconstructed geometry of the arm is smaller than the true arm, texture for the arm will be backprojected onto the surface behind the arm (see Figure 7.31). In order to best reproduce the input images, the texture basis encodes this texture variation. On the other hand, in order to get the best novel viewpoint renderings, these misrepresented textures should be avoided. Although it is possible to filter out pixels near depth discontinuities, it is not straightforward to choose a single filter width for all possible geometric errors, implying that these artifacts are unavoidable. Furthermore, these rendering artifacts in novel viewpoints would exist in the view-dependent renderings even if the original input images were used for texturing.

In this section, the combined model of appearance and geometric deformation has been used to represent and render free-viewpoint performance animations. Although a significant compression in model size has been achieved, it may be possible to obtain a better compression ratio while having a higher signal to noise ratio, for example, through JPEG or MPEG encoding of the textures or input images. If transmission size is the main concern, the texture basis can be further compressed with

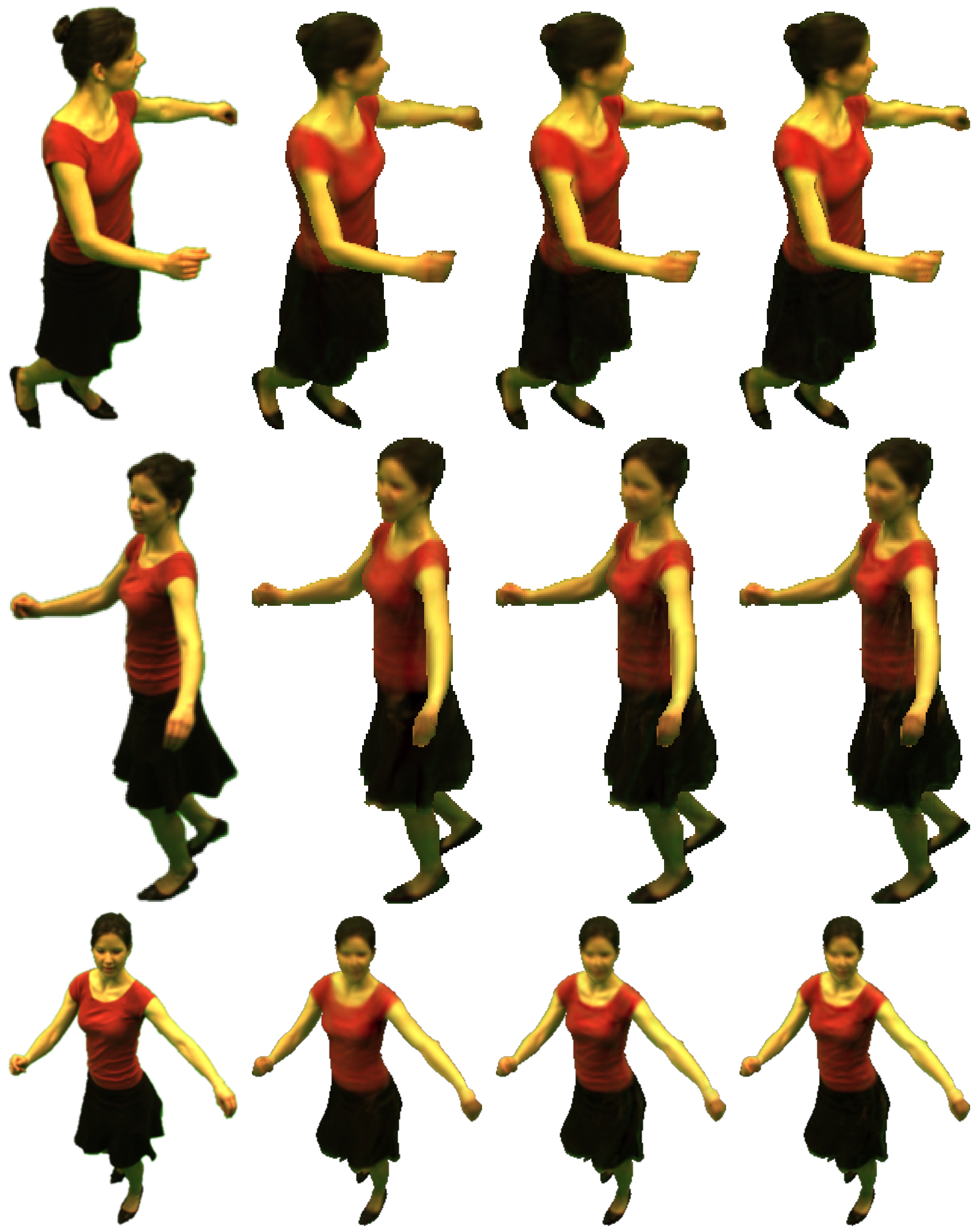


Figure 7.29: Rendered results for the *Samba* sequence using various numbers of basis functions. File sizes: 7MB, 13MB, 20MB.

such lossy image encoding methods. The later basis vectors represent higher order variation, and thus contribute less to the overall render quality and can likely be compressed more than the first few basis elements.

A key benefit of using the basis encoding is that the texture and geometry bases can be stored uncompressed in memory at runtime, as opposed to MPEG or JPEG representations that would

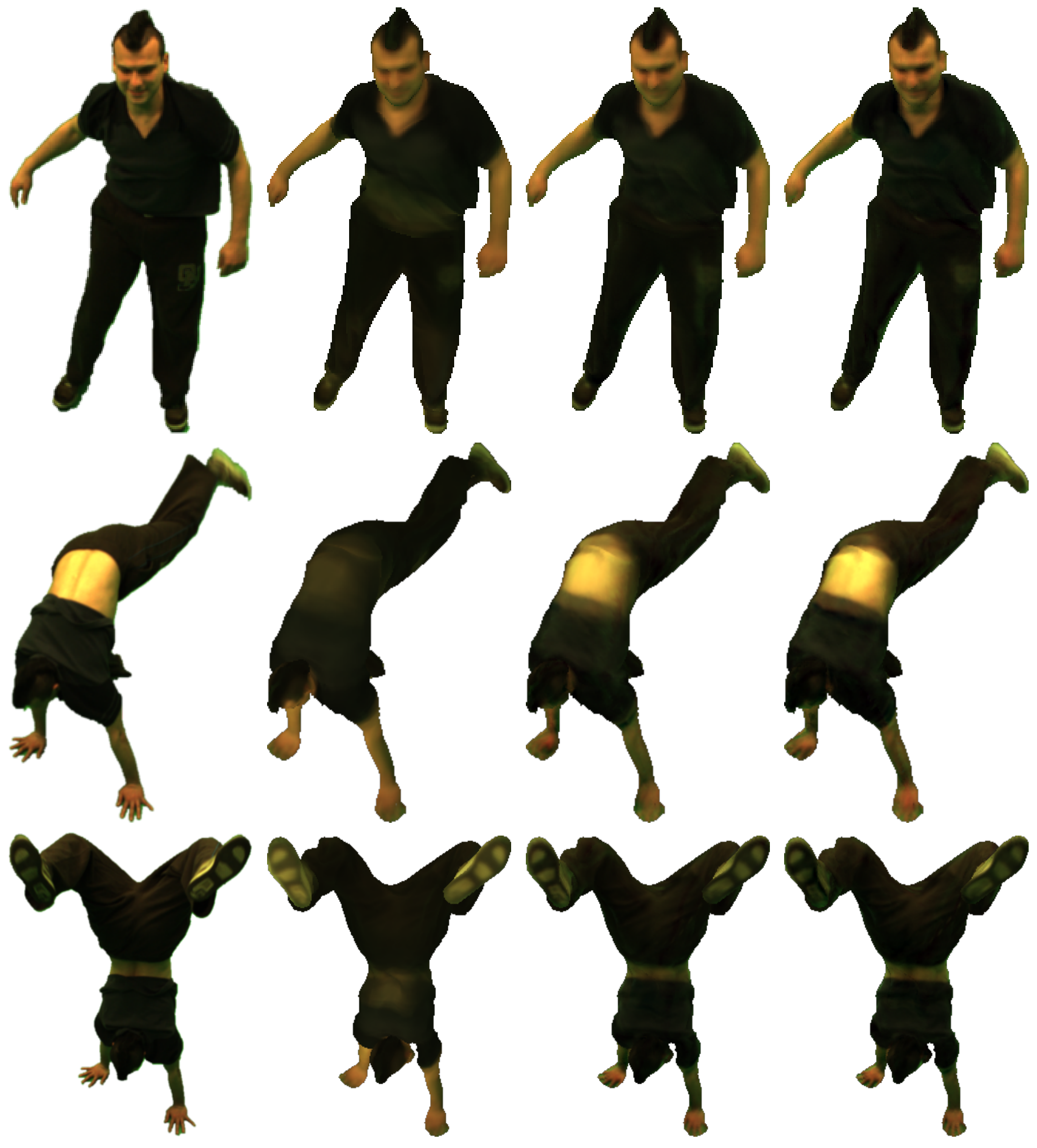


Figure 7.30: Rendered results for the *Handstand* sequence using various numbers of basis functions. File sizes: 800KB, 7MB, 13MB.

need to uncompress individual frames and could likely only store a subset of the entire sequence in memory.

Another benefit is that rendering from the texture basis is also efficient and can easily be implemented with software acceleration (e.g., through SIMD instructions, see Appendix E), or graphics hardware acceleration (through shaders or CUDA). Figure 7.32 gives example framerates for the most expensive part of rendering (the texture blending) for various implementations.

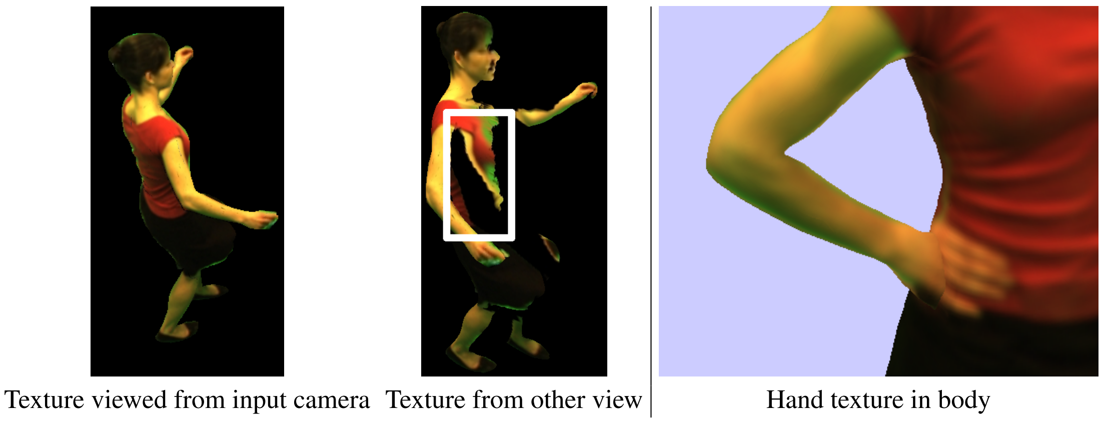


Figure 7.31: A single texture warped to uv-space and viewed from the camera image looks correct regardless of inaccurate geometry. However, when viewed from another viewpoint errors in the geometry cause colors to get mapped to the wrong location as is the case in the center image above. In order to best reproduce the input images, these artifacts will be encoded in the texture basis. A similar case occurs in the *Samba* sequence when the hand penetrates the body, but the texture basis is capable of reproducing the hand texture.

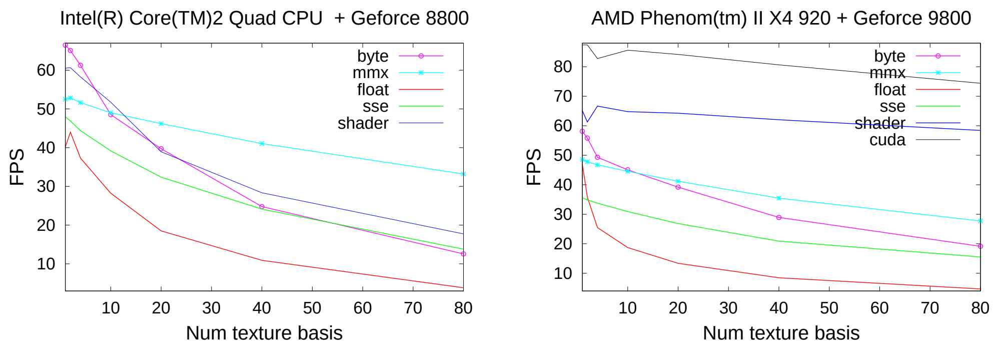


Figure 7.32: Modulating the texture basis is the most time consuming part of rendering. The above tables show the frames-per-second (FPS) of texture basis modulation for various implementations on different hardware. If available, a CUDA implementation provides superior performance. A CPU-only MMX implementation also performs well.

7.10 Discussion

In this chapter, a number of experiments were performed on reduced versions of the pose- and view-dependent model. These experiments motivate the use of the kinematic component, a pose-dependent geometric component, and a pose- and view-dependent texture component. The complete model was then used to acquire pose- and view-dependent models of a human face, an upper body, and a lower body that can be rendered under novel pose and view conditions. Furthermore, the complete model has other applications in compression of free-viewpoint performances, and acquisition of other non-rigid objects.

Through the design and implementation of these experiments, a number of general guidelines for acquisition and model building can be defined.

	Ease of Capture due to Acquisition Setup			
Configuration	few views few poses	few views many poses	many views few poses	many views many poses
		face, upper body, legs	*accordion*	*robotic arm*
	◄··························	··························	··························	··························►
	harder to capture cheaper setup			easier to capture expensive setup

Table 7.2: In general, the model is easier to capture and has better quality with many views and many poses. However, such conditions often require expensive hardware setups.

7.10.1 Acquisition

For acquisition, there are a number of important considerations regarding the design of the input motion and the configuration of the cameras used during acquisition. First, the input motion should avoid observing the same pose multiple times in the input sequence (§7.2); any duplicates should be removed before building the model. For example, in the stair walk sequence (§7.8.1), only the upward motion of the knee was used. Downward motion would be reanimated using poses with similar frames. Removing frames from the sequence in this way means that there can be some discontinuity between the time different parts of the model are acquired. For example, in the stairs sequence, the input observations between the time the left leg goes from up to down are not used, but the right leg starts moving up after the left leg goes down, implying that region in the crotch shared by both legs could become inconsistent. A final consideration for the input motion is that depth discontinuities of the geometry relative to the input views should be avoided if possible. As illustrated in Section 7.9.2, such discontinuities cause artifacts in the texture and are hard to eliminate simply by filtering.

The cameras used during acquisition must be accurately calibrated with respect to position, color, and time. The camera configuration depends on the model being captured and hardware availability. In general, as illustrated in Table 7.2, the use of many views and a dense sampling of poses is ideal to both acquire the input geometry as well as to model pose and view variation smoothly (e.g., the accordion §7.9.1, or the robot arm captures §7.5). For the robot arm captures, a view separation of 10 degrees is ideal, while a denser sampling than 10 degrees seemed unnecessary. Similarly, a dense acquisition of pose is also desirable, with the effect of sparse sampling of pose being similar to sparsely sampled view. However, separating the input views by 10 degrees would often require an impractical number of cameras, or it would require robotic capture.

When few viewpoints are available, the setup of cameras during acquisition should take into account how the model will be rendered (e.g., frontal viewpoints for the legs). A conflicting requirement in this few view case, is that the cameras should be separated enough to allow tracking through the use of silhouettes.

Surface considerations also come into play when designing the camera setup. For example, if the surface is not view-dependent and the geometry is accurate, the view variation will be negligible.

	Ease of Modeling due to Inputs and Object Properties		
Shape & correspondence	poor shape no correspond. *upper body, legs*	good shape poor correspond. *robotic arm* *accordion*	accurate shape dense correspond. *deforming bump*
Number of freedoms	many *upper body, legs*	*face*	few *robotic arm*
Pose-dependency	baggy clothing *pajama cloth*		skin, tighter clothing *face, legs, corduroy cloth*
	◄ harder to model		easier to model ►

Table 7.3: Creating good quality models is directly influenced by good quality structure and inputs. More complicated objects with more freedoms are harder to model, as are objects that are likely to stray from pose-dependency.

This means that the closely arranged views are not necessary to obtain the texture information. Of course, the maximum separation of the cameras may be limited by the the surface reconstruction method. For example, if stereo is being used, the views cannot be too widely separated.

The acquisition method has been designed to obtain a dense sampling of pose. However, for objects that have limited deformations in pose, which was the case for the accordion, dense pose sampling can be exchanged for dense sampling of viewpoint, for example, by using turntable acquisition of a discrete set of poses as in §7.9.1.

A final consideration for acquisition is with respect to lighting. As lighting gets baked into the texture, the input sequence should be designed (or acquired) in similar lighting conditions as the rendering will occur.

7.10.2 Modeling

With respect to the model, the experiments have demonstrated that, in general, all components are important. However, the kinematic component can be dropped if the underlying object does not require it (e.g., the face). The ability to obtain an accurate model is highly dependent on the quality of the input and the complexity of the object (Table 7.3) Ideally, correct correspondence of cloth shifting could be obtained, which gives dense accurate correspondence of the surface over time. In practice, it is hard to obtain these cloth shifts, due mostly to change in appearance and occlusion of the surface. It is still useful to obtain stereo displacements that make the surface photo-consistent without temporal correspondence. Although the improved surface estimate does nothing to reduce variation across time, a better surface allows for more accurate view interpolation.

Silhouettes are another reliable cue to obtain temporal displacements. A model built with such displacements will be capable of producing deformations that give accurate silhouettes. If silhouette or stereo displacements cannot be reliably obtained, it is best to use only the texture and kinematic components of the model. When using silhouette displacements, it can be useful to integrate the updated displacements into the tracking module (in order to reduce jitter). Furthermore, whenever using geometric displacements, the displacements should be projected onto the lower dimensional linear basis before warping textures. This ensures that the rendered model will best agree with the input images.

One of the most important components of the model are the choices made for the pose and view interpolation. The texture and geometry basis typically encode all of the information necessary for achieving good renderings, but the interpolation (or mapping) from pose and view to texture must be done correctly. The decisions involved in the interpolation of the model include splitting the model into parts, defining the influencing parameters of the parts, and defining the interpolators that map influencing parameters to texture and geometry coefficients. The division of the object into parts often comes naturally for kinematic objects (e.g., splitting the lower body into the components of the legs). However, depending on the desired rendering, the decision of which pose parameters affect each part are important to get right. Again, for kinematic components, the affecting parameters of a part are typically the joint angles of the parent of the part combined with those of the child. However, depending on the input and desired re-rendering, other influencing parameters may be included. For example, correct modeling of cloth and light variation on the shin required the shin part to be influenced also by the hip angle (§ 7.8.2).

The last component of interpolation is the choice of the interpolant and the space where interpolation is performed. As discussed in Section 3.3.2, the interpolation is easier to perform by doing the pose and view interpolation separately. A decision on whether to use a linear projection for pose can be made by analyzing the dimensionality of the input pose animation. The type of projection used in view interpolation is easily determined by the camera configuration (e.g., a ring of cameras should use a 1D projection, whereas a 2D sampling of the view hemisphere should use a spherical mapping).

As for the interpolant, nearest neighbor interpolation will guarantee high fidelity but is suitable only for rendering static images. When rendering animations, the interpolant must produce a smoothly varying texture as a function of pose. Problems due to incorrect interpolation manifest as popping in texture (something that is hard to visualize in this thesis). In practice, linear interpolation works well for animations, and it guarantees that the rendered textures will not stray wildly from the input textures. This is especially important outside the convex hull of the input poses, where interpolants like RBFs may give results that differ too much from the input textures.

8 Conclusion

The focus of this thesis has been image-based modeling of pose- and view-dependent appearance and pose-dependent geometry for dynamic human subjects. In this chapter, the model and contributions are reviewed, limitations of the proposed model are identified, and several areas for future work are discussed.

8.1 Summary

The pose-/view-dependent model and the vision acquisition pipeline presented in this thesis extends our three-tier modeling paradigm for static objects [116] with a temporal or pose dimension. The three-tier paradigm used a coarse macro-scale base geometry, a meso-scale refined geometry, and a micro-level view-dependent appearance basis. The compact model presented in this thesis also has three-tiers, with each tier successively refining the detail of the geometry, motion, and appearance. At the macro level, a proxy geometry coupled with a moving kinematic skeleton is used to model coarse geometric effects. Meso-scale geometric deformations are modeled during acquisition and rendering by vertex displacements, and fine-scale surface appearance is modeled by a micro-level texture basis. This paradigm breaks down the complicated acquisition process into manageable tasks; the compact representation has pieces that parallel these three-tiers.

This compact model can be acquired from a multi-view training sequence and can then be rendered under novel pose and view conditions while exhibiting the rich deformations that existed in the training sequence. The proposed model and acquisition pipeline resulted in several individually useful contributions.

8.1.1 Pose- and View-dependent Model

The compact model of pose-dependent geometry and pose- and view-dependent appearance for kinematic models is based on the extensive use of linear subspaces for modeling geometric/appearance variation. In the vision and graphics literature, linear subspaces have been used to model appearance (e.g., [61, 174]) or geometric variation (e.g., [148]). Although the existing methods model both appearance and geometric variation (e.g., [62, 183, 182, 34]), they do not use a kinematic model or represent appearance changes due to pose. The contribution of the proposed model lies in the combined modeling of appearance and geometric variation on a kinematic model. The geometric component encodes cloth shifting and geometric shape changes, for example, to make silhouettes agree. The appearance component models view-dependent effects, due to incorrect geometry or specularities, and fine-scale pose-dependent effects, due to cloth and geometry movement not encoded in the geometric component.

In order to achieve view-dependent appearance, the kinematic object is divided into multiple overlapping regions, with each part using a local view vector. Pose- and view-dependent appearance synthesis uses the training sequence as input for scattered data interpolation. As joint angles and view direction are not commensurate, an interpolation scheme that favors pose over viewpoint has also been proposed. The interpolation scheme uses a data projection of pose to first interpolate view-based texture coefficients for each view, followed by a separate interpolation of these view-based texture coefficients as a function of view.

The end result is a model that is capable of generalizing a wide range of computer graphics models including simple single textured skinned models, view-dependent static models, free-viewpoint performance acquisitions, and the full pose- and view-dependent dynamic models.

8.1.2 Static Model Acquisition

The basic component of the proposed model is a static geometry that is attached to a kinematic skeleton. This geometry could be obtained by any means (e.g., stereo, shape from silhouette, or manually).

In Chapter 4, an interleaved silhouette-based tracking and refinement procedure was proposed to acquire human models while compensating for unintended subject motion. This solution is low cost, requiring as few as two cameras, and obtained estimates of human geometry and texture without requiring obtrusive structural supports.

Acquisition of human geometries in this manner has applications outside the scope of this thesis. For example, studies on population statistics or representations of human geometric variation [9, 212, 213] could utilize the propose tracking and refinement as an inexpensive way to obtain input geometries of humans.

8.1.3 Coarse-Scale Joint Tracking

In order to acquire the coarse-scale joint tracking of the model (both for the training sequence and potentially for novel animations), an energy-based formulation was used to track linear-blend skinned meshes. The tracking framework is general purpose and can be used to obtain motion capture of rigid objects, human skeletons, or even non-rigid animations (e.g., facial animation).

Within this energy framework, two silhouette data terms and an SSD intensity-based data term were combined with temporal smoothness and pose prior terms. The silhouette terms are preferable when available, but the intensity-based term is useful when they are not.

The energy was minimized using local optimization, with the focus being on efficiency. For this reason, efficient methods for both evaluating the silhouette objective functions and optimizing the energies were presented. Analytic derivatives and GPU acceleration were used to bring the SSD minimization to near real-time, making it practical for interactive applications. The experiments demonstrate that the local optimization is sufficient in most cases, and if not, the efficiency of the methods eases the time commitment of an operator who can monitor and correct the tracking algorithm.

On real input sequences, silhouette tracking using the iterative-closest-point (ICP) term was faster than the XOR-based term, but the XOR term worked better in the case of noisy silhouettes. The pose prior also aided the tracking when silhouettes were poor but was unnecessary otherwise.

8.1.4 Fine-Scale Surface Motion

For acquiring the fine-scale time-dependent surface geometry and temporal correspondences of the geometry, methods were developed with the focus being on how to best make use of the coarse tracking with a limited number of viewpoints. In order to accomplish this task, a simpler problem of reconstructing both surface motion and structure from a single moving camera was described. For this reduced problem, a constant velocity condition was imposed on the flow, allowing reconstructing of both the depth and flow of the surface from as few as 3 time frames using a variational algorithm.

This simple version of the problem is extended to a multi-view configuration, where instead of assuming known camera motion, the known tracked motion of the base skinned mesh is used. Further, instead of formulating the problem of reconstructing depth and flow on the image plane, the temporal surface correspondence and surface displacements are represented directly as displacements from the skinned base mesh. In this more general setting, the 3D surface motion is represented using a low dimensional temporal basis (e.g., a cosine basis). Through this basis representation, the temporal surface motion is constrained while the number of variables of motion are reduced. Using this formulation of displacement relative to a known moving proxy surface best makes use of the known tracked motion, and the basis constraint allows for reconstruction when only few views are available. Finally, this representation generalizes two-frame optic flow, scene flow, and displacement map reconstruction from a proxy mesh.

8.1.5 Animating and Rendering the Compact Model

Through the experiments with the compact model, it was demonstrated that clothing can be represented as pose dependent. The experiments also validated the geometric component of the compact model as it reduces some of the burden of the texture component when it comes to representing view variation (e.g., if geometry is accurate, fewer input views and fewer texture basis elements are needed).

Qualitative results of pose- and view-dependent synthesis using the proposed pipeline were demonstrated for a face, an upper body, and legs. An alternative to the multi-camera pipeline acquisition is to use a small set of input poses and a dense set of views (e.g., through turntable acquisition). This was demonstrated on the accordion.

The proposed model was also tested in compression applications for free-viewpoint performance capture. The trade-off in both model size (and render time) is dependent on the number of geometric and texture basis elements that are retained in the model. Finally, the experimental analysis allowed the identification of general guidelines regarding camera placement and the requirements of the training sequence during acquisition.

8.2 Limitations

The proposed compact model and its acquisition have some limitations. The acquisition of dense surface correspondences is complicated by lighting changes, self-occlusions, and cloth folding. These complications are primarily due to the use of an intensity-based correspondence measure. Although feature-based methods may be more appropriate, they would only work when the surface has sufficient texture. However, having a more photo-consistent surface at each frame is more important for view interpolation than temporal correspondences, meaning that traditional stereo methods can be used; temporal texture shifts can be modeled by the appearance component.

When synthesizing novel animation and viewpoint there are several limitations. First, the synthesized images will exhibit the illumination conditions at the time of capture, i.e., light is baked into the model. In order to insert a captured model into a synthetic scene, the illumination during capture must be adjusted appropriately. An alternative, not yet explored, is to capture and model illumination variation in the model. One approach for obtaining this illumination variation without making the acquisition more tedious would be to use the time multiplexed illumination common in photometric stereo [129].

It is hard to acquire a dense sampling of all poses of a human as the full pose space for humans is combinatorial in the number of joint parameters. While such a dense sampling of poses is possible using a robotic manipulator, in practice the training sequence of a human is likely to contain a sparse non-uniform sampling of poses. To ensure decent renderings with sparsely sampled training poses, the poses in the training sequence should lie close to the desired animation sequence. This often means that the acquisition sequence has to be specifically designed for the application for which it is being captured.

Similarly, extrapolation of view is limited to local regions around the input viewpoints. If it is desirable to render from many viewpoints, then many viewpoints are required during the acquisition. Too few views result in poor view interpolation and inaccurate geometry. Turntable acquisition can be used to reduce the number of cameras required, but this type of capture only works when the object of interest is inanimate and can be posed in different poses when placed on the turntable.

As with other image-based methods, and unlike traditional computer graphics, it is hard to apply artist controlled edits. Although one simple type of transformation edit was applied to the pose-dependent interpolation space (7.8.2), any artistic feedback must be directed during the acquisition.

The applicability of the model is limited to objects and scenes where the motion and appearance can be represented as a function of some measurable variable. For human capture, a range of tighter fitting, thick fabrics will obey this pose-dependent assumption. On the other hand, the model cannot represent dynamic effects such as waving cloth (e.g., baggy clothes or dresses) because they are not solely dependent on pose. It may be possible to model these effects with different interpolation parameters, such as using velocity in addition to the joint pose. However, adding extra parameters to the interpolators would necessitate a denser sampling of motions, but dense sampling of poses alone is hard enough without the addition of an extra dimension.

8.3 Future Work

The focus throughout this work was to recover the necessary inputs of the model when only a few viewpoints were available. The methods presented in Chapter 6 tried to utilize intra-camera observations in order to perform reconstructions under these conditions. To do so the presented methods assumed known coarse surface tracking was available. If the tracking was incorrect, these methods could not recover. Instead of relying blindly on the coarse tracking results, salient features (e.g., corner points) could be used when possible to both aid the tracking and improve the reconstruction. Further, such feature-points could also be used to aid the motion recovery for the static turntable acquisition.

Another direction for improvement of the dense temporal surface tracking would be to integrate a silhouette constraint directly into the surface refinement. The joint angle tracking can be used to aid the reconstruction, but accurate tracking requires an accurate model of surface deformation. For this reason it would be interesting to couple the tracking and flow estimation into a single formulation, which would allow each process to mutually benefit from the output of the other. Along a similar direction, it would be useful to explore the use of the final pose- and view-dependent compact model in tracking new sequences. The improved pose-based geometry and more accurate appearance should result in more accurate tracking regardless of whether silhouette or appearance cues are used.

One of the design decisions involved when acquiring a model is to manually decide what pose parameters influence the appearance/geometry and which method is used for the interpolation. Although the influencing pose parameters are most naturally something that is chosen by the operator, the best interpolation method (e.g., which projection to use, how many projected pose dimensions to keep) could likely be identified automatically.

There are also several implementation details of the model that could be further investigated. For example, when building the appearance basis, texture pixels that are not visible for a particular view are simply replaced with the mean value of the texel in all visible views and poses. Instead, in a multi-view setting, it may be better to fill in these missing regions using the information available from the same time instant using a different view only. An additional improvement related to the PCA of textures could come from using a multi-resolution hierarchical PCA instead of performing a single PCA on the high resolution textures. Such a strategy would further compress the model.

Finally, it would be useful to identify a small sequence of human actions that can be used capture a complete human model that can be animated through arbitrary poses. Ideally, the compact model could be learnt by some arbitrary motion of a human subject, but as identified earlier, such arbitrary motion would likely suffer from duplicate poses that observed the human surface with different deformations—a situation which makes pose-based interpolation impossible. Separating the model into upper and lower body parts, as done in the current experiments, makes the design and acquisition of such a sequence less tedious. Further, the guidelines identified for acquisition in this thesis (e.g., the requirement on many views and dense pose samples) and the interpolation methods described are a significant step towards this goal.

Applications in graphics and computer vision often rely on a variety of complementing techniques to make up a complete system. Many of the techniques are adapted to modeling or representing a specific effect. The photo-realistic pose- and view-dependent model proposed in this thesis is useful for modeling pose-dependent effects on kinematic objects directly from images. With applications ranging from vision-based tracking to graphics applications requiring real-time renderings of realistic humans, this compact model is yet another useful tool in the graphics and vision practitioners toolbox.

Bibliography

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. *M. Landy and J. A. Movshon, (eds) Computational Models of Visual Processing*, 1991.

[2] E. Aganj, J.-P. Pons, F. Segonne, and R. Keriven. Spatio-temporal shape from silhouette using four-dimensional delaunay meshing. In *IEEE International Conference on Computer Vision (ICCV '07)*, pages 1–8, Oct. 2007.

[3] Naveed Ahmed, Edilson de Aguiar, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Automatic generation of personalized human avatars from multi-view video. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 257–260, New York, NY, USA, 2005. ACM.

[4] Tomas Akenine-Möller and Eric Haines. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, 2nd edition, 2002.

[5] M. Alexa and W. Müller. Representing animations by principal components. *Computer Graphics Forum*, 19(3):411–418, 2000.

[6] Marc Alexa. Linear combination of transformations. *ACM Trans. Graph.*, 21(3):380–387, 2002.

[7] Marc Alexa. Mesh editing based on discrete laplace and poisson models. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 51–59, New York, NY, USA, 2006. ACM.

[8] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM Trans. Graph.*, 21(3):612–619, 2002.

[9] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: reconstruction and parameterization from range scans. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 587–594, New York, NY, USA, 2003. ACM.

[10] 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 *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 147–156, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[11] C. Allne, J-P. Pons, and R. Keriven. Seamless image-based texture atlases using multi-band blending. In *Proceedings of International Conference on Pattern Recognition (ICPR'08)*, pages 1–4, Tampa, US, Dec 2008.

[12] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: shape completion and animation of people. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 408–416, New York, NY, USA, 2005. ACM Press.

[13] Javier Iglesia Aparicio and Jaime Gómez García-Bermejo. An approach for determining phong reflectance parameters from real objects. In *Proceedings of International Conference on Pattern Recognition (ICPR'00)*, pages 3572–3575, 2000.

[14] J. Arnabat, S. Casanovas, and G. Medioni. 3d modeling from turntable sequences using dense stereo carving and multi-view consistency. In *Proceedings of International Conference on Pattern Recognition (ICPR'04)*, volume 4, pages 36–39, August 2004.

[15] Amaury Aubel, Ronan Boulic, and Daniel Thalmann. Animated impostors for real-time display of numerous virtual humans. In *VW '98: Proceedings of the First International Conference on Virtual Worlds*, pages 14–28, London, UK, 1998. Springer-Verlag.

[16] Shai Avidan and Amnon Shashua. Non-rigid parallax for 3d linear motion. In *Proceedings of the 1998 DARPA Image Understanding Workshop*, pages 199–201, 1998.

[17] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 1:1090, 2001.

[18] A. O. Balan, L. Sigal, and M. J. Black. A quantitative evaluation of video-based 3d person tracking. In *ICCCN '05: Proceedings of the 14th International Conference on Computer Communications and Networks*, pages 349–356, Washington, DC, USA, 2005. IEEE Computer Society.

[19] Alexandru Balan, Leonid Sigal, Michael Black, James Davis, and Horst Haussecker. Detailed human shape and pose from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, Minneapolis, Minnesota, USA, June 2007.

[20] A.O. Bălan and M.J. Black. The naked truth: Estimating body shape under clothing. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, pages 15–29. Springer-Verlag, 2008.

[21] Luca Ballan, Gabriel J. Brostow, Jens Puwein, and Marc Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. In *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, pages 1–11, Los Angeles, July 2010.

[22] Luca Ballan and Guido Maria Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT'08*, Atlanta, GA, USA, June 2008.

[23] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series*, pages 43–54. SIGGRAPH, 1998.

[24] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. *ACM Trans. Graph.*, 22(3):862–870, 2003.

[25] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72, 2007.

[26] Tali Basha, Yael Moses, and Nahum Kiryati. Multi-view scene flow estimation: A view centered variational approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pages 1506–1513, 2010.

[27] Adam Baumberg. Blending images for texturing 3d models. In *British Machine Vision Conference (BMVC '02)*, pages 404–413, 2002.

[28] Adam Baumberg, Alex Lyons, and Richard Taylor. 3D S.O.M. - a commercial software solution to 3d scanning. In *Proceedings of Vision, Video, and Graphics (VVG'03)*, pages 41–48, July 2003.

[29] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[30] Peter N. Belhumeur and Gregory D. Hager. Tracking in 3d: Image variability decomposition for recovering object pose and illumination. *Pattern Analysis and Applications*, 2(1):82–91, 1999.

[31] Peter N. Belhumeur, David J. Kriegman, and Alan L. Yuille. The bas-relief ambiguity. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, 00:1060, 1997.

[32] Kiran Bhat, Christopher D. Twigg, Jessica K Hodgins, Pradeep Khosla, Zoran Popovic, and Steven Seitz. Estimating cloth simulation parameters from video. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 37 – 51, July 2003.

[33] Neil Birkbeck, Dana Cobzas, and Martin Jagersand. Object centered stereo: Displacement map estimation using texture and shading. In *3DPVT'06*, pages 790–797, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[34] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 187–194, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[35] Thomas Bonfort and Peter Sturm. Voxel carving for specular surfaces. In *IEEE International Conference on Computer Vision (ICCV '03)*. IEEE CSP, October 2003.

[36] Edmond Boyer. On using silhouettes for camera calibration. In *Asian Conference on Computer Vision (ACCV '06)*, 2006.

[37] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 1–9, New York, NY, USA, 2008. ACM.

[38] Matthieu Bray, Pushmeet Kohli, and Philip H. S. Torr. Posecut: Simultaneous segmentation and 3d pose estimation of humans using dynamic graph-cuts. In *ECCV (2)*, pages 642–655, 2006.

[39] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, volume 2, pages 690–696, 2000.

[40] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '98)*, pages 8–15, 1998.

[41] Christoph Bregler, Jitendra Malik, and Katherine Pullen. Twist based acquisition and tracking of animal and human kinematics. *Int. J. Comput. Vision*, 56:179–194, February 2004.

[42] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 28–36. Eurographics Association, 2003.

[43] Thomas Brox, Bodo Rosenhahn, Daniel Cremers, and Hans-Peter Seidel. Nonparametric density estimation with adaptive, anisotropic kernels for human motion tracking. In *Proceedings of the 2nd conference on Human motion: understanding, modeling, capture and animation*, pages 152–165, Berlin, Heidelberg, 2007. Springer-Verlag.

[44] Marcus A. Brubaker, David J. Fleet, and Aaron Hertzmann. Physics-based person tracking using simplified lower-body dynamics. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, Minneapolis, Minnesota, USA, June 2007.

[45] Andres Bruhn and Joachim Weickert. Towards ultimate motion estimation: Combining highest accuracy with real-time performance. In *IEEE International Conference on Computer Vision (ICCV '05) - Volume 1*, pages 749–755, Washington, DC, USA, 2005. IEEE Computer Society.

[46] Peter J. Burt and Edward H. Adelson. A multiresolution spline with application to image mosaics. *ACM Trans. Graph.*, 2:217–236, October 1983.

[47] Cedric Cagniart, Edmond Boyer, and Slobdodan Ilic. Iterative mesh deformation for dense surface tracking. In *IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM '09)*, 2009.

[48] S. Capell, S. Green, B. Curless, T. Duchamp, and Z. Popovic. Interactive skeleton-driven dynamic deformations. *ACM Trans. Graph.*, 21(3):586–593, 2002.

[49] Rodrigo L. Carceroni and Kiriakos N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3d motion, shape and reflectance. *Int. J. Comput. Vision*, 49(2-3):175–214, 2002.

[50] Charles-Félix Chabert, Per Einarsson, Andrew Jones, Bruce Lamond, Wan-Chun Ma, Sebastian Sylwan, Tim Hawkins, and Paul Debevec. Relighting human locomotion with flowed reflectance fields. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 76, New York, NY, USA, 2006. ACM.

[51] J. E. Chadwick, D. R. Haumann, and R. E. Parent. Layered construction for deformable animated characters. In *SIGGRAPH '89: Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 243–252, New York, NY, USA, 1989. ACM.

[52] Shenchang Eric Chen. QuickTime VR — an image-based approach to virtual environment navigation. *Computer Graphics*, 29(Annual Conference Series):29–38, 1995.

[53] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 447–456, New York, NY, USA, 2002. ACM.

[54] Yisheng Chen, J. Lee, R. Parent, and R. Machiraju. Markerless monocular motion capture using image features and physical constraints. In *CGI '05: Proceedings of the Computer Graphics International 2005*, pages 36–43, Washington, DC, USA, 2005. IEEE Computer Society.

[55] Kong Man Cheung, Simon Baker, Jessica K Hodgins, and Takeo Kanade. Markerless human motion transfer. In *Proceedings of the 2nd International Symposium on 3D Data Processing, Visualization and Transmission*, September 2004.

[56] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, 62(3):221 – 247, May 2005.

[57] Kong Man Cheung, Simon Baker, and Takeo Kanade. Shape-from-silhouette across time: Part ii: Applications to human modeling and markerless motion tracking. *International Journal of Computer Vision*, 63(3):225 – 245, August 2005.

[58] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but responsive cloth. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 604–611, New York, NY, USA, 2002. ACM.

[59] Haili Chui and Anand Rangarajan. A new point matching algorithm for non-rigid registration. volume 89, pages 114–141, New York, NY, USA, February 2003. Elsevier Science Inc.

[60] CMU graphics lab motion capture database. http://mocap.cs.cmu.edu/.

[61] Dana Cobzas, Keith Yerex, and Martin Jägersand. Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. *Computer Graphics Forum*, 21(3), 2002.

[62] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *Lecture Notes in Computer Science*, 1407:484–498, 1998.

[63] Frederic Cordier and Nadia Magnenat-Thalmann. A data-driven approach for real-time clothes simulation. *Computer Graphics Forum*, 24(2):173–183, 2005.

[64] Jérôme Courchay, Jean-Philippe Pons, Pascal Monasse, and Renaud Keriven. Dense and accurate spatio-temporal multi-view stereovision. In Hongbin Zha, Rin ichiro Taniguchi, and Stephen J. Maybank, editors, *ACCV (2)*, volume 5995 of *Lecture Notes in Computer Science*, pages 11–22. Springer, 2009.

[65] W. Bruce Culbertson, Thomas Malzbender, and Gregory G. Slabaugh. Generalized voxel coloring. In *ICCV '99: Proceedings of the International Workshop on Vision Algorithms*, pages 100–115, London, UK, 2000. Springer-Verlag.

[66] Lawrence D. Cutler, Reid Gershbein, Xiaohuan Corina Wang, Cassidy Curtis, Erwan Maigret, Luca Prasso, and Peter Farson. An art-directed wrinkle system for CG character clothing and skin. *Graph. Models*, 69:219–230, September 2007.

[67] *The Twenty-Third IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2010, San Francisco, CA, USA, 13-18 June 2010*. IEEE, 2010.

[68] Cyberware ™. http://www.cyberware.com/.

[69] S. Dambreville, R. Sandhu, A.Yezzi, and A. Tannenbaum. Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. In *European Conference on Computer Vision (ECCV '08)*, volume 5303, pages 169–182, 2008.

[70] Per-Erik Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[71] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, pages 1–10, New York, NY, USA, 2008. ACM.

[72] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Marker-less 3d feature tracking for mesh-based human motion capture. In Elgammal et al. [81], pages 1–15.

[73] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proceedings of SIGGRAPH 96*, pages 11–20, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[74] Philippe Decaudin, Dan Julius, Jamie Wither, Laurence Boissieux, Alla Sheffer, and Marie-Paule Cani. Virtual garments: A fully geometric approach for clothing design. *Computer Graphics Forum (Eurographics'06 proc.)*, 25(3), sep 2006.

[75] Quentin Delamarre and Olivier Faugeras. 3d articulated models and multiview tracking with physical forces. *Comput. Vis. Image Underst.*, 81(3):328–357, 2001.

[76] M. Desbrun, M. Meyer, and P. Alliez. Intrinsic parameterizations of surface meshes. *Computer Graphics Forum*, 21(3):209–218, 2002.

[77] Jonathan Deutscher and Ian Reid. Articulated body motion capture by stochastic search. *Int. J. Comput. Vision*, 61(2):185–205, 2005.

[78] Frederic Devernay, Diana Mateus, and Matthieu Guilbert. Multi-camera scene flow by tracking 3-d points and surfels. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2203–2212, 2006.

[79] Ye Duan, Liu Yang, Hong Qin, and Dimitris Samaras. Shape reconstruction from 3d and 2d data using pde-based deformable surfaces. In *ECCV (3)*, pages 238–251, 2004.

[80] Martin Eisemann, Bert De Decker, Marcus Magnor, Philippe Bekaert, Edilson de Aguiar, Naveed Ahmed, Christian Theobalt, and Anita Sellent. Floating Textures. *Computer Graphics Forum (Proc. Eurographics EG'08)*, 27(2):409–418, 4 2008.

[81] Ahmed M. Elgammal, Bodo Rosenhahn, and Reinhard Klette, editors. *Human Motion - Understanding, Modeling, Capture and Animation, Second Workshop, Human Motion 2007, Rio de Janeiro, Brazil, October 20, 2007, Proceedings*, volume 4814 of *Lecture Notes in Computer Science*. Springer, 2007.

[82] Luis Baumela Enrique Muoz, Jos M. Buenaposada. A direct approach for efficiently tracking with 3d morphable models. In *IEEE International Conference on Computer Vision (ICCV '09)*, September 2009.

[83] Carlos Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. In *4th International Conference on 3D Digital Imaging and Modeling (3DIM '03)*, pages 46–53, October 2003.

[84] Carlos Hernández Esteban and F. Schmitt. A snake approach for high quality image-based 3d object modeling. In *2nd IEEE Workshop on Variational, Geometric and Level Set Methods in Computer Vision*, pages 241–248, October 2003.

[85] Olivier Faugeras and Renaud Keriven. Variatonal principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Transactions on Image Processing*, 7:336–344, Mar 1998.

[86] Jean-Sébastien Franco and Edmond Boyer. Exact polyhedral visual hulls. In *British Machine Vision Conference (BMVC '03)*, pages 329–338, September 2003. Norwich, UK.

[87] Oren Freifeld, Alex Weiss, Silvia Zuffi, and Michael J. Black. Contour people: A parameterized model of 2d articulated human shape. In CVPR 2010 [67], pages 639–646.

[88] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: combining multi-image stereo and shading. *Int. J. Comput. Vision*, 16(1):35–55, 1995.

[89] Y. Furukawa and J. Ponce. Dense 3d motion capture for human faces. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:1674–1681, 2009.

[90] Yasutaka Furukawa and Jean Ponce. Dense 3d motion capture from synchronized video streams. In *CVPR*. IEEE Computer Society, 2008.

[91] Yasutaka Furukawa and Jean Ponce. Accurate camera calibration from multi-view stereo and bundle adjustment. *Int. J. Comput. Vision*, 84(3):257–268, 2009.

[92] Yasutaka Furukawa, Amit Sethi, Jean Ponce, and David Kriegman. Structure and motion from images of smooth textureless objects. In *European Conference on Computer Vision (ECCV '04)*, 2004.

[93] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[94] Juergen Gall, Bodo Rosenhahn, and Hans P. Seidel. Drift-free tracking of rigid and articulated objects. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*. IEEE Computer Society, 2008.

[95] Juergen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, 2009.

[96] Jürgen Gall, Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. Optimization and filtering for human motion capture - a multi-layer framework. *International Journal of Computer Vision*, 87:75–92, 2010.

[97] Jürgen Gall, Carsten Stoll, Edilson de Aguiar, Christian Theobalt, Bodo Rosenhahn, and Hans-Peter Seidel. Motion capture using joint skeleton tracking and surface estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pages 1746–1753, Miami, USA, 2009. IEEE Computer Society.

[98] Jaime Gómez García-Bermejo and J. López Coronado. F. J. Díaz Pernas. An approach for determining bidirectional reflectance parameters from range and brightness data. In *IEEE International Conference on Image Processing (ICIP'96)*, pages 41–44, 1996.

[99] Ravi Garg, Luis Pizarro, Lourdes Agapito, and Daniel R uckert. Dense multi-frame optic flow for non-rigid objects using subspace constraints. In *Asian Conference on Computer Vision (ACCV '10)*, pages 460–473, 2010.

[100] D. M. Gavrila and L. S. Davis. 3-d model-based tracking of humans in action: a multi-view approach. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '96)*, page 73, Washington, DC, USA, 1996. IEEE Computer Society.

[101] Bastian Goldlücke and Daniel Cremers. Superresolution texture maps for multiview reconstruction. In *IEEE International Conference on Computer Vision (ICCV '09)*, pages 1677–1684, 2009.

[102] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *SIGGRAPH '96: Proceedings of the 23th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1996. ACM Press.

[103] Peng Guan, Oren Freifeld, and Michael J. Black. A 2d human body model dressed in eigen clothing. In *Proceedings of the 11th European conference on Computer vision: Part I*, ECCV'10, pages 285–298, Berlin, Heidelberg, 2010. Springer-Verlag.

[104] P. Gunawardane, O. Wang, S. Scher, I. Rickard, J. Davis, and T. Malzbender. Optimized image sampling for view and light interpolation. In *Proceedings of the International Symposium on Virtual Reality, Archeology and Cultural Heritage (VAST)*, pages 93–100. Eurographics Association, 2009.

[105] Z. Guo and K. Cheong Wong. Skinning with deformable chunks. *Computer Graphics Forum*, 24(3):373–381, 2005.

[106] Sunil Hadap, Endre Bangerter, Pascal Volino, and Nadia Magnenat-Thalmann. Animating wrinkles on clothes. In *VIS '99: Proceedings of the conference on Visualization '99*, pages 175–182, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

[107] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[108] Nils Hasler, Hanno Ackermann, Bodo Rosenhahn, Thorsten Thormählen, and Hans-Peter Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In CVPR 2010 [67], pages 1823–1830.

[109] Nils Hasler, Mark Asbach, Bodo Rosenhahn, Jens rainer Ohm, and Hans peter Seidel. Physically based tracking of cloth. In *VMV*, pages 49–56, 2006.

[110] Nils Hasler, Bodo Rosenhahn, Thorsten Thormählen, Michael Wand, Jürgen Gall, and Hans-Peter Seidel. Markerless motion capture with unsynchronized moving cameras. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '09)*, pages 224 – 231, Miami, USA, June 2009. IEEE Computer Society.

[111] C. Hernandez, F. Schmitt, and R. Cipolla. Silhouette coherence for camera calibration under circular motion. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, 29(2):343–349, February 2007.

[112] Adrian Hilton and Jonathan Starck. Multiple view reconstruction of people. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pages 357–364, Washington, DC, USA, 2004. IEEE Computer Society.

[113] Thanarat Horprasert, David Harwood, and Larry S. Davis. A statistical approach for real-time robust background subtraction and shadow detection. In *ICCV'99 Frame-Rate Workshop*, pages 1–19, 1999.

[114] Frederik Huguet and Frederic Devernay. A variational method for scene flow estimation from stereo sequences. In *ICCV*, pages 1–7. IEEE, 2007.

[115] Katsushi Ikeuchi and Kosuke Sato. Determining reflectance properties of an object using range and brightness images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(11):1139–1153, 1991.

[116] Martin Jagersand, Neil Birkbeck, and Dana Cobzas. A three-tier hierarchical model for capturing and rendering of 3d geometry and appearance from 2d images. In *3DPVT'08*, 2008.

[117] D.L. James and D.K. Pai. DyRT: dynamic response textures for real time deformation simulation with graphics hardware. *ACM Trans. Graph.*, 21(3):582–585, 2002.

[118] Doug L. James and Kayvon Fatahalian. Precomputing interactive dynamic deformable scenes. *ACM Trans. Graph.*, 22:879–887, 2003.

[119] Doug L. James and Christopher D. Twigg. Skinning mesh animations. *ACM Trans. Graph.*, 24(3):399–407, July 2005.

[120] Zsolt Janko and Jean-Philippe Pons. Spatio-temporal image-based texture atlases for dynamic 3-d models. In *IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM '09)*, pages 1646 –1653, 2009.

[121] Hailin Jin, Stefano Soatto, and Anthony J. Yezzi. Stereoscopic shading: Integrating multi-frame shape cues in a variational framework. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, volume 1, pages 1169–1176, 2000.

[122] Hailin Jin, Stefano Soatto, and Anthony J. Yezzi. Multi-view stereo beyond lambert. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 171–178, June 2003.

[123] Hailin Jin, Anthony J. Yezzi, and Stefano Soatto. Variational multiframe stereo in the presence of specular reflections. In *3DPVT'02*, pages 626–631, 2002.

[124] Shanon X. Ju, Michael J. Black, and Yaser Yacoob. Cardboard people: A parameterized model of articulated image motion. In *FG '96: Proceedings of the 2nd International Conference on Automatic Face and Gesture Recognition (FG '96)*, page 38, Washington, DC, USA, 1996. IEEE Computer Society.

[125] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Skinning with dual quaternions. In *I3D '07: Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 39–46, New York, NY, USA, 2007. ACM.

[126] Ladislav Kavan, Peter-Pike Sloan, and Carol O'Sullivan. Fast and Efficient Skinning of Animated Meshes. *Computer Graphics Forum (Eurographics 2010)*, 29(2):327–336, 2010.

[127] Ladislav Kavan and Jiří Žára. Spherical blend skinning: a real-time deformation of articulated models. In *I3D '05: Proceedings of the 2005 symposium on Interactive 3D graphics and games*, pages 9–16, New York, NY, USA, 2005. ACM.

[128] Roland Kehl, Matthieu Bray, and Luc Van Gool. Full body tracking from multiple views using stochastic sampling. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*, pages 129–136, Washington, DC, USA, 2005. IEEE Computer Society.

[129] Hyeongwoo Kim, Bennett Wilburn, and Moshe Ben-Ezra. Photometric stereo for dynamic surface orientations. In *European Conference on Computer Vision (ECCV '10) - Part I*, pages 59–72, 2010.

[130] T.Y. Kim and E. Vendrovsky. DrivenShape: a data-driven approach for shape deformation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 49–55. Eurographics Association, 2008.

[131] K. Kolev and D. Cremers. Integration of multiview stereo and silhouettes via convex functionals on convex domains. In *European Conference on Computer Vision (ECCV '08)*, pages 752–765, Marseille, France, October 2008.

[132] K. Kolev, T. Pock, and D. Cremers. Anisotropic minimal surfaces integrating photoconsistency and normal information for multiview stereo. In *European Conference on Computer Vision (ECCV '10)*, pages III: 538–551, 2010.

[133] Koji Komatsu. Human skin model capable of natural shape variation. *The Visual Computer*, 3(5):265–271, 1988.

[134] Paul G. Kry, Doug James, and Dinesh Pai. Eigenskin: Real time large deformation character skinning in hardware. In *Proceedings of the ACM SIGGRAPH Symposium on Computer Animation*, pages 153 – 160, July 2002.

[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, pages 355–363, Aire-la-Ville, Switzerland, Switzerland, 2004. Eurographics Association.

[136] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. *Int. J. Comput. Vision*, 38(3):199–218, 2000.

[137] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 117–126, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[138] Caroline Larboulette and Marie-Paule Cani. Real-time dynamic wrinkles. In *Computer Graphics International 2004 (CGI '04)*, pages 522–525, Crète, Grèce, June 2004. IEEE.

[139] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(2):150–162, 1994.

[140] C. Lei, X. Chen, and Y.H. Yang. A new multiview spacetime-consistent depth recovery framework for free viewpoint video rendering. In *IEEE International Conference on Computer Vision (ICCV '09)*, 2009.

[141] Victor Lempitsky and Denis Ivanov. Seamless mosaicing of image-based texture maps. pages 1–6, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[142] Jerome Edward Lengyel. Compression of time-dependent geometry. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, I3D '99, pages 89–95, New York, NY, USA, 1999. ACM.

[143] Hendrik P. A. Lensch, Michael Goesele, Jan Kautz, Wolfgang Heidrich, and Hans-Peter Seidel. Image-based reconstruction of spatially varying materials. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 103–114, London, UK, 2001. Springer-Verlag.

[144] Hendrik P.A. Lensch, Wolfgang Heidrich, and Hans-Peter Seidel. A silhouette-based algorithm for texture registration and stitching. *Graph. Models*, 63:245–262, July 2001.

[145] A. Levin, D. Lischinski, and Y. Weiss. A closed-form solution to natural image matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, pages 228–242, 2007.

[146] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH '96: Proceedings of the 23th annual conference on Computer graphics and interactive techniques*, pages 31–42, New York, NY, USA, 1996. ACM Press.

[147] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 362–371, New York, NY, USA, 2002. ACM Press.

[148] J. P. Lewis, Matt Cordner, and Nickson Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 165–172, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[149] Rui Li and Stan Sclaroff. Multi-scale 3d scene flow from binocular stereo sequences. *Comput. Vis. Image Underst.*, 110(1):75–90, 2008.

[150] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 163–169, New York, NY, USA, 1987. ACM Press.

[151] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60:91–110, November 2004.

[152] Nadia Magnenat-Thalmann, Hyewon Seo, and Frederic Cordier. Automatic modeling of animatable virtual humans - a survey. In *4th International Conference on 3D Digital Imaging and Modeling (3DIM '03)*, page 2, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[153] Nadia Magnenat-Thalmann and Daniel Thalmann. *Human body deformations using joint-dependent local operators and finite-element theory*, pages 243–262. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[154] Marcus Magnor and Bastian Goldlucke. Spacetime-coherent geometry reconstruction from multiple video streams. In *3DPVT '04: Proceedings of the 3D Data Processing, Visualization, and Transmission, 2nd International Symposium*, pages 365–372, Washington, DC, USA, 2004. IEEE Computer Society.

[155] Russell A. Manning and Charles R. Dyer. Interpolating view and scene motion by dynamic view morphing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, volume 1, pages 388–394, 1999.

[156] A. Marquina and S.J. Osher. Image super-resolution by TV-regularization and Bregman iteration. *Journal of Scientific Computing*, 37(3):367–382, 2008.

[157] Lucie Masson, Michel Dhome, and Frederic Jurie. Robust real time tracking of 3d objects. In *ICPR '04: Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04) Volume 4*, pages 252–255, Washington, DC, USA, 2004. IEEE Computer Society.

[158] Y. Matsumoto, K. Fujimura, and T. Kitamura. Shape-from-silhouette/stereo and its application to 3-d digitizer. In *DCGI '99: Proceedings of the 8th International Conference on Discrete Geometry for Computer Imagery*, pages 177–190, London, UK, 1999. Springer-Verlag.

[159] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 115–126, London, UK, 2001. Springer-Verlag.

[160] Wojciech Matusik, Chris Buehler, Ramesh Raskar, Steven J. Gortler, and Leonard McMillan. Image-based visual hulls. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[161] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic brdf measurement. In *EGRW '03: Proceedings of the 14th Eurographics workshop on Rendering*, pages 241–247. Eurographics Association, 2003.

[162] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. *ACM Trans. Graph.*, 21(3):427–437, 2002.

[163] Brice Michoud, Erwan Guillou, and Saida Bouakaz. Real-time and Markerless Full-Body Human Motion Capture. In *Groupe de Travail Animation et Simulation 2007 (GTAS'07)*, June 2007.

[164] G. Miller, J. Starck, and A. Hilton. Projective surface refinement for free-viewpoint video. *European Conference on Visual Media Production (CVMP)*, 2006.

[165] Minpack. http://www.netlib.org/minpack/.

[166] J. Mitchelson and A. Hilton. Wand-based calibration of multiple cameras. In *British Machine Vision Association workshop on Multiple Views*, May 2002.

[167] Thomas B. Moeslund and Erik Granum. Multiple cues used in model-based human motion capture. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, page 362, Washington, DC, USA, 2000. IEEE Computer Society.

[168] Alex Mohr and Michael Gleicher. Building efficient, accurate character skins from examples. *ACM Trans. Graph.*, 22(3):562–568, 2003.

[169] M. Müller. Hierarchical position based dynamics. In *Proceedings of Virtual Reality Interactions and Physical Simulations (VRIPhys2008)*, November 2008.

[170] Matthias Müller and Nuttapong Chentanez. Wrinkle meshes. In *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '10, pages 85–92, Aire-la-Ville, Switzerland, Switzerland, 2010. Eurographics Association.

[171] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson. Physically based deformable models in computer graphics. *Computer Graphics Forum*, 25(4):809–836, 2006.

[172] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In *Proceedings of ACM GRAPHITE*, pages 381–389, 2006.

[173] Tal Nir, Alfred M. Bruckstein, and Ron Kimmel. Over-parameterized variational optical flow. *Int. J. Comput. Vision*, 76:205–216, February 2008.

[174] K. Nishino, Yoichi Sato, and Katsushi Ikeuchi. Eigen-texture method: appearance compression based on 3d model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '99)*, volume 1, pages 618 – 624, June 1999.

[175] K. Ogawara, Xiaolu Li, and K. Ikeuchi. Marker-less human motion estimation using articulated deformable model. *Robotics and Automation, 2007 IEEE International Conference on*, pages 46–51, 10-14 April 2007.

[176] Organic motion. http://www.organicmotion.com.

[177] Stanley Osher and James A Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[178] Hyun Soo Park, Takaaki Shiratori, Iain Matthews, and Yaser Ajmal Sheikh. 3d reconstruction of a moving point from a series of 2d projections. In *European Conference on Computer Vision (ECCV '10)*, September 2010.

[179] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Trans. Graph.*, 25(3):881–889, 2006.

[180] Sang Il Park and Jessica K. Hodgins. Data-driven modeling of skin and muscle deformation. In *ACM SIGGRAPH 2008 papers*, SIGGRAPH '08, pages 96:1–96:6, New York, NY, USA, 2008. ACM.

[181] Bui Tuong Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[182] Frederic Pighin, Richard Szeliski, and David H. Salesin. Resynthesizing facial animation through 3d model-based tracking. In *IEEE International Conference on Computer Vision (ICCV '99)*, pages 143 – 150 vol.1, Los Alamitos, CA, USA, 1999. IEEE Computer Society.

[183] Frederic H. Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 75–84, 1998.

[184] Ralf Plänkers and Pascal Fua. Articulated soft objects for multiview shape and motion capture. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1182–1187, 2003.

[185] Mark Pollefeys, R. Koch, and L. Van Gool. A simple and efficient rectification method for general motion. In *IEEE International Conference on Computer Vision (ICCV '99)*, pages 496–501, 1999.

[186] Jean-Philippe Pons, Renaud Keriven, and Olivier Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, pages 822–827, 2005.

[187] Tiberiu Popa, Qingnan Zhou, Derek Bradley, Vladislav Kraevoy, Hongbo Fu, Alla Sheffer, and Wolfgang Heidrich. Wrinkling captured garments using space-time data-driven deformation. *Computer Graphics Forum (Proc. Eurographics)*, 28(2), 2009.

[188] Ronald Poppe. Vision-based human motion analysis: An overview. *Comput. Vis. Image Underst.*, 108:4–18, October 2007.

[189] Victor Prisacariu and Ian Reid. Pwp3d: Real-time segmentation and tracking of 3d objects. In *British Machine Vision Conference (BMVC '09)*, September 2009.

[190] David Pritchard and Wolfgang Heidrich. Cloth motion capture. In *SIGGRAPH '03: ACM SIGGRAPH 2003 Sketches & Applications*, pages 1–1, New York, NY, USA, 2003. ACM.

[191] PTI phoenix technologies incorporated. http://www.ptiphoenix.com/.

[192] T. Rhee, JP Lewis, and U. Neumann. Real-Time Weighted Pose-Space Deformation on the GPU. *Computer Graphics Forum*, 25(3):439–448, 2006.

[193] C. Rocchini, P. Cignoni, F. Ganovelli, C. Montani, P. Pingi, and R. Scopigno. Marching Intersections: An efficient resampling algorithm for surface management. In *SMI '01: Proceedings of the International Conference on Shape Modeling & Applications*, page 296, Washington, DC, USA, 2001. IEEE Computer Society.

[194] Sami Romdhani and Thomas Vetter. Efficient, robust and accurate fitting of a 3d morphable model. In *IEEE International Conference on Computer Vision (ICCV '03)*, page 59, Washington, DC, USA, 2003. IEEE Computer Society.

[195] B. Rosenhahn, T. Brox, and J. Weickert. Three-dimensional shape knowledge for joint image segmentation and pose tracking. *International Journal of Computer Vision*, 73(3):243–262, July 2007.

[196] B. Rosenhahn, C. Schmaltz, T. Brox, J. Weickert, D. Cremers, and H.-P. Seidel. Markerless motion capture of man-machine interaction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pages 1–8, June 2008.

[197] Bodo Rosenhahn, Thomas Brox, Daniel Cremers, and Hans-Peter Seidel. Online smoothing for markerless motion capture. In Fred Hamprecht, Christoph Schnörr, and Bernd Jaehne, editors, *Pattern Recognition*, volume 4713 of *Lecture Notes in Computer Science*, pages 163–172, Heidelberg, Germany, 2007. Springer.

[198] Bodo Rosenhahn, Uwe G. Kersting, Katie Powell, and Hans-Peter Seidel. Cloth x-ray: Mocap of people wearing textiles. In Katrin Franke, Klaus-Robert Mller, Bertram Nickolay, and Ralf Schfer, editors, *DAGM-Symposium*, volume 4174 of *Lecture Notes in Computer Science*, pages 495–504. Springer, 2006.

[199] Bodo Rosenhahn, Reinhard Klette, and Gerald Sommer. Silhouette based human motion estimation. In Carl Edward Rasmussen, Heinrich H. Bülthoff, Bernhard Schölkopf, and Martin A. Giese, editors, *DAGM-Symposium*, volume 3175 of *Lecture Notes in Computer Science*, pages 294–301. Springer, 2004.

[200] Hideo Saito, Kazuko Omata, and Shinji Ozawa. Recovery of shape and surface reflectance of specular object from rotation of light source. In *2nd International Conference on 3D Digital Imaging and Modeling (3DIM '99)*, pages 526–535. IEEE Computer Society, 1999.

[201] Mathieu Salzmann, Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Closed-form solution to non-rigid 3d surface registration. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (4)*, volume 5305 of *Lecture Notes in Computer Science*, pages 581–594. Springer, 2008.

[202] Mathieu Salzmann, Julien Pilet, Slobodan Ilic, and Pascal Fua. Surface deformation models for nonrigid 3d shape recovery. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29:1481–1487, August 2007.

[203] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. *ACM Trans. Graph.*, 22(3):578–586, 2003.

[204] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 379–387, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[205] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int. J. Comput. Vision*, 47(1-3):7–42, 2002.

[206] Hartmut Schirmacher, Wolfgang Heidrich, Martin Rubick, Detlef Schiron, and Hans-Peter Seidel. Image-based BRDF reconstruction. In *Proceedings of the 4th Conference on Vision, Modeling, and Visualization (VMV-99)*, pages 285–292, November 1999.

[207] Christian Schmaltz, Bodo Rosenhahn, Thomas Brox, Daniel Cremers, Joachim Weickert, Lennart Wietzke, and Gerald Sommer. Region-based pose tracking. In *IbPRIA (2)'07*, pages 56–63, 2007.

[208] A. Schödl, R. Szeliski, D.H. Salesin, and I. Essa. Video textures. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498. ACM Press/Addison-Wesley Publishing Co., 2000.

[209] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor. Garment motion capture using color-coded patterns. *Computer Graphics Forum (Proc. Eurographics EG'05)*, 24(3):439–448, August 2005.

[210] Steven M. Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '06)*, pages 519–528, Washington, DC, USA, 2006. IEEE Computer Society.

[211] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 1067, Washington, DC, USA, 1997. IEEE Computer Society.

[212] Hyewon Seo and Nadia Magnenat-Thalmann. An automatic modeling of human bodies from sizing parameters. In *I3D '03: Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 19–26, New York, NY, USA, 2003. ACM.

[213] Hyewon Seo and Nadia Magnenat-Thalmann. An example-based approach to human body manipulation. *Graph. Models*, 66(1):1–23, 2004.

[214] Wolfgang Sepp. A direct method for real-time tracking in 3-d under variable illumination. In Walter G. Kropatsch, Robert Sablatnig, and Allan Hanbury, editors, *DAGM-Symposium*, volume 3663 of *Lecture Notes in Computer Science*, pages 246–253. Springer, 2005.

[215] Wolfgang Sepp. Efficient tracking in 6-dof based on the image-constancy assumption in 3-d. In *ICPR '06: Proceedings of the 18th International Conference on Pattern Recognition*, pages 59–62, Washington, DC, USA, 2006. IEEE Computer Society.

[216] Xiaohan Shi, Kun Zhou, Yiying Tong, Mathieu Desbrun, Hujun Bao, and Baining Guo. Example-based dynamic skinning in real time. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '08, pages 1–8, New York, NY, USA, 2008. ACM.

[217] Hedvig Sidenbladh, Michael J. Black, and David J. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *ECCV (2)*, pages 702–718, 2000.

[218] Hedvig Sidenbladh, Michael J. Black, and Leonid Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 784–800, London, UK, 2002. Springer-Verlag.

[219] Hedvig Sidenbladh, Fernando de la Torre, and Michael J. Black. A framework for modeling the appearance of 3d articulated figures. In *Automatic Face and Gesture Recognition (FG '00)*, page 368, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[220] S. Sinha and M. Pollefeys. Camera network calibration from dynamic silhouettes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '04)*, pages I–195 – I–202 Vol.1, 2004.

[221] Peter-Pike J. Sloan, III Charles F. Rose, and Michael F. Cohen. Shape by example. In *I3D '01: Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 135–143, New York, NY, USA, 2001. ACM.

[222] Cristian Sminchisescu and Bill Triggs. Estimating articulated human motion with covariance scaled sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003.

[223] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM SIGGRAPH 2006 Papers*, SIGGRAPH '06, pages 835–846, New York, NY, USA, 2006. ACM.

[224] Stefano Soatto, Anthony J. Yezzi, and Hailin Jin. Tales of shape and radiance in multi-view stereo. In *IEEE International Conference on Computer Vision (ICCV '03)*, pages 974–981, 2003.

[225] J. Starck, A. Hilton, and J. Illingworth. Human shape estimation in a multi-camera studio. *British Machine Vision Conference (BMVC)*, pages 573–582, 2001.

[226] J. Starck, Adrian Hilton, and John Illingworth. Reconstruction of animated models from images using constrained deformable surfaces. In *DGCI '02: Proceedings of the 10th International Conference on Discrete Geometry for Computer Imagery*, pages 382–391, London, UK, 2002. Springer-Verlag.

[227] J. Starck, A. Maki, S. Nobuhara, A. Hilton, and T. Matsuyama. The multiple-camera 3-d production studio. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(6):856–869, June 2009.

[228] J. Starck, G. Miller, and A. Hilton. Video-based character animation. In *SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 49–58, New York, NY, USA, 2005. ACM.

[229] J. Starck, G. Miller, and A. Hilton. Volumetric stereo with silhouette and feature constraints. *British Machine Vision Conference (BMVC)*, 3:1189–1198, 2006.

[230] Jonathan Starck and Adrian Hilton. Model-based multiple view reconstruction of people. In *IEEE International Conference on Computer Vision (ICCV '03)*, volume 2, page 915, Los Alamitos, CA, USA, 2003. IEEE Computer Society.

[231] Jonathan Starck and Adrian Hilton. Towards a 3d virtual studio for human appearance capture. In *VVG'03*, pages 17–24, 2003.

[232] Jonathan Starck and Adrian Hilton. Spherical matching for temporal correspondence of non-rigid surfaces. In *IEEE International Conference on Computer Vision (ICCV '05)- Volume 2*, ICCV '05, pages 1387–1394, Washington, DC, USA, 2005. IEEE Computer Society.

[233] Jonathan Starck and Adrian Hilton. Correspondence labelling for wide-timeframe free-form surface matching. In *IEEE International Conference on Computer Vision (ICCV '07)*, pages 1–8, 2007.

[234] Jonathan Starck and Adrian Hilton. Surface capture for performance-based animation. *IEEE Comput. Graph. Appl.*, 27(3):21–31, 2007.

[235] Carsten Stoll, Juergen Gall, Edilson de Aguiar, Sebastian Thrun, and Christian Theobalt. Video-based reconstruction of animatable human characters. *ACM Trans. Graph.*, 29:139:1–139:10, December 2010.

[236] Narayanan Sundaram, Thomas Brox, and Kurt Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *European Conference on Computer Vision (ECCV '10) - Part I*, pages 438–451, 2010.

[237] A. Sundaresan and R. Chellappa. Multi-camera tracking of articulated human motion using motion and shape cues. In *ACCV06*, pages II:131–140, 2006.

[238] Martin Sunkel, Bodo Rosenhahn, and Hans-Peter Seidel. Silhouette based generic model adaptation for marker-less motion capturing. In Elgammal et al. [81], pages 119–135.

[239] Richard Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Underst.*, 58(1):23–32, 1993.

[240] Marco Tarini, Marco Callieri, Claudio Montani, Claudio Rocchini, Karin Olsson, and Therese Persson. Marching Intersections: An efficient approach to shape-from-silhouette. In *Proceedings of the Conference on Vision, Modeling, and Visualization (VMV 2002)*, pages 255–262, November 2002.

[241] Franco Tecchia and Yiorgos Chrysanthou. Real-time rendering of densely populated urban environments. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 83–88, London, UK, 2000. Springer-Verlag.

[242] C. Theobalt, J. Carranza, M. Magnor, and H.-P. Seidel. A parallel framework for silhouette-based human motion capture. *Proc. Vision, Modeling, and Visualization (VMV-2003),* Munich, Germany, pages 207–214, November 2003.

[243] Christian Theobalt, Naveed Ahmed, Hendrik Lensch, Marcus Magnor, and Hans-Peter Seidel. Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):663–674, 2007.

[244] Christian Theobalt, Joel Carranza, Marcus A. Magnor, and Hans-Peter Seidel. Combining 3d flow fields with silhouette-based human motion capture for immersive video. *Graphical Models*, 66(6):333–351, 2004.

[245] Christian Theobalt, Edilson de Aguiar, Marcus Magnor, and Hans-Peter Seidel. *Reconstructing Human Shape, Motion and Appearance from Multi-view Video*, chapter Reconstructing Human Shape, Motion and Appearance from Multi-view Video, pages 29–58. Springer, Heidelberg, Germany, November 2007.

[246] Kenneth E. Torrance and E.M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Journal of the Optical Society of America*, 57(9):1105–1114, September 1967.

[247] Lorenzo Torresani and Christoph Bregler. Space-time tracking. In *European Conference on Computer Vision (ECCV '02)*, pages 801–812, 2002.

[248] Lorenzo Torresani, Aaron Hertzmann, and Christoph Bregler. Learning non-rigid 3d shape from 2d motion. In *NIPS*, 2003.

[249] P. Tresadern and I. Reid. Articulated structure from motion by factorization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05)*, 2005.

[250] Tony Tung, Shohei Nobuhara, and Takashi Matsuyama. Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo. In *IEEE International Conference on Computer Vision (ICCV '09)*, pages 1709–1716, 2009.

[251] S. Ullman. Maximizing rigidity: the incremental recovery of 3-d structure from rigid and nonrigid motion. *Perception*, 13(3):255–274, 1984.

[252] Raquel Urtasun, David J. Fleet, Aaron Hertzmann, and Pascal Fua. Priors for people tracking from small training sets. In *ICCV '05: Proceedings of the Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 403–410, Washington, DC, USA, 2005. IEEE Computer Society.

[253] Kiran Varanasi, Andrei Zaharescu, Edmond Boyer, and Radu P. Horaud. Temporal surface tracking using mesh evolution. In *European Conference on Computer Vision (ECCV '08)*, volume Part II of *LNCS*, pages 30–43, Marseille, France, October 2008. Springer-Verlag.

[254] M. Alex O. Vasilescu and Demetri Terzopoulos. TensorTextures: multilinear image-based rendering. *ACM Trans. Graph.*, 23(3):336–342, 2004.

[255] Sundar Vedula, Simon Baker, and Takeo Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Trans. Graph.*, 24(2):240 – 261, April 2005.

[256] Sundar Vedula, Simon Baker, Peter Rander, Robert Collins, and Takeo Kanade. Three-dimensional scene flow. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):475–480, 2005.

[257] Sundar Vedula, Simon Baker, Steven Seitz, and Takeo Kanade. Shape and motion carving in 6d. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '00)*, June 2000.

[258] Vicon. http://www.vicon.com/.

[259] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Trans. Graph.*, 27(3):1–9, 2008.

[260] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 28(5):174:1–174:11, December 2009.

[261] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '05) - Volume 2*, pages 391–398, Washington, DC, USA, 2005. IEEE Computer Society.

[262] George Vogiatzis, Philip Torr, S.M. Seitz, and Roberto Cipolla. Reconstructing relief surfaces. In *British Machine Vision Conference (BMVC '04)*, pages 117–126, 2004.

[263] Michael Wand, Bart Adams, Maksim Ovsjanikov, Alexander Berner, Martin Bokeloh, Philipp Jenke, Leonidas Guibas, Hans-Peter Seidel, and Andreas Schilling. Efficient reconstruction of nonrigid shape and motion from real-time 3d scanner data. *ACM Trans. Graph.*, 28(2):1–15, 2009.

[264] Michael Wand, Philipp Jenke, Qixing Huang, Martin Bokeloh, Leonidas Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing*, pages 49–58, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[265] Huamin Wang, , Ravi Ramamoorthi, and James F. O'Brien. Data-driven elastic models for cloth: Modeling and measurement. *ACM Trans. Graph.*, 30(4):71:1–11, July 2011. Proceedings of ACM SIGGRAPH 2011, Vancouver, BC Canada.

[266] Huamin Wang, Florian Hecht, Ravi Ramamoorthi, and James F. O'Brien. Example-based wrinkle synthesis for clothing animation. In *ACM SIGGRAPH 2008 Papers*, SIGGRAPH '10, pages 107:1–8, July 2010.

[267] Xiaohuan Corina Wang and Cary Phillips. Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 129–138, New York, NY, USA, 2002. ACM.

[268] Martin Weber, Andrew Blake, and Roberto Cipolla. Towards a complete dense geometric and photometric reconstruction under varying pose and illumination. In *British Machine Vision Conference (BMVC '02)*, 2002.

[269] O. Weber, O. Sorkine, Y. Lipman, and C. Gotsman. Context-Aware Skeletal Shape Deformation. *Computer Graphics Forum*, 26(3):265–274, 2007.

[270] Andreas Wedel, Clemens Rabe, Tobi Vaudrey, Thomas Brox, Uwe Franke, and Daniel Cremers. Efficient dense scene flow from sparse or dense stereo data. In *European Conference on Computer Vision (ECCV '08)*, pages 739–751, Berlin, Heidelberg, 2008. Springer-Verlag.

[271] Sebastian Weik. A passive full body scanner using shape from silhouettes. In *Pattern Recognition, International Conference on*, volume 1, page 1750, Los Alamitos, CA, USA, 2000. IEEE Computer Society.

[272] Daniel Weinland, Remi Ronfard, and Edmond Boyer. Free viewpoint action recognition using motion history volumes. *Comput. Vis. Image Underst.*, 104:249–257, November 2006.

[273] Ryan White, Keenan Crane, and D. A. Forsyth. Capturing and animating occluded cloth. *ACM Trans. Graph.*, 26, July 2007.

[274] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[275] Robert J. Woodham. Photometric method for determining surface orientation from multiple images. *Optical Engineering*, 19(1):139–144, 1980.

[276] J. Xiao, C. Rao, and M. Shah. View interpolation for dynamic scenes. In *Proc. EUROGRAPHICS'02*, pages 153–162, 2002.

[277] Yilei Xu and Amit Roy-Chowdhury. Inverse compositional estimation of 3d pose and lighting in dynamic scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30:1300–1307, 2008.

[278] Jingyu Yan and Marc Pollefeys. Articulated motion segmentation using RANSAC with priors. In *ICCV Workshop on Dynamical Vision*, 2005.

[279] Ruigang Yang, Marc Pollefeys, and Greg Welch. Dealing with textureless regions and specular highlights-a progressive space carving scheme using a novel photo-consistency measure. In *IEEE International Conference on Computer Vision (ICCV '03)*, page 576, Washington, DC, USA, 2003. IEEE Computer Society.

[280] Anthony J. Yezzi and Stefano Soatto. Stereoscopic segmentation. In *IEEE International Conference on Computer Vision (ICCV '01)*, pages 59–66, 2001.

[281] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L1 range image integration. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07)*, pages 1–8. IEEE, 2007.

[282] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao. Recovering consistent video depth maps via bundle optimization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pages 1–8, 2008.

[283] Huaifeng Zhang, Jan Cech, Radim Sara, Fuchao Wu, and Zhanyi Hu. A linear trinocular rectification method for accurate stereoscopic matching. In *British Machine Vision Conference (BMVC '03)*, pages 281–290, 2003.

[284] Li Zhang, Brian Curless, and Steven M. Seitz. Spacetime stereo: Shape recovery for dynamic scenes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 367–374, June 2003.

[285] Ye Zhang and Chandra Kambhamettu. Integrated 3d scene flow and structure recovery from multiview image sequences. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2674, 2000.

[286] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *Int. J. Comput. Vision*, 13:119–152, October 1994.

[287] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *IEEE International Conference on Computer Vision (ICCV '99)*, pages 666–673, September 1999.

[288] Ciyou Zhu, Richard H. Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23:550–560, December 1997.

[289] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 600–608, New York, NY, USA, 2004. ACM.

A OpenGL Projection Matrix

When synthesizing images (e.g., for tracking) and performing projective texture mapping in OpenGL, the OpenGL camera matrix needs to be consistent with our pinhole camera parameterization. In this section, the OpenGL projection and modelview matrix corresponding to a camera with projection matrix $\mathbf{P} = \mathbf{K}\mathbb{I}_{3\times 4}\mathbf{E}$ are derived (Eq. 2.5).

OpenGL performs the perspective projection in 3D homogeneous coordinates. The OpenGL projection matrix takes on the following form:

$$\begin{bmatrix} X_D \\ Y_D \\ Z_D \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{\hat{X}}{\hat{W}} \\ \frac{\hat{Y}}{\hat{W}} \\ \frac{\hat{Z}}{\hat{W}} \\ 1 \end{bmatrix} \simeq \begin{bmatrix} \hat{X} \\ \hat{Y} \\ \hat{Z} \\ \hat{W} \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & 0 \\ p_{21} & p_{22} & p_{23} & 0 \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & p_{43} & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{A.1}$$

Where $[X_D, Y_D, Z_D, 1]^T$ are the normalized device coordinates of a point, and $[X, Y, Z, 1]$ is a point in the eye coordinate system (e.g., after transformation by the modelview matrix). The normalized device coordinates lie in the canonical 3D cube $[-1, 1] \times [-1, 1] \times [-1, 1]$ (see Figure A.1).

The above formulation still performs a perspective division by the input Z point. Therefore,

$$\hat{W} = Z \rightarrow p_{43} = 1. \tag{A.2}$$

Also, the minimum, z_{mn}, and maximum, z_{mx}, depth values should map to -1 and 1 respectively:

$$\frac{p_{33}z_{mn} + p_{34}}{z_{mn}} = -1 \text{ and} \tag{A.3}$$

$$\frac{p_{33}z_{mx} + p_{34}}{z_{mx}} = 1, \tag{A.4}$$

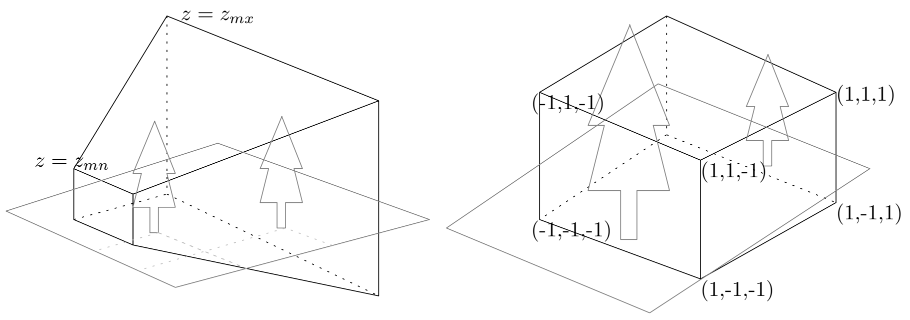


Figure A.1: In OpenGL, the view-frustum (left) is transformed into normalized device coordinates (right) through the perspective transformation.

which gives

$$p_{33} = \frac{z_{mn} + z_{mx}}{z_{mx} - z_{mn}} \text{ and} \tag{A.5}$$

$$p_{34} = 2\frac{z_{mn} z_{mx}}{z_{mx} - z_{mn}}. \tag{A.6}$$

The normalized point coordinates are transformed to window coordinates, X_W, Y_W, Z_W, where the measure is now pixel values and positively increasing depth values in the range of $[0, 1]$. The mapping to window coordinates takes the width, w, and height, h, of the viewport (or image window). The x component is

$$\begin{aligned} X_w &= \left(\frac{\hat{X}}{\hat{Z}} + 1\right)\frac{w}{2} & \text{(A.7)} \\ &= \left(\frac{p_{11}X + p_{12}Y + p_{13}Z}{Z}\right)\frac{w}{2} + \frac{w}{2} & \text{(A.8)} \\ &= \left(p_{11}\frac{w}{2}\right)\frac{X}{Z} + \left(p_{12}\frac{w}{2}\right)\frac{Y}{Z} + \left(p_{13}\frac{w}{2} + \frac{w}{2}\right) & \text{(A.9)} \\ &= f_x\frac{X}{Z} + \zeta\frac{Y}{Z} + p_x. & \text{(A.10)} \end{aligned}$$

The final two equations give the relationship between the GL matrix and the camera internals, $\mathbf{K}$. The GL matrix elements can be solved independently, giving

$$p_{11} = 2\frac{f_x}{w},\ p_{12} = 2\frac{\zeta}{w}, \text{ and } p_{13} = 2\frac{p_x}{w} - 1. \tag{A.11}$$

Similar expansion of the y component gives

$$p_{21} = 0,\ p_{22} = 2\frac{f_y}{h}, \text{ and } p_{23} = 2\frac{p_y}{h} - 1. \tag{A.12}$$

The above equations give an OpenGL projection matrix that can be used with the perspective camera parameters. When the modelview matrix is set to the camera externals, $\mathbf{E}$, OpenGL can be used to render from the camera viewpoint.[1] However, this formulation will give an image where the $(0, 0)$ pixel is at the bottom left hand side of the screen—meaning that the image is flipped vertically. This formulation is fine when the rendered image is being read back with a glReadPixels call.

If the image is to be displayed on screen through OpenGL, the above formulation should be changed by mapping $p_{43} = -1$ and deriving the other elements. This will give a projection matrix that expects the scene to be along the negative z direction of the camera. For proper rendering, the OpenGL modelview matrix should be set to $\mathbf{R}_x(180°)\mathbf{E}$, where $\mathbf{R}_x$ is a rotation around the x axis.

[1] Similarly, the derived projection matrix and externals can be used to build a texture matrix for projective texturing.

B Linearization and Solution of the Monoflow Euler-Lagrange Equations

Our solution to Eqs 6.11-6.14 follows the derivation for optic flow [45]. First, a fixed point iteration is defined:

$$\begin{aligned}
\sum_{i\in\mathcal{N}(j)} \Psi'_d((I_{ij}^{k+1})^2)I_{ij}^{k+1}I_{id}^{k} - \alpha\nabla\cdot(\Psi'_s(|\nabla d^{k+1}|^2)\nabla d^{k+1}) = 0\\
\sum_{i\in\mathcal{N}(j)} \Psi'_d((I_{ij}^{k+1})^2)I_{ij}^{k+1}I_{iu}^{k} - \beta\nabla\cdot(\Psi'_u(|\nabla \mathbf{o}^{k+1}|^2)\nabla u^{k+1}) = 0\\
\sum_{i\in\mathcal{N}(j)} \Psi'_d((I_{ij}^{k+1})^2)I_{ij}^{k+1}I_{iv}^{k} - \beta\nabla\cdot(\Psi'_u(|\nabla \mathbf{o}^{k+1}|^2)\nabla v^{k+1}) = 0\\
\sum_{i\in\mathcal{N}(j)} \Psi'_d((I_{ij}^{k+1})^2)I_{ij}^{k+1}I_{iw}^{k} - \beta\nabla\cdot(\Psi'_u(|\nabla \mathbf{o}^{k+1}|^2)\nabla w^{k+1}) = 0\\
|\nabla\mathbf{o}^{k+1}|^2 := |\nabla u^{k+1}|^2 + |\nabla v^{k+1}|^2 + |\nabla w^{k+1}|^2.
\end{aligned}$$

Letting $d^{k+1} = d^k + \delta d^k$, $u^{k+1} = u^k + \delta u^k$, $v^{k+1} = v^k + \delta v^k$, and, $w^{k+1} = w^k + \delta w^k$, the data term can be linearized as

$$\begin{aligned}
I_{ij}^{k+1} =& I_i(\underset{j\to i}{W}(\mathbf{d}^{k+1})) - I_j\\
\approx& I_i(\underset{j\to i}{W}(\mathbf{d}^{k})) - I_j +\\
& I_{id}\delta d^k + I_{iu}\delta u^k + I_{iv}\delta v^k + I_{iw}\delta w^k\\
=& I_{ij}^k + I_{id}\delta d^k + I_{iu}\delta u^k + I_{iv}\delta v^k + I_{iw}\delta w^k
\end{aligned}$$

Letting

$$\nabla I_i^k = [I_{id}^k\ I_{iu}^k\ I_{iv}^k\ I_{iw}^k\ I_{ij}^k]^T,$$

and

$$\delta\mathbf{d}^k = [\delta d^k\ \delta u^k\ \delta v^k\ \delta w^k\ 1]^T,$$

we have $I_{ij}^{k+1} \approx (\nabla I_i^k)^T \delta \mathbf{d}^k$. Defining, $S_i^k = \nabla I_i^k (\nabla I_i^k)^T$, the Euler-Lagrange equations become:

$$
\begin{aligned}
&\sum_{i \in \mathcal{N}(j)} \Psi_d'(\cdot)[S_i^k]_1 \delta \mathbf{d}^k - \alpha \nabla \cdot (\Psi_s'^k(\cdot) \nabla d^{k+1}) = 0 \\
&\sum_{i \in \mathcal{N}(j)} \Psi_d'(\cdot)[S_i^k]_2 \delta \mathbf{d}^k - \beta \nabla \cdot (\Psi_u'^k(\cdot) \nabla u^{k+1}) = 0 \\
&\sum_{i \in \mathcal{N}(j)} \Psi_d'(\cdot)[S_i^k]_3 \delta \mathbf{d}^k - \beta \nabla \cdot (\Psi_u'^k(\cdot) \nabla v^{k+1}) = 0 \\
&\sum_{i \in \mathcal{N}(j)} \Psi_d'(\cdot)[S_i^k]_4 \delta \mathbf{d}^k - \beta \nabla \cdot (\Psi_u'^k(\cdot) \nabla w^{k+1}) = 0 \\
&\Psi_d'^k(\cdot) := \Psi_d'((\delta \mathbf{d}^k)^T S_i^k \delta \mathbf{d}^k) \\
&\Psi_s'^k(\cdot) := \Psi_s'(|\nabla (d^k + \delta d^k)|^2) \\
&\Psi_u'^k(\cdot) := \Psi_u'(|\nabla u^{k+1}|^2 + |\nabla v^{k+1}|^2 + |\nabla w^{k+1}|^2)
\end{aligned}
$$

where $[S_i^k]_n$ is the n^{th} row of S_i^k. After discretization the above system is solved with geometric multi-grid methods [45].

C Linearization and Solution of the Basis Flow Euler-Lagrange Equations

Following optic flow [45], the Euler-Lagrange equations (Eqs 6.36 & 6.37) are solved with fixed point iterations:

$$\begin{aligned}\sum_{t=1}^{T}\sum_{j\neq i} w_{ijt}\Psi'((I_{ijt}^{h+1})^2)I_{ijt}^{h+1}\frac{\partial}{\partial d}I_{ijt}^{h}+ \\ \alpha\sum_{t=2}^{T} w_{it1}\Psi'((I_{it1}^{h+1})^2)I_{it1}^{h+1}\frac{\partial}{\partial d}I_{it1}^{h}- \\ \beta_1\nabla\cdot(\Psi'(|\nabla d^{h+1}|^2)\nabla d^{h+1}) = 0,\end{aligned} \tag{C.1}$$

$$\begin{aligned}\sum_{t=1}^{T}\sum_{j\neq i} w_{ijt}\Psi'((I_{ijt}^{h+1})^2)I_{ijt}^{h+1}\frac{\partial}{\partial \lambda_k}I_{ijt}^{h}+ \\ \alpha\sum_{t=2}^{T} w_{it1}\Psi'((I_{it1}^{h+1})^2)I_{it1}^{h+1}\frac{\partial}{\partial \lambda_k}I_{it1}^{h}- \\ \beta_2\nabla\cdot(\Psi'(|\nabla \mathbf{o}^{h+1}|^2)\sum_{c=1}^{3}\nabla o_c^{h+1}[\mathcal{B}_k(t)]_c) = 0.\end{aligned} \tag{C.2}$$

Defining the update, $d^{h+1} = d^h + \delta d^h$, $\lambda_k^{h+1} = \lambda_k^h + \delta\lambda_k^h$, $1 \leq k \leq H$, the data terms can now be linearized:

$$\begin{aligned}I_{ijt}^{h+1} =& I_{i,t}(\Pi_i(\mathbf{x}^{h+1})) - I_{j,t}(\Pi_j(\mathbf{x}^{h+1})) \\ \approx& I_{i,t}(\Pi_i(\mathbf{x}^{h})) - I_{j,t}(\Pi_j(\mathbf{x}^{h}))+ \\ & (I_{ijt}^h)\delta d^h + \sum_k (I_{ijt}^h)_{\lambda_k}\delta\lambda_k \\ =& (\nabla I_{ijt}^h)^\intercal \delta\mathbf{D}^h.\end{aligned} \tag{C.3}$$

Where

$$\nabla I_{ijt}^h = [(I_{ijt}^h), (I_{ijt}^h)_{\lambda_1}, \cdots, (I_{ijt}^h)_{\lambda_H}, I_{ijt}^h]^\intercal, \tag{C.4}$$

$$\delta\mathbf{D}^h = [\delta d_k^h, \delta\lambda_1^h, \cdots, \delta\lambda_H^h, 1]^\intercal. \tag{C.5}$$

The terms for I_{it1} are defined analogously. Defining the data tensors $S_{ijt}^h = (\nabla I_{ijt}^h)(\nabla I_{ijt}^h)^\intercal$ and $S_{it1}^h = (\nabla I_{it1}^h)(\nabla I_{it1}^h)^\intercal$ the linearized equations are

$$\begin{aligned}\sum_{t=1}^{T}\sum_{j\neq i} w_{ijt}\Psi'_{ijt}(\cdot)[S_{ijt}^h]_1\delta\mathbf{D}^h + \alpha\sum_{t=2}^{T} w_{it1}\Psi'_{it1}(\cdot)[S_{ijt}^h]_1\delta\mathbf{D}^h \\ -\beta_1\nabla\cdot(\Psi'(|\nabla d^{h+1}|^2)\nabla d^{h+1}) = 0,\end{aligned}$$

$$
\begin{aligned}
&\sum_{t=1}^{T}\sum_{j\neq i} w_{ijt}\Psi'_{ijt}(\cdot)[S^h_{ijt}]_{k+1}\delta\mathbf{D}^h + \\
&\quad \alpha\sum_{t=2}^{T} w_{it1}\Psi'_{it1}(\cdot)[S^h_{ijt}]_{k+1}\delta\mathbf{D}^h - \\
&\quad \beta_2\nabla\cdot(\Psi'(|\nabla\mathbf{o}^{h+1}|^2)\sum_{c=1}^{3}\nabla o_c^{h+1}[\mathcal{B}_k(t)]_c) = 0, \\
&\quad \Psi'_{ijt}(\cdot) = \Psi'((\delta\mathbf{D}^h)^\intercal S_{ijt}\delta\mathbf{D}^h), \\
&\quad \Psi'_{it1}(\cdot) = \Psi'((\delta\mathbf{D}^h)^\intercal S_{it1}\delta\mathbf{D}^h).
\end{aligned}
$$

The discretization and geometric multi-grid optimization follow the framework for optic flow [45].

D Backprojected Intra-view 2D Optical Flow

In this simple approach of obtaining dense 3D deformation correspondences, optic flow is solved independently for each image using a variational method with strong regularization [45]. The result is a set of 2D flow fields defining the flow between $t-1$ and t: $V^+_{i,t}$. The individual flows are composed to obtain flow vectors mapping from time 1 to t: $V_{i,t} = V_{i,t-1} \odot V^+_{i,t}$, where $(V_i \odot V_j)(\mathbf{x}) = V_j(\mathbf{x} + V_i(\mathbf{x})) + V_i(\mathbf{x})$ (see Figure D.1).

Assuming the surface at $t = 1$ is accurate (an approximation can be obtained from vision, e.g., [186]), the displaced vertices at time t are obtained by triangulating the 2D points flowed from $t = 1$. For example, given a 3D mesh vertex, $\mathbf{x}$, at time $t = 1$, the displacement to any time t can be obtained as

$$\mathbf{d}_t = \underbrace{(\operatorname*{argmin}_{\hat{\mathbf{x}}} \sum_{i=1}^{C} (\Pi(\mathbf{P}_i\hat{\mathbf{x}}) - (V_{i,t}(\Pi(\mathbf{P}_i\mathbf{x})) + \Pi(\mathbf{P}_i\mathbf{x})))^2)}_{\text{Triangulated displaced point}} - \mathbf{x}. \tag{D.1}$$

This method does not directly take photo-consistency into account in the latter frames, and can become problematic for surfaces that exhibit overlapping. Composing the flow in the above manner can also introduce drift, so it is best used for short sequences.

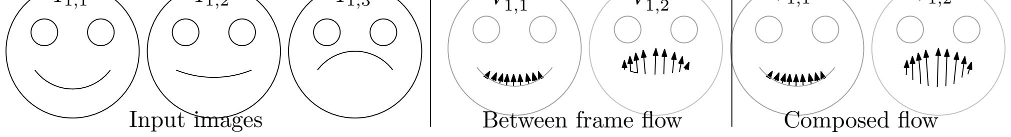


Figure D.1: Flow fields between frames are composed to get flow from time $t = 1$.

E SIMD Texture Modulation

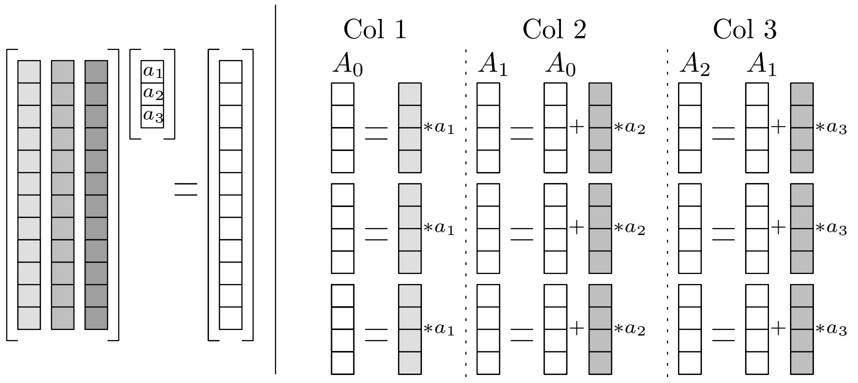


Figure E.1: The texture modulation is a large matrix multiplication. Like normal matrix multiplication, a SIMD implementation uses an accumulator, A, to aggregate the multiplication of the columns into with the coefficients a_i. Each multiplication (and addition on later columns) processes 4 units each.

The texture modulation is a matrix multiplication that is the most expensive part of rendering. SIMD operations can increase the efficiency of this multiplication by operating on several elements at once. When using SSE, the SIMD operands include 4 floating point values. Figure E.1 illustrates how the matrix multiplication can be broken into floating point chunks that are 4 elements wide. Figure E.2 illustrates the corresponding implementation through the use of SSE intrinsics.

Instead of a simple SSE floating point implementation, an MMX implementation (which operates on integer data) can be used to reduce the storage. When using integer data for the basis, care needs to be taken when converting the floating point values to the corresponding integers. The basis vectors should first be scaled such that each column of the basis is in the range $[-1, 1]$. The corresponding coefficients should be scaled by the inverse of the scale factor. The basis and coefficients can then be mapped to integers in the range of $[-128, 128)$ (e.g., by multiplying by 128). The MMX instruction set can then be used to modulate the basis and accumulate the results, ensuring temporary results are properly sign extended. In MMX the multiplication of two single-byte operands results in a two-byte short integer result. The resulting aggregated results (shorts) can be converted to floating point values by dividing by 128^2. See the code listing for the corresponding MMX implementation Figure E.3.

```
// Clear the results (e.g., memset(result, 0, sizeof(float)*m);
// Iterate over each column
for (int j=0; j<n; j++) {
   __m128 c = _mm_set_ps(a[j], a[j], a[j], a[j]);

   // Iterate down the column
   for (int i=0; i<m; i+=4) {
      // Load 4 elements from column at i
      __m128 d = _mm_loadu_ps(basis + i + j*stride);

      // Perform multiplication
      __m128 scaled = _mm_mul_ps(d, c);

      // Load current accumulated result
      d = _mm_loadu_ps(result + i);

      // Accumulate
      __m128 A = _mm_add_ps(d, scaled);

      // Store accumulation into result
      _mm_storeu_ps(result + i, A);
   }
}
```

Figure E.2: SIMD basis modulation code listing.

```
// Clear the accumulation vector, e.g., memset(result, 0, sizeof(short)*m);
_mm_empty(); // Clear MMX state.

// Iterate over all basis elements (basis is stored in shorts)
for (int j=0; j<n; j++) {
   int cin = scoeff[j]; // assume ccoeff is already in range [-128, 128)

    // Load the coefficient into the 4 short locations.
    __m64 c = _mm_cvtsi32_si64(cin);
    c = _m_punpcklwd(c, c);
    c = _m_punpckldq(c, c);

    // Iterate down the column
    for (int i=0; i<m; i+=4) {
      // Load 4 elements into lower 8 bytes [0, 0, 0, 0, b3, b2, b1, b0]
      __m64 din = _mm_cvtsi32_si64(*((int*)(basis + i + j*stride)));

      // Load into upper portions [b3, b3, b2, b2, b1, b1, b0, b0]
      __m64 d = _m_punpcklbw(din, din);

      // perform sign extension by rotating right 8 bits: [b3; b2; b1; b0]
      d = _m_psraw(d, _mm_cvtsi64x_si64(8));

      // Scale the basis [b3; b2; b1; b0] * c
      __m64 scaled = _m_pmullw(d, c);

      // Accumulate with existing 4 shorts.
      __m64 A = _mm_add_pi16(_mm_cvtsi64x_si64(*(long long*)(result + i)), scaled);

      *((long long *)(result + i)) = (long long)A; // Store the resulting 4 shorts.
   }
}
```

Figure E.3: MMX basis modulation code listing.