Reconstruction and Analysis of Dynamic Shapes

by

Daniel Vlasic

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2010

Author ....[illegible].......
Department of Electrical Engineering and Computer Science
April 30, 2010

Certified by.......[illegible].......
Dr. Jovan Popović
Associate Professor
Thesis Supervisor

Accepted by.......[illegible].......
Professor Terry P. Orlando
Chairman, Department Committee on Graduate Students

Reconstruction and Analysis of Dynamic Shapes

by

Daniel Vlasic

Submitted to the Department of Electrical Engineering and Computer Science
on April 30, 2010, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science and Engineering

Abstract

Motion capture has revolutionized entertainment and influenced fields as diverse as the arts, sports, and medicine. This is despite the limitation that it tracks only a small set of surface points. On the other hand, 3D scanning techniques digitize complete surfaces of static objects, but are not applicable to moving shapes. I present methods that overcome both limitations, and can obtain the moving geometry of dynamic shapes (such as people and clothes in motion) and analyze it in order to advance computer animation. Further understanding of dynamic shapes will enable various industries to enhance virtual characters, advance robot locomotion, improve sports performance, and aid in medical rehabilitation, thus directly affecting our daily lives.

My methods efficiently recover much of the expressiveness of dynamic shapes from the silhouettes alone. Furthermore, the reconstruction quality is greatly improved by including surface orientations (normals). In order to make reconstruction more practical, I strive to capture dynamic shapes in their natural environment, which I do by using hybrid inertial and acoustic sensors. After capture, the reconstructed dynamic shapes are analyzed in order to enhance their utility. My algorithms then allow animators to generate novel motions, such as transferring facial performances from one actor onto another using multilinear models. The presented research provides some of the first and most accurate reconstructions of complex moving surfaces, and is among the few approaches that establish a relationship between different dynamic shapes.

Thesis Supervisor: Dr. Jovan Popović
Title: Associate Professor

Acknowledgments

I dedicate this thesis to my family. They believed in me and supported me throughout this endeavor. I am grateful to my adviser Jovan for letting me explore many different topics over many different years. I thank my wife Ana for making those years more enjoyable. I appreciate the guidance of my mentors, the discussions with my collaborators, the feedback from my colleagues, the money from my funding sources, and the friendship of my friends.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

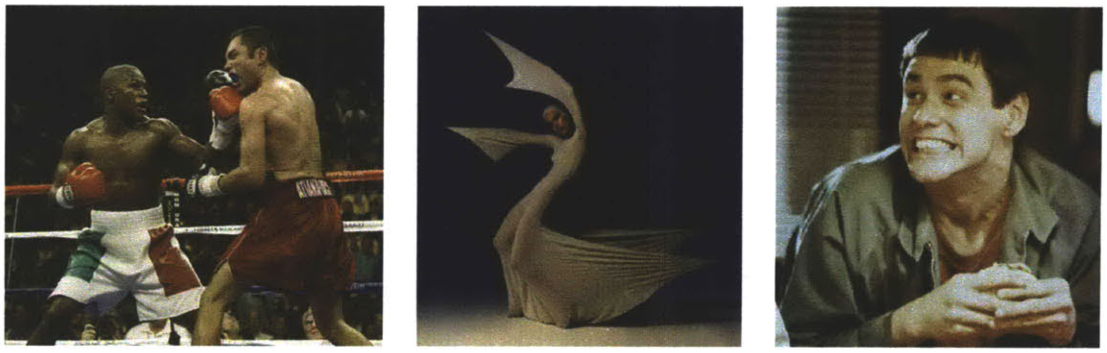


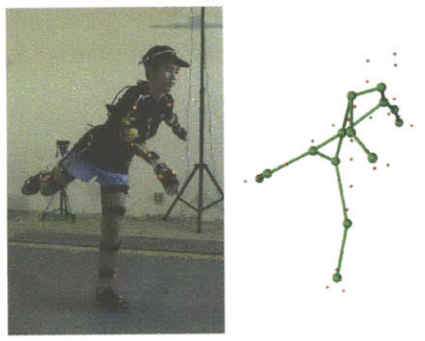
Figure 1-1: Dynamic shapes are complex and include bulging muscles, flapping clothes, and subtle facial expressions (pictures are from floydmayweatherjr.org, geocities.com/Athens/3067, and movie "Dumb and Dumber"). Understanding these complexities has implications in entertainment, education, sports, robotics, and medicine.

The ability to understand dynamic shapes, such as people and clothes in motion, impacts many industries. This is clearly evident from the success of the motion capture technology (Metamotion, PhaseSpace, Vicon, Xsens). Even though *mocap* is able to assess only a small number of surface points on these dynamic shapes (Figure 1-2 left), it has revolutionized entertainment and influenced fields as diverse as the arts, sports, and medicine. It is used to produce life-like body and face motions for virtual characters in computer games (EA Sports), animated movies (Disney, Pixar) and feature films (ILM). It is enabling new forms of human-computer interaction, where users can control virtual characters [57] or electronic devices (GypsyMIDI) with their body motions. Athletes use it to analyze and improve their sports performance (`www.iClub.net`). Physicians are exploring its application to physical rehabilitation (`www.ndigital.com`).

Understanding dynamic shapes more comprehensively, and analyzing their complete surfaces rather than just a few points, would enhance all the abovementioned applications and transform other fields as well. In computer vision, capturing and analyzing scenes is fundamental, and often those scenes contain dynamic shapes. Understanding them helps us control appliances with hand gestures [68, 52], track people [37, 56, 144], or recognize faces (`www.omron.com/r_d/coretech/vision/okao.html`). In robotics, researchers have engineered robots that can serve us refreshments (`world.honda.com/ASIMO`), carry our backpacks (`www.bostondynamics.com`), and even imitate our facial expressions (`www.hansonrobotics.com`). In order to better interact with people and perform human tasks such as repair satellites, manipulate paper, or fold laundry, robots need to understand how dynamic shapes deform. In biomechanics, understanding the motion of bones and muscles can help people move more efficiently (`advbiom.com`), improve their sports performance (`www.retul.com`), and avoid injuries (`www.exponent.com/injury_causation`). Finally, in medicine, understanding dynamic shapes can aid rehabilitation (`www.mercurybiomechanics.com`), track tumor growths [135], even analyze fetal heart beats [63].

Although dynamic shapes are important in many fields, they are complex and hard to model (Figure 1-1). They come in a large variety of forms and sizes, and deform in unique and complicated manners. Our muscles bulge, our clothes dynamically flap, our facial expressions are finely tuned to convey many subtle emotions. Coping with this complexity stands in the way of enhanced and broader usage of dynamic shapes in computer systems today.

While the coarse motion estimates obtained with mocap cannot fully represent the complexities of dynamic shapes, sophisticated surface scanning methods (Figure 1-2 right) attempt to do so by recovering complete surfaces of objects to within a millimeter precision [93, 71, 77, 79]. However, only a limited number of them can



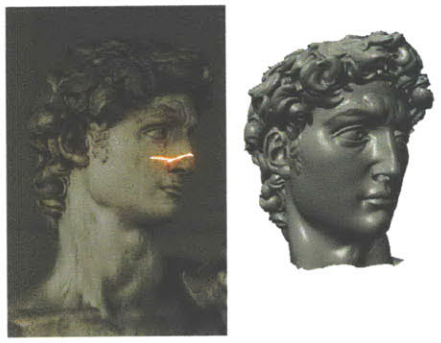


Figure 1-2: Traditional motion capture (left) tracks only a small number of points on dynamic shapes, yet is extensively used in entertainment, sports, education, and biomechanics (picture taken from [Kirk 2005]). At the same time, static surface reconstruction (right) has reached high level of sophistication, but is not applicable to moving shapes (picture taken from [Levoy 2000]). We want to reconstruct and analyze detailed moving surfaces of dynamic shapes.

possibly be adapted to dynamic shapes. For example, binocular and multi-view stereo offer convenience and dynamic capture, although they provide detailed data only in high-texture regions. Active 3D scanning based on structured light provides high-quality static meshes, but does not scale well to moving scenes or large working volumes. Photometric stereo (active shape from shading) offers high-quality geometric detail in the form of normal maps, but it remains difficult to combine the normal maps from multiple views. Template-based regularization may be applied to any of these methods, often with visually-compelling results, but the ultimate quality depends on the similarity of the actual surface to the template.

Because of their complexity, dynamic shapes are also hard to animate. This is most often done by hand using keyframing and procedural deformations. Adding detail with this approach is tedious because it requires setting hundreds of parameters. As a result, most animated characters appear to wear skin-tight garments that move stiffly without flowing. Procedural approaches that take physics into account can generate these details with more ease and efficiency, but they are difficult to control when the goal is to match a particular motion or performance.

In this thesis *we provide tools for reconstruction and analysis of moving surfaces,*

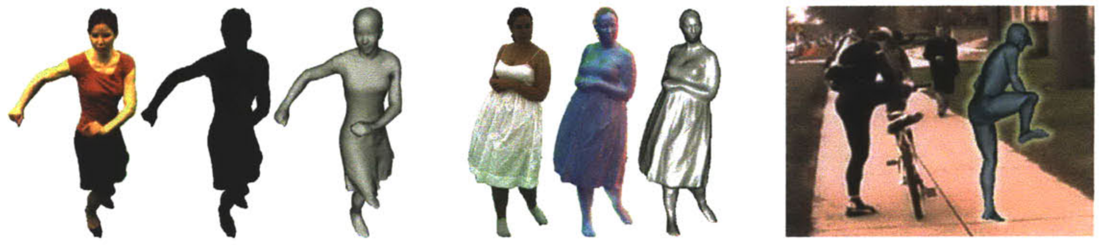


Figure 1-3: We provide tools to reconstruct dynamic shapes very efficiently from silhouettes (left), very precisely from normal maps (middle), and very practically from hybrid sensors (right).

opening new possibilities for application of complex dynamic shapes. Mirroring the traditional mo-cap, we capture the detailed moving surfaces of real objects in order to obtain compelling high-quality models. The ability to modify the recorded shapes is also vitally important. They need to be edited, transformed, interpolated, and recomposed so that new high-quality surfaces can be animated as easily as skeletons.

In Chapter 2 [150], we describe a system that reconstructs the moving surface of people and dynamic clothes by molding a template shape to fit multi-view silhouettes (Figure 1-3 left). Our method is one of the first and most efficient ways to obtain detailed surfaces for fast and complex motions. We show that the silhouettes contain enough geometric information to plausibly reconstruct body articulation, bulging muscles, and flapping clothes. The output mesh animations are fully corresponded making them suitable for texturing, editing, and model fitting. They can be readily used for virtual characters in games and animated movies, as well as to analyze complex human motions such as breakdancing.

In Chapter 3 [152], we enhance the quality of the reconstructed moving surfaces by also processing multi-view normal maps (Figure 1-3 middle). While normal maps require additional hardware resources (i.e. controlled lighting), they add detailed information in the interiors of the silhouttes, which allows us to obtain correct folds on the clothes to within millimeters of the true surface, even without the a-priori shape template. As a result, we can capture arbitrary moving surfaces such as actors with props and shapes with changing topology. While this method does not produce

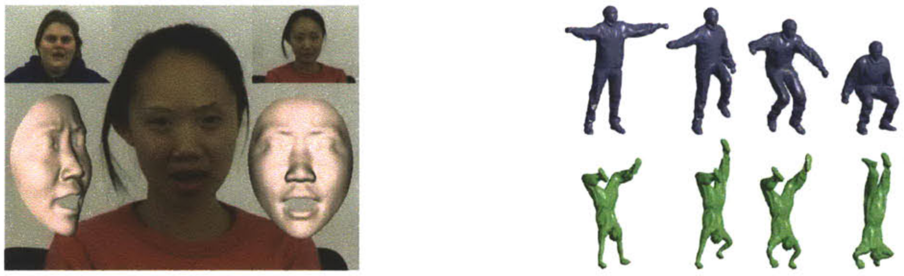


Figure 1-4: We provide methods to transfer facial performances between different faces (left), as well as full-body motions between different characters (right).

watertight corresponded surfaces, it yields data ideal for the geometry processing community, who can aggregate our surfaces into watertight and corresponded moving meshes. The high precision of our output meshes means that they can be used in high-end feature film production, as well as for various biometric and material measurements.

In Chapter 4 [149], we explore a possible approach to capturing dynamic shapes in their natural environments using hybrid inertial and acoustic sensors (Figure 1-3 right). Our method combines intertial measurements of different points on the body with acoustic time-of-flight between them in order to recover high-precision high-frequency body pose. Hybrid sensors make capture more practical by not relying on expensive and highly-controlled studio settings. They could allow anyone to capture motions anywhere for long periods of time, making this approach ideal for biomedical analysis and physical rehabilitation without extrenuously burdening the users.

In Chapter 5 [151], we move from reconstruction onto analysis of dynamic shapes with a goal to edit, combine and learn from the captured surface motions in order to generate novel surface motions and extend the usefulness of the captured data. We demonstrate a method that fits a multilinear model to the captured faces in order to transfer performances from one face to another (Figure 1-4 left). The multilinear model simplifies the process of face animation by naturally encoding individual idiosyncracies (such as style of smiling) and providing separate sets of parameters for identity, expression and speech. This means that we can change one set of parameters (e.g. identity) while keeping the others (expression and speech) fixed. In this manner, our method can animate the digital face of an expensive actor with a performance of a cheap actor, making sure that the likeness, individuality and mannerism of the expensive actor is preserved.

In Chapter 6, we summarize our findings and discuss possible future directions, including improved surface capture by further processing the reconstructions from normals (Chapter 3), and transferring not just facial performances, but full-body motions between very different characters (Figure 1-4 right) [12].

The presented methods can be used to improve virtual characters in entertainment and education, as well as to simplify the laborious process of animating them. They show potential for enhancing human-computer interaction and improving the application of biomechanics to medicine and rehabilitation. As motion capture has become commonplace across a wide variety of fields, moving surfaces will become an enhanced way of capturing, viewing, analyzing and editing dynamic shapes rich with surface details.

Chapter 2

Reconstruction of Dynamic Shapes from Silhouettes

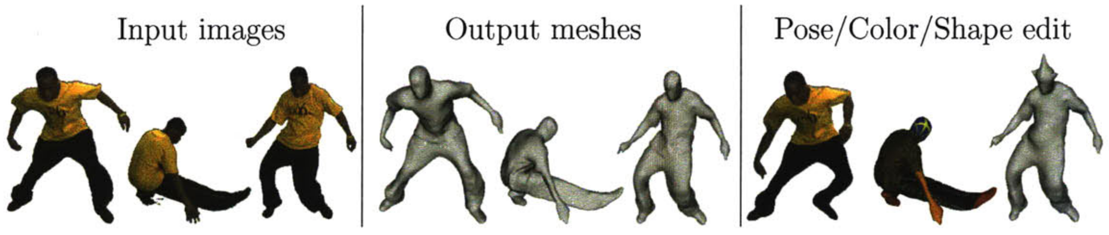


Figure 2-1: Our methods for pose tracking and non-rigid shape matching make it possible to extract a mesh animation with full correspondence (middle) from multi-view video data (left), allowing easy editing of pose, texture and geometry (right).

In this chapter we address the challenging problem of reconstructing the moving geometry of dynamic shapes (e.g. human actors) with a template-based approach. We "mold" a 3D template of the subject to fit the silhouettes obtained from multiple cameras surrounding the scene. This method efficiently obtains consistently-parameterized moving meshes that are suitable for visualization and editing. However, the surface detail in-between the contours is interpolated from the template and does not exactly match the observed motions. Nevertheless, the reconstructed surfaces convey the flavor and the dynamics of the captured actors by recovering their body pose, deforming skin and flapping clothes.

Our system processes a set of synchronized background-subtracted videos that provide a record of a human performance from multiple viewpoints. The silhouette from each viewpoint corresponds to a cone of rays from the camera origin through all points of the subject. The intersection of these cones approximates the subject's shape and is called the visual hull. Our method first uses the visual hulls to track the skeletal pose of the performer. In especially difficult frames, the user can specify constraints for joint positions, allowing for more robust tracking than is possible with fully automatic methods. Our system then deforms a template mesh of the performer to fit the recovered pose and the silhouettes at each frame. The output is suitable for editing because the template ensures frame-to-frame correspondence. Figure 2-1 shows a few representative results and some sample edits.

Our pipeline makes use of two novel techniques. The first is a geometric pose tracking method fast enough to incorporate interactively provided user constraints. This is essential because no current automatic method can track the skeleton perfectly across all frames in a setting such as ours: motions are fast and complex, performers wear loose clothing, textures are poor, and the visual hull is ambiguous (e.g., crossing hands). The second novel technique is an iterative method for deforming a template mesh to match the observed silhouettes while preserving the detail in the template. This allows us to capture the secondary deformations, such as flapping clothing, that make the motion appear natural. Together these techniques enable us to process more challenging data faster and obtain higher quality results than previously possible.

2.1 Previous Work

Traditional motion capture estimates skeleton motion using information from markers or sensors on the body. While these systems are accurate, they still require manual adjustments to clean up recorded data. Furthermore, recording requires skin-tight garments to ensure that markers move rigidly with the corresponding limb. Markerless motion capture addresses these limitations by estimating pose directly from multi-view video [99, 37, 140, 49]. However, these methods are less reliable when limbs are not clearly distinguishable (e.g., when arms are close to the body). Manual clean-up is often necessary, but the above methods do not discuss ways to assist in this process.

Pose estimation does not capture the fine detail on the surface of the shape. Richer templates with more degrees of freedom can capture time-varying geometric detail better. Sand et al. [127] capture some of those effects with silhouette-bounded needles radiating from each bone, without maintaining frame-to-frame correspondence. Park and Hodgins [109] improve on these results to record detailed bulging and jiggling using a traditional motion-capture system with many markers. Other researchers have shown how to design patterns that can be printed on clothing to enable highly detailed estimation of garment motion [128, 165]. Markers, either attached or printed, make it less convenient to record arbitrary human performances in casual clothing. Purely vision-based techniques rely on texture cues to track motion of a few carefully selected points [51, 50]. These approaches are not as reliable on fast motions, especially when there are few textures cues (e.g., pants in Figure 2-1).

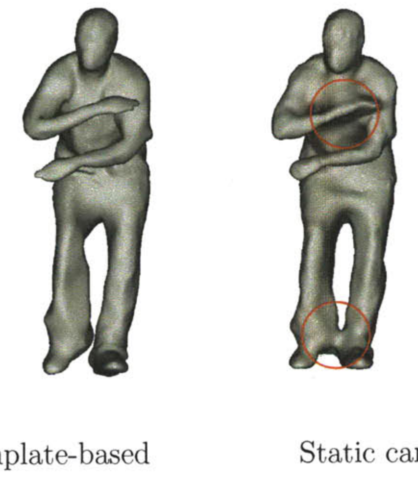


Figure 2-2: Applying static carving techniques can result in topology issues, such as the connected feet and arm in this frame while template-based reconstruction ensures a consistent topology.

The multi-view stereo literature provides a powerful collection of tools for recovering a single static mesh [129]. Some of the best methods produce extremely accurate results but they are computationally expensive, requiring an hour of computation or more for a single frame [82, 70, 61]. Faster carving methods sacrifice quality but capture meshes at speeds more practical for processing video streams [122, 134, 75]. These methods do not make assumptions about the observed deformations and geometry, allowing them to capture arbitrary deforming surfaces, not just articulated characters. They, however, do not always reconstruct the topology correctly and produce uncorresponded results (Figure 2-2). Our approach is much faster and it generates mesh animations with frame-to-frame vertex correspondence.

Our approach maintains correspondence by matching a single mesh template to every frame in the sequence. Using a template mesh leads to faster algorithms and allows interpolation of regions where data is unavailable. Articulated templates are often used to improve pose tracking, but they do not deform to capture details in the video [33, 138, 37, 6, 39, 9]. A template based on implicit surfaces [117] can capture time-varying detail, but does not produce corresponded meshes. To learn a skinning model, Allen et al. [4] obtain corresponded shapes for different poses by matching a subdivision surface template to range data.

Our processing pipeline is most similar to the work of de Aguiar and colleagues [48] who also estimate the skeleton configuration before deforming the template mesh to match the data. Both approaches use Laplacian coordinates to preserve mesh detail while satisfying silhouette constraints (their method also uses texture constraints). A critical difference is that they sample the mesh and drive the samples towards the visual hull, while we sample the contours and pull only the closest mesh points to our samples. Because most of the correct surface is not on the visual hull, their method tends to distort the shape and needs strong regularization as well as reliable texture cues. In contrast, we only pull those vertices to the visual hull that are likely to be on it and can therefore match the contours more closely. Hence, our method performs significantly better on images with poor texture cues and strong silhouette cues. Additionally, our method is over an order of magnitude faster, which allowed us to develop an interactive user interface to assist pose correction. These differences enable us to capture mesh animations even for fast and complex human motions.

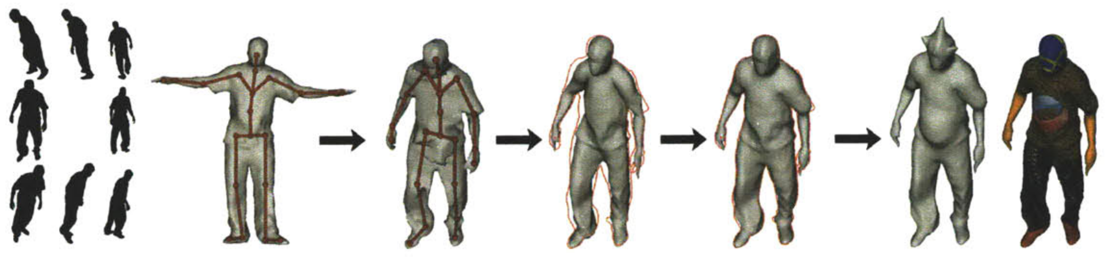


Figure 2-3: From left to right, our system starts with a stream of silhouette videos and a rigged template mesh. At every frame, it fits the skeleton to the visual hull, deforms the template using linear blend skinning (LBS), and adjusts the deformed template to fit the silhouettes. The user can then easily edit the geometry or texture of the entire motion.

2.2 Overview

A multi-view studio provides a set of synchronized high-definition silhouette videos by recording a performance from several angles using multiple calibrated cameras. Our software pipeline, shown in Figure 2-3, also uses a template mesh rigged with a skeleton that matches the physical dimensions of the performer: the skeleton is positioned within the template mesh and each vertex is assigned a weight that is used to deform the template with linear blend skinning (LBS). Our software outputs a sequence of joint configurations and vertex positions to represent the pose and shape of the performer in every frame of the multi-view sequence.

The software pipeline proceeds in two stages: skeleton tracking and surface refinement. In the first stage (Section 2.3), the system optimizes the fit of the skeleton to the visual hull at each time step. The system tracks most simple motions automatically, but may fail on very complex motions. In such cases an easy manual intervention can be used to correct the skeleton. In the second stage (Section 2.4), our system deforms the template according to the skeleton and adjusts its surface to better match the silhouettes.

2.3 Pose Estimation

During the pose estimation stage, we fit the skeleton to the visual hull in each frame. Our objective is to position the bones deeply into the visual hull and to maintain the temporal smoothness. Evaluating this objective requires computing the distance to the visual hull and we develop an efficient method for this task. Additionally, our objective incorporates information provided by the user for especially difficult frames. To optimize the objective, we iterate through the frames in order and find the best pose for each frame, while providing visual feedback of the progress to the user and allowing user corrections. To propagate user constraints backwards, we make another pass over the frames in reverse order.

2.3.1 Objective Function

Pose estimation recovers the pose by minimizing an objective function that is a weighted combination of four terms. The first term E_D pushes the bones towards the medial axis of the visual hull, the second term E_T improves the temporal smoothness of the motion, the third term E_R pulls the end effectors into the extremities, and the fourth term E_U enforces the user constraints:

$$\arg\min_{\boldsymbol{\theta}} \left(w_D E_D(\boldsymbol{\theta}) + w_T E_T(\boldsymbol{\theta}) + w_R E_R(\boldsymbol{\theta}) + w_U E_U(\boldsymbol{\theta}) \right),$$

where $\boldsymbol{\theta}$ represents the degrees of freedom of the skeleton and the scalar weights w determine the relative importance of the depth, temporal, refinement, and user terms. The vector $\boldsymbol{\theta}$ consists of joint angles (in radians) and the root translation (in meters). Joint limits constrain it to lie in a range.

Depth (E_D) The true deformed object is completely contained inside the visual hull. Therefore, we require that in a good fit, the distance from a point on the skeleton to the visual hull surface be no less than the distance from the corresponding point on the template skeleton to the surface of the template. Let $\mathbf{p}_i$ be samples on the bones of the skeleton and let r_i be corresponding distances to the template surface (so if $\mathbf{p}_1$ is in the middle of the femur, r_1 is roughly the thigh radius at that point). In a particular frame, let $d(\mathbf{p})$ be the distance from a point $\mathbf{p}$ to the visual hull surface. We wish to penalize $d(\mathbf{p}_i) < r_i$, so we set

$$E_D(\boldsymbol{\theta}) = \sum_i \gamma(d(\mathbf{p}_i(\boldsymbol{\theta})) - r_i),$$

where $\gamma(x)$ is a smooth function that is approximately $-x$ when x is negative and approximately 0 when x is positive. Using four evenly spaced samples per bone provides a good compromise between performance and accuracy.

Temporal Smoothness (E_T) During the forward pass, we enforce smoothness by penalizing deviation from the previous frame: $E_T(\boldsymbol{\theta}_t) = ||\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}||^2$. During the reverse pass, we penalize deviation from the next and the previous frame:

$$E_T(\boldsymbol{\theta}_t) = ||\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1}||^2 + ||\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t+1}||^2.$$

Refinement From Shape Estimation (E_R) Because the depth term tends to pull the bones to the inside of the visual hull, the end effectors (hands, feet, and head) often do not extend all the way into the extremities. To enhance the skeleton fit, we use our shape estimation algorithm (described in Section 2.4) to move these joints into place. After an initial optimization with w_R set to 0, we run an iteration of shape estimation. This pulls the template vertices towards the correct surface locations. For each end effector j, we look at the portion of the surface that is nearly rigidly bound to that joint. Let $\Delta\mathbf{m}_j$ be the vector by which the center of mass of that portion moves as a result of shape estimation. We translate each joint by $\Delta\mathbf{m}_j$ by setting

$$E_R(\boldsymbol{\theta}) = \sum_j ||\mathbf{q}_j(\boldsymbol{\theta}) - (\mathbf{q}_j(\boldsymbol{\theta}_0) + \Delta\mathbf{m}_j)||^2,$$

where $\mathbf{q}_j(\boldsymbol{\theta})$ is the position of joint j and $\boldsymbol{\theta}_0$ is the joint angle vector after the initial optimization. We repeat this process twice to obtain the final pose.

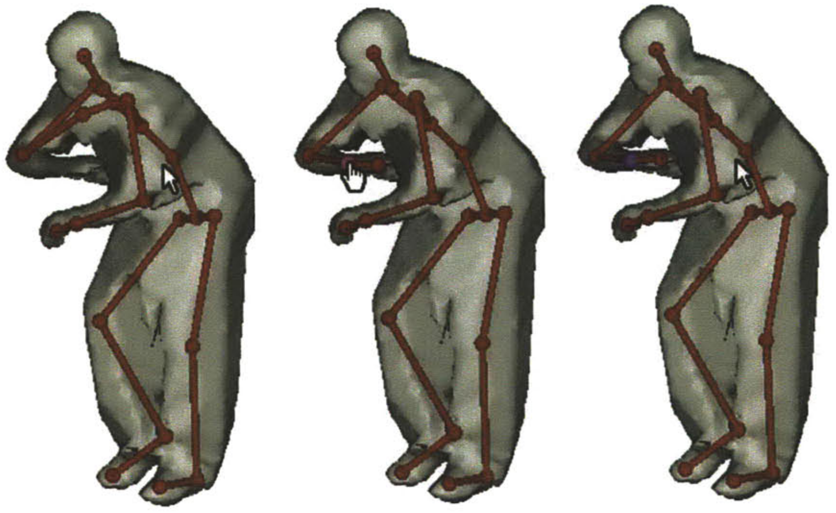


Figure 2-4: An incorrectly fit skeleton is fixed by the user dragging the wrist joint. Our system repositions the hand joint automatically to fit the visual hull.

User Constraints (E_U) When the user interrupts the forward pass and drags a joint j to a new position on the screen, our system interprets that as a constraint on the vertex to lie on the ray that projects to that position. We incorporate this constraint by adding a point-to-ray squared distance term to E_U:

$$E_U(\boldsymbol{\theta}) = \sum_{u \in U} \|(\mathbf{q}_{j_u}(\boldsymbol{\theta}) - \mathbf{o}_u) \times \mathbf{r}_u\|^2,$$

where U is the set of user constraints, j_u, $\mathbf{o}_u$ and $\mathbf{r}_u$ are the joint index, ray origin and ray unit direction, respectively, for constraint u. To constrain a joint completely, the user needs to specify the constraint from two different views. The user is also allowed to remove the refinement constraints.

Weights Our rationale in choosing the weights is that we want the depth term to be stronger than the temporal smoothness term, the estimation-from-refinement constraints to override the depth term, and the user constraints to override everything. Because the error terms E_D, E_T, E_R, and E_U have units of m, rad^2 (our conversion constant between radians and meters is 1), m^2, and m^2, our weights have inverse units. The actual weights we use are $w_D = 5$, $w_T = 1$, $w_R = 500$, $w_U = 5000$, but as discussed in Section 2.5.2, our method is not overly sensitive to their precise values.

2.3.2 Distance Evaluation

Without optimization, the most expensive part of the objective function evaluation would be computing $d(\mathbf{p_i})$ for the depth term. To ensure interactive performance for user corrections, the evaluation of the distance to the visual hull must be fast. Because computing a full 3D distance field would take too much time per frame, we make use of the fact that the visual hull is the intersection of eight cones and most of the evaluations are inside the intersection.

In general, if we have several objects, the distance field inside their intersection is the minimum of their individual distance fields. So if we could quickly evaluate the distance field inside each cone, we would be able to quickly evaluate the distance field inside the visual hull. Let $d_i(\mathbf{p})$ be the distance to the boundary of cone i and let $\mathbf{o}_i$ be the origin of that cone. Note that for an arbitrary scalar a, $d_i(\mathbf{o}_i + a(\mathbf{p} - \mathbf{o}_i)) = a\, d_i(\mathbf{p})$. This lets us compute d_i at every point on the camera image plane, and then evaluate d_i everywhere else just by scaling. This formulation corrects an earlier observation that uses image-space distances instead of point-ray distances [59].

To compute d_i on the image plane, we adapt a vector distance transform algorithm [43]. We initialize d_i to zero on the contour pixels and rather than propagating a 2D vector to the nearest pixel during the scans, we propagate the nearest ray.

When the distance needs to be computed outside the visual hull, that skeleton joint is far from where it should be and we don't need as much accuracy. In this case, we use a KD-tree to compute the distance to a polygonal approximation of the visual hull. We extract this polygonal approximation by evaluating the distance field near the visual hull boundary and then running marching cubes.

2.3.3 Processing

Because our objective function is nonlinear in joint angles, we use SNOPT [74] to find a local minimum from an initial guess. The first frame has no good initial guess and needs manual constraints to fit the skeleton properly. The second frame is initialized to the solution for the first frame. During the rest of the forward pass, we use the linear prediction from the previous two frames $2\boldsymbol{\theta}_{t-1} - \boldsymbol{\theta}_{t-2}$ as an initial guess for frame t.

The user can interrupt the tracking if it fails and view the scene from any direction (including original camera views). The user can then fix the pose by dragging joints to their correct positions on the screen (Figure 2-4), defining rays on which those joints must lie. These constraints are incorporated into the objective function and the skeleton is reoptimized several times per second. This enables the user to position a joint interactively and observe the effect on the rest of the skeleton.

After all frames have been processed, our system performs an additional optimization pass over all the frames in reverse order during which the user does not specify additional constraints. The initial guess for a frame during this pass is the solution for that frame from the forward pass. The temporal smoothness term in the objective function allows user constraints and other information to be propagated backwards during the reverse pass.

2.4 Shape Estimation

After recovering skeletal poses for the whole sequence, shape estimation deforms the template mesh for each frame. A naive way of doing this is to put the template into the skeleton pose using linear blend skinning. While resembling the true shape in overall deformation, the resulting shape does not respect the silhouettes and exhibits artifacts at bent joints (Figure 2-5, left). We designed an iterative method for non-rigid shape matching using Laplacian coordinates that corrects this problem. The algorithm begins with a smoothed version of the LBS mesh as the initial guess. At each iteration, it reintroduces part of the original template detail and also introduces

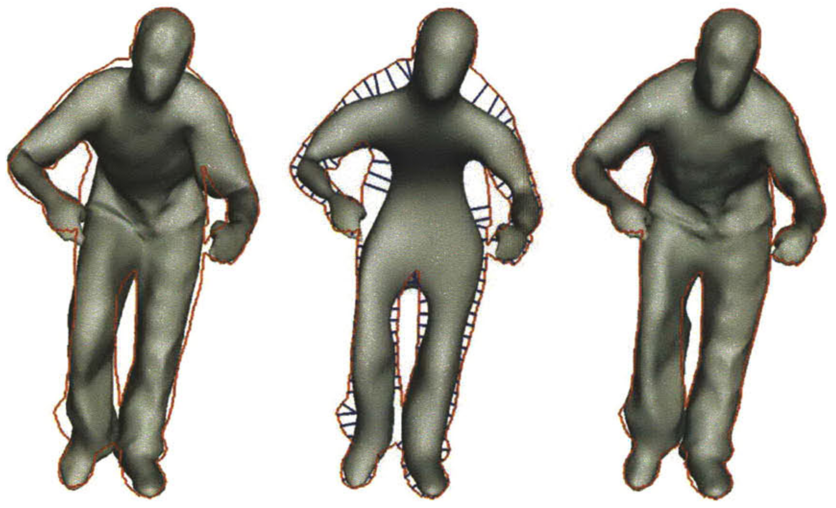


Figure 2-5: The template deformed with LBS suffers from artifacts around joints and does not fit the silhouettes. Our method smooths it and constrains it to the contours, gradually reintroducing detail. The result after several iterations has the detail of the original template, fits the silhouettes, and does not have LBS artifacts.

vertex position constraints that bring the shape closer to the contours in each camera. To enhance temporal consistency we interleave a bilateral filter on the meshes with these iterations. The resulting shapes match the silhouettes while still resembling the undeformed template.

2.4.1 Laplacian Coordinates

We convert our template mesh to Laplacian coordinates [3, 97] in order to represent the iteratively deforming shape. Let $\mathbf{V}_\mathrm{T}$ be the $n \times 3$ matrix storing the template mesh vertex coordinates. Then Laplacian coordinates $\mathbf{L}$ are generated with:

$$\mathbf{L} = \mathbf{\Delta} \mathbf{V}_\mathrm{T}$$

where $\mathbf{\Delta}$ is the $n \times n$ mesh Laplacian matrix (cotangent weights work best [101]). Laplacian coordinates are well-suited for our algorithm because they encode all of the geometric detail of our template. In addition, they let us constrain a subset of the

Algorithm 1 Shape Estimation

```
1: $k \leftarrow$ number of frames
2: for $t = 1$ to $k$ do
3:    $(\mathbf{C}_{\mathrm{LBS}}, \mathbf{P}_{\mathrm{LBS}}) \leftarrow$ nearly-rigid vertices
4:    $\mathbf{V}_t \leftarrow$ solve Equation 2.1
5: end for
6: $\mathbf{V}_{1..k} \leftarrow$ TemporalFilter($\mathbf{V}_{1..k}$)
7: for $w_L = 0$ to 1, step 0.2: do
8:    for $t = 1$ to $k$ do
9:       $\mathbf{L}' \leftarrow$ Laplacian coordinates rotated using LBS
10:      $(\mathbf{C}_{\mathrm{SIL}}, \mathbf{P}_{\mathrm{SIL}}) \leftarrow$ silhouette constraints using current $\mathbf{V}_t$
11:      $\mathbf{V}_t \leftarrow$ solve Equation 2.2
12:   end for
13:   $\mathbf{V}_{1..k} \leftarrow$ TemporalFilter($\mathbf{V}_{1..k}$)
14: end for
```

vertices and the remaining vertices will be appropriately interpolated. Furthermore, we can control the smoothness of the shape by scaling the coordinate vectors.

We recover the Euclidean vertex coordinates $\mathbf{V}$ from the Laplacian coordinates and vertex position constraints by solving the following linear least squares system:

$$\underset{\mathbf{V}}{\arg\min}\left(\|\mathbf{\Delta V} - w_L\mathbf{L}\|^2 + w_C\|\mathbf{CV} - \mathbf{P}\|^2\right),$$

where the $m \times 3$ matrix $\mathbf{P}$ contains the target positions of m constrained vertices, with the rows of the $m \times n$ matrix $\mathbf{C}$ having 1's at the corresponding columns. To constrain a point on the mesh that is not a vertex, we put its barycentric coordinates into the appropriate row of $\mathbf{C}$ instead. The first term ensures detail preservation, with the weight w_L determining how smooth the resulting mesh will be (0 is the smoothest, while 1 incorporates the full detail of the original mesh). The second term ensures that the constraints are satisfied, and the weight w_C determines how much it is weighted compared to the Laplacian coordinates term. We use soft constraints rather than hard ones so that we can balance fitting to the contours against the reproduction of detail from the template mesh.

2.4.2 Non-Rigid Shape Matching

We do not know initially which points on the mesh should be constrained to which points on the contours. We determine this iteratively (Algorithm 1), starting from a mesh deformed using only the recovered pose. The initial and final shapes for one of the frames are shown in Figure 2-5. The mesh does not match the silhouette perfectly because of our use of soft constraints and silhouette sampling.

Initialization (Lines 2–5) The LBS-deformed mesh is a poor initial guess for shape estimation because of skinning artifacts near the joints. However, the vertices that are attached mostly to a single bone are acceptable, as they deform almost rigidly. Therefore, for our initial guess, we constrain the *nearly-rigid* vertices (whose LBS weight is at least .95 for a single bone) and smoothly interpolate the rest of the mesh by scaling the Laplacian coordinates to zero. We solve:

$$\arg\min_{\mathbf{V}} \left(\|\mathbf{\Delta V}\|^2 + w_C \|\mathbf{C}_{\text{LBS}}\mathbf{V} - \mathbf{P}_{\text{LBS}}\|^2 \right), \tag{2.1}$$

where we set $w_C = 1000$, and fill $\mathbf{P}_{\text{LBS}}$ and $\mathbf{C}_{\text{LBS}}$ with the nearly-rigid LBS vertices and their indices. This results in the non-rigid regions being smoothed-out, as shown in Figure 2-5 (middle), alleviating problems such as over-folding and self-intersections.

Iteration (Lines 7–14) After obtaining the initial shape, we perform several iterations that bring it into better agreement with the silhouettes. We compute each subsequent shape estimate by solving

$$\arg\min_{\mathbf{V}} \left(\|\mathbf{\Delta V} - w_L \mathbf{L}'\|^2 + w_C \|\mathbf{C}_{\text{SIL}}\mathbf{V} - \mathbf{P}_{\text{SIL}}\|^2 \right), \tag{2.2}$$

where $w_C = 1000$ and $\mathbf{C}_{\text{SIL}}$ is filled with barycentric coordinates of surface points that are constrained to points on contour rays (as described below). The target locations are stored in $\mathbf{P}_{\text{SIL}}$. Rows of $\mathbf{L}'$ contain transformed Laplacian coordinates $\boldsymbol{\ell}'_i$ for the current frame. This is necessary because changes in the subject's pose rotate portions

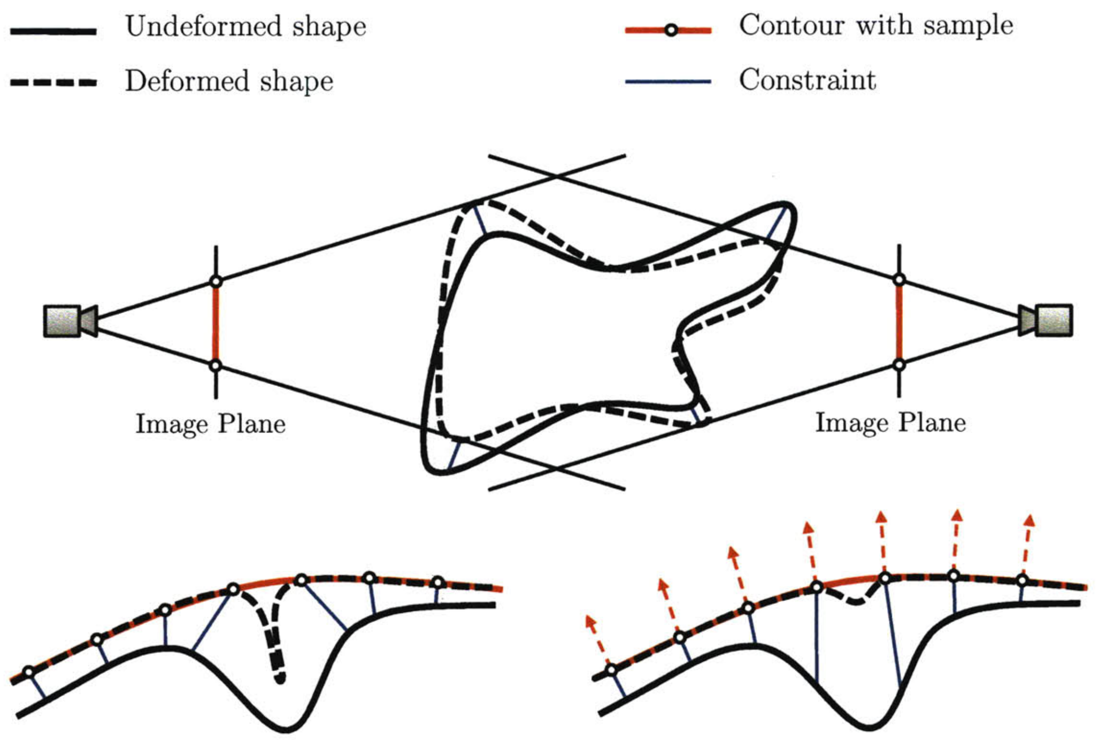


Figure 2-6: We deform the template to fit the silhouettes by constraining at most one surface point to each contour sample ray (top). Adjusting the template by pulling the closest surface point toward each contour sample (bottom left) exacerbates folding effects, while preferring the points in the direction normal to the contour (bottom right) helps diminish these effects.

of the surface and Laplacian coordinates are not rotation-invariant. For vertex i, we transform its Laplacian coordinate vector $\boldsymbol{\ell}_i$ (i^{th} row of $\mathbf{L}$) by a linear combination of the bone rotations:

$$\boldsymbol{\ell}'_i = \sum_j b_i^j \mathbf{R}^j(\boldsymbol{\ell}_i),$$

where $\mathbf{R}^j$ is the rotation of bone j, and b_i^j is that bone's LBS weight for vertex i. During our six iterations, we gradually increase w_L from 0 (completely smooth) to 1 (full original detail).

Silhouette Constraints (Line 10) At every iteration we use the silhouette contours to determine $\mathbf{C}_{\text{SIL}}$, the surface points that need to be constrained, and $\mathbf{P}_{\text{SIL}}$, the locations on the contour rays to which they should be constrained. Every ray from a camera origin through the contour is tangent to the subject. We therefore sample the contours (every 10 pixels) and look for constraints to make the estimated shape tangent to each sampled ray. For each ray, we find a point on the mesh that should be pulled towards that ray. If the ray is outside the reconstructed shape, we use the closest point on the shape to the ray. If the ray intersects the reconstructed shape, we find the point on the ray that's deepest inside the shape and use the point on the shape closest to it. Figure 2-6 (top) illustrates both of these cases. To prevent the surface from folding in on itself (Figure 2-6, bottom), we distort the distance to the shape by preferring vertices that are in the normal direction of the ray (obtained using the outward facing silhouette normal). We scale the distance metric by 0.25 in this direction. To avoid incorrect constraints, we do not constrain points that are farther than 5cm (in the scaled metric) from the ray or whose mesh normal differs by more than 90° from the silhouette normal. The resulting closest points are used to fill in $\mathbf{P}_{\mathrm{SIL}}$ and $\mathbf{C}_{\mathrm{SIL}}$ in Equation 2.2. Figure 2-5 (middle) shows the contour constraints pulling on various surface points for a frame in one camera.

Temporal Filtering (Lines 6 and 13) Minor tracking errors and temporal inconsistency in the contour samples causes the estimated shapes in neighboring frames to exhibit "texture sliding" [7] even if the subject is stationary: though the surface appears smooth and fixed, the underlying triangulation wobbles around. Simply applying temporal smoothing on the mesh vertices works poorly because it causes large deviations from the data when the subject is moving quickly. We therefore apply a bilateral smoothing filter:

$$\mathbf{v}_i^t \leftarrow \mathbf{v}_i^t + \left(\frac{\mathbf{v}_i^{t-1} - 2\mathbf{v}_i^t + \mathbf{v}_i^{t+1}}{4}\right) e^{\frac{-\|\mathbf{v}_i^{t+1}-\mathbf{v}_i^t\|^2 - \|\mathbf{v}_i^{t-1}-\mathbf{v}_i^t\|^2}{\sigma^2}},$$

where $\mathbf{v}_i^t$ is the vertex i at time t. In our implementation $\sigma = 0.07$m. We apply a pass of this filter twice after each shape estimation iteration. Texture sliding artifacts are most visible when the object is static, and our bilateral filter greatly reduces them.

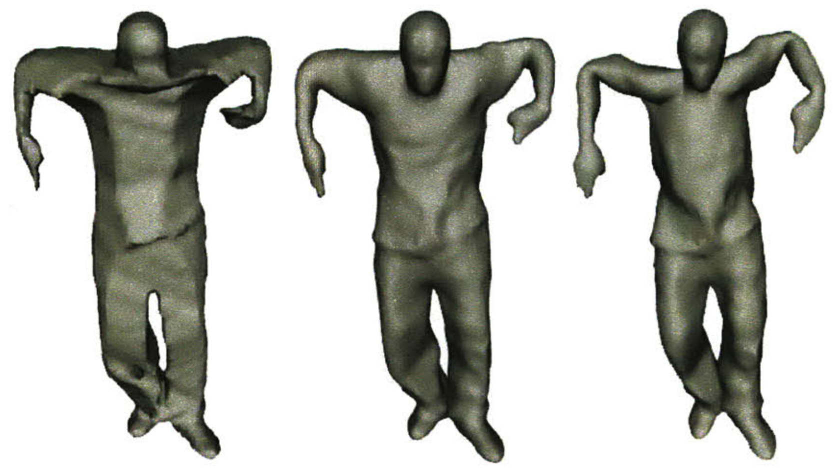


Figure 2-7: Our reconstruction has more detail than the visual hull and fits the data better than the template deformed using LBS.

2.5 Results

We tested our method on eleven sequences with five subjects captured in two different studios. The accompanying video demonstrates the ability of our method to produce meshes that capture fine motion details for very fast and challenging motions. Figure 2-10 shows reconstructed shapes from five of these sequences. We evaluate our algorithm according to several criteria: computation efficiency, the amount of user interaction needed, and robustness to parameter changes. We demonstrate the utility of our resulting meshes by applying a static texture and a geometric deformation to some sequences.

2.5.1 Experimental Setup

We have data sets from two different sources. In both cases the setup consists of a ring of eight cameras looking down at the performer. The first data set was provided by Starck and Hilton [134]. Their video streams are recorded at 25 FPS at 1920 by 1080 pixel resolution. The second source was our studio where we capture video at 1600 by 1200 resolution also at 25 FPS. We calibrated our cameras using an LED and software by Svoboda et al. [137].

The template mesh may be obtained by various means, such as 3D scanning [93], static multi-view stereo of a single frame [129], or manual modeling (as in [109]). For the Starck and Hilton data set, we use a good frame obtained by their multi-view stereo method [134] as the template. For our data sets we use a 3D scan of the performer (see Figure 2-10) obtained with a Cyberware scanner. Each template mesh was decimated to 10,000 vertices.

We manually built 40-DOF skeletons to fit the dimensions of each performer and embedded them within the template meshes (Figure 2-3, left). We attached each mesh to its skeleton with linear blend skinning using weights generated with Pinocchio [11].

2.5.2 Performance

Computation The processing was done on a 2.4 GHz Intel Core 2 Duo with 2GB of RAM. For the 175 frame Samba sequence, the forward pass of pose estimation ran at 4.3 seconds per frame while the backward pass added another 3.3 seconds per frame (some computations are reused). Shape estimation took 4.8 seconds per frame, making the total processing time about 12.4 seconds per frame, or 36 minutes for the whole sequence. User supervision is only necessary for the forward pass, which took a total of 12.5 minutes for this sequence. Timings for all of the processed sequences are shown in Table 2.1.

Accuracy Our method enforces the surface to fit the silhouettes and retain the detail of the template. The average silhouette error (percentage mismatch between the reconstructed and captured silhouettes) is below 8% for our motions, and below 15% for Starck and Hilton motions. The average distortion (percentage edge length change between the reconstructed surface and the template) is below 13% for all motions. Per-sequence errors are reported in the rightmost columns of Table 2.1. We were unable to assess the 3D accuracy of the reconstructed surfaces, as there are no ground truth moving surfaces for our data.

Sequence	Total Frames	Fixed Frames	Total Time	Frame Time	Silhouette Error	Edge Error
Flashkick	250	4	57m	13.7s	11.64%	12.65%
Headstand	250	7	61m	14.6s	12.46%	11.50%
Kickup	220	1	53m	14.5s	12.73%	12.51%
Lock	250	13	64m	15.4s	14.10%	11.22%
Walkpose	66	0	16m	14.5s	11.05%	10.64%
Bouncing	175	4	40m	13.7s	6.11%	10.14%
Crane	175	0	32m	11.0s	6.27%	8.69%
Handstand	175	2	40m	13.7s	6.67%	13.27%
Jumping	150	2	32m	12.8s	7.18%	10.76%
Samba	175	0	36m	12.3s	7.44%	8.83%
Swing	150	5	33m	13.2s	7.78%	10.40%

Table 2.1: We tested our algorithm on eleven different motions. We captured the bottom six sequences, while the top five were provided by Starck and Hilton. For each sequence we report the total number of frames, the number of trames requiring user intervention, the total processing time, the per-frame processing time, the average silhouette error (the percentage of the reconstructed silhouette that did not agree with the captured silhouette, and the average edge error (the percentage change in edge length as compared to the template).

User Interaction The provided interface allows an experienced user to fix a frame with a poorly fitted skeleton in under 20 seconds most of the time. As the user drags a joint constraint, the solver updates the skeleton several times per second, providing immediate feedback. Keyboard shortcuts let the user inspect the fit overlaid on the video images, allowing quick evaluation of the skeleton fit quality. We tracked each sequence and provided manual constraints necessary to reconstruct it correctly. Table 2.1 shows the number of frames that needed user constraints.

Sensitivity to Constants Our method depends on a number of manually chosen constants. These include error function weights for the pose estimation and the silhouette constraint parameters for the shape estimation. There was no need for extensive tuning of their values and we did not run into complicated interactions between them. We experimented with random changes to several of these parameters and found that the results are reasonable even when they are varied over a range of almost an order of magnitude.

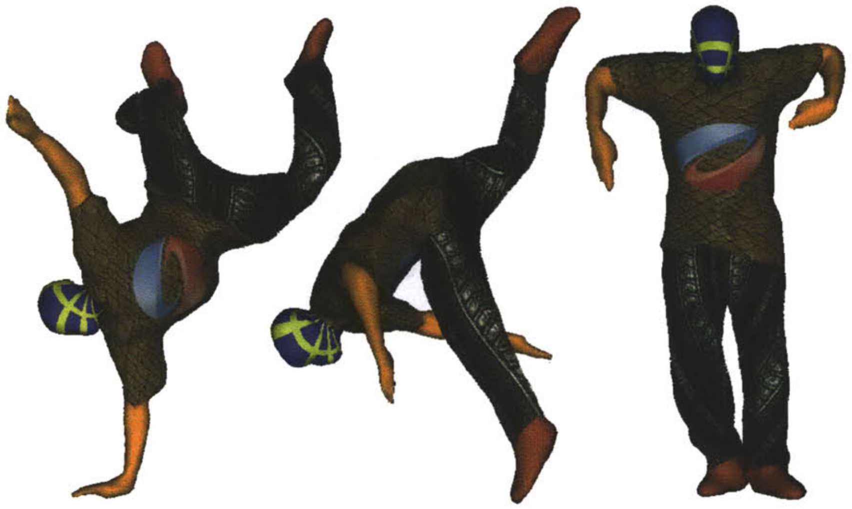


Figure 2-8: Frame-to-frame correspondence allows a static texture to be easily applied to the entire motion.

2.5.3 Editing

The reconstructed mesh sequence is ideal for editing the entire motion. A texture applied to the template is trivially propagated throughout the motion (Figure 2-8). A geometry edit to the mesh can be propagated through all or part of the motion using deformation transfer (see Figure 2-9) [136] or Kircher and Garland's method [87]. It is also possible to train a better skinning model than LBS such as [157] and use it with new skeletal data or to generate new poses (Figure 2-1, right), using deformation transfer to apply details that the skinning model does not capture.

2.6 Discussion

Our approach demonstrates that by taking advantage of geometric information in the silhouettes, we can obtain detailed mesh animations with full correspondence and correct topology. The output mesh animations are suitable for editing with methods such as texturing and deformation transfer. The key insight is to use skeletal pose estimation for gross deformation followed by iterative non-rigid shape matching to fit the image data. Moreover, an interactive mechanism for correcting occasional pose

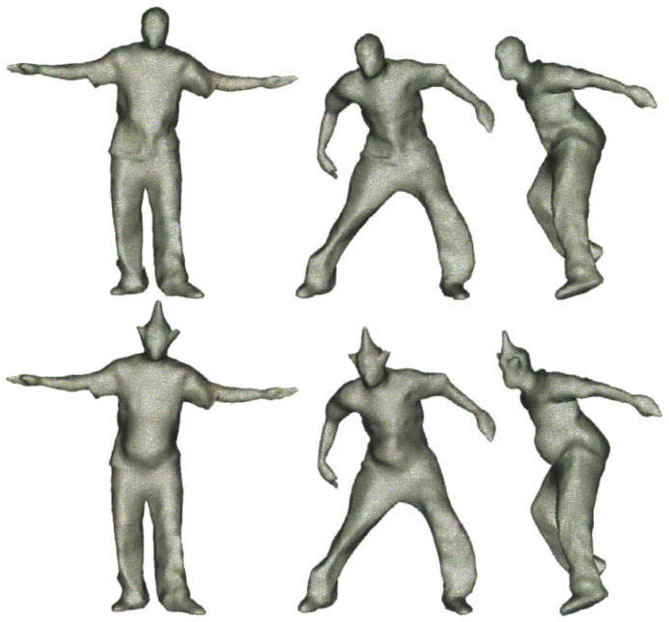


Figure 2-9: Top row: the template mesh and two frames of animation. Bottom row: the template mesh geometry is edited and the change is propagated to the two frames using deformation transfer.

estimation mistakes allowed us to obtain high-quality tracking of complex sequences with little effort. Our results show that silhouettes alone can convey rich and complex visual details despite inaccuracies away from the contours.

Limitations. Because our method only relies on silhouettes, it is completely immune to color noise, lighting, and color calibration problems. However, this reliance leads to two limitations. First, our method cannot reproduce the surface accurately away from the contours: it has to rely on the template to interpolate geometric information. This is especially problematic for unarticulated objects such as long scarves, faces, or flowing hair. Using color information in a manner that does not sacrifice robustness would improve their reconstruction. Second, our method is sensitive to errors in the silhouettes, and will produce incorrect geometry when visual hulls are noisy. However, this is not a serious problem in a studio setting where chroma-keying methods can be used to obtain clean silhouettes.

We found that the quality of the output animation strongly depends on the quality of the template. While a space-carved template can convey the character of the motion convincingly, the final appearance is much better for templates with accurate and high-resolution detail, such as those obtained with a laser scanner. Additionally, pose estimation is sensitive to the accuracy of the template skeleton proportions and degrees of freedom. Fine-tuning the skeleton is currently a manual task, but it may be possible to extract an improved skeleton automatically from our output sequences and use it to improve tracking of the same performer.

Another issue we have not completely addressed is texture sliding, which is still occasionally visible despite bilateral filtering. We believe that this problem traces back to temporally inconsistent ray constraints used in shape estimation. A possible remedy is to permit motions of a constrained point within the tangent plane of the silhouette cone. This would allow more natural configurations of the mesh and the resulting optimization procedure would still be efficient to solve.

Future Work. An interesting application of our results would be to learn a model of shape motion that takes clothing and dynamics into account. Such a model could simplify mesh animation from skeletal motion capture data and retain the details acquired during performance. Another possible future direction would be to provide an interface to edit and clean up the output mesh sequences, similar to our user interface for pose correction. Standard mesh editing techniques are not well suited for this purpose because they do not account for temporal consistency. An integrated mesh-editing framework could take advantage of the video data to assist the user and improve reconstruction.

We hope that we can support further improvement of mesh capture and processing techniques by sharing our data with other researchers (`http://graphics.csail.mit.edu/mesh_animation/`).

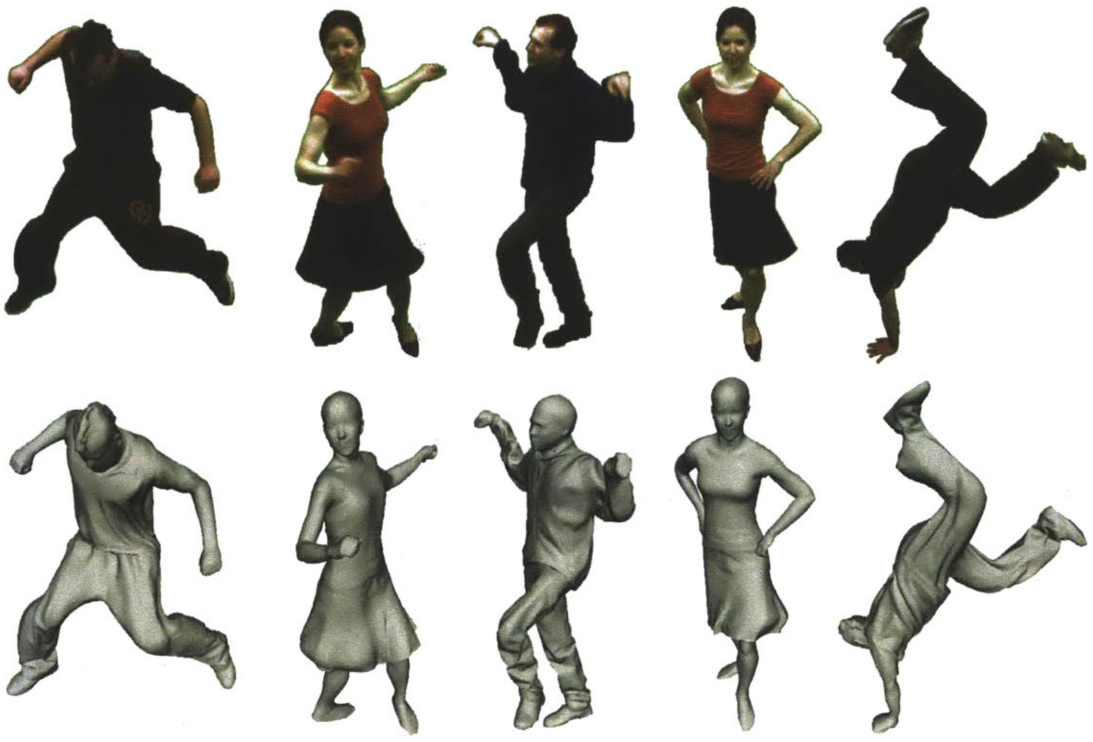


Figure 2-10: Background-subtracted video frames (top) and the corresponding recovered shapes (bottom) for three subjects in five sequences.

Chapter 3

Reconstruction of Dynamic Shapes from Photometric Normals

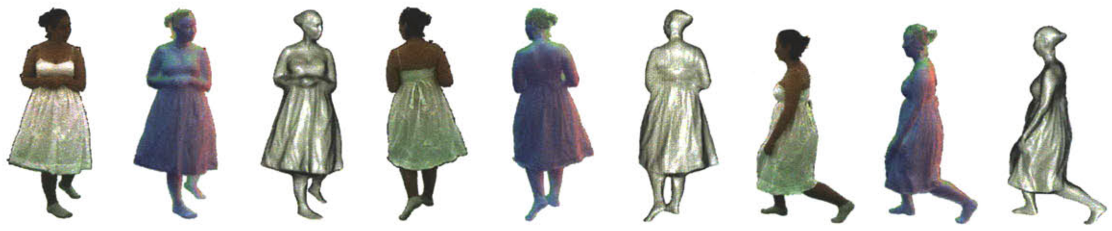


Figure 3-1: Our system rapidly acquires images under varying illumination in order to compute photometric normals from multiple viewpoints. The normals are then used to reconstruct detailed mesh sequences of dynamic shapes such as human performers.

In the previous chapter we have obtained moving 3D surfaces by deforming a template shape to fit the silhouettes from multiple views. This approach was very efficient and yielded compelling reconstructions of deforming skin and flapping clothes. However, since the silhouettes do not convey any surface information in their interior, the reconstructed surface detail came mainly from the template. In this chapter, we describe a *practical* system that measures the photometric normals and uses them to reconstruct the detailed moving surfaces of dynamic shapes. Since the normal maps (as opposed to silhouettes) contain an abundance of surface detail, we no longer need a template shape, which makes this approach more flexible. In addition, the reconstructed surface detail closely matches the true detail, i.e. we obtain the correct folds and wrinkles on the clothes. The large amount of measured data increases the memory and computation requirements of this method, while the lack of a template means that, due to visibility, portions of the surface cannot be reliably recovered. Nevertheless, this method yields moving surfaces of unprecedented quality.

Our system captures highly detailed 3D geometry of an actor's performance at sixty frames per second. We begin by acquiring normal maps from a small number of views (i.e., 8 or 9), using a novel variant of photometric stereo based on a small set of view-independent time-multiplexed light patterns produced by a large lighting dome (Section 3.2.1). In contrast to many existing systems, we use only four spherical lighting patterns (Section 3.2.2) to obtain a frame of geometry, which offers numerous practical advantages in acquiring detailed geometrry of full-body performances. The spherical nature of the lighting patterns ensures that each camera, irrespective of its relative location, gains photometric information at each frame. This makes the patterns particularly well-suited for capturing performances from all viewpoints. Furthermore, the large extent of the lighting patterns improves the robustness of the light/camera calibration, as well as allowing the irradiance to be better distributed than with bright point-light sources, and improving actor comfort.

Our reconstruction pipeline is split into several stages to make the large amount of data more manageable. Our algorithm processes the multi-view normal maps together with the corresponding silhouettes to produce high-resolution meshes independently for each time frame. We first integrate each normal map to obtain an initial surface per view, and estimate the locations of depth discontinuities (Section 3.3.1). The visual hull, obtained as the intersection of silhouettes, provides constraints on the integration as well as an initial rough estimate for depth discontinuities. Next, we match the partial surfaces and deform them to improve their mutual fit (Section 3.3.2). We evaluate several matching metrics, including ones based on local shape rather than color. Finally, we use volumetric merging to combine the separate views into a single surface (Section 3.3.3). We use this surface as an improved proxy replacing the visual hull in a second pass. The final meshes are computed independently between time frames, and recover the majority of a dynamic shape with high quality at sixty frames per second.

Compared to existing systems, our work represents an advance in the combination of high frame rate (60 Hz.), large working volume (human-size), and high spatial resolution (millimeter-scale) necessary for practical, high-quality performance capture. To achieve this, we make contributions in both the capture and reconstruction stages of our pipeline. First, we obtain high-quality high-frame rate normal maps using photometric stereo with spherical lighting. This photometric configuration is more comfortable to the subject, and is easier to construct, calibrate, and time-multiplex than other active illumination schemes. Second, we present a pipeline partitioned into stages that avoids expensive data-intensive optimization strategies on which some recent methods rely. This is more efficient in storage and computation, is trivially parallelizable, and scales well for large data volumes. Third, we demonstrate that matching neighboring views using a surface-based metric yields better and more robust correspondences than image-based approaches, for a sparse view sampling with a wide baseline such as ours.

We anticipate that data from our system (`http://graphics.csail.mit.edu/dynamic_shape/`) will have immediate impact on several problems within geometry processing and physical animation. First, because of our high resolution and our ability to capture surfaces without color detail, we obtain data suitable for analyzing and validating physical simulation models for materials such as cloth. Second, our data is ideally suited as input to algorithms that perform temporal registration and merging of geometry, which have recently received considerable interest in the geometry processing community [156, 83, 95, 131, 170, 155]. Looking ahead, we believe that it will be the combination of capture systems such as ours and temporal processing algorithms that will enable detailed full-body performance capture, resulting in even more believable virtual humans in movies and games.

3.1 Previous Work

Our methods are related to work from the following four research areas: (1) multi-view stereo and volumetric carving, (2) real-time structured light, (3) template-based approaches for performance capture, and (4) multi-view photometric stereo. We will describe the most relevant work in each of these categories.

Multi-view (Wide-baseline) Stereo and Volumetric Carving Multi-view stereo and volumetric space carving approaches use multiple, sparsely spaced cameras to observe a scene from different viewpoints. The color information is employed to carve out the space until the scene is photo-consistent [130] or the color information is matched along multiple epipolar lines [107]. Recent approaches additionally employ global optimization to take into account smoothness constraints as well as the silhouettes of the object [61, 154, 70, 82]. A comprehensive evaluation of many approaches has been conducted by Seitz and colleagues [129].

Several characteristics of these methods make it difficult to apply them to high-quality dynamic motion capture. In particular, their performance is highest with dense texture and many camera viewpoints. However, for dynamic capture the number of camera locations is often limited (as opposed to static capture, for which a single camera may be moved to many locations), and the viewpoints must contain separate physical cameras (whose geometric and photometric properties must be calibrated). Although there have been systems that use more than 50 cameras [122], a typical system for dynamic scene acquisition [134] might use only 8 cameras. Due to these challenges, reconstruction quality for moving surfaces has typically been lower than for static objects.

Real-time Structured Light Several methods capture dynamic facial performance geometry using structured illumination, usually emitted from a video projector onto the subject, and observed from a single viewpoint [125, 45, 171, 172]. While these systems can achieve impressive results for small (e.g., face size) working volumes, extending them to the two-meter working volumes required for full-body performance capture and to multi-view point capture is difficult due to numerous technical limitations of video projectors, such as limited spatial resolution, limited depth of field, and diminishing light levels with increasing working volume.

Template-based Approaches for Performance Capture Geometric templates (3D models of the subjects) can be used to aid in performance capture, for example by deforming them to match (possibly sparse) multiview data, or using them for hole-filling and parameterization. Caranza et al. [33] use a generic template and deform it using silhouette data from different view points. Theobalt et al. [138] further estimate surface reflectance and dynamic normal maps. Corazza et al. [40] also use a generic template, but deform it to the visual hull. Bradley et al. [22] combine multi-view stereo and a template to obtain moving garments. Starck and Hilton [133] employ silhouettes, stereo, and feature cues to refine a generic humanoid model. Balan and colleagues [9] use silhouettes from multiple viewpoints to estimate parameters of SCAPE [6]—a low-dimensional geometric body model derived from a large collection of static 3D scans. Zhang and colleagues [171] capture facial performances using a multi-camera and projector system. The resulting 3D geometry is regularized with a 3D face template. Most recently, some approaches have used high-quality, person-specific static 3D scans as templates [50, 150, 47]. These are deformed by tracking points on the surface of the object or by using silhouettes and photometric constraints.

The main advantage of template-based methods is that the prior model provides the correct connectivity and topology for the output mesh sequence. This minimizes major artifacts, while also reducing the search space and consequently running time. Furthermore, temporal correspondence is automatically provided since the meshes for all frames share the same parameterization. Therefore, missing data can be interpolated from other frames. However, template-based approaches have significant disadvantages. The quality of the output is low when generic 3D templates are used. The templates do not capture many person-specific details. Using high-quality, person-specific templates requires using an additional 3D range scanner (e.g., Cyberware). Furthermore, deformation details (such as cloth folds) can be baked into the template, and it is difficult to modify or remove them as the corresponding geometric features appear and disappear during the performance. Finally, template-based approaches cannot deal with atypical geometry and props often used during performances.

Multi-view Photometric Stereo The work on multi-view photometric stereo is the most similar to ours. Bernardini et al. [15] combine 3D range data with multiple normal maps acquired from different viewpoints. These normal maps are used directly during rendering and are not fused with the range data. Nehab and colleagues [104] fuse range data with a single normal map obtained using photometric stereo. The resulting geometry preserves high-frequency details from the normal maps while taking on the overall shape (i.e., low frequencies) of the range data. Nevertheless, this method still requires initial 3D geometry of reasonable quality, and does not address how to combine data from multiple normal maps.

Ahmed et al. [2] use a template and silhouettes to estimate the large-scale geometry of a performance. Geometric details are added by estimating normals, simultaneously with reflectance properties. In contrast, we do not require geometric templates or reflectance estimation. Campbell et al. [31] jointly incorporate data captured from multiple views to reconstruct a single shape using a volumetric approach. While elegant in theory, these approaches scale poorly to larger datasets. For example, to obtain millimeter precision in a two meter working volume requires a 2000^3 volumetric grid, which easily occupies several gigabytes of memory. Furthermore, solving a large volumetric optimization rapidly becomes computationally intensive.

Lim et al. [96] use a sparse set of 3D features to construct a rough depth-map. Using this depth-map, normal directions are computed for each depth-map location. While the quality of the results is high, the method relies on the existence of detectable features, which are not always readily available. Joshi and Kriegman [85] employ a graph-cut method to find dense correspondences based on a multi-view matching cost function that combines multi-view and photometric stereo. A depth-map from these dense correspondences is fused with photometric normals to yield a high quality shape using a non-iterative method. Unlike the method presented here, this method requires having a large number of views (at least 3) of each surface point, and is limited to 2.5D shapes.

The most similar methods to ours are those by Vogiatzis et al. [153], and Hernandez et al. [79]. These methods combine shading and silhouettes from different viewpoints, acquired with a turntable, to derive a 3D geometric model. In contrast, our approach captures dynamic scenes and requires fewer views (8 to 9 as opposed to 36). The method of Vogiatzis et al. performs a local search (essentially gradient descent) to establish correspondences between views. While this is appropriate if adjacent viewpoints are spatially nearby, it can fail to converge to the right correspondence if cameras are 45 degrees apart, as in our system. Furthermore, these methods rely on accurate silhouettes, which can be difficult to obtain in multi-camera setups with active illumination. Finally, our approach explicitly deals with surface discontinuities, which is necessary to obtain correct surfaces for nontrivial performances.

Summary We argue that practical capture of dynamic geometry requires a method that:

- reconstructs full 3D geometry (as compared to 2.5D depth maps) using a small number of views.
- does not use templates, and can handle arbitrary topology.
- remains computationally tractable for high working-volume-to-resolution ratios (several thousand to one) and high frame rates.
- handles fast and complex motion.

The tradeoffs and design decisions made in designing our system are based specifically on satisfying *all* of these requirements.

3.2 Hardware System and Image Processing

In this section we will discuss our hardware system, and the necessary image processing to compute high-quality normal maps for multi-view dynamic performance capture. We start by describing the hardware setup and calibration in Subsection 3.2.1. Next, in Subsection 3.2.2 we introduce a novel photometric normal estimation algorithm and its corresponding active illumination conditions. We conclude this section by a detailed discussion of additional optimization strategies and implementation details.

3.2.1 Acquisition Setup and Calibration

Setup We employ a variant of photometric stereo to compute per-camera and per-pixel normal information. This requires an active illumination setup. We use a device similar to the system built by Einarsson and colleagues [58]. This lighting device consists of the top two-thirds of an 8-meter, 6th-frequency geodesic sphere with 1,200 regularly-spaced individually controllable light sources, of which 901 are on the sphere and the rest are placed on the floor. A central area is reserved for the subject, and therefore it does not contain any floor lights. We capture dynamic performances at a 1024×1024 resolution with eight Vision Research V5.1 cameras. The cameras are placed on the sphere around the subject, at an approximate height of 1.7 meters relative to the central performance area. An optional ninth camera looks down onto the performer from the top of the dome. The performances are captured at a constant rate of 240fps, and the geometry is acquired at an effective rate of 60fps. Figure 3-2 shows our capture setup, with two selected cameras marked in red and the performance area marked in green.

Calibration Our system requires geometric and photometric calibration of all cameras. We use the LED waving technique of Svoboda et al. [137] in order to calibrate the intrinsic and extrinsic camera parameters. We photometrically calibrate the cameras by capturing a Macbeth Color Checker under uniform illumination and then solve for the optimal color transfer matrix for each camera.

Silhouettes Our geometry processing algorithms require silhouettes and corresponding visual hulls of the subject in order to provide an initial guess for the surface

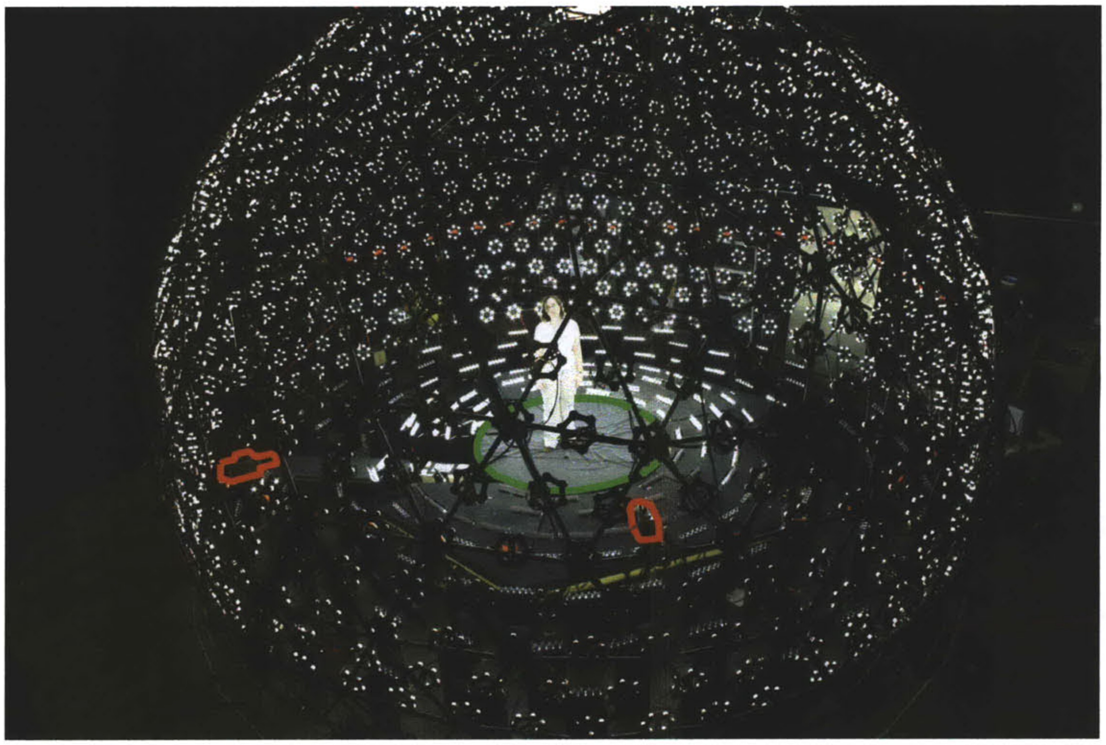


Figure 3-2: Our acquisition setup consists of 1200 individually controllable light sources. Eight cameras (two of which are marked in red) are placed around the setup aimed at the performance area (marked in green). An additional ninth camera looks down from the top of the dome onto the performance area.

reconstruction. We use a combination of background subtraction and chroma-keying to automatically extract approximate silhouettes. Though higher quality could be obtained with user assistance, this would be impractical (because so many frames need to be processed) and also unnecessary, since the resulting visual hulls are only used as a rough guide in the initial phase of the geometry reconstruction. However, it is important that the silhouettes not contain spurious holes, so small gaps in the foreground are detected and filled by comparing color statistics (i.e., average and variance) inside and outside the hole.

3.2.2 Multi-view Photometric Normals

Illumination Design Simultaneously acquiring images for computing photometric normals from multiple viewpoints imposes specific conditions on the design of active illumination patterns. First, capturing data-streams from multiple cameras produces a huge amount of data. Our main objective is to minimize the number of required lighting conditions, and thus the number of captured frames. This allows us to maximize an effective frame rate of our capture system and enable using non-specialized (i.e., high-speed) cameras at sufficiently high resolutions. Second, longer exposure times cause motion blur which degrades the quality of the normal estimation. Therefore, to minimize motion blur, we need to minimize camera exposure, and consequently maximize the light levels on the subject. Large area light sources make it easier to maintain a high total light intensity, while remaining comfortable for the subject. Third, to obtain high quality normal estimates, we would like to maximize the signal-to-noise ratio of our measurements.

While conventional photometric stereo [168] is able to estimate normals in a wide range of applications, it has the disadvantage that it only uses a single light source at a time to illuminate the subject. This can result in a low signal-to-noise ratio (i.e., many pixels with low intensity values). Furthermore, a significant number of light sources needs to be placed around the subject in order to obtain a sufficient number of unoccluded directional samples for each pixel in each camera. Recently, Ma et al. [98] proposed to use spherical gradient illumination to compute per-pixel normal information. The spherical gradient patterns cover the full sphere of incident lighting directions. They are well suited for multi-camera systems and result in a better signal-to-noise ratio. However, these patterns require careful calibration of the emitted illumination conditions such that they exactly conform to the theoretical gradients (both geometrically and radiometrically).

Inspired by [98], we propose a novel set of binary half-on illumination patterns that cover half of the sphere of incident directions. They are tailored to efficiently compute photometric normals for multi-view performance captures. Specifically, we

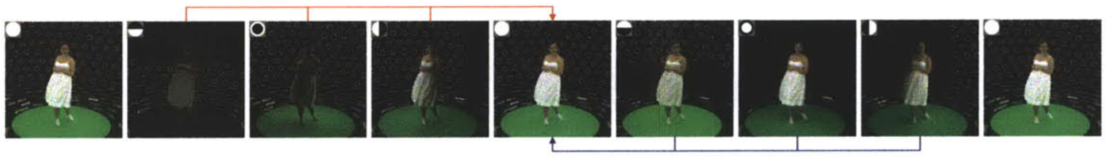


Figure 3-3: Captured frames under binary half-on illumination patterns. A complete set plus an additional full-on condition is shown. The insets depict the illumination condition used in each frame. The red and blue arrows indicate the forward and backward motion compensation respectively. High-quality geometry is reconstructed for every full-on tracking frame.

employ two sets of 3 illumination patterns. The first set consists of three patterns $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$ defined by

$$\mathbf{X}(x, y, z) = \begin{cases} 1, & \text{if } x > 0, \\ 0, & \text{otherwise,} \end{cases} \tag{3.1}$$

with $x^2 + y^2 + z^2 = 1$. $\mathbf{Y}$ and $\mathbf{Z}$ are similarly defined. The second set consists of the complements $\bar{\mathbf{X}}, \bar{\mathbf{Y}}$, and $\bar{\mathbf{Z}}$ of the first set, i.e. $\bar{\mathbf{X}}(x, y, z) = 1 - \mathbf{X}(x, y, z)$. We also add a full-on tracking frame $\mathbf{F}$ once every four frames in order to improve the temporal alignment and to compensate for motion over the multiple lighting conditions. To summarize, we illuminate the subject repeatedly with the following eight illumination patterns: $[\mathbf{X}, \mathbf{Y}, \mathbf{Z}, \mathbf{F}, \bar{\mathbf{X}}, \bar{\mathbf{Y}}, \bar{\mathbf{Z}}, \mathbf{F}]$. Figure 3-3 shows a subject under these illumination conditions.

Normal Estimation Let us initially assume that there is no subject motion during eight consecutive frames; furthermore, let's assume that all surfaces are diffuse. We can reconstruct a normal for each surface point based on the observed radiance under eight lighting conditions. This requires solving a system with three unknowns: the normal direction (2 unknowns), and surface albedo (1 unknown). While it is possible to compute an analytical solution for this system, the solution will be dependent on how accurately the physically emitted illumination conditions match the assumptions (in intensity and geometrical configuration).

Instead, we use a data-driven approach that improves robustness and facilitates calibration. By capturing a known shape with a known BRDF during a calibration step, we can establish a relationship between the observed radiance and normal direction. In our case we use a grey diffuse sphere, and treat the conversion from observed radiance to normal direction as a multi-dimensional lookup problem where the key is defined as $\frac{\mathbf{k}}{||\mathbf{k}||}$, with $\mathbf{k} = [I_{\mathbf{X}} - I_{\bar{\mathbf{X}}}, I_{\mathbf{Y}} - I_{\bar{\mathbf{Y}}}, I_{\mathbf{Z}} - I_{\bar{\mathbf{Z}}}]$, and I_p is the observed radiance under illumination $p \in \{\mathbf{X}, \bar{\mathbf{X}}, \mathbf{Y}, \bar{\mathbf{Y}}, \mathbf{Z}, \bar{\mathbf{Z}}\}$. Normalizing the lookup key removes any dependence of surface albedo from the key.

During calibration we capture a grey diffuse sphere under the binary half-on illumination conditions. For each camera view we extract the sphere's pixels, and create a vector similar to $\mathbf{k}$. We then store these vectors in a kD-tree together with their respective normals. When estimating the normals of a performance frame, we create a similar normalized vector for each camera pixel, search for the best match in the kD-tree, and retrieve the associated normal. In order to further improve the quality and minimize the effects of measurement noise during calibration, we search for the N best matches, and compute the output normal $\mathbf{n}$ as the weighted sum:

$$\mathbf{n} = \sum_{i}^{N} \left(\max_{j \leq N} ||\mathbf{k} - \mathbf{k}_j|| - ||\mathbf{k} - \mathbf{k}_i|| \right) \mathbf{n}_i, \tag{3.2}$$

and renormalize it.

Our normal estimation algorithm has a number of advantages. First, the illumination conditions are binary and therefore they are easier to create in practice. Second, the number and positioning of the cameras is independent of the number and orientation of the lighting conditions. For any possible camera location, all lighting conditions yield sufficient information to compute photometric normals. Finally, this procedure requires very little calibration: a single photograph per camera, per lighting condition of a calibration object with known geometry. The calibration and normal computation is robust to modest variations in light source intensities and light source distribution. Furthermore, the computed photometric normals are in camera space, and thus *independent* of any (multi-view) camera calibration, further improving the robustness of the calibration. In light of the necessary complexity of our setup, this data-driven approach with easily calibratable sub-parts makes the whole acquisition process better manageable.

The presented data-driven method shares some similarities with [80], where an object of known geometry is used to assist in determining photometric normals. The main difference is that they assume a known (point source) lighting configuration and unknown BRDF, while we assume a Lambertian BRDF and an unknown lighting configuration.

3.2.3 Implementation

In this section we discuss additional implementation details that are necessary in order to compute high-quality normal maps.

Multiple Calibration Spheres In the previous section, we have assumed that the incident illumination generated by a given illumination pattern is the same in the whole performance area (i.e., the lights sources are at infinity). However, in the current system this assumption does not hold. For example, the lower third of the light-sphere is placed much closer to the subject. This creates a significantly different illumination depending on the distance to the floor. In order to compensate for this effect, we use multiple normal lookup tables, which depend on the position in the performance volume. We capture images of the grey spheres at 7 different heights. During normal estimation, we compute an output normal by linearly interpolating between normals computed from the two closest in height calibration spheres.

Motion Compensation In the previous section we have also assumed that the subject does not move during the capture of eight illumination conditions required to compute normal maps. In practice, this assumption also does not hold. In order to compensate for subject motion, we compute both forward and backward optical flow between consecutive tracking frames. By assuming a linear motion between full-on tracking frames, we flow the intermediate images under the binary illumination conditions to the central tracking frame. A normal map is then computer for every tracking frame. In our implementation we use the variational approach by Brox et al. [30] to compute the optical flow. We show the direction of the optical flow to a single key frame in Figure 3-3. The forward and backward flows are illustrated using the red and blue arrows, respectively. Note that because we have a tracking frame every 4 frames, and the two sets of illumination conditions are complementary to each other, we can compute 2 sets of normal maps per 8 frame cycle.

Optical flow is able to correct for most of the subject's motions. However, flow computation can fail near or at occlusion boundaries. Therefore, we estimate the confidence for both forward and backward flow. The flow confidence is computed for each pixel as the L^2 error of the difference between the tracking frame and the flowed neighboring tracking frame. If this error is below some threshold, we compute the normal as detailed before. Otherwise, the normal is not computed and marked as invalid. When reconstructing the final geometry, we rely on normal estimates from different viewpoints and hole filling to correct for invalid marked normals.

Impact of Albedo Another factor that can negatively impact the quality of the estimated photometric normals is the low albedo of the surface points. In particular, camera noise dominates when imaging surface points with low albedo. In this case, the estimated photometric normals become noisy. Similarly, oversaturated pixels lead to incorrectly computed normals. Therefore, we only estimate normals for pixels that have normalized intensities between 0.03 and 0.97. Since albedo is a view-independent quantity, we rely on hole filling to deal with surface points with low albedo.

3.3 Reconstruction

The multi-view normal maps reproduce the high-resolution geometric detail present in the surface. We combine the information from the normal maps to reconstruct a complete 3D mesh in a three-stage process. First, for each view separately, we integrate a surface from the normal map. During this process, the visual hull acts as a "proxy": it provides rough constraints for the overall position of the integrated surface. Second, we use a similarity metric based on illumination, reconstructed local surface shape, or both to match neighboring views and smoothly deform the integrated surfaces in order to improve their fit. Again, we use the visual hull as a proxy, this time to constrain the possible matches. Finally, we merge the matched surfaces into a single mesh and optionally fill in the holes. The resulting closed mesh is now an excellent approximation to the true shape of the surface, and in particular is a better surface proxy than the visual hull. Therefore, we may repeat all three stages of our reconstruction algorithm, using the new proxy mesh instead of the visual hull wherever needed. The output of the second pass of our algorithm is the final mesh, as we have observed little quality improvement from further passes.

3.3.1 Single-View Surface Reconstruction

We begin by integrating individual normal maps into partial surface meshes. We represent the surface as a depth image, centered at the corresponding camera viewpoint (i.e., the value at each pixel determines how far along the camera ray the surface point lies).

We pose the integration problem as an optimization process, in which the depth values are the unknowns and the observed normals provide constraints. In particular, we enforce that the 3D vector between two neighboring depth samples i and j be perpendicular to the average of the measured normals at those pixels. With only the normals as constraints, the reconstruction problem would be ill-posed: it would be possible to move the surface forward or back while still satisfying the constraints. Therefore, we use the visual hull or the proxy mesh to provide (soft) depth constraints on the reconstruction. The optimization is formulated as a linear system:

$$\arg\min_{\mathbf{z}} \sum_{i,j} \left((\mathbf{n}_i + \mathbf{n}_j)^{\top} (\mathbf{r}_i z_i - \mathbf{r}_j z_j) \right)^2 + \alpha \sum_i (z_i - \bar{z}_i)^2, \tag{3.3}$$

where for a pixel i: z_i is the distance to the surface along the corresponding ray direction $\mathbf{r}_i$, $\mathbf{n}_i$ is the measured normal, and $\bar{z}_i$ is a possible depth constraint at that pixel — the depth of the visual hull or the proxy, if available. The parameter

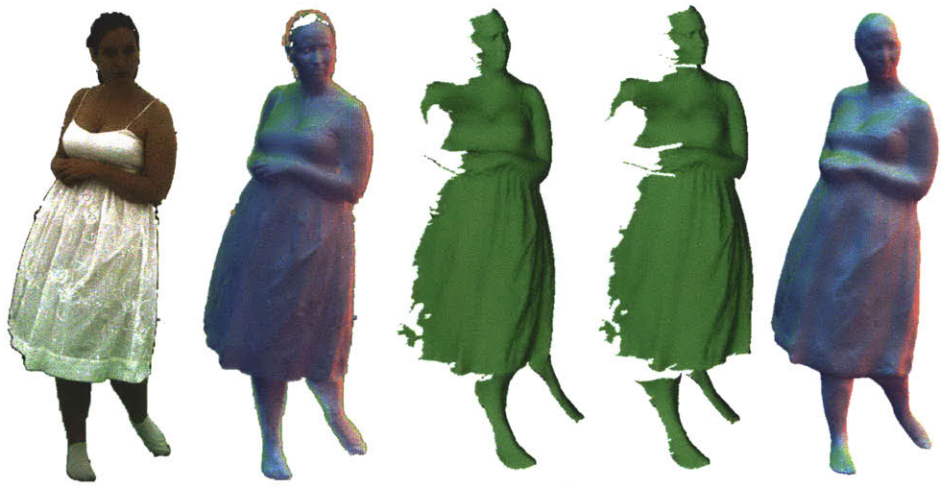


Figure 3-4: For a particular view (a), we use the normal map (b) in order to integrate the initial surface (c). Better surfaces are obtained when we detect large depth discontinuities (d). The normal maps after reconstruction are smoother than the original normal maps, but remove the initial bias, which can be seen in the legs (e).

α determines the relative strengths of the normal and depth constraints. We set it to be low (10^{-6}), corresponding approximately to the inverse of the number of our surface points. In the second pass, we expect the proxy mesh to be more accurate, so we increase the weight on the depth constraints by setting α to 10^{-5}.

The optimization process that computes surfaces according to Equation 3.3 does not intrinsically take into account depth discontinuities. However, integrating normals across depth discontinuities may cause significant distortions, as is evident in Figure 3-4c, where the legs are connected to the rim of the dress. In order to avoid this issue, we must remove the pixels straddling the depth discontinuities from the linear system (Equation 3.3). However, detecting depth discontinuities is a difficult problem. We have experimented with a variety of heuristics (e.g., maxima of color and normal gradients, local integrability measures), and have found that the following two simple strategies produce good results. First, large visual hull discontinuities are usually located near the true discontinuities. Removing pixels along them helps keep the surface free of large distortions. In addition to the large visual hull depth

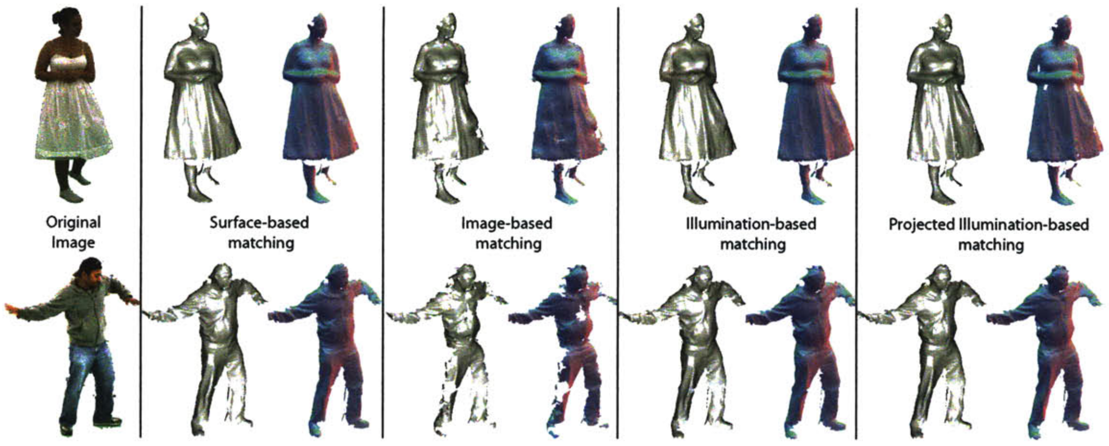

Original Image
Surface-based matching
Image-based matching
Illumination-based matching
Projected Illumination-based matching

Figure 3-5: Results (surfaces and normals) of several matching strategies on two data sets. Our surface-based metric yields the overall best results, especially in regions that lack texture or are shadowed. This is visible in the first row by zooming in and comparing the reconstructed legs. In the second row, only the surface-based metric does not push the waist inward.

discontinuities, our matching algorithm (described in the next subsection) identifies more precise discontinuities, defined as locations at which the surface jumps by more than 1 cm per pixel. Together, these heuristics alleviate depth discontinuity issues and allow the legs to integrate closer to the center of the skirt in our example (Figure 3-4d). In the second pass, all the detected discontinuities are closer to their true locations, making the integrated mesh even more precise.

Overall our method is fairly robust with respect to the quality of the visual hull. The accuracy of the visual hull mainly plays a role in determining the depth discontinuities. The closer the visual hull is to the true surface, the more complete our reconstruction will be. If the visual hull's depth discontinuities are far from the true depth discontinuities, then the pixels that are mistakenly connected to a wrong part of the performer (e.g., the leg connected to skirt) will be disregarded (and thus lost from the reconstruction) by the matching phase described below.

3.3.2 Pairwise Matching

Individual integrated surfaces contain the high-frequency details that are present in the normal maps. However, these surfaces rarely match exactly because of the bias in the normals (due to self-occlusion for example), and so cannot yet be merged into a single model. To correct for this distortion, we warp the surface based on matches computed between pairs of neighboring views.

Metrics for Matching We have experimented with a number of matching metrics to perform the correspondences, two based on images alone and two that rely on the integrated surfaces. Our first image-based metric uses only pixel windows under the spherical full-on illumination condition, and therefore reduces to traditional stereo matching. Our second metric is based on comparing an "image stack" of the six illumination conditions. We have found that this increases robustness in many areas that have little color texture, but significant geometric variation.

In practice, the performance of both image-based metrics is limited by different amounts of foreshortening between different views: different views of a region on a surface will not, in general, appear the same from different cameras. Therefore, we have also compared metrics that match the integrated surfaces. Since the distortion is low-frequency, we can assume that a small 3D surface patch in one view and the corresponding patch in the neighboring view will differ only by a rigid transformation. Therefore, we can compare small surface patches (e.g., 5×5 pixels) by computing the mean surface-to-surface distance under the optimal rigid-body alignment. Specifically, given a pair of surfaces ("left" and "right"), we compute the matching error between a point on the left surface and a point on the right surface as follows. First, we find a window of depth samples around the left point. Then, we render the right surface from the left camera's point of view, and find a window of depth samples around the projection of the right point. We assume that the two windows of points correspond, solve for the optimal rigid-body alignment, and compute the mean distance between the pairs of points in the windows. Then, to obtain a symmetric matching score, we repeat the computation with the roles of left and right reversed (i.e., rendering the left mesh into the right camera and finding the windows of samples there). We take the maximum of the two mean distances as the matching error.

Our final matching metric also relies on aligning surface patches, but takes the matching error to be the difference between image stacks (under different illumination) projected onto the surface. In areas of significant color detail, this improves discriminability over surface-distance-based matching, while retaining the advantage of compensating for foreshortening. However, we have found that in areas of little texture the surface-distance-based metric is superior.

We have compared these four metrics on a variety of datasets, as illustrated in Figure 3-5. As expected, the illumination-stack metric yields better surfaces than simple image-based matching, but, due to the wide baseline, still produces wrong matches. The projected-illumination metric further improves the reconstruction, but exhibits artifacts in shadowed regions. On the whole, we have found that the surface-based metric usually yields the least noisy surfaces.

Global Correspondence To find the best match for a point in the left view, we only need to explore the points in the right view that project to the corresponding epipolar line. Therefore, a simple matching strategy would be to simply take the point with minimum matching error as the correspondence. A more robust approach that takes into account surface continuity considers a whole epipolar plane (e.g., a plane passing through a point in the left view and both of the camera centers) at the same time. We sample this plane by stepping one pixel at a time in each view. We evaluate our matching error on the resulting grid and find the lowest-cost path from one corner to the opposite corner (Figure 3-6). In particular, we use the 4-step method described by Criminisi et al. [41] to perform the search. In addition to finding the best matches, this algorithm detects depth discontinuities. We use these depth discontinuities during surface integration. Note that we do not enforce smoothness *across* the sampled planes directly. However, some smoothness is indirectly incorporated because each surface patch spans several epipolar planes.

Good surface matches define absolute depth constraints at certain pixels in each of the views. After computing these matches for all views, we deform the integrated surface to fit these constraints while preserving its high-frequency detail. We achieve

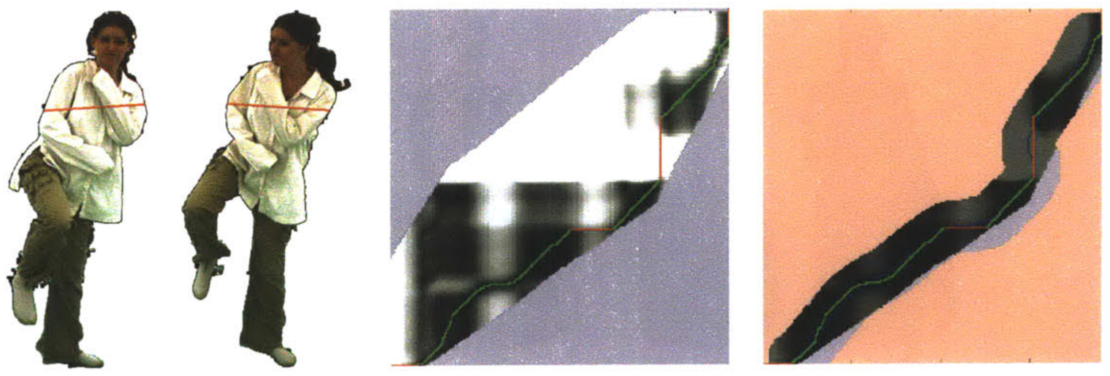


Figure 3-6: We match the surfaces in neighboring views one epipolar plane at a time (a). Each plane defines a grid of matching errors, where the lowest cost path yields the surface matches (b,c). The green portions of this path denote good matches, while red lines denote depth discontinuities. In the second pass (c), we use the proxy mesh (blue line) to guide this search and only evaluate the nearby matching errors. Light blue denotes points outside the visual hull, while points in pink are not considered in the second pass.

this with a thin-plate offset:

$$\arg\min_{\mathbf{d}} \sum_i \sum_j (d_i - d_j)^2 + \beta \sum_i (d_i - \bar{d}_i)^2, \tag{3.4}$$

where the depth offset d_i of each pixel i should be equal to the offset $\bar{d}_i$ obtained from the matching. These offsets are smoothly interpolated by pulling d_i toward the centroid of its neighboring offsets d_j. Because the thin-plate offset may introduce significant deformations in parts of the mesh that are far from the constraints, we discard these regions (using a threshold on distance to the nearest constraint point).

Applying the depth offsets obtained from Equation 3.4 aligns all surfaces much closer (Figure 3-7). The surfaces are still not perfectly aligned, since the matches are computed on a pixel grid. We address the remaining misalignment errors in the surface merging stage.

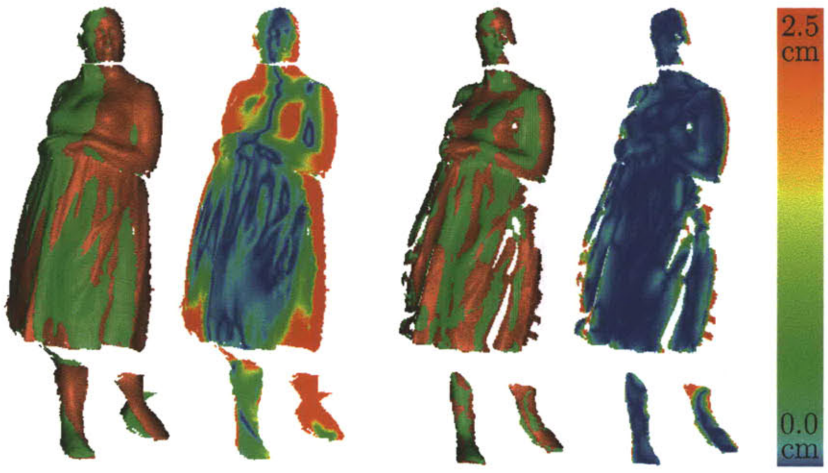


Figure 3-7: The integrated surfaces before matching (left) are far from each other, while the deformed surfaces after matching (right) are much closer to each other. Within each pair, the leftmost visualization shows the two meshes (in different colors) overlaid on each other, while the rightmost visualization is a color-coding of mesh-to-mesh distance.

3.3.3 Multi-view Surface Reconstruction

After the integration, matching, and deformation stages are performed on all views, we must merge the aligned, yet still logically separate surfaces into a single mesh. In addition, we must account for the regions in which no data was available, by performing hole-filling.

Merging We merge the eight aligned surfaces using the Volumetric Range Image Processing (VRIP) algorithm [42]. This is a volumetric method that allows for some residual misalignment between scans by averaging signed-distance ramps along the line-of-sight of each mesh. We set the ramp-length to 6 cm. to allow for worst-case misalignment, and reconstruct using a 2 mm. voxel size, which approximately matches the average resolution of our raw data. We also modify the weight computation of VRIP: in addition to weighting each point dependent on its distance to the nearest mesh boundary (to provide for smooth blending) and the cosine of its normal with the view direction (to downweight foreshortened data), we also include a term inversely proportional to the distance between the sample and its camera. The latter down-

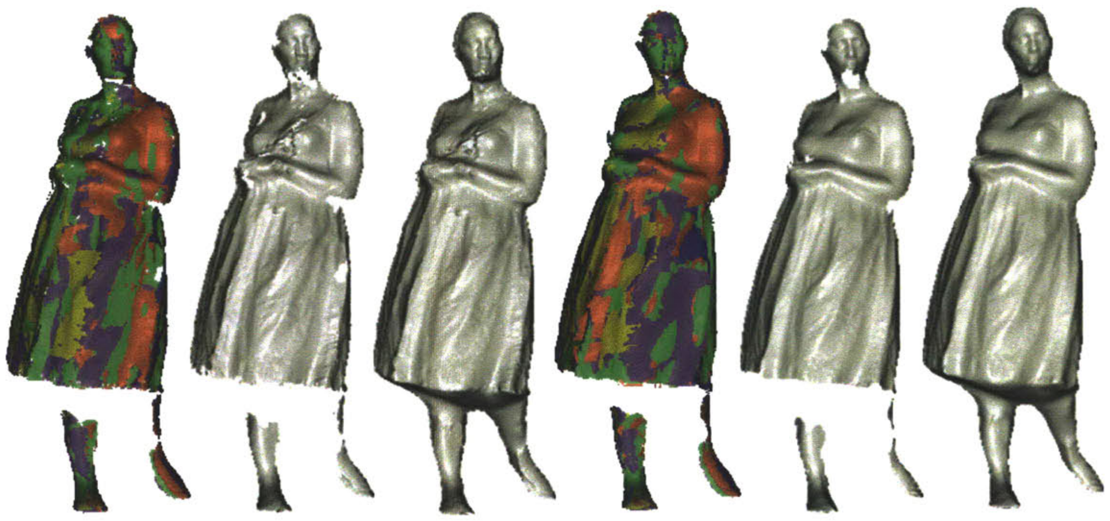


Figure 3-8: From left: false-color visualization of eight meshes after integration, matching, and deformation; result of initial volumetric merging; result of hole-filling; similar visualizations after second-pass matching.

weights regions of sparsely-sampled data, and is of greater benefit in our setup (in which the distance to the camera can vary significantly) than in typical 3D scanners. A sample result of the merging step is shown in Figure 3-8. In the first column, we show the meshes that are inputs to the merging, with the output of VRIP in the second column.

Hole Filling As can be seen, the merged mesh contains regions in which there is no surface, typically due to occlusion. Several approaches have been proposed to fill such holes, using techniques such as space carving [42] and volumetric diffusion [44]. In our case, we wish to combine information similar to that used in the two above techniques: we would like to fill small holes smoothly, yet for larger holes we wish to use the information present in the visual hull.

Our hole-filling approach uses the Poisson Surface Reconstruction algorithm of Kazhdan et al. [86]. As input, we provide oriented point samples taken from both the VRIP reconstruction and the visual hull, giving significantly lower weight (0.01) to the latter. We have found that this produces smooth fills, and draws the final surface towards the visual hull in regions of significant missing data. Moreover, the method guarantees a watertight manifold output, as shown in the result in Figure 3-8, third column. This mesh may now be used as the proxy mesh, instead of the visual hull, in the second pass of matching. The result of this second pass, together with the merged meshes, are shown in the right half of Figure 3-8.

We have also experimented with using Poisson Surface Reconstruction for both merging and hole-filling simultaneously (by using samples from the original meshes rather than the VRIP reconstruction). However, we have found that in regions of significant residual misalignment this results in more smoothing. This is because the Poisson problem inherently treats the influence of each point as isotropic, and hence does not preserve detail as well as VRIP's oriented signed-distance ramps when the merged surface is far from the original samples. We therefore believe that the combination of the two algorithms produces better results than either alone.

Using the visual hull for hole filling is noisy and can yield incorrect topology. This is why we only use it in the first pass, to produce a watertight proxy surface for the second pass. Our final meshes are suitable for filling using more sophisticated approaches, which exploit temporal coherence to aggregate information from multiple frames. This is an active area of research [156, 83, 95, 131, 170, 155], and lies outside the scope of this paper.

3.4 Results

We have acquired and processed five different sequences, including people wearing loose clothing, long skirts, and even a subject covered with a linen sheet. The reconstructions are presented throughout this paper and in Figures 3-1 and 3-9, which show a number of individual frames from our sequences. Note that the normals used for rendering are the geometrical (computed) normals, not the (measured) normals from the normals maps. Reprojecting and embossing photometrically measured normals would yield even better results, but would not represent the *true* quality of the reconstructeed geometry. Additional results, including complete animations, are presented

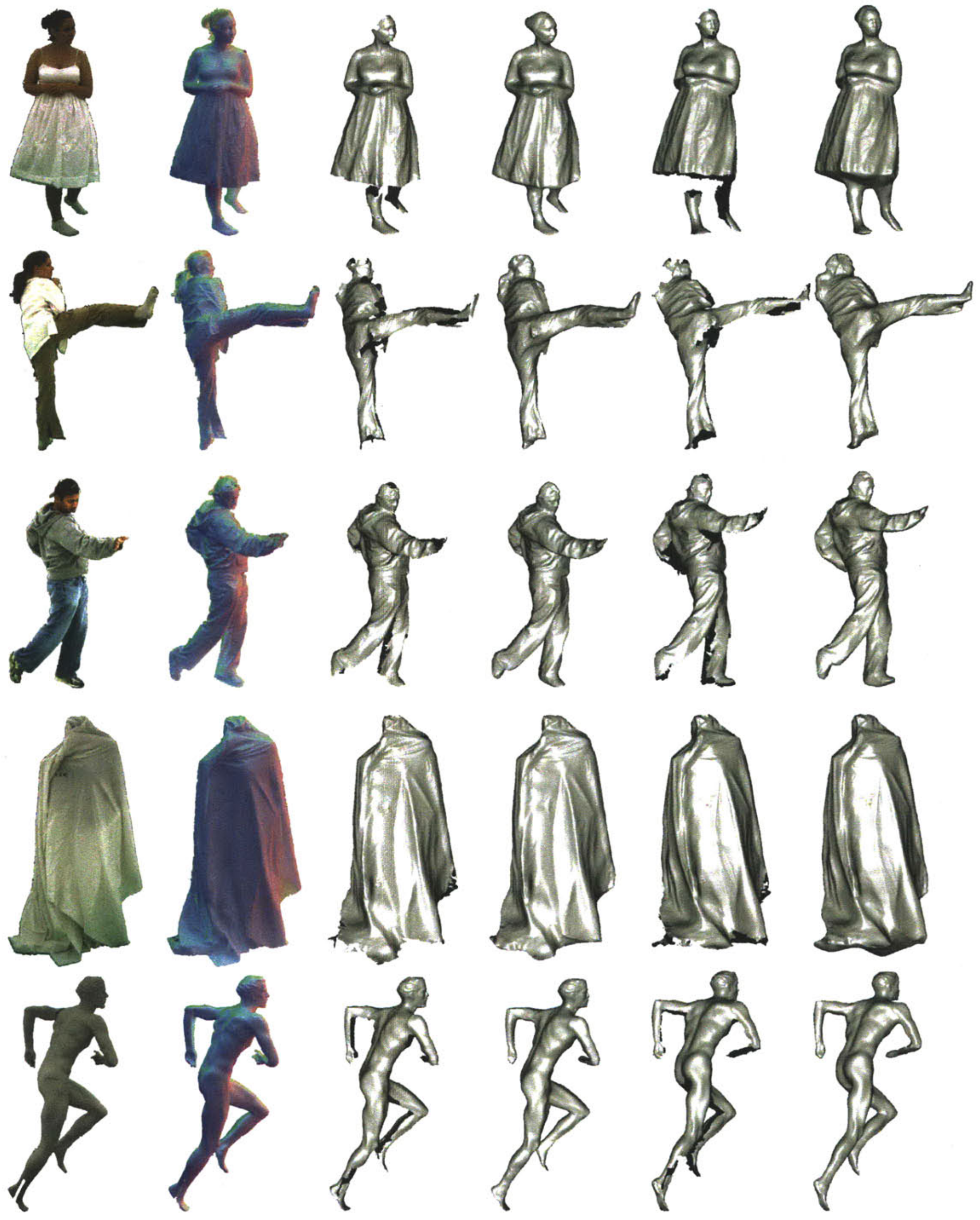


Figure 3-9: Each row shows an original image, the corresponding normal map, the reconstructed surface, the hole-filled surface, as well as a novel view of the reconstructed and hole-filled surface for a captured performance. Note that hole-filling was not applied to the versions of these results shown in the accompanying video.

Sequence	Coverage	Normal deviation	Normal deviation (local)
Abhijeet	85.90%	15.00°	1.89°
Ghost	92.26%	14.03°	1.26°
Jay	81.27%	14.76°	2.04°
Saskia	88.77%	12.18°	1.41°

Table 3.1: For each of our four test sequences we report (1) the percentage of measured normals that yielded an actual surface, (2) the angular deviation between the reconstructed and measured normals, and (3) the angular deviation between the normal difference within a vertex neighborhood and the corresponding difference in the measured normals.

in the supplementary video.

These sequences demonstrate that our system can correctly handle characters performing fast motions, as well as non-articulated characters for which template-based approaches fail. Many of our sequences have few or no textured areas, making them challenging for any type of stereo matching algorithm. In contrast, the analogous difficult situation for our algorithm is surfaces that lack geometric detail. However, in these regions, the smoothing performed by our algorithms is the correct action.

Computation Time We typically run the software on a 2.4 GHz PC with at least 2GB of RAM. The total computation time to obtain one final mesh is about an hour, and no user assistance is required. However, most parts of our code have not been optimized or parallelized, which leads us to believe that this time can be significantly reduced. In the current implementation, typical running times of the parts of the pipeline are: normal map computation with motion compensation: 50 min; first pass: 8 min; second pass: 5 min; and merging: 2 min. Normal map computation is dominated by the optical flow (two per camera), while the reconstruction spends most of its time computing the surface matches. While the total processing time is an hour per frame, it should be noted that this includes processing of the raw 240Hz video streams using non-optimized optical flow code. Using a GPU based optical implementation, or a less accurate but more efficient algorithm, could greatly reduce this pre-processing time. The actual processing time starting from the normal maps is only 10 to 15 minutes per frame.

Accuracy Table 3.1 summarizes several accuracy measurements. For each sequence we report the overall percentage of the measured normals that is covered by the reconstructed surfaces. About $10 - -15\%$ of the measured samples are not reconstructed due to matching difficulties. We also report the normal deviation between the measured and the reconstructed normals. This number is larger than desired (up to $15°$) due to the initial normal bias (Figure 3-4 b and e). The final column reports the local normal deviation: specifically, the angle between a vertex and its neighbors is compared to the angle between the measured normal for that vertex and the measured normals for the neighbors. As expected, this deviation is smaller (less than $2°$).

3.5 Discussion

In this work we have made the deliberate choice of not integrating multiple steps together to obtain a potentially more unified or optimized pipeline. Capturing and processing high quality detailed performance geometry is a complex task that requires significant amounts of hardware, calibration and processing, and rapidly produces huge amounts of data. We therefore try to keep our system as modular as possible. This greatly improves the robustness of the calibration, eases data management, and increases the amount of parallelism in the processing.

Our pipeline currently does not enforce temporal correspondences nor apply temporal filtering. However, we have observed only minor flickering in the matched regions of our meshes. This suggests that our reconstructions are quite close to the true surfaces. While temporal filtering could potentially be used to further smooth the results, it would also remove desirable details of mesh animations. Temporal registration of the acquired mesh sequences could improve the surface coverage by accumulating information through time. However, this is still a very active area of research [103, 126, 112, 35]. The choice and evaluation of a particular algorithm, or design of a new algorithm, falls outside the scope of this work. Nevertheless, we designed our acquisition and data processing pipeline so that these algorithms can operate as a post-process on our data.

Even though we do not perform any temporal processing, there is little flickering noticable when playing back the processed geometries of a performance. There is some flicker visible near the boundaries of each single-view surface, because data is increasingly bad there, and because the decision of whether to mark a pixel as a discontinuity is binary and independent per frame. Away from the boundary, however, the flicker is very low, showing that we are getting close to the true geometry. Since our resolution is very high (1,000,000 triangles, or 1 triangle per pixel), differences in triangulation do not produce noticeable artifacts either. Finally, the temporal coherence of the computed normals maps is very good to begin with, which reduces the need for temporal regularization.

A second type of coherence not currently exploited by our method is inter-scanline continuity. This could be enforced by using a Markov-Random Field formulation and employing an optimization method such as Belief Propagation or min-cut. However, this would require large amounts of memory and computational power.

Limitations. The computed normal maps show significant high frequency detail. However, these normal maps also have a significant bias. This bias is different then the bias observed in traditional photometric stereo (i.e., point light sources versus area light sources), and is especially present in the concavities, since in these regions a smaller-than-expected portion of the sphere of lights illuminates the surface. Furthermore, the normals are typically incorrect in the hair region (due to the complex scattering in the hair volume, which is not consistent with the Lambertian assumption made by photometric stereo), and the generally low albedo of hair. The normal maps are also noisy in areas of low albedo and in areas that were not properly aligned during the motion compensation stage. Figure 3-4e compares the acquired input normals from one of the viewpoints and the corresponding normal map of the final model. Observe the differences in the left leg, where the captured normals were corrupted by the shadow cast by the dress. This figure also demonstrates that our reconstruction pipeline introduces some smoothing of fine details: the original normal map is sharper than the reconstruction. This smoothing is introduced at several stages during the pipeline, including the surface integration, matching, and scan merging. Most of the original detail, however, is retained.

The current hardware setup is fairly complex, but it should be noted that the setup described is designed as a research prototype, not specialized to the task at hand, and can be refined to reduce complexity and cost. First, not every light source needs to be individually controllable. Only eight distinct groups require individual control, corresponding to the eight quadrants of the sphere of incident directions. Each illumination pattern would then light half of the groups at the same time. Second, using only 10% (i.e., 120) of the number of light sources is sufficient to accurately infer photometric normals from diffuse reflections. Both observations allow us to greatly simplify the hardware setup. Furthermore, we carefully designed the illumination conditions to require only a modest level of control and accuracy. Individual light source intensities do not need to match due to our data-driven scheme to infer normals. The only hard requirement is the ability to quickly toggle lights on and off, which is automatically achieved by using LEDs. Nevertheless, we expect our processing pipeline to work with alternative setups equally well. For example, six individually triggered flash lights aimed away from the subject at the surrounding walls could achieve similar illumination conditions as used in this paper, and could be directly used to estimate normals, because of the data-driven nature of our normal estimation algorithm. Finally, even though we employ high-speed cameras, our method is specifically designed to work at moderate frame rates that fall in the range of what is possible with readily-available hardware such as Point Grey's Grasshopper, which can capture at 200Hz with a 640x480 resolution.

Future Work. Though 3D capture of dynamic performances remains a challenging problem, this paper makes progress towards acquiring high-quality mesh animations of real-life performances. Our system uses novel hardware and image processing algorithms to obtain high-quality normal maps and silhouettes from multiple viewpoints at video rates. The surface reconstruction algorithms process this data to derive high-quality mesh sequences. The resulting mesh sequences can be used in biomechanics to analyze complex motions, in computer games to create next-generation characters, and in movies to create digital doubles.

We take advantage of the high-frequency geometric information present in photometric-stereo normal maps; therefore, our method significantly outperforms multi-view stereo techniques that produce overly smooth surfaces due to lack of texture or photometric calibration errors. Furthermore, our method does not require geometric templates as input and thus it is not restricted by their limitations.

The data produced by the system (`http://graphics.csail.mit.edu/dynamic_shape/`) will stimulate more work in the geometry processing community, whose research endeavors take a sequence of incomplete moving surfaces and produce the best-fitting, congruent, water-tight moving mesh. Further processing of these mesh sequences will prove challenging due to the shear amount of data our system is producing (each mesh contains more than 500,000 vertices).

Chapter 4

Reconstruction of Dynamic Shapes from Hybrid Sensors

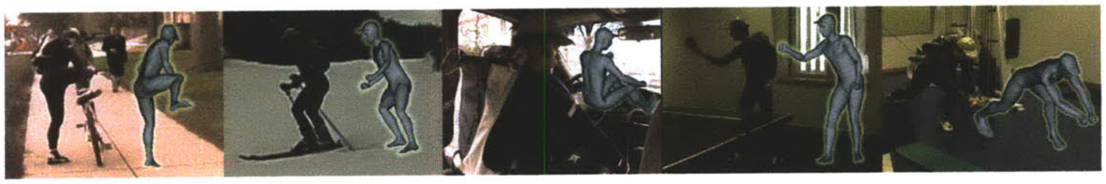


Figure 4-1: Traditional motion-capture systems excel at recording motions within lab-like environments but struggle with recording outdoor activities such as skiing, biking, and driving. This limitation led us to design a wearable motion-capture system that records human activity in both indoor and outdoor environments.

While the quality of systems that capture point and surface motions is ever increasing, the majority of existing acquisition systems inhibit broader use of motion analysis by requiring data collection within restrictive lab-like environments. As a result, motions such as skiing and driving are simply never acquired, while others like cycling and playing football are not recorded in their natural competitive setting. Furthermore, recording the activities, routines, and motions of a human for an entire day is still challenging.

In this chapter, we explore the design of a wearable self-contained system that is capable of recording and reconstructing everyday activities such as walking, biking, and exercising. Our design minimizes discomfort and maximizes recording time by prioritizing light-weight components with low power requirements. It records acoustic and inertial information from sensors worn on the body. Inertial measurements are provided by miniature gyroscopes and accelerometers no more than a few millimeters in size. Each sensor also includes a miniature microphone, which is used to record distances between pairs of sensors on the body. These distance measurements reduce the drift common to purely inertial systems.

The reconstruction algorithm estimates body postures by combining inertial and distance measurements with an Extended Kalman Filter that incorporates the body's joint structure. Although it lacks the information to recover global translation and rotation, our approach reconstructs sequences of full body postures that are visually similar to the original motions.

Our system is not the first acoustic-inertial tracker, but it is the first such system capable of reconstructing configurations for the entire body. We show that these reconstructions are most accurate when combining information from all three sensor types: gyroscopes, accelerometers, and distance measurements. The best reconstructions are still not perfect, but their quality, along with the small size and improved versatility, suggest that our system may lead to new applications in augmented reality, human-computer interaction, and other fields.

4.1 Previous Work

Several motion capture technologies have been proposed in the last two decades. The advantages and disadvantages of the dominant approaches are argued in several excellent surveys [100, 69, 81, 162]. In this brief summary, we review optical, image-based, mechanical, magnetic, inertial, acoustic, and hybrid systems, mentioning a few exemplary systems in each category.

Optical motion capture systems [167, 19] and modern systems manufactured by Vicon (*vicon.com*), Codamotion (*codamotion.com*), and PhaseSpace (*phasespace.com*), track retro-reflective markers or light-emitting diodes placed on the body. Exact 3D marker locations are computed from the images recorded by the surrounding cameras using triangulation methods. These systems are favored in the computer-animation community and the film industry because of their exceptional accuracy and extremely fast update rates. The major disadvantages of this approach are extreme cost and lack of portability. To reduce cost and improve portability, some systems use a small number of markers in conjunction with standard video cameras. For example, Yokokohji and colleagues [169] capture arm motions with a head-mounted camera.

Image-based systems [26, 46, 36] use computer vision techniques to obtain motion parameters directly from video footage without using special markers. These approaches are less accurate than optical systems, however, they are more affordable and more portable. Still, they are not entirely self-contained since they rely on one or more external cameras. Furthermore, they suffer from line-of-sight problems, especially in the case of monocular video.

Mechanical systems, such as Meta Motion's Gypsy™ (*metamotion.com*), require performers to wear exoskeletons. These systems measure joint angles directly (e.g., using electric resistance), rather than estimating the positions of points on the body, and can record motions almost anywhere. Exoskeletons are uncomfortable to wear for extended time periods and impede motion, although these problems are alleviated in some of the modern systems, such as Measurand's ShapeWrap™ (*measurand.com*).

Magnetic systems, such as MotionStar® by Ascension Technology Corporation (*ascension-tech.com*), detect the position and orientation using a magnetic field (either the Earth's magnetic field or the field generated by a large coil). These systems offer good accuracy and medium update rates with no line-of-sight problems. However, they are expensive, have high power consumption, and are sensitive to the presence of metallic objects in the environment.

Inertial motion capture systems, such as Moven (*xsens.com*) and ALERT system (*verhaert.com*), measure rotation of the joint angles using gyroscopes or accelerometers placed on each body limb [102]. Like the mechanical systems, they are portable, but cannot measure positions and distances directly for applications that must sample the geometry of objects in the environment. More importantly, the measurements drift by significant amounts over extended time periods. In addition, the motion of the root cannot be reliably recovered from inertial sensors alone, although in some cases this problem can be alleviated by detecting foot plants [64].

Acoustic systems use the time-of-flight of an audio signal to compute the marker locations. Most current systems are not portable and handle only a small number of markers. With the Bat system [159], an ultrasonic pulse emitter is worn by a user, while multiple receivers are placed at fixed locations in the environment. A system by Hazas and Ward [78] extends ultrasonic capabilities by using broadband signals; Vallidis [145] alleviates occlusion problems with a spread-spectrum approach; Olson and colleagues [108] are able to track receivers without known emitter locations. The Cricket location system [120] fills the environment with a number of ultrasonic beacons that send pulses along with RF signals at random times in order to minimize possible signal interference. This allows multiple receivers to be localized independently. A similar system is presented by Randell and Muller [121], in which the beacons emit pulses in succession using a central controller. Lastly, the WearTrack system [65], developed for augmented reality applications, uses one ultrasonic beacon placed on the user's finger and three fixed detectors placed on the head-mounted display. This system can track the location of the finger with respect to the display, based on time-of-flight measurements.

Hybrid systems combine multiple sensor types to alleviate their individual shortcomings. They aim to improve performance, rather than decrease cost and increase portability. For example, an acoustic-inertial system, Constellation^{TM}, has been developed for indoor tracking applications [66]. The system corrects inertial drift using ultrasonic time-of-flight measurements to compute exact distances between receivers and ultrasonic beacons placed at known locations. Another acoustic-inertial system [160] uses a wrist-worn microphone and a 3-axis accelerometer for gesture recognition. Similarly, MERG sensors [8] enable inertial-magnetic systems that account for the drift by using a reference magnetic field. In the same manner, Hy-BIRD^{TM} by Ascension Technology Corporation (*ascension-tech.com*) combines optical and inertial technologies to tackle occlusion problems. Finally, a combination of image-based and inertial tracking is used for sign language recognition [25].

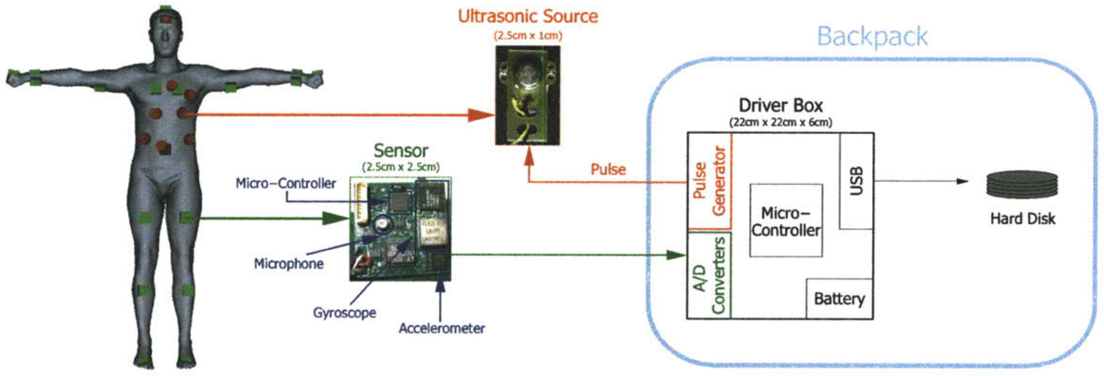


Figure 4-2: System Prototype. Our system consists of an array of small, low-cost, low-power ultrasonic sources and detectors (microphones) placed on the body (left). The ultrasonic sources (top-middle) sequentially emit ultrasonic pulses, which are received by the microphones (bottom-middle) and processed to yield distance measurements for all source-microphone pairs. To increase precision and the sampling rate, as well as to alleviate visibility problems, each sensor board is also equipped with a 3-axis gyroscope and a 3-axis accelerometer that measure rotation rates and linear accelerations respectively (bottom-middle). The data collection is managed by a small driver box (right) using a laptop hard disk for storage; both the driver box and the laptop are carried by the user in a backpack.

4.2 System Prototype

Our wearable motion-capture system consists of ultrasonic and inertial subsystems, a driver box that controls their operation, and a storage device that records the data. During operation, the data from the two independent subsystems (the ultrasonic subsystem used for distance measurements and the inertial subsystem used for measurements of accelerations and rotation rates) are acquired at each sensor board, encoded, and jointly transmitted to the driver box. The driver box samples the signals from all the sensor boards and transfers them onto the storage drive for off-line signal processing and pose estimation. This section describes our hardware and its operation at a high level. More details, including individual part numbers, can be found in the document by Adelsberger [1].

4.2.1 System Components

As illustrated in Figure 4-2, our system is controlled by a custom-built driver box connected to a laptop using a USB interface. The driver box is also connected to eight ultrasonic sources and eighteen sensor boards using shielded 3-wire cables. All sensors are attached to the user's garment. The driver box provides pulse signals to the ultrasonic sources, polls data from the sensors, and provides power to the sensor boards. It is powered by a rechargeable Lithium-Ion battery pack with 4.4AHr capacity, which provides several hours of safe operation (the current drawn by our system is 1.5A). A laptop records the data to a hard drive but is not used for any processing. We envision a commercial system that replaces the driver box and laptop, both of which are currently carried in a backpack, with a single iPod-sized unit.

After examining the current state of transmitter/receiver technology, we have determined that only acoustic (and in particular ultrasonic) components offer high precision, low cost, and small size. Therefore, our signal sources employ off-the-shelf piezoelectric transducers (Figure 4-2 top-center) to emit pulses at ultrasonic frequencies (40 kHz). They are mounted onto small plastic plates, attached to the garment, and wired to the pulse-generating driver box. The pulses are detected by conventional audio microphones (Figure 4-2, bottom-center). Although they do not exhibit optimal response in the 40 kHz range, they are able to clearly detect our ultrasonic pulses while offering several advantages over ultrasonic detectors: they are small in size (2.5mm^3); they have a wide-angle response — there is no need for accurate alignment with the sources; and they have a wide bandwidth — we do not need to tune them to the exact frequency of the ultrasonic source. We arranged the ultrasonic sources such that they see most of the microphones most of the time: seven sources around the chest and belly pointing forward, with the eighth source on the brim of a hat pointing down (Figure 4-2, left).

In addition to the microphone, each sensor board (Figure 4-2, bottom-center) is equipped with a 3-axis rotation rate sensing unit (the gyroscope), and a 3-axis linear acceleration sensing unit (the accelerometer). Their measurements enhance the pre-

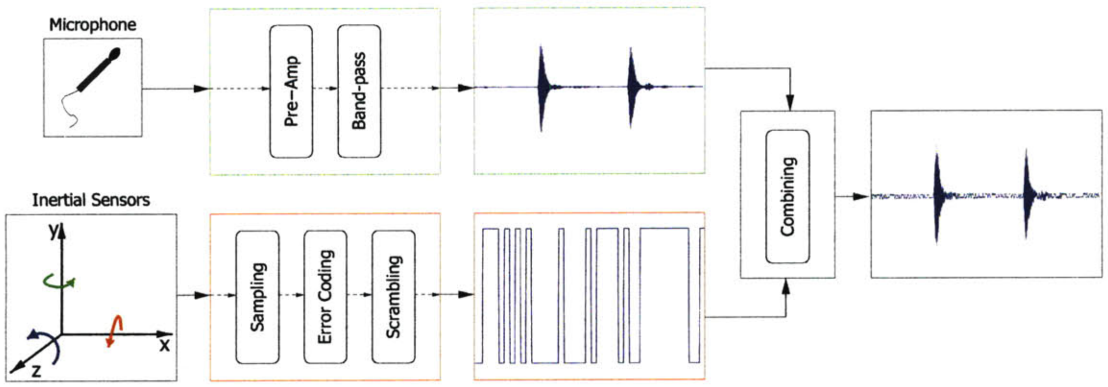


Figure 4-3: Sensor Operation. Our sensor boards combine acoustic and inertial data, and send the resulting signal to the driver box, which samples and stores it onto a hard disk. The acoustic signal sensed by the microphones (top) is amplified and filtered to enhance the quality of the ultrasonic pulses. At the same time, the inertial data from the gyroscopes and accelerometers (bottom) is digitally encoded by the on-board micro-processor, ensuring faithful reconstruction with error correction code and scrambling. The analog acoustic signal and the 13kHz digital inertial signal are multiplexed together and transmitted on a single wire.

cision and frame rate of the ultrasonic components. Furthermore, they alleviate the line-of-sight problems associated with acoustic signals. An on-board micro-controller collects the inertial data, combines it with the acoustic signal, and sends it to the driver box.

The driver box has three main tasks: to generate pulses that drive each ultrasonic source, sample the data from each of the sensor boards, and provide power to all inertial and ultrasonic components. As a result, all of our data is perfectly synchronized (we know exactly when the pulses are emitted with respect to each sensor signal). The sampling rate of the A/D converters in the driver box is about 150kHz, well above the Nyquist rate of the 40kHz ultrasonic pulses and the 13kbps inertial data (see below). In addition, the box houses a USB hub through which the sampled signals from each sensor board are transferred to a hard disk.

4.2.2 Ultrasonic Operation

Our ultrasonic subsystem operates similarly to a conventional acoustic ranging system, where there is a single source and a single detector. At regular intervals, the source emits a short burst of ultrasonic energy (a "pulse"), which is subsequently sensed by the detector. For example, our pulses are ten cycles wide at 40 kHz. The observed time delay ("time of flight") between the emission of the pulse and its detection is proportional to the distance between the two.

As the signal propagates through the air and bounces off objects in the environment, the detector will record several pulses at different times. The earliest detected pulse is the one that corresponds to the direct line-of-sight (LOS) and should be used to determine distance. The subsequent reflected pulses generally will be progressively weaker as they have to travel further through the air.

In our system, we also need to distinguish between pulses emitted by different sources. To accomplish this, the sources emit pulses at different times in a round-robin fashion (similarly to [121]). The time separation between pulses from different sources must be long enough to ensure that reflected pulses from one source are not mistaken for the LOS pulse from the next source in the sequence. We have selected a conservative time interval of about 8 ms between the subsequent pulses. At the average speed of sound, this corresponds to a distance of about 2.75m the pulse will travel before another pulse is emitted by another source. We have found this to be sufficient to ensure that the LOS pulse is considerably stronger than any reflected pulse from a previous source. Since our system includes eight sources, each individual source emits pulses at 64 ms intervals.

The microphone on each of our sensor boards senses the ultrasonic pulses from all the visible ultrasonic sources. As the top row of Figure 4-3 visualizes, the corresponding analog signal is amplified and filtered in order to enhance its quality in the 40kHz range. The resulting analog signal, together with the digital inertial signal, is sent to the driver box and and stored on the laptop's hard disk.

4.2.3 Inertial Operation

Our inertial subsystem operates independently of the ultrasonic components. The gyroscopes and accelerometers measure rotational rates and accelerations. The microprocessor on each sensor board samples them as 10-bit quantities and accumulates

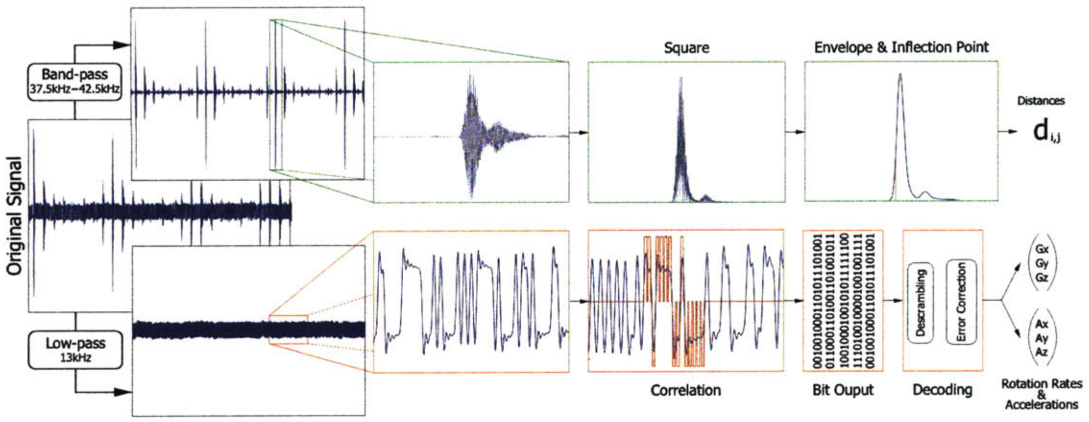


Figure 4-4: Signal Processing. The stored data is processed off-line to yield distance measurements from the analog acoustic data (top), and the rotation rates and accelerations from the digital inertial data (bottom). Top: After isolating the acoustic signal by band-passing the original signal around 40kHz, we compute its power envelope. The inflection points of the power envelope provide a robust indication of detected pulse locations, which in turn provide estimates of distances between ultrasonic sources and microphones. Bottom: The digital signal is isolated by low-passing below 13kHz, after which the scrambling pattern is used to identify the beginnings of different samples. The descrambling and error correction recover the encoded rotation rates and accelerations.

several readings to obtain more precise 12-bit quantities. We increase the frame rate by not sending the internal 12-bit values. Instead, we turn them into 6-bit values using delta modulation to maintain good precision. In addition, we employ error-correction coding, enabling the amplitude of the digital data to be much lower than that of the acoustic signal and therefore causing less interference. Finally, we interleave and scramble our data with a pseudo-random sequence, which helps with error correction and prevents the baseline drift. The resulting values from the three gyroscope axes and the three accelerometer axes are encoded as a 13kbps digital stream and multiplexed with the analog acoustic signal. The sampling rate of the inertial data is 140Hz.

4.3 Signal Processing

In the signal processing stage, our system extracts distance and inertial measurements from the stored sensor signals. It extracts pulse locations from the sampled acoustic signals and converts them into distances. It also converts the digitally encoded inertial sensor voltages into accelerations and angular velocities. To obtain accurate measurements for both these steps, we perform precise calibration of our sensors. In the following section we overview both signal processing steps as well as the calibration procedure, and refer the reader to [1] for more details.

4.3.1 Ultrasonic Processing

We process the stored signal to extract distance measurements, noting that the microphone data are perfectly synchronized with the emitted pulses. Therefore, we can partition the signal into frames, where each frame corresponds to the maximum distance traveled by the pulse (2.75 m, which translates to 1120 samples). As visualized in the top row of Figure 4-4, we first band-pass the signal to eliminate all frequencies that are outside the range of our ultrasonic sources (these include the multiplexed 13kBps digital data). Based on the specifications of the ultrasonic source, we use a filter that is centered at 40kHz and has a width of 5kHz. Second, we square the signal, since we are more interested in its power than in the signal itself. Third, we extract the envelope of the signal power by applying a low-pass filter with a cut-off frequency of 30kHz. We observe that tracking the location of the peak does not provide the most precise distance measurement since the gradient of the signal power envelope is low. Instead, we compute the inflection point—the point where the gradient is the largest. In our case, this point is positioned about 40 samples after the start of the pulse. We perform a calibration for each ultrasonic source to compute the exact offset in the number of samples.

The power envelope of the signal can contain multiple peaks due to reflection. Furthermore, it can contain no useful peaks if there is no direct LOS between the source and the detector. Therefore, with each distance measurement we associate a confidence measure w ranging from 0 (no useful measurement) to 1 (a correct measurement). We represent the confidence measure w as a product of three factors: signal strength (w_s), temporal continuity (w_t), and an angle between the ultrasonic source normal and the detector (w_a).

To incorporate the signal strength factor, we first ensure that the signal is well above the estimated ambient noise, which we assume to be a zero-mean Gaussian. We also normalize the peak values by multiplying them by their squared distances since the signal strength of a spherical wave is proportional to the inverse of the squared radius. If the resulting value is above a predetermined threshold, then w_s is set to 1. Otherwise, w_s decreases with the squared inverse of the difference between the threshold and the peak value. The temporal continuity measure w_t is equal to 1 if the difference between the corresponding peak in two neighboring time instants is within a threshold, and decreases to zero as that difference grows. The angular confidence measure w_a is computed based on the current estimates of the sensor locations. In our implementation, w_a is set to 1 unless the angle between the ultrasonic source normal and the vector toward the sensor is greater than 90 degrees, in which case it is set to 0, ensuring that the microphone is within the field-of-view of the ultrasonic source. Figure 4-5 plots the confidence values for several source-sensor pairs throughout a 30-second motion exercising all joints. In general, the quality of the signal between each source-sensor pair varies with time, depending on the motion.

4.3.2 Inertial Processing

Along with distances, we also extract the digitally encoded inertial data from the stored signal. As depicted in the bottom row of Figure 4-4, we first isolate the inertial portion of the signal by low-passing it with a cut-off frequency of 13kHz. We use the known scrambling pattern to lock onto the beginning of each new inertial sample and use thresholding to recover the bits of data. Those bits are then descrambled and error-corrected to yield the rotation rate readings for the three gyroscope axes, and the acceleration readings for the three accelerometer axes.

Even though the inertial readings are represented as 6-bit values, delta-modulation

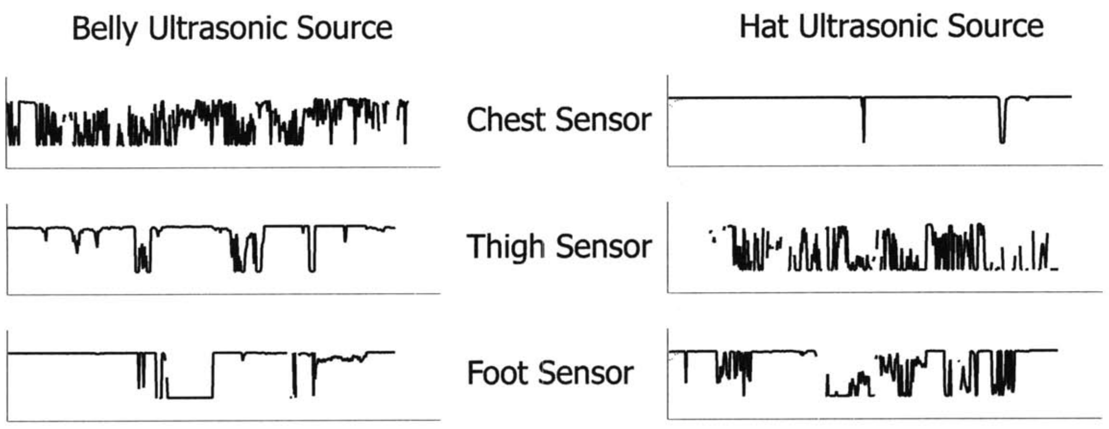


Figure 4-5: Confidence values for several source-sensor pairs are plotted over a 30-second motion exercising all joints. The belly source (left column) has a better line-of-sight with the thigh sensor (second row), while the hat source (right column) has a better view of the chest sensor (first row). They both see the foot sensor (third row) equally well but for different reasons: the belly source is positioned closer, while the hat source is better oriented.

ensures that the overall precision is not degraded. For example, if the 6-bit round-off underestimates the acceleration in one frame, it will compensate by overestimating in the next frame; if we were to double-integrate the delta-modulated acceleration, the resulting position would have more than 6 bits of precision.

4.3.3 Calibration

The processed values from the ultrasonic signal correspond to the number of samples between the pulse emission and the detected inflection point, while the values from the inertial signal correspond to the voltages given by the inertial sensors. To convert these values to meaningful distances, accelerations, and rotation rates, we carefully calibrated all the components of our system.

For the ultrasonic components, we identified the offsets between the detected inflection points and the true pulse beginnings. These offsets differ from source to source, but are fairly constant across different microphones. This is because our microphones are of much better quality and sensing capabilities than our sources. We found the offsets by affixing the ultrasonic sources and microphones at several different locations (ranging from 30cm to 90cm apart), calculating the resulting inflection points, and measuring the actual distances using a FARO arm contact digitizer (*faro.com*). We obtained offsets for each ultrasonic source, yielding a distance error of 2.35mm mean and 1.92mm standard deviation according to our leave-one-out experiments.

For the accelerometers and gyroscopes, we identified zero crossings and slopes to convert their voltages to physical values assuming a linear model. Using a level, we aligned each axis of the accelerometer along the gravity and opposite gravity. Accumulating over a period of time, we obtained accurate estimates of $\pm g$, and the zero crossing as their mean. We then affixed each gyroscope to a turntable, orienting each of its axes both up and down. The turntable was rotating at 45rpm with 0.1% accuracy, enabling us to find the voltages corresponding to ±45rpm. We averaged these two values to obtain the zero crossing.

4.4 Pose Estimation

Our system recovers body poses using angular velocities from gyroscopes, accelerations from accelerometers, and distances from the acoustic subsystem. While some approaches use algorithms specialized to only one kind of observation (e.g., [106, 139, 88]), we employ the Extended Kalman Filter [72] to combine information from all three sensor types. Extended Kalman Filter (EKF) provides a convenient, efficient, and elegant framework for combining different types of measurements to recover the state of a given system [163]. It incorporates a model of the system dynamics with its indirect observations to yield pose estimates. On a high level, EKF evaluates the system dynamics to evolve the system state until the next observation, then uses this observation to improve its estimate of the system state.

4.4.1 System Dynamics

The body structure provides constraints that aid the recovery of its joint configuration, or pose. The pose of an articulated body is specified by the joint angles that describe the configuration of the shoulders, elbows, and other body joints. We use a single vector θ to assemble all joint angles in the body. This joint structure determines the forward-kinematics equations $\mathbf{F}(\theta)$ and $\mathbb{F}(\theta)$, which are used to compute position and orientation of any sensor.

The system state $\mathbf{x}$ contains joint angles, their velocities, and accelerations (θ, $\dot{\theta}$, $\ddot{\theta}$). Because we do not know the internal muscle forces, we assume that the change in accelerations between frame $k-1$ and frame k is governed by the zero-mean Gaussian noise $\mathbf{w}$:

$$\mathbf{x}_k = \begin{bmatrix} \theta \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix}_k = \begin{bmatrix} \theta + \dot{\theta} \\ \dot{\theta} + \ddot{\theta} \\ \ddot{\theta} + \mathbf{w} \end{bmatrix}_{k-1} = f(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}), \tag{4.1}$$

We hand tune the standard deviation of the noise term by tracking several motions. Setting this value to 0.04 rad/s^2 works well for most of our examples.

4.4.2 System Observations

Accelerometers provide acceleration readings in the local coordinate frame of each sensor. They sense the Earth's gravity as an upward acceleration of g even when the body is stationary. To derive sensor accelerations as a function of body joints, we first express the position of each sensor through forward kinematics as $p = \mathbf{F}(\theta)$. Differentiating with respect to time and applying the chain rule yields an expression for velocity $v = \mathbf{J}\dot{\theta}$, with $\mathbf{J} = d\mathbf{F}/d\theta$ being the positional forward-kinematics Jacobian. Differentiating once again, we calculate the accelerations as $a = \mathbf{J}\ddot{\theta} + \dot{\mathbf{J}}\dot{\theta}$. After rotating into the sensor coordinate frame, we express acceleration observations at frame k with the following function h:

$$\mathbf{z}_k = [z_i]_k = \left[\mathtt{rot}\left\{\dot{\mathbf{J}}_i\dot{\theta} + \mathbf{J}_i\ddot{\theta} - g\right\} + v_i\right]_k = h(\mathbf{x}_k, \mathbf{v}_k), \tag{4.2}$$

where $\mathtt{rot}\,\{\cdot\}$ denotes the rotation from the global coordinate frame to the coordinate frame of sensor i, and the standard deviation of the Gaussian noise $\mathbf{v}$ corresponds to the accelerometer precision of about 0.02m/s^2.

Gyroscopes measure angular velocity in the local frame of each sensor. To derive the angular velocity of each sensor as a function of body joints, we begin with the forward kinematics equation for the orientation quaternion: $q = \mathbb{F}(\theta)$. Taking a time derivative and applying the chain rule, we get $\dot{q} = \mathbb{J}\dot{\theta}$, where $\mathbb{J} = d\mathbb{F}/d\theta$ is the orientational forward-kinematics Jacobian. To get angular velocity, we multiply by the orientation quaternion conjugate and double the vector part of the resulting quaternion: $\omega = 2(q^*\dot{q})_{\mathsf{vec}} = 2(\mathbb{F}^*\mathbb{J}\dot{\theta})_{\mathsf{vec}}$. In the sensor coordinate frame, angular velocity observations at frame k are defined by the following function h:

$$\mathbf{z}_k = [z_i]_k = \left[\mathtt{rot}\left\{2(\mathbb{F}_i^*\mathbb{J}_i\dot{\theta})_{\mathsf{vec}}\right\} + v_i\right]_k = h(\mathbf{x}_k, \mathbf{v}_k) \tag{4.3}$$

where $\mathtt{rot}\{\cdot\}$ denotes the rotation from the global coordinate frame to the coordinate frame of sensor i, and the standard deviation of the Gaussian noise $\mathbf{v}$ corresponds to the gyroscope precision of about 0.002 rad/s.

The ultrasonic subsystem provides distances between a sensor i and a source j for all source-sensor pairs. The position of both the source and the sensor can be computed as a function of joint angles using the positional forward kinematics function $\mathbf{F}$. Since the distance observation timings are not synchronized with the inertial observations, we process at frame k all the distances that were measured between frame $k-1$ and k. The following function h expresses a set of distance observations in terms of the system state:

$$\mathbf{z}_k = [z_{ij}]_k = [\|\mathbf{F}(\theta)_i - \mathbf{F}(\theta)_j\| + v_{ij}]_k = h(\mathbf{x}_k, \mathbf{v}_k), \tag{4.4}$$

where the standard deviation of the Gaussian noise $\mathbf{v}$ corresponds to the ultrasonic subsystem precision of about 2.5mm, and is further divided by the confidence of each distance measurement.

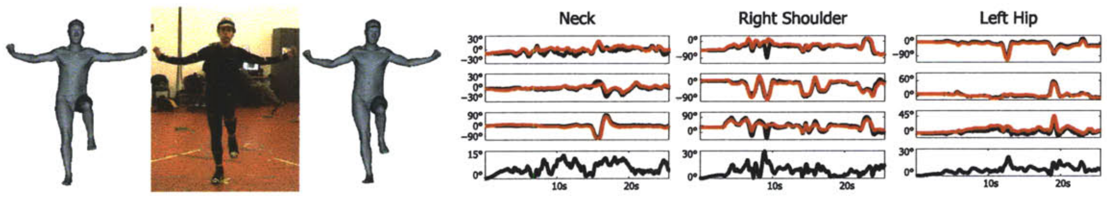


Figure 4-6: Comparison to the Vicon system. Left: one frame of the captured motion with our reconstruction to its left and Vicon reconstruction to its right. Right: a graph of a few 3-DOF joints (neck, right shoulder, and left hip) over 30 seconds of motion. The top three plots visualize the joint angles as reconstructed by us (red) and Vicon (black), while the bottom plot shows the orientation difference between our reconstruction and that of the Vicon system.

4.4.3 Extended Kalman Filter

EKF incorporates the evolution of the underlying system state $\mathbf{x}$ along with the observations of the system. Each set of observations coming from accelerometers, gyroscopes, or the acoustic subsystem, is processed sequentially using the appropriate formulations of h and the corresponding measurement noise $\mathbf{v}$.

The Kalman time update step evolves the system until it reaches the next observation, yielding an *a priori* (before observation) estimate of the system state $\mathbf{x}$ and its covariance $\mathbf{P}$:

$$\mathbf{x}_k^- = f(\mathbf{x}_{k-1}, \mathbf{0}) \tag{4.5}$$

$$\mathbf{P}_k^- = \mathbf{A}_k \mathbf{P}_{k-1} \mathbf{A}_k^\top + \mathbf{W}_k \mathbf{Q} \mathbf{W}_k^\top, \tag{4.6}$$

where $^-$ stands for the *a priori* estimate, $\mathbf{Q}$ is the system noise covariance, $\mathbf{A} = \partial f / \partial \mathbf{x}$ is the Jacobian of f with respect to the system parameters, and $\mathbf{W} = \partial f / \partial \mathbf{w}$ is the Jacobian of f with respect to the system noise parameters.

The Kalman observation step uses the observation $\mathbf{z}_k$ to improve on the *a priori* estimates of the state $\mathbf{x}_k^-$ and its covariance $\mathbf{P}_k^-$:

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} (\mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\mathrm{T} + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k^\mathrm{T})^{-1} \tag{4.7}$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k (\mathbf{z}_k - h(\mathbf{x}_k^-, \mathbf{0})) \tag{4.8}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-, \tag{4.9}$$

where $\mathbf{K}$ is the Kalman gain chosen to minimize the *a posteriori* state covariance, $\mathbf{R}$ is the measurement noise covariance, $\mathbf{H} = \partial h / \partial \mathbf{x}$ is the Jacobian of h with respect to the system parameters, and $\mathbf{V} = \partial h / \partial \mathbf{v}$ is the Jacobian of h with respect to the measurement noise parameters.

In our system, the noise covariance matrices ($\mathbf{Q}$, $\mathbf{R}$) are all diagonal, and their elements are the variances corresponding to the aforementioned standard deviations. All the Jacobian matrices ($\mathbf{A}, \mathbf{W}, \mathbf{H}, \mathbf{V}$) were computed analytically for speed.

4.4.4 Initialization

EKF is unstable if there are misalignments in the first frame because most of our measurements are incremental. We alleviate this problem by providing good estimates of the initial pose and the location of each body sensor. Our human mesh and its skeleton match the proportions of the subject, and the sensors are manually specified as rigid transforms in the coordinate frames of their parent bones. In addition, the subject begins each motion by holding a specified "rest" pose. As a result, the average accelerometer reading during that pose should be aligned with gravity (pointing upward). We exploit this fact to improve the pose of our model and tune the sensor orientations using a gradient descent approach. We additionally refine the inertial sensor offsets with an objective function which assures that the readings of each sensor during the initial pose integrate to zero.

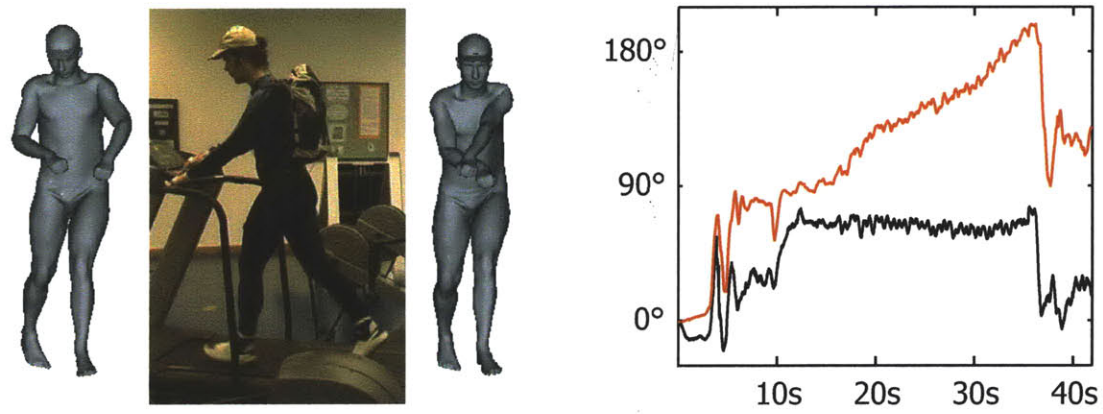


Figure 4-7: In the treadmill motion above, the distance measurements enabled us to avoid pose drift in the shoulder joint. Left: One frame of the treadmill motion where the reconstruction without distances (right) drifts, while the one with distances (left) does not. Right: A graph plotting the left shoulder joint angle for the reconstruction with (black) and without distances (red).

4.5 Results

Our system is capable of acquiring motions ranging from biking and driving, to skiing, table tennis, and weight lifting (Figure 4-1). The accompanying video demonstrates its capability to reconstruct visible subtleties in recorded motions. The results are processed at a rate of 10 frames per second and visualized without any post-processing using an automatically skinned mesh [11].

We evaluated the accuracy of our system by comparing it with Vicon's optical motion capture system, which is known for its exceptional precision. Since we could not collocate the optical markers with our sensors without interfering with the ultrasonic line-of-sight, we placed them around the body according to the suggestions in the Vicon manual. We used Vicon software to fit the same skeleton used by our system to the optical markers, and started all the reconstructions with an identical initial pose. To remove the effects of root drift in our reconstructions, we provided the root transform from Vicon's reconstruction to our Kalman filter at each frame. Although neither reconstruction is perfect, Vicon matches the original motions better, thus we treat it as ground truth in our analysis.

The left side of Figure 4-6 shows one frame of a 30-second motion, with our recon-

Sensors	Neck		Right Shoulder		Left Hip	
	$\mu(^\circ)$	$\sigma(^\circ)$	$\mu(^\circ)$	$\sigma(^\circ)$	$\mu(^\circ)$	$\sigma(^\circ)$
A	159.7	147.7	168.6	105.9	184.0	146.0
A+D	74.0	125.4	178.6	110.9	165.5	144.5
D	120.2	93.4	54.7	37.0	117.4	79.2
G	25.9	15.9	10.1	7.0	8.3	8.0
G+D	18.4	7.4	8.1	5.8	9.8	5.4
A+G	9.5	3.4	10.9	6.4	5.6	4.0
A+G+D	5.7	2.9	8.0	5.0	6.6	3.8

Table 4.1: Different combinations of our sensors (accelerometers **A**, gyroscopes **G**, and ultrasonic distances **D**) yield varying reconstruction quality as compared to the Vicon system. We report the mean μ and the standard deviation σ of the orientation differences between our reconstruction and Vicon's reconstruction for three 3-DOF joints of the skeleton.

struction on the left and Vicon's on the right. Qualitatively, Vicon's reconstruction matches the original motion better, although we are also able to reconstruct motion nuances such as slight wrist flicks. In addition, optical motion capture is able to recover the root transform without drift. The right side of Figure 4-6 shows plots of the neck, right shoulder, and left hip joints over the course of this motion. We visualize these three joints because they each have three degrees of freedom and are directly under the root in the skeletal hierarchy, and therefore free of parent-inherited errors. The top three plots for each joint compare the three Euler angles directly, with our reconstruction drawn in red and Vicon's in black. The bottom plot for each joint shows the orientation difference between the two reconstructions.

Our sensing capabilities have led us to explore multiple facets of our pose recovery system. We have reconstructed motions using various combinations of our sensors. Table 4.1 summarizes our findings for the same 30-second motion visualized in Figure 4-6. Compared to the Vicon reconstruction, the Extended Kalman Filter performs poorly with just accelerometers, just distances, or a combination of the two. Gyroscopes, on the other hand, provide much better reconstructions on their own, and yield even better results in conjunction with either the distances or the accelerations. The best results require using information from all available sensors. Although distance measurements do not have a dramatic effect on the reconstruction of activity in Table 4.1, we have observed significant drift in several experiments. For example, distance measurements are critical for an accurate reconstruction of the treadmill motion in Figure 4-7.

4.6 Discussion

We have presented a wearable motion capture system prototype that is entirely self-contained and capable of operating for extended periods of time in a large variety of environments. Our system acquires distance measurements between a set of ultrasonic sources and a set of sensors. Each sensor is augmented with inertial measurements in order to improve precision and sampling rate, as well as to alleviate line-of-sight problems. We have shown how to compute the pose of a human model directly from these measurements using the Extended Kalman Filter and qualitatively validated the performance of the system against a high-quality optical motion capture system. An attractive feature of our system is its low cost, with the current component price of around \$3,000 (excluding the laptop). We believe that a much smaller version of the system with more sensors could be mass-produced for only a few hundred dollars, implying that this type of system could be owned and used on a daily basis by almost anyone.

Limitations. Due to inherent physical limitations of our hardware components, we were unable to reconstruct high-impact motions such as jumping or hard kicking. Though sensors with wider sensing ranges are available to the detriment of precision, we have chosen ours to maximize the trade-off between coverage and precision. Other types of motions that we are unable to acquire with the current prototype include interaction between multiple subjects, such as dancing. By changing the ultrasonic source frequency of the partner, we could track each subject without interference, as well as obtain distance measurements between points on two different subjects.

Our distance measurements depend on the speed of sound, which is affected by temperature and, to a lesser extent, humidity. To obtain a more precise speed of sound, one could use a digital thermometer or a calibration device prior to each capture session. Distance measurements may also be affected by the presence of ultrasonic noise in the environment, but we have not experienced these problems in our experiments.

Perhaps the most significant limitation of our system is the lack of direct measurements of the root transformation. Our reconstructions exhibit drift in both global translation and rotation. We show that distance measurements help but they do not eliminate the problem entirely: over time, root drift propagates to other joints as well. Some of our experiments show that if we detect foot plants we can determine the global position and orientation relative to the ground plane. Another approach would be to use accelerometer readings for a vertical reference whenever the body is not accelerating. A more general solution could rely on additional sensors that perform absolute measurements (e.g., a GPS or magnetometer) to better constrain the root. These measurements can be incorporated into the EKF. Our evaluations, for example, incorporate root transforms obtained by the Vicon system. We could also compute absolute root measurements using ceiling-mounted ultrasonic sources [66] or image-based tracking. Lastly, optimization, which is slower than Kalman filtering, might converge to a better solution for the same sensor data.

Future Work. There are many possibilities for future work. First, we could employ additional sensors to better constrain the root transform. Next, we could decrease the size and cost of our system. The driver box in particular has been built almost entirely from off-the-shelf components. By designing custom circuitry with only the necessary hardware, we could turn it into an iPod-like device that powers the system and stores the captured data, removing the need for a laptop. Another direction would be to perform all of the processing on-line, which might be achieved by accumulating several measurements before feeding them to the EKF to improve the speed. This would allow the system to be used as an input device in a variety of augmented-reality applications.

We should enrich motion repositories with varied data sets to further understand human motion. Restrictive recording requirements limit the scope of current motion data sets, which prevents the broader application of motion processing. An inexpensive and versatile motion-capture system would enable the collection of extremely large data sets. This enhanced infrastructure could then support large-scale analysis of human motion, including its style, efficiency, and adaptability. The analysis of daily human motion could even extend beyond computer graphics, and help prevent repetitive stress injuries, quicken rehabilitation, and enable design of improved computer interfaces.

Chapter 5

Analysis of Dynamic Shapes for Face Transfer

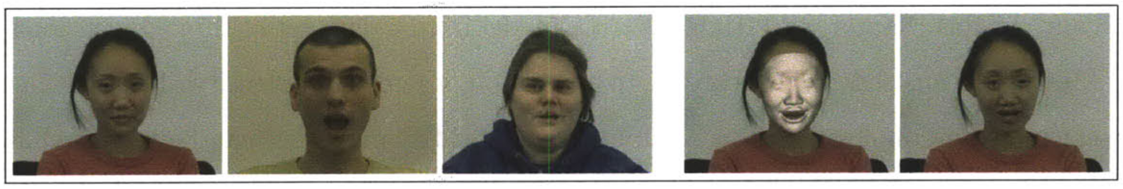


Figure 5-1: Face Transfer with multilinear models gives animators decoupled control over facial attributes such as identity, expression, and viseme. In this example, we combine pose and identity from the first frame, surprised expression from the second, and a viseme (mouth articulation for a sound midway between "oo" and "ee") from the third. The resulting composite is blended back into the original frame.

As the technology for reconstructing 3D surfaces of dynamic shapes advances, the demand for thechniques that analyze such data in order to advance various applications will grow. In this chapter we describe a method that analyzes captured 3D face shapes in order to learn the relationship between different face shapes. It can then transfer facial performance between different faces, enabling even untrained users to create realistic 3D face animations.

Performance-driven animation has a growing role in film production because it allows actors to express content and mood naturally, and because the resulting animations have a degree of realism that is hard to obtain from synthesis methods [123].

The search for the highest quality motions has led to complex, expensive, and hard-to-use systems. This chapter introduces new techniques for producing compelling facial animations that are inexpensive, practical, versatile, and well suited for editing performances and retargeting to new characters.

Face Transfer extracts performances from ordinary video footage, allowing the transfer of facial action of actors who are unavailable for detailed measurement, instrumentation, or for re-recording with specialized scanning equipment. Expressions, visemes (speech-related mouth articulations), and head motions are extracted automatically, along with a performance-driven texture function. With this information in hand, our system can either rewrite the original footage with adjusted expressions and visemes or transfer the performance to a different face in a different footage.

Multilinear models are ideally suited for this application because they can describe face variations with separable attributes that can be estimated from video automatically. In this chapter, we estimate such a model from a data set of three-dimensional (3D) face scans that vary according to expression, viseme, and identity. The multilinear model decouples the three attributes (i.e., identity or viseme can be varied while expression remains constant) and encodes them consistently. Thus the attribute vector that encodes a smile for one person encodes a smile for every face spanned by the model, regardless of identity or viseme. Yet the model captures the fact that every person smiles in a slightly different way. Separability and consistency are the key properties that enable the transfer of a performance from one face to another without a change in content.

Contributions. This chapter describes a general, controllable, and practical system for facial animation. It estimates a multilinear model of human faces by examining geometric variations between 3D face scans. In principle, given a large and varied data set, the model can generate any face, any expression, any viseme. As proof of concept, we estimate the model from a couple of geometric data sets: one with 15 identities and 10 expressions, and another with 16 identities, 5 expressions, and 5 visemes. Existing estimation algorithms require perfect one-to-one correspondence between all meshes, and a mesh for every possible combination of expression, viseme, and identity. Because acquiring the full Cartesian product of meshes and putting them into dense correspondence is extremely difficult, this chapter introduces methods for populating the Cartesian product from a sparse sampling of faces, and for placing unstructured face scans into correspondence with minimal cross-coupling artifacts.

By linking the multilinear model to optical flow, we obtain a single-camera tracker that estimates performance parameters and detailed 3D geometry from video recordings. The model defines a mapping from performance parameters back to 3D shape, thus we can arbitrarily mix pose, identity, expressions, and visemes from two or more videos and render the result back into a target video. As a result, the system provides an intuitive interface for both animators (via separably controllable attributes) and performers (via acting). And because it does not require performers to wear visible facial markers or to be recorded by special face-scanning equipment, it is an inexpensive and easy-to-use facial animation system.

5.1 Related Work

Realistic facial animation remains a fundamental challenge in computer graphics. Beginning with Parke's pioneering work [110], desire for improved realism has driven researchers to extend geometric models [111] with physical models of facial anatomy [161, 92] and to combine them with non-linear finite element methods [89] in systems that could be used for planning facial surgeries. In parallel, Williams presented a compelling argument [166] in favor of performance-driven facial animation, which anticipated techniques for tracking head motions and facial expressions in video [94, 60, 54, 116]. A more expensive alternative could use a 3D scanning technique [171], if the performance can be re-recorded with such a system.

Much of the ensuing work on face estimation and tracking relied on the observation that variation in faces is well approximated by a linear subspace of low dimension [132]. These techniques estimate either linear coefficients for known basis shapes [14, 24] or both the basis shapes and the coefficients, simultaneously [28, 142]. In computer graphics, the combination of accurate 3D geometry with linear texture models [115, 21] produced striking results. In addition, Blanz and Vetter [21] presented a process for estimating the shape of a face in a single photograph, and a set of controls for intuitive manipulation of appearance attributes (thin/fat, feminine/masculine).

These and other estimation techniques share a common challenge of decoupling the attributes responsible for observed variations. As an early example, Pentland and Sclaroff estimate geometry of deformable objects by decoupling linear elastic equations into orthogonal vibration modes [113]. In this case, modal analysis uses eigen decomposition to compute the independent vibration modes. Similar factorizations are also relied upon to separate variations due to pose and lighting, pose and expression, identity and lighting, or style and content in general [67, 28, 55, 73, 32].

A technical limitation of these formulations is that each pair of factors must be considered in isolation; they cannot easily decouple variations due to a combination of more than two factors. The extension of such two-mode analysis to more modes of variation was first introduced by Tucker [143] and later formalized and improved on by Kroonenberg and de Leeuw [91]. These techniques were successfully applied to multilinear analysis of images [146, 147].

This chapter describes multilinear analysis of *three-dimensional (3D)* data sets and generalizes face-tracking techniques to create a unique performance-driven system for animation of any face, any expression, and any viseme. In consideration of similar needs, Bregler and colleagues introduced a two-dimensional method for transferring mouth shapes from one performance to another [27]. The method is ideal for film dubbing—a problem that could also be solved without performance by first learning the mouth shapes on a canonical data set and then generating new shapes for different texts [62]. These methods are difficult to use for general performance-driven animation because they cannot change emotions of a face. Although the problem can be resolved by decoupling emotion and content via two-mode analysis [38], all three techniques are view specific, which presents difficulties when view, illumination, or both have to change.

Our Face Transfer learns a model of 3D facial geometry variations in order to

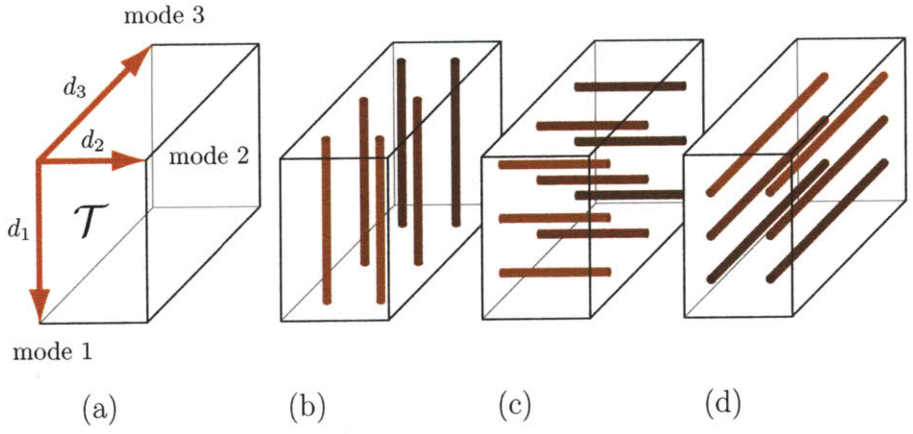


Figure 5-2: In (a) we show a 3^{rd}-order (3-mode) tensor $\mathcal{T}$ whose modes have d_1, d_2, and d_3 elements respectively. Depending on how we look at the data within the tensor, we can identify three mode spaces. By viewing the data as vectors parallel to the first mode (b), we define mode-1 space as the span of those vectors. Similarly, mode-2 space is spanned by vectors parallel to the second mode (c), and mode-3 space by vectors in the third mode (d).

infer a particular face shape from 2D images. Previous work combines identity and expression spaces by copying deformations from one subject onto the geometry of other faces [55, 20, 34]. Expression cloning [105, 136] improves on this process but does not account for actor-specific idiosyncrasies that can be revealed by statistical analysis of the entire data set (i.e., the mesh vertex displacements that produce a smile should depend on who is smiling and on what they are saying at the same time). Other powerful models of human faces have been explored [158] at the cost of making the estimation and transfer of model parameters more difficult. This chapter describes a method that incorporates all such information through multilinear analysis, which naturally accommodates variations along multiple attributes.

5.2 Multilinear Algebra

Multilinear algebra is a higher order generalization of linear algebra. In this section we provide insight behind the basic concepts needed for understanding of our Face Transfer system. De Lathauwer's dissertation [53] provides a comprehensive treatment of this topic. Concise overviews have also been published in the graphics and vision literature [146, 147].

Tensors. The basic mathematical object of multilinear algebra is the tensor, a natural generalization of vectors (1^{st} order tensors) and matrices (2^{nd} order tensors) to multiple indices. An N^{th}-order tensor can be thought of as a block of data indexed by N indices: $\mathcal{T} = (t_{i_1 i_2 \ldots i_N})$. Figure 5-2 shows a 3^{rd}-order (or 3-mode) tensor with a total of $d_1 \times d_2 \times d_3$ elements. Different modes usually correspond to particular attributes of the data (e.g, expression, identity, etc.).

Mode Spaces. A matrix has two characteristic spaces, row and column space; a tensor has one for each mode, hence we call them *mode spaces.* The $d_1 \times d_2 \times d_3$ 3-tensor in Figure 5-2 has three mode spaces. Viewing the data as a set of d_1-dimensional vectors stored parallel to the first axis (Figure 5-2b), we can define the mode-1 space as the span of those vectors. Similarly, mode-2 space is defined as the span of the vectors stored parallel to the second axis, each of size d_2 (Figure 5-2c). Finally, mode-3 space is spanned by vectors in the third mode, of dimensionality d_3 (Figure 5-2d). Multilinear algebra revolves around the analysis and manipulation of these spaces.

Mode-n Product. The most obvious way of manipulating mode spaces is via linear transformation, officially referred to as the *mode-n product.* It is defined between a tensor $\mathcal{T}$ and a matrix $\mathbf{M}$ for a specific mode n, and is written as a multiplication with a subscript: $\mathcal{T} \times_n \mathbf{M}$. This notation indicates a linear transformation of vectors in $\mathcal{T}$'s mode-n space by the matrix $\mathbf{M}$. Concretely, $\mathcal{T} \times_2 \mathbf{M}$ would replace each mode-2 vector $\mathbf{v}$ (Figure 5-2c) with a transformed vector $\mathbf{Mv}$.

Tensor Decomposition. One particularly useful linear transformation of mode data is the N-mode singular value decomposition (N-mode SVD). It rotates the mode spaces of a *data tensor* $\mathcal{T}$ producing a *core tensor* $\mathcal{C}$, whose variance monotonically decreases from first to last element in each mode (analogous to matrix SVD). This enables us to truncate the insignificant components and get a reduced model of our data.

Mathematically, N-mode SVD can be expressed with mode products

$$\mathcal{T} \times_1 \mathbf{U}_1^{\mathsf{T}} \times_2 \mathbf{U}_2^{\mathsf{T}} \times_3 \mathbf{U}_3^{\mathsf{T}} \cdots \times_N \mathbf{U}_N^{\mathsf{T}} = \mathcal{C} \tag{5.1}$$

$$\Longrightarrow \quad \mathcal{T} = \mathcal{C} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \times_3 \mathbf{U}_3 \cdots \times_N \mathbf{U}_N, \tag{5.2}$$

where $\mathcal{T}$ is the data tensor, $\mathcal{C}$ is the core tensor, and $\mathbf{U}_i$'s (or more precisely their transposes) rotate the mode spaces. Each $\mathbf{U}_i$ is an orthonormal matrix whose columns contain left singular vectors of the ith mode space, and can be computed via regular SVD of those spaces [53]. Since variance is concentrated in one corner of the core tensor, data can be approximated by

$$\mathcal{T} \simeq \mathcal{C}_{reduced} \times_1 \check{\mathbf{U}}_1 \times_2 \check{\mathbf{U}}_2 \times_3 \check{\mathbf{U}}_3 \cdots \times_N \check{\mathbf{U}}_N \,, \tag{5.3}$$

where $\check{\mathbf{U}}_i$'s are truncated versions of $\mathbf{U}_i$'s with last few columns removed. This truncation generally yields high quality approximations but it is not optimal—one of several matrix-SVD properties that do not generalize in multilinear algebra. One can obtain a better approximation with further refinement of $\check{\mathbf{U}}_i$'s and $\mathcal{C}_{reduced}$ via alternating least squares [53].

5.3 Multilinear Face Model

To construct the multilinear face model, we first acquire a range of 3D face scans, put them in full correspondence, appropriately arrange them into a data tensor (Figure 5-3), and use the N-mode SVD to compute a model that captures the face geometry and its variation due to attributes such as identity and expression.

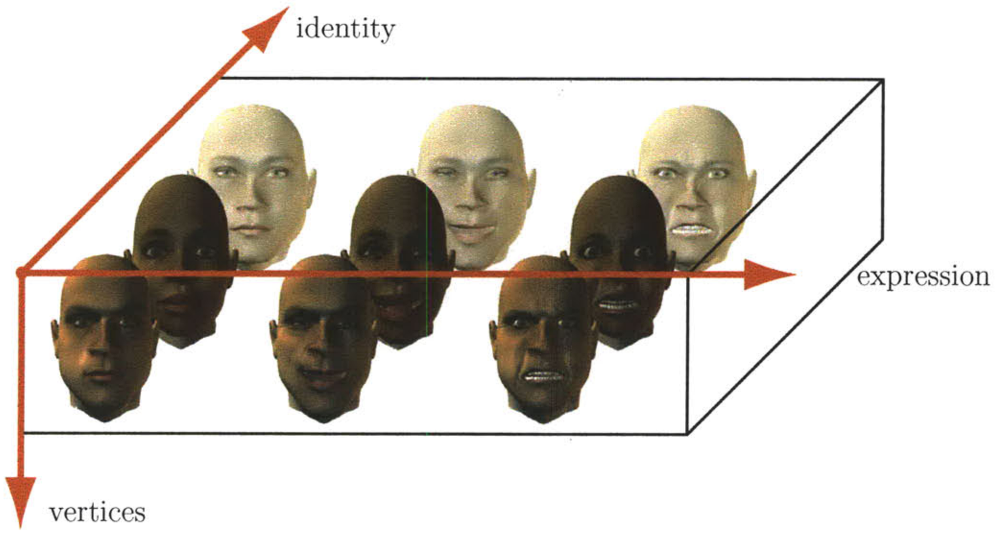


Figure 5-3: Data tensor for a bilinear model that varies with identity and expression; the first mode contains vertices, while the second and third modes correspond to expression and identity respectively. The data is arranged so that each slice along the second mode contains the same expression (in different identities) and each slice along the third mode contains the same identity (in different expressions). In our trilinear experiments we have added a fourth mode, where scans in each slice share the same viseme.

5.3.1 Face Data

We demonstrate our proof-of-concept system on two separate face models: a bilinear model, and a trilinear model. Both were estimated from detailed 3D scans ($\sim$ 30K vertices) acquired with 3dMD/3Q's structured light scanner (`http://www.3dmd.com/`) in a process similar to regular flash photography, although our methods would apply equally to other geometric data sets such as motion capture. As a preprocess, the scans were smoothed using the bilateral filter [84] to eliminate some of the capture noise. The subject pool included men, women, Caucasians, and Asians, from the mid-20s to mid-50s.

Bilinear model. 15 subjects were scanned performing the same 10 facial expressions. The expressions were picked for their familiarity as well as distinctiveness, and include neutral, smile, frown, surprise, anger, and others. The scans were assembled

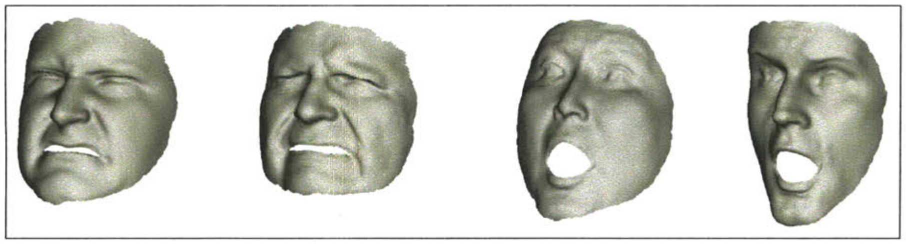


Figure 5-4: Several faces generated by manipulating the parameters of the bilinear model. The left two faces show our attempt of expressing disgust, an expression that was not in the database. They only differ in identity, to demonstrate the separability of our parameters. The right two faces show surprise for two novel identities, illustrating how the expression adjusts to identity.

into a third order (3-mode) data tensor (30K vertices $\times$ 10 expressions $\times$ 15 identities). After N-mode SVD reduction, the resulting bilinear model offers 6 knobs for manipulating expression and 9 for identity.

Trilinear model. 16 subjects were asked to perform 5 visemes in 5 different expressions (neutral, smiling, scowling, surprised, and sad). The visemes correspond to the boldfaced sounds in **m**an, c**ar**, **ee**l, t**oo**, and **sh**e. Principal components analysis of detailed speech motion capture indicated that these five expressions broadly span the space of lip shapes, and should give a good approximate basis for all other visemes—with the possible exception of exaggerated fricatives. The resulting fourth order (4-mode) data tensor (30K vertices $\times$ 5 visemes $\times$ 5 expressions $\times$ 16 identities) was decomposed to yield a trilinear model providing 4 knobs for viseme, 4 for expression, and 16 for identity (we have kept the number of knobs large since our data sets were small).

5.3.2 Correspondence

Training meshes that are not placed in perfect correspondence can considerably muddle the question of how to displace vertices to change one attribute versus another (e.g. identity versus expression), and thus the multilinear analysis may not give a model with good separability. We show here how to put a set of unstructured face scans into correspondence suitable for multilinear analysis.

Despite rapid advances in automatic parameterization of meshes (e.g., [119, 76]), it took considerable experimentation to place many facial scans into detailed correspondence. The principal complicating factors are that the scans do not have congruent mesh boundaries, and the problem of matching widely varied lip deformations does not appear to be well served by conformal maps or local isometric constraints. This made it necessary to mark a small number of feature points in order to bootstrap correspondence-finding across large deformations.

We developed a protocol for a template-fitting procedure [5, 136], which seeks a minimal deformation of a parameterized template mesh that fits the surface implied by the scan. The optimization objective, minimized with gradient descent, balances overall surface similarity, proximity of manually selected feature points on the two surfaces, and proximity of reference vertices to the nearest point on the scanned surface. We manually specified 42 reference points on a reference facial mesh and on a neutral (m-viseme) scan. After rigidly aligning the template and the scan with Procrustes' alignment, we deformed the template mesh into the scan: at first, weighing the marked correspondences heavily and afterwards emphasizing vertex proximity. For the trilinear model, the remaining m-viseme (closed-mouth) scans were marked with 21 features around eyebrows and lips, rigidly aligned to upper-face geometry on the appropriate neutral scans, and then non-rigidly put into correspondence as above. Finally, all other viseme scans were similarly put into correspondence with the appropriate closed-mouth scan, using the 18 features marked around the lips.

5.3.3 Face Model

Equation (5.3) shows how to approximate the data tensor by mode-multiplying a smaller core tensor with a number of truncated orthogonal matrices. Since our goal is to output vertices as a function of attribute parameters, we can decompose the data tensor without factoring along the mode that corresponds to vertices (mode-1),

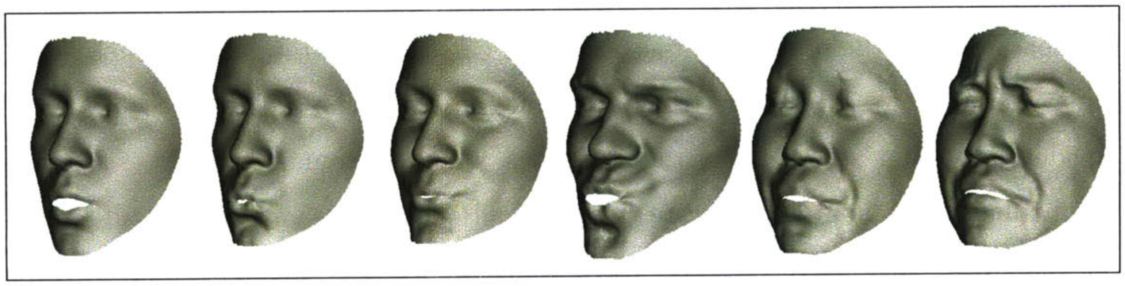


Figure 5-5: Faces generated by manipulating the parameters of the trilinear model. Left to right: producing the 'oo' sound, trying to whistle, breaking into a smile, two changes of identity, then adding a scowl.

changing Equation (5.3) to:

$$\mathcal{T} \simeq \mathcal{M} \times_2 \check{\mathbf{U}}_2 \times_3 \check{\mathbf{U}}_3 \cdots \times_N \check{\mathbf{U}}_N \,, \tag{5.4}$$

where $\mathcal{M}$ can now be called the *multilinear model* of face geometry. Mode-multiplying $\mathcal{M}$ with $\check{\mathbf{U}}_i$'s approximates the original data. In particular, mode-multiplying it with one row from each $\check{\mathbf{U}}_i$ reconstructs exactly one original face (the one corresponding to the attribute parameters contained in that row). Therefore, to generate an arbitrary interpolation (or extrapolation) of original faces, we can mode-multiply the model with a linear combination of rows for each $\check{\mathbf{U}}_i$. We can write

$$\mathbf{f} = \mathcal{M} \times_2 \mathbf{w_2}^{\top} \times_3 \mathbf{w_3}^{\top} \cdots \times_N \mathbf{w_N}^{\top} \,, \tag{5.5}$$

where $\mathbf{w_i}$ is a column vector of parameters (weights) for the attribute corresponding to i^{th} mode, and $\mathbf{f}$ is a column vector of vertices describing the resulting face.

5.3.4 Missing Data

Building the multilinear model from a set of face scans requires capturing the full Cartesian product of different face attributes, (i.e., all expressions and visemes need to be captured for each person). Producing a full data tensor is not always practical for large data sets. For example, a certain person might have trouble performing some expressions on cue, or a researcher might add a new expression to the database but be unable reach all the previous subjects. In our case, data corruption and subsequent unavailability of a subject led to an incomplete tensor. The problem becomes more evident if we add *age* as one of the attributes, where we cannot expect to scan each individual throughout their entire lives. In all these cases, we would still like to include a person's successful scans in the model, and fill in the missing ones with the most likely candidates. This process is known as imputation.

There are many possible schemes for estimating a model from incomplete data. A naive imputation would find a complete sub-tensor, use it to estimate a smaller model, use that to predict a missing face, use that to augment the data set, and repeat. In a more sophisticated Bayesian setting, we would treat the missing data as hidden variables to be MAP estimated (imputed) or marginalized out. Both approaches require many iterations over a huge data set; Bayesian methods are particularly expensive and generally require approximations for tractability. With MAP estimation and naive imputation, the results can be highly dependent on the order of operations. Because it fails to exploit all available constraints, the naive imputative scheme generally produces inferior results.

Here we use an imputative scheme that exploits more available constraints than the naive one, producing better results. The main intuition, which we formalize below, is that any optimization criteria can be linearized in a particular tensor mode, where it yields a matrix factorization problem with missing values. Then we leverage existing factorization schemes for incomplete matrices, where known values contribute a set of linear constraints on the missing values. These constraints are then combined and solved in the least-squares sense.

Description. Our algorithm consists of two steps. First, for each mode we assemble an *incomplete* matrix whose columns are the corresponding mode vectors. We then seek a subspace decomposition that best reconstructs the known values of that matrix. The decomposition and the known values provide a set of linear constraints for the missing values. This can be done with off-the-shelf imputative matrix factorizations (e.g., PPCA [141], SPCA [124], or ISVD [23]). Typically these algorithms estimate a low-rank subspace from the complete vectors of the mode and use that to predict missing values in the incomplete columns (and/or update the subspace). In our experiments we used the standard PPCA formulation for filling in missing values, which reduces to a system of linear equations that relate unknown values to known values through the estimated mean and covariance of the vectors in the mode space. Second, the linear constraints are combined through the missing elements, because they are shared across all groups of modal vectors and must be filled in with consistent values. To that end, we collect the linear equations that determine a particular missing value in all the modes, and solve them together. For example, if two missing values co-occur in some mode vector, then they must be jointly estimated. We update the mean and covariance for each decomposition and repeat the two steps until convergence.

Evaluation. Figure 5-6 contrasts the results of this method with faces predicted by our generalization of the simple method proposed by Blanz and colleagues [20]. In their formulation the same displacement vectors that make one person smile are copied over onto every other identity. Because our data set includes smiles for more than one person, we extend that approach to copy their average. In this particular example, 15% of real faces where held out of the trilinear data set and predicted by our imputation scheme and the simple averaging scheme. Note how the multilinear prediction is closer to the truth in most examples, even predicting some individual idiosyncrasies in puckers and smiles. The simple averaging scheme, however, seems to do a better job at keeping the lips sealed for closed-mouth faces (bottom row of Figure 5-6). We could obtain better results by preferentially weighting detail around the mouth.

In our earlier trilinear experiments, we found that ISVD-based imputations predicted how faces vary from the mean with less than 9% relative error (Frobenius norm of the total error divided by the norm of the held-out face variations) for up to 50% missing data. In general, the predictions are drawn towards the mean of the known data. Closed-mouth expressions, which are under-represented in our data and thus lie

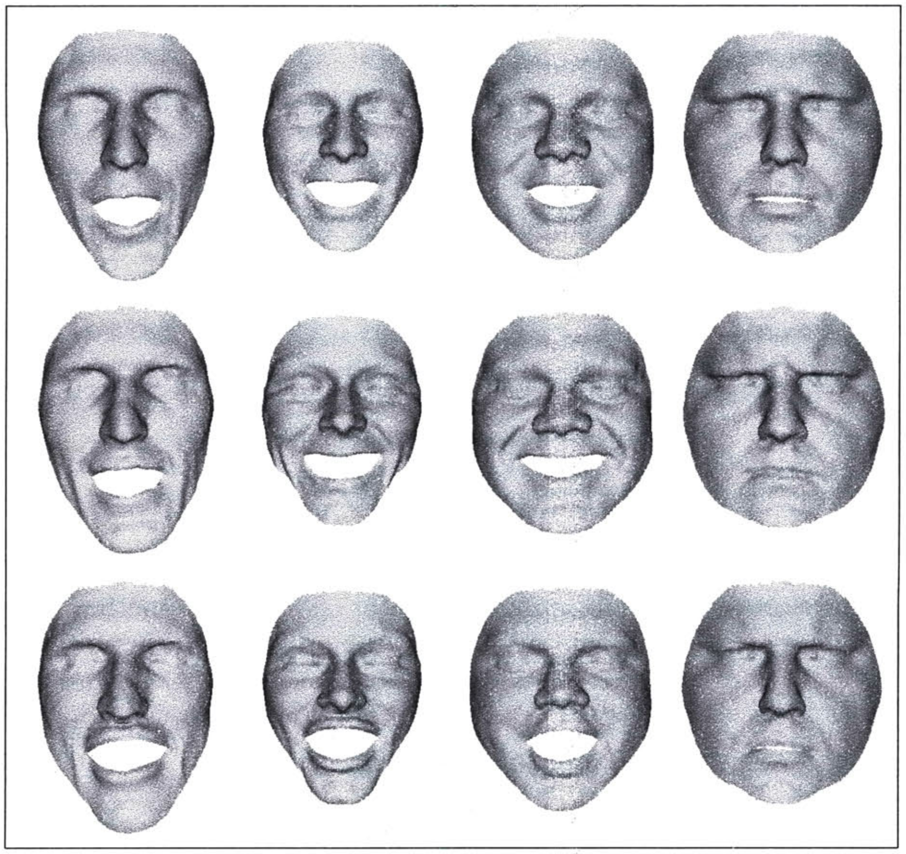


Figure 5-6: From top to bottom: Prediction of held-out faces with our imputation scheme (on the trilinear model), the actual face, and a simple averaging scheme.

far from the mean, were not predicted as well as other expressions. That can be fixed by reweighting the data. Tests performed on synthetic data indicate that the quality of imputation increases as the data set grows in size, even if significant portions of it are missing. The reason why is that if the data is truly low-dimensional in each of the modes, the missing samples will fall within the span and density of the known ones.

Probabilistic Interpretation. The above algorithm fills in missing data by approximating the true multilinear distribution. The form of this approximation is made precise by a probabilistic interpretation, which starts from a multilinear generative model

$$\mathcal{T} = \mathcal{M} \times_2 \check{\mathbf{U}}_2 \times_3 \check{\mathbf{U}}_3 \cdots \times_N \check{\mathbf{U}}_N + \nu,$$

where $\mathcal{T}$ and $\mathcal{M}$ are the data and model tensors, $\check{\mathbf{U}}_i$ is the i-th modal subspace, and ν is a Gaussian noise source. Filling in missing data according to this model is computationally expensive. Instead, we approximate the true likelihood with a geometric average of Gaussians

$$p(\mathcal{T}|\mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N) \approx \prod_{j=2}^N q_j(\mathcal{T}, \mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N)^{1/N}.$$

Each Gaussian $q_j(\mathcal{T}, \mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N) \doteq \mathcal{N}(\mathcal{T}|\check{\mathbf{U}}_j\mathbf{J}_j, \sigma_j^2)$ is found by fixing $\{\check{\mathbf{U}}_i\}_{i\neq j}$ and turning the tensor Equation (5.4) into matrix form: $\mathbf{T}_j = \check{\mathbf{U}}_j\mathbf{J}_j$. Here, columns of $\mathbf{T}_j$ are the mode-j vectors of $\mathcal{T}$, and the columns of $\mathbf{J}_j$ are the mode-j vectors of $\mathcal{M} \times_2 \check{\mathbf{U}}_2 \cdots \times_{j-1} \check{\mathbf{U}}_{j-1} \times_{j+1} \check{\mathbf{U}}_{j+1} \cdots \times_N \check{\mathbf{U}}_N$. The resulting likelihood becomes:

$$p(\mathcal{T}|\mathcal{M}, \{\check{\mathbf{U}}_i\}_{i=2}^N) \approx \prod_{j=2}^N \mathcal{N}(\mathcal{T}|\check{\mathbf{U}}_j\mathbf{J}_j, \sigma_j^2)^{1/N},$$

which can be maximized efficiently.

Taking logarithms and discarding constant factors such as N and σ_j, we seek to minimize the sum-squared error

$$\sum_{j=2}^N \|\mathbf{T}_j - \check{\mathbf{U}}_j\mathbf{J}_j\|_F^2$$

Each term of the summation presents a matrix factorization problem with missing values, where $\check{\mathbf{U}}_j$ and $\mathbf{J}_j$ are treated as unknown factors of the incomplete matrix $\mathbf{T}_j$, and are solved for using PPCA as described above.

5.4 Face Transfer

One produces animations from a multilinear model by varying the attribute parameters (the elements of the $\mathbf{w}_i$'s) as if they were dials, and generating mesh coordinates from Equation 5.5. The N-mode SVD conveniently gives groups of dials that separately control identity, expression and viseme. Within each group, the dials do not correspond to semantically meaningful deformations (such as smile or frown), but rather reflect the deformations that account for most variance. However, the dials can be "tuned" to reflect deformations of interest through a linear transform of each $\mathbf{w}_i$. This approach was successfully applied in [5] to make their body shape dials correspond to height and weight. A similar linear scheme was employed in [21]. In general, dial-based systems are currently used on most of the deformable models in production, but only skilled animators can create believable animations (or even stills) with them. To give similar power to a casual user, we have devised a method that automatically sets model parameters from given video data. With this tool, a user can enact a performance in front of a camera, and have it automatically transferred to the model.

5.4.1 Face Tracking

To link the parameters of a multilinear model to video data, we use optical flow in conjunction with the weak-perspective camera model. Using the symmetric Kanade-Lucas-Tomasi formulation [18], we express the frame-to-frame motion of a tracked point with a linear system:

$$\mathbf{Z}\mathbf{d} = \mathbf{Z}(\mathbf{p} - \mathbf{p}_0) = \mathbf{e}\,. \tag{5.6}$$

Here, the 2-vector $\mathbf{d}$ describes the image-space motion of the point, also expressed as the difference between the point's true location $\mathbf{p}$ and its current best guess $\mathbf{p}_0$ (if we have no guess, then $\mathbf{p}_0$ is the location from the previous frame). Matrix $\mathbf{Z}$ and vector $\mathbf{e}$ contain spatial and temporal intensity gradient information in the surrounding region

[18].

Using a weak-perspective imaging model, the point position $\mathbf{p}$ can be expanded in terms of rigid head-motion parameters and nonrigid facial shape parameters, which are constrained by the multilinear model:

$$\mathbf{Z}(s\mathbf{R}\mathbf{f}_i + \mathbf{t} - \mathbf{p}_0) = \mathbf{e}\,, \tag{5.7}$$

where the rigid parameters consist of scale factor s, the first two rows of a 3D rotation matrix $\mathbf{R}$, and the image-space translation $\mathbf{t}$. The 3D shape $\mathbf{f}$ comes from the multilinear model through Equation (5.5), with $\mathbf{f}_i$ indicating the ith 3D vertex being tracked.

Solving for the pose and all the multilinear weights from a pair of frames using Equation (5.7) is not a well-constrained problem. To simplify the computation, we use a coordinate-descent method: we let only one of the face attributes vary at a time by fixing all the others to their current guesses. This transforms the multilinear problem into a linear one, as described below, which we solve with standard techniques that simultaneously compute the rigid pose along with the linear weights from a pair of frames [14, 24].

When we fix all but one attribute of the multilinear model, thereby making $\mathbf{f}$ linear, Equation (5.7) turns into

$$\mathbf{Z}(s\mathbf{R}\mathbf{M}_{m,i}\mathbf{w}_m + \mathbf{t} - \mathbf{p}_0) = \mathbf{e}\,, \tag{5.8}$$

where m is the mode corresponding to the non-fixed attribute. $\mathbf{w}_m$ is a vector of weights for that attribute, and $\mathbf{M}_{m,i}$ is the corresponding linear basis for the tracked vertex i obtained from

$$\mathbf{M}_m = \mathcal{M} \times_2 \mathbf{w}_2^{\top} \cdots \times_{(m-1)} \mathbf{w}_{(m-1)}^{\top} \times_{(m+1)} \mathbf{w}_{(m+1)}^{\top} \cdots \times_N \mathbf{w}_N^{\top}\,. \tag{5.9}$$

To get a well constrained solution for the per-frame pose (scale, rotation, and translation) as well as the model's attribute parameters (expression, identity, etc.), we track a number of vertices and stack the resulting linear equations into one big system. For each pair of neighboring video frames we assemble a set of linear systems, each one applying Equation (5.8) to one of the tracked vertices. If the currently tracked attribute varies from frame to frame (such as expression does), we solve the set of linear systems and proceed to the next pair of neighboring frames. If, on the other hand, the attribute is constant across all frames (like identity), we accumulate the mentioned linear systems from each pair of frames and solve them together as one combined system. Solving the entire system for a pair of frames and a thousand tracked points completes in about a millisecond. Taking into account several levels of multi-scale and several passes through the whole video, the tracking process averages at one frame per second.

5.4.2 Initialization

The method described above, since it is based on tracking, needs to be initialized with the first frame alignment (pose and all the weights of the multilinear model). We accomplish this by specifying a small number of feature points which are then used to position the face geometry. The correspondences can be either user-provided (which gives more flexibility and power) or automatically detected (which avoids user intervention). We have experimented with the automatic feature detector developed by [148], and found that it is robust and precise enough in locating a number of key features (eye corners, nose tip, mouth corners) to give a good approximating alignment in most cases. Imperfect alignment can be improved by tracking the first few frames back and forth until the model *snaps* into a better location. Other more powerful automated alignment approaches that take in account texture and lighting, such as the one described in [21], could also be adapted to multilinear models.

5.5 Results

Multilinear models provide a convenient control of facial attributes. Figures 5-4 and 5-5 show example manipulations of our bilinear and trilinear models.

Face Transfer infers the attribute parameters automatically by tracking the face in a video. Figure 5-7A shows a few example frames acquired by tracking a face outside of our data set. The resulting 3D shapes, shown below each frame, are generated by the bilinear model with the mouth shapes closed-off for texturing. Because our system tracks approximately a thousand vertices, the process is less sensitive to localized intensity changes (e.g., around the furrow above the lip). Once we obtain the 3D geometry, we can lift the texture from the video by assigning pixel colors to the corresponding mesh vertices. A simple performance-driven texture function can be obtained with the weighted sum of nearest neighbors.

All advantages of our system are combined in video rewrite applications, where a performance is lifted from a video, altered, and seamlessly rendered back into the video. In Figure 5-7B, we use the bilinear model to change a person's identity while retaining the expressions from the original performance. From left to right, the figure shows the original video, the recovered 3D shape, the modified 3D shape, and its textured overlay over the original video (without blending and with the simple texture model). Note how the style of smiling changes with identity. Figure 5-7C shows a post-production example, where a repeat performance is transferred onto the old footage. From left to right, we present the original video frame, a frame from a new video, the new geometry, and the final modified frame (without blending). Here, we combine the pose from the original video with the expression from the new video to modify original expressions.

Our most challenging face transfer uses the trilinear model to transfer expressions and visemes of a singing performance. In Figure 5-7D, the left two images show the frames from two input videos: a target video for a subject in our data set and a source video of a singing performance from a novel subject. The third image shows the final composite (along with the matching geometry as seen from two viewpoints) that combines the expressions and visemes from the source performance with the identity shown in the target video. For best visual results, we blend [114] the face texture from the target video (constant throughout the sequence); the mouth texture from the source video; and the background texture, which includes the eyes peeking

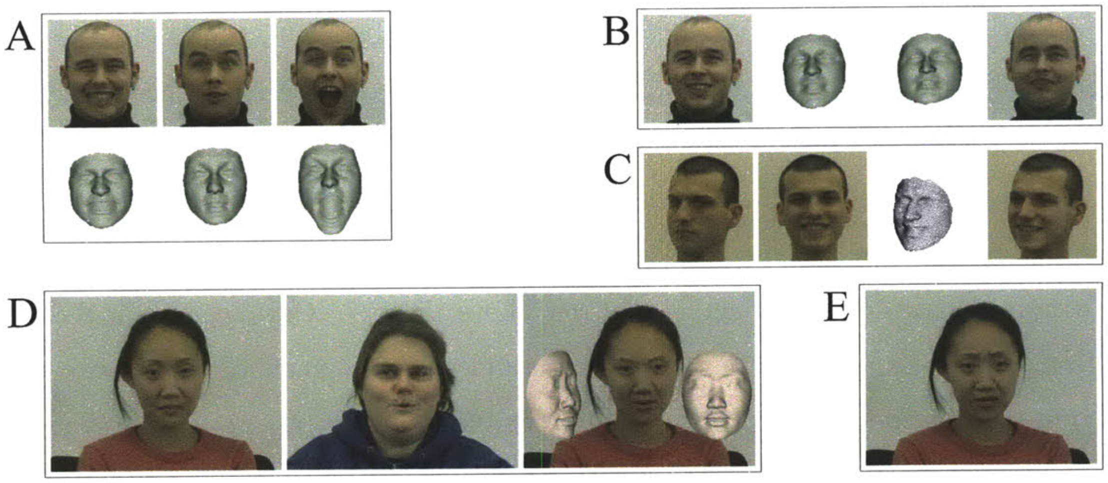


Figure 5-7: Several examples of our system: **(A)** A few frames of a bilinear-model tracking of a novel face and the corresponding 3D shapes below. **(B)** Changing the identity parameters of a performance tracked with the bilinear model. **(C)** Transferring a performance of a known identity from one video to another using the bilinear model. In each example the mouth gap was closed to allow for texturing. **(D)** Using the trilinear model to copy a singing performance from one video to another (left-to-right: original video, singing video, and the result). **(E)** Altering the performance from the previous example by adding a frown.

through the holes in the transferred geometry. In Figure 5-7E, we manually add a frown to the geometry in the final image; the frown texture was lifted from another frame of the same subject. With a similar technique, we can also combine facial attributes from several videos as shown in Figure 5-1, where the pose, expressions, and visemes are mixed from three different input videos.

5.6 Discussion

To summarize, we have shown how to estimate a highly detailed face model from an incomplete set of face scans. The model is multilinear, and thus has the key property of separability: different attributes, such as identity and expression, can be manipulated independently. Thus we can change the identity and expression, but keep the smile. Even more useful, the new smile is in the style idiosyncratic to the new identity.

What makes this multilinear model a practical tool for animation is that we connect it directly to video, showing how to recover a time-series of poses and attribute parameters (expressions and visemes), plus a performance-driven texture function for an actor's face.

Our methods greatly simplify the editing of identity, performance, and facial texture in video, enabling video rewrite applications such as performance animation (puppetry) and actor replacement. In addition, the model offers a rich source of synthetic actors that can be controlled via video.

Limitations. Perhaps our most remarkable empirical result is that even with a model estimated from a rather tiny data set, we can produce videorealistic results for new source and target subjects. Further improvements can be expected when we move to a wider variety of subjects and facial configurations.

We also see algorithmic opportunities to make aspects of the system more automatic and robust. Correspondence between scans might be improved with some of the methods shown in [90]. First-frame alignment would benefit from a successful application of a method such as [21]. Optical flow-based tracking, which is inherently vulnerable to imaging factors such as lighting, occlusions and specular reflections, can be made more robust with edge and corner constraints as demonstrated in [54].

Future Work. In a production setting, the scan data would need to be expanded to contain shape and texture information for the ears, neck, and hair, so that we can make a larger range of head pose changes. Motions of eyes, eyelids, tongues, and teeth, are currently modeled in the texture function or not at all; this will either require more video data or a better solution. Finally, the texture function lifted from video is performance specific, in that we made no effort to remove variations due to lighting. Since we do estimate 3D shape, it may be possible to estimate and remove lighting, given sufficiently long videos.

An intriguing prospect is that one could now build a multilinear model representing a vertex×identity×expression×viseme×*age* data tensor—without having to capture each individual's face at every stage of their life. The model would provide animators with control dials for each of the listed attributes, so they could change an actor's age along with their appearance and performance.

Chapter 6

Conclusion

Dynamic shapes, such as people and clothes in motion, are important in fields ranging from entertainment to medicine. Due to their complexity, however, they are not extensively used in these fields. To address this need, we have attempted to understand dynamic shapes by reconstructing and analyzing their moving geometry. We have shown how to reconstruct moving surfaces of dynamic shapes very efficiently from silhouettes, very accurately from normals, and very pracically from hybrid sensors. We have also demonstrated how to re-animate these dynamic shapes under user control. The methods presented in this thesis might help advance machine vision, robotics and biomechanics, thus impacting our daily lives.

Reconstruction of dynamic shapes from silhouettes (Chapter 2) efficiently yields fully-corresponded high-quality moving meshes with secondary motions such as flapping clothes. These desirable qualities stem from the robustness of silhouette extraction, as well as our non-rigid surface deformation algorithm that molds an articulated template to fit the silhouettes while preserving the template's appearance. We have demonstrated that geometric information in the silhouettes is sufficient to describe the overall pose, the skin deformations, and the dynamic clothing for fast and complex motions such as breakdancing. However, because we only use the information at the contours, the surface in-between the contours is not real, but interpolated from the template. As a result, the reconstructed surface folds do not match the actual folds on the garment. To improve the reconstruction quality (at a penalty to efficiency), one should incorporate other visual cues such as color [47], shading [118], or orientation (Chapter 3). Nevertheless, our output mesh animations convey the flavor and the dynamics of the true motions, and are suitable for creating richer animated characters in computer games and movies.

Reconstruction of dynamic shapes from normals (Chapter 3) produces moving surfaces with unprecedented accuracy and temporal resolution without relying on a template. Using a more elaborate setup (controllable lighting) and more involved processing, we have captured detailed normal maps of dynamic shapes, which complement the silhouette information in-between the contours. They have allowed us to integrate partial surfaces from each camera view and merge them into a single high-resolution mesh with full 360° coverage of the scene. Our output surfaces are incomplete in the regions with visibility issues but otherwise come to within millimeters of the true surface, recovering the true folds on the garments. Furthermore, these surfaces are ideally suited for the sophisticated geometry processing algorithms [83, 95, 170, 155] that are able to aggregate a sequence of incomplete meshes into a single complete and corresponded moving mesh. Our data will drive the advancement of even more powerful geometry processing methods, resulting in moving meshes of fidelity so high that they would be readily usable for accurate biometric measurements in medicine, as well as for digital actors in feature films.

Reconstruction of dynamic shapes from hybrid sensors (Chapter 4) can potentially enable surface capture in natural environments. This is important for motions such as skiing or baseball, which cannot be performed well within easily instrumentable studios. We have demonstrated that inexpensive inertial and acoustic sensors can be used to capture body pose for such motions outside the controlled studio environments. The inertial sensors provide fast updates and high resolution, while the acoustic sensors alleviate the inertial drift. While our system does not recover the global translation and rotation, they can be obtained by using external acoustic sensors [120] where possible, or including additional position (GPS) [29] and orientation (compass) [8] sensors. Practical systems, with their affordability and flexibility, will bring motion capture to consumers everywhere, enabling people to enhance their human-computer interaction, improve their sports performance, avoid injuries due to motion and posture, even carry out physical rehabilitation.

Analysis of the reconstructed dynamic shapes increases their utility and enhances standard animation tasks. In Chapter 5 we have learned the relation between surfaces of different faces in order to transfer motions between them. With a simple multilinear model we were able represent individual idiosyncrasies such as style of smiling, modify facial performances, and transfer them from one video to another. However, our model only describes the large-scale face motion, and does not accurately model fine wrinkles and pore-sized detail. It also represents only the geometric surface of a face, and none of its photometric properties. Both the fine surface detail [17] and the color and reflectance [164] should receive equal treatment if we are to generate photorealistic faces usable for digital actors. Enhanced face models built from many examples will allow for a wide range of applications, from adding a smile to someone's performance, to animating digital models of expensive actors with performances of cheap actors.

Future research in reconstruction of dynamic shapes promises capturing complex dynamic shapes with millimeter precision within several years. We will be able to obtain intricate details on the surface of skin and clothes, including subtle facial expressions, i.e. laser-scan quality [93] for moving surfaces. As a result, we will improve virtual humans and facilitate deeper understanding of dynamic shapes. With the advancement of makerless motion capture [10] and rotation-invariant shape representations [12], we might be able to fit an articulated dynamic shape model to the silhouettes in real-time. This would improve human-computer interaction, allowing our whole bodies to become an input device with many subtle degrees of freedom. For example, one would be able to animate digital characters by enacting their motions, or control cloth animation by simply folding and waving real cloth. Finally, we should combine different sensors in order to capture dynamic shapes and their interaction outside the studios. This would not only increase the range of motions we can reconstruct, but also allow for tracking people in the comfort of their homes for the purpose of, for example, physical rehabilitation.

On the analysis side, we strive to process the captured surfaces in order to learn how they move and deform. We should construct art-directable digital models that incorporate the individual traits of dynamic shapes, i.e. if an artist bends the elbow of a character, the muscles should bulge and the sleeve should fold and flap as they did in reality. These models should also be controlled and displayed interactively for applications such as computer games. We have demonstrated these capabilities for faces (Chapter 5), but, outside the scope of this thesis, we have analogously learned the relation between full-body surfaces of different characters using a novel shape representation called Patch-based Linear Rotation Invariant Coordinates [12]. It has enabled us to transfer motions between very different characters, where the semantics of the mapping are user-defined (i.e. users can map a regular walk onto a hand-walk by providing the appropriate examples). Both of these methods (for faces and full bodies) learn only static shape relations and ignore the dynamics. To get the most out of our models, we should also include the dynamics. One approach to doing so would be to fit the parameters of a physical simulation, such as cloth simulation [16], to our reconstructed data. Novel motions would be generated by a physical simulation, which is very general but hard to control. A more appropriate approach would be to learn the dynamics as a function of our control parameters [13] in a purely data-driven manner. This way the surface would mimic the deformations in the example motions to generate novel motions. Te key properties of these enhanced models should be control and efficiency. They would be ideal for animators, and would potentially alleviate some of the limitations of the reconstruction methods by enabling capture in real-time or capture outside the studios.

The *Holy Grail* in this line of research (Figure 6-1) is to observe dynamic shapes move and interact in nature, reconstruct their moving surfaces, compute detailed models that can be used to analyze the captured motions, and manipulate those models to produce new motions that retain the style and dynamics of the captured shapes. We should strive to close the loop and use the reconstructed models in order to capture novel motions more easily and efficiently. We believe that adapting our techniques to utilize sophisticated dynamic shape models and combine many different

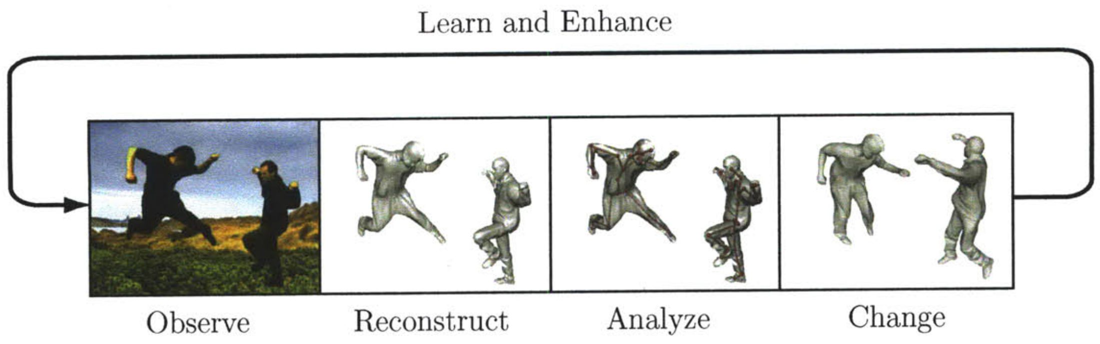


Figure 6-1: We want to observe dynamic shapes interact in nature, reconstruct their moving surfaces, analyze their motions, and change their performances. By learning powerful models of dynamic shapes from several example motions, we can enhance the reconstruction and analysis of these shapes anywhere.

sensors (inertial, acoustic, GPS, compass) in conjuction with cameras and projectors could enable the reconstruction of not only body pose (Chapter 4), but complete moving surfaces, in natural surroundings. We envision reconstructing a number of representative high-quality motions of a dynamic shape within the studio, then using this data in order to efficiently capture this shape anywhere.

Once we reach that goal, we will obtain enhanced dynamic 3D content for computer games, virtual worlds, animated movies, and feature films. We will significantly impact computer vision, improve the way robots understand their environment, and inspire new research in biomechanics. We might be able to assess artistic performances in new ways, or enable a new generation of clothes shopping. We could enhance human-computer interaction and allow our body to become a powerful input device. We could aid rehabilitation or prevent injury by analyzing someone's motion throughout a day. We might be even able to record several days of activity or motions throughout the whole life. As motion capture has become commonplace across a wide variety of fields, moving surfaces will become an enhanced way of capturing, viewing, analyzing and editing dynamic shapes rich with surface details.

Bibliography

[1] Rolf Adelsberger. Practical motion capture in natural surroundings. Master's thesis, ETH Zürich, Zürich, Switzerland, 2007.

[2] Naveed Ahmed, Christian Theobalt, Petat Dobre, Hans-Peter Seidel, and Sebastian Thrun. Robust fusion of dynamic shape and normal capture for high-quality reconstruction of time-varying geometry. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[3] Marc Alexa. Differential coordinates for local mesh morphing and deformation. *The Visual Computer*, 19(2):105–114, 2003.

[4] Brett Allen, Brian Curless, and Zoran Popović. Articulated body deformation from range scan data. *ACM Transactions on Graphics*, 21(3):612–619, July 2002.

[5] Brett Allen, Brian Curless, and Zoran Popović. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphics*, 22(3):587–594, July 2003.

[6] Dragomir Anguelov, Praveen Srinivasan, Daphne Koller, Sebastian Thrun, Jim Rodgers, and James Davis. Scape: Shape Completion and Animation of People. *ACM Transactions on Graphics*, 24(3):408–416, August 2005.

[7] N. Anuar and I. Guskov. Extracting animated meshes with adaptive motion estimation. In *Workshop on Vision, Modeling, and Visualization*, pages 63–71, 2004.

[8] Eric Robert Bachmann. *Inertial and Magnetic Tracking of Limb Segment Orientation for Inserting Humans Into Synthetic Environments.* PhD thesis, Naval Postgraduate School, Monterey, California, 2000.

[9] Alexandru O. Balan, Leonid Sigal, Michael J. Black, James E. Davis, and Horst W. Haussecker. Detailed human shape and pose from images. In *Computer Vision and Pattern Recognition*, 2007.

[10] 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 *3D Data Processing, Visualization and Transmission*, June 2008.

[11] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics*, 26(3), 2007.

[12] Ilya Baran, Daniel Vlasic, Eitan Grinspun, and Jovan Popović. Semantic deformation transfer. *ACM Transactions on Graphics*, 28(3)::–:, 2009.

[13] Jernej Barbič, Marco da Silva, and Jovan Popović. Deformable object animation using reduced optimal control. *ACM Transactions on Graphics (SIGGRAPH 2009)*, 28(3), 2009.

[14] B. Bascle and A. Blake. Separability of pose and expression in facial tracking and animation. In *International Conference on Computer Vision (ICCV)*, pages 323–328, 1998.

[15] Fausto Bernardini, Holly Rushmeier, Ioana M. Martin, Joshua Mittleman, and Gabriel Taubin. Building a digital model of michelangelo's florentine pietà. *IEEE Computer Graphics & Applications*, 22(1):59–67, January/February 2002.

[16] Kiran S. Bhat, Christopher D. Twigg, Jessica K. Hodgins, Pradeep K. Khosla, Zoran Popović, and Steven M. Seitz. Estimating cloth simulation parameters from video. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 37–51, 2003.

[17] Bernd Bickel, Mario Botsch, Roland Angst, Wojciech Matusik, Miguel Otaduy, Hanspeter Pfister, and Markus Gross. Multi-scale capture of facial geometry and motion. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 26(3):33, 2007.

[18] Stan Birchfield. KLT: An implementation of the kanade-lucas-tomasi feature tracker. http://www.ces.clemson.edu /~stb/, 1996.

[19] Thomas Gary Bishop. *Self-Tracker: A Smart Optical Sensor on Silicon.* PhD thesis, University of North Carolina at Chapel Hill, 1984.

[20] Volker Blanz, C. Basso, T. Poggio, and T. Vetter. Reanimating faces in images and video. *Computer Graphics Forum*, 22(3):641–650, September 2003.

[21] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 187–194, August 1999.

[22] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless garment capture. *ACM Transactions on Graphics*, 27(3):99:1–99:9, August 2008.

[23] Matthew E. Brand. Incremental singular value decomposition of uncertain data with missing values. In *European Conference on Computer Vision (ECCV)*, volume 2350, pages 707–720, 2002.

[24] Matthew E. Brand and R. Bhotika. Flexible flow for 3D nonrigid tracking and shape recovery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 315–322, 2001.

[25] Helene Brashear, Thad Starner, Paul Lukowicz, and Holger Junker. Using multiple sensors for mobile sign language recognition. In *International Symposium on Wearable Computers*, pages 45–53, 2003.

[26] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Conference on Computer Vision and Pattern Recognition*, pages 8–15, 1998.

[27] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 353–360, August 1997.

[28] Christoph Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 690–696, 2000.

[29] Matthew Brodie, Alan Walmsley, and Wyatt Page. Fusion motion capture: a prototype system using inertial measurement units and gps for the biomechanical analysis of ski racing. 1(1):17–28, 2008.

[30] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proceedings of the 8th European Conference on Computer Vision*, pages 25–36, 2004.

[31] N.D.F. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Automatic 3d object segmentation in multiple views using volumetric graph-cuts. In *British Machine Vision Conference*, 2007.

[32] Yong Cao, Petros Faloutsos, and Frédéric Pighin. Unsupervised learning for speech motion editing. In *Eurographics/SIGGRAPH Symposium on Computer animation (SCA)*, pages 225–231, 2003.

[33] Joel Carranza, Christian Theobalt, Marcus A. Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics*, 22(3):569–577, July 2003.

[34] Jin-xiang Chai, Jing Xiao, and Jessica Hodgins. Vision-based control of 3D facial animation. In *Eurographics/SIGGRAPH Symposium on Computer Animation (SCA)*, pages 193–206, 2003.

[35] Will Chang and Matthias Zwicker. Automatic registration for articulated shapes. *Computer Graphics Forum (Proceedings of SGP 2008)*, 27(5):1459–1468, 2008.

[36] Yisheng Chen, J. Lee, R. Parent, and R. Machiraju. Markerless monocular motion capture using image features and physical constraints. In *Computer Graphics International*, pages 36–43, June 2005.

[37] 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, July 2005.

[38] Erika S. Chuang, Hrishikesh Deshpande, and Chris Bregler. Facial expression space learning. In *Pacific Conference on Computer Graphics and Applications (PG)*, pages 68–76, 2002.

[39] S. Corazza, Lars Mündermann, A. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, July 2006.

[40] S. Corazza, Lars Mündermann, A. Chaudhari, T. Demattio, C. Cobelli, and T. P. Andriacchi. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals of Biomedical Engineering*, 34(6):1019–1029, July 2006.

[41] A. Criminisi, A. Blake, C. Rother, J. Shotton, and P. H. Torr. Efficient dense stereo with occlusions for new view-synthesis by four-state dynamic programming. *Int. Journal of Computer Vision*, 71(1):89–110, 2007.

[42] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 303–312, August 1996.

[43] P.-E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, 14:227–248, 1980.

[44] James Davis, Steven R. Marschner, Matt Garr, and Marc Levoy. Filling holes in complex surfaces using volumetric diffusion. In *Symposium on 3D Data Processing, Visualization, and Transmission*, pages 428–438, 2002.

[45] James Davis, Ravi Ramamoorthi, and Szymon Rusinkiewicz. Spacetime stereo: A unifying framework for depth from triangulation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 196–302, 2005.

[46] Andrew J. Davison, Jonathan Deutscher, and Ian D. Reid. Markerless motion capture of complex full-body movement for character animation. In *Computer Animation and Simulation*, pages 3–14, 2001.

[47] Edilson de Aguiar, Carsten Stoll, Christian Theobalt, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. Performance capture from sparse multi-view video. *ACM Transactions on Graphics*, 27(3):98:1–98:10, August 2008.

[48] Edilson de Aguiar, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Reconstructing human shape and motion from multi-view video. In *Conference on Visual Media Production*, pages 44–51, December 2005.

[49] Edilson de Aguiar, Christian Theobalt, Marcus Magnor, Holger Theisel, and Hans-Peter Seidel. M3: marker-free model reconstruction and motion tracking from 3D voxel data. In *Pacific Conference on Computer Graphics and Applications*, pages 101–110, October 2004.

[50] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *Computer Vision and Pattern Recognition*, 2007.

[51] Edilson de Aguiar, Christian Theobalt, Carsten Stoll, and Hans-Peter Seidel. Rapid animation of laser-scanned humans. In *Virtual Reality*, pages 223–226, 2007.

[52] Martin de La Gorce, Nikos Paragios, and David J. Fleet. Model-based hand tracking with texture, shading and self-occlusions. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[53] Lieven De Lathauwer. *Signal Processing based on Multilinear Algebra.* PhD thesis, Katholieke Universiteit Leuven, Belgium, 1997.

[54] Douglas DeCarlo and Dimitris Metaxas. The integration of optical flow and deformable models with applications to human face shape and motion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 231–238, 1996.

[55] Douglas DeCarlo and Dimitris Metaxas. Optical flow constraints on deformable models with applications to face tracking. *International Journal of Computer Vision*, 38(2):99–127, 2000.

[56] Jonathan Deutscher and Ian Reid. Articulated body motion capture by stochastic search. *Int. Journal of Computer Vision*, 61(2):185–205, 2005.

[57] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. In *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, pages 409–416, 2003.

[58] P. Einarsson, C-F. Chabert, A. Jones, W-C Ma, B. Lamond, T. Hawkins, M. Bolas, S. Sylwan, and P. Debevec. Relighting human locomotion with flowed reflectance fields. In *Proc. of Eurographics Symposium on Rendering*, pages 183–194, 2006.

[59] Ali Erol, George Bebis, Richard D. Boyle, and Mircea Nicolescu. Visual hull construction using adaptive sampling. In *Workshop on Applications of Computer Vision*, pages 234–241, 2005.

[60] Irfan Essa, S. Basu, T. Darrell, and Alex Pentland. Modeling, tracking and interactive animation of faces and heads: Using input from video. In *Computer Animation '96*, pages 68–79, June 1996.

[61] Carlos Hernández Esteban and Francis Schmitt. Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392, December 2004.

[62] Tony Ezzat and Tomaso Poggio. Visual speech synthesis by morphing visemes. *International Journal of Computer Vision*, 38(1):45–57, 2000.

[63] M. A. Fogel, D. R. Wilson, A. Flake, M. Johnson, D. Cohen, G. McNeal, Z. Y. Tian, and J. Rychik. Preliminary investigations into a new method of functional assessment of the fetal heart using a novel application of 'real-time' cardiac magnetic resonance imaging. *Fetal Diagnosis and Therapy*, 20:475–480, 2005.

[64] Eric Foxlin. Pedestrian tracking with shoe-mounted inertial sensors. *Computer Graphics and Applications*, 25(6):38–46, 2005.

[65] Eric Foxlin and Michael Harrington. Weartrack: A self-referenced head and hand tracker for wearable computers and portable VR. In *International Symposium on Wearable Computers*, pages 155–162, 2000.

[66] Eric Foxlin, Michael Harrington, and George Pfeifer. Constellation: A wide-range wireless motion-tracking system for augmented reality and virtual set applications. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 371–378, July 1998.

[67] W. T. Freeman and J. B. Tenenbaum. Learning bilinear models for two factor problems in vision. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 554–560, 1997.

[68] William T. Freeman and Craig D. Weissman. Television control by hand gestures. Technical Report TR94-24, Mitsubishi Electric Research Laboratories, 1994.

[69] W. Frey. Off-the-shelf, real-time, human body motion capture for synthetic environments. Technical Report NPSCS-96-003, Naval Postgraduate School, Monterey, California, 1996.

[70] Y. Furukawa and J. Ponce. Carved visual hulls for image-based modeling. In *European Conference on Computer Vision*, pages 564–577, 2006.

[71] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[72] Arthur Gelb, editor. *Applied Optimal Estimation*. MIT Press, Cambridge, Massachusetts, 1974.

[73] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(6):643–660, 2001.

[74] Philip E. Gill, Walter Murray, and Michael A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[75] B. Goldlücke, I. Ihrke, C. Linz, and M. Magnor. Weighted minimal hypersurface reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1194–1208, July 2007.

[76] Craig Gotsman, Xianfeng Gu, and Alla Sheffer. Fundamentals of spherical parameterization for 3D meshes. *ACM Transactions on Graphics*, 22(3):358–363, July 2003.

[77] Martin Habbecke and Leif Kobbelt. A surface-growing approach to multi-view stereo reconstruction. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[78] Mike Hazas and Andy Ward. A novel broadband ultrasonic location system. In *International Conference on Ubiquitous Computing*, pages 264–280, 2002.

[79] Carlos Hernandez, George Vogiatzis, and Roberto Cipolla. Multiview photometric stereo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(3):548–554, 2008.

[80] Aaron Hertzmann and Steven M. Seitz. Example-based photometric stereo: Shape reconstruction with general, varying brdfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(8):1254–1264, 2005.

[81] Jeffrey Hightower and Gaetano Borriello. Location systems for ubiquitous computing. *Computer*, 34(8):57–66, August 2001.

[82] Alexander Hornung and Leif Kobbelt. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *Computer Vision and Pattern Recognition*, pages 503–510, 2006.

[83] Qi-Xing Huang, Bart Adams, Martin Wicke, and Leonidas J. Guibas. Non-rigid registration under isometric deformations. *Computer Graphics Forum (Proc. SGP'08)*, 27(5):1449–1457, 2008.

[84] Thouis R. Jones, Frédo Durand, and Mathieu Desbrun. Non-iterative, feature-preserving mesh smoothing. *ACM Transactions on Graphics*, 22(3):943–949, July 2003.

[85] N. Joshi and D. Kriegman. Shape from varying illumination and viewpoint. In *International Conference on Computer Vision*, 2007.

[86] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, 2006.

[87] Scott Kircher and Michael Garland. Editing arbitrarily deforming surface animations. *ACM Transactions on Graphics*, 25(3):1098–1107, July 2006.

[88] Adam G. Kirk, James F. OBrien, and David A. Forsyth. Skeletal parameter estimation from optical motion capture data. In *Conference on Computer Vision and Pattern Recognition*, pages 782–788, 2005.

[89] R. M. Koch, M. H. Gross, F. R. Carls, D. F. von Büren, G. Fankhauser, and Y. Parish. Simulating facial surgery using finite element methods. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 421–428, August 1996.

[90] Vladislav Kraevoy and Alla Sheffer. Cross-parameterization and compatible remeshing of 3D models. *ACM Transactions on Graphics*, 23(3):861–869, August 2004.

[91] P. M. Kroonenberg and J. De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45:69–97, 1980.

[92] Yuencheng Lee, Demetri Terzopoulos, and Keith Waters. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 95*, Computer Graphics Proceedings, Annual Conference Series, pages 55–62, August 1995.

[93] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In *Proceedings of ACM SIGGRAPH 2000*, Annual Conference Series, pages 131–144, July 2000.

[94] H. Li, P. Roivainen, and R. Forchheimer. 3-D motion estimation in model-based facial image coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 15(6):545–555, 1993.

[95] Hao Li, Robert W. Sumner, and Mark Pauly. Global correspondence optimization for non-rigid registration of depth scans. *Computer Graphics Forum*, 27(5):1421–1430, 2008.

[96] J. Lim, J. Ho, M.-H.Yang, and D. Kriegman. Passive photometric stereo from motion. In *International Conference on Computer Vision*, 2005.

[97] Yaron Lipman, Olga Sorkine, Daniel Cohen-Or, David Levin, Christian Rössl, and Hans-Peter Seidel. Differential coordinates for interactive mesh editing. In *Shape Modeling International*, pages 181–190, 2004.

[98] Wan-Chun Ma, Tim Hawkins, Pieter Peers, Charles-Felix Chabert, Malte Weiss, and Paul Debevec. Rapid acquisition of specular and diffuse normal maps from polarized spherical gradient illumination. In *Rendering Techniques*, pages 183–194, 2007.

[99] Clément Ménier, Edmond Boyer, and Bruno Raffin. 3D skeleton-based body pose recovery. In *3D Data Processing, Visualization and Transmission*, pages 389–396, 2006.

[100] Kenneth Meyer, Hugh L. Applewhite, and Frank A. Biocca. A survey of position-trackers. *Presence*, 1(2):173–200, 1992.

[101] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*, pages 35–57. Springer, 2002.

[102] Nathan Miller, Odest Chadwicke Jenkins, Marcelo Kallmann, and Maja J. Matrić. Motion capture from inertial sensing for untethered humanoid teleoperation. In *International Conference of Humanoid Robotics*, pages 547–565, Nov 2004.

[103] N. J. Mitra, S. Flory, M. Ovsjanikov, N. Gelfand, L. Guibas, and H. Pottmann. Dynamic geometry registration. In *Proc. Symposium on Geometry Processing*, pages 173–182, 2007.

[104] Diego Nehab, Szymon Rusinkiewicz, James Davis, and Ravi Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *ACM Transactions on Graphics*, 24(3):536–543, August 2005.

[105] Jun-yong Noh and Ulrich Neumann. Expression cloning. In *Proceedings of SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 277–288, August 2001.

[106] James O'Brien, Robert Bodenheimer, Gabriel Brostow, and Jessica Hodgins. Automatic joint parameter estimation from magnetic motion capture data. In *Graphics Interface*, pages 53–60, 2000.

[107] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[108] E. Olson, J. Leonard, and S. Teller. Robust range-only beacon localization. *Journal of Oceanic Engineering*, 31(4):949–958, October 2006.

[109] Sang Il Park and Jessica K. Hodgins. Capturing and animating skin deformation in human motion. *ACM Transactions on Graphics*, 25(3):881–889, July 2006.

[110] F. I. Parke. *A parametric model for human faces.* PhD thesis, University of Utah, Salt Lake City, Utah, December 1974.

[111] F. I. Parke. Parameterized models for facial animation. *IEEE Computer Graphics & Applications*, 2:61–68, November 1982.

[112] Yuri Pekelny and Craig Gotsman. Articulated object reconstruction and markerless motion capture from depth video. *Computer Graphics Forum*, 27(2):399–408, April 2008.

[113] Alex Pentland and Stan Sclaroff. Closed-form solutions for physically based shape modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 13(7):715–729, 1991.

[114] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics*, 22(3):313–318, July 2003.

[115] Frédéric Pighin, Jamie Hecker, Dani Lischinski, Richard Szeliski, and David H. Salesin. Synthesizing realistic facial expressions from photographs. In *Proceedings of SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 75–84, July 1998.

[116] Frederic H. Pighin, Richard Szeliski, and David Salesin. Resynthesizing facial animation through 3d model-based tracking. In *International Conference on Computer Vision (ICCV)*, pages 143–150, 1999.

[117] Ralf Plänkers and Pascal Fua. Articulated soft objects for video-based body modeling. In *International Conference on Computer Vision*, pages 394–401, 2001.

[118] 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*, 28(2), 2009.

[119] Emil Praun and Hugues Hoppe. Spherical parameterization and remeshing. *ACM Transactions on Graphics*, 22(3):340–349, July 2003.

[120] Nissanka Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *International Conference on Mobile Computing and Networking*, pages 32–43, August 2000.

[121] Cliff Randell and Henk L. Muller. Low cost indoor positioning system. In *International Conference on Ubiquitous Computing*, pages 42–48, 2001.

[122] Peter W. Rander, PJ Narayanan, and Takeo Kanade. Virtualized reality: Constructing time-varying virtual worlds from real world events. In *IEEE Visualization*, pages 277–284, November 1997.

[123] Barbara Robertson. Locomotion. *Computer Graphics World*, December 2004.

[124] Sam Roweis. EM algorithms for PCA and SPCA. In *Advances in neural information processing systems 10 (NIPS)*, pages 626–632, 1997.

[125] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3D model acquisition. *ACM Transactions on Graphics (Proc. ACM SIGGRAPH)*, 21(3):438–446, July 2002.

[126] Ryusuke Sagawa, Nanaho Osawa, and Yasushi Yagi. Deformable registration of textured range images by using texture and shape features. In *3DIM '07: Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling*, pages 65–72, 2007.

[127] Peter Sand, Leonard McMillan, and Jovan Popović. Continuous capture of skin deformation. *ACM Transactions on Graphics*, 22(3):578–586, July 2003.

[128] V. Scholz, T. Stich, M. Keckeisen, M. Wacker, and M. Magnor. Garment motion capture using color-coded patterns. *Computer Graphics Forum*, 24(3):439–448, August 2005.

[129] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Computer Vision and Pattern Recognition*, pages 519–528, June 2006.

[130] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[131] Andrei Sharf, Dan A. Alcantara, Thomas Lewiner, Chen Greif, Alla Sheffer, Nina Amenta, and Daniel Cohen-Or. Space-time surface reconstruction using incompressible flow. *ACM Trans. Graph.*, 27(5):1–10, 2008.

[132] L. Sirovich and M. Kirby. Low dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*, 4:519–524, 1987.

[133] J. Starck and A. Hilton. Model-based multiple view reconstruction of people. In *International Conference on Computer Vision*, pages 915–922, 2003.

[134] J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.

[135] C. Studholme, C. S. Drapaca, B. Iordanova, and V. Cardenas. Deformation-based mapping of volume change from serial brain mri in the presence of local tissue contrast change. *IEEE Transactions on Medical Imaging*, 25(5):626–639, May 2006.

[136] Robert W. Sumner and Jovan Popović. Deformation transfer for triangle meshes. *ACM Transactions on Graphics*, 23(3):399–405, August 2004.

[137] Tomáš Svoboda, Daniel Martinec, and Tomáš Pajdla. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments*, 14(4):407–422, August 2005.

[138] 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, July/August 2007.

[139] Christian Theobalt, Edilson de Aguiar, Marcus Magnor, Holger Theisel, and Hans-Peter Seidel. Marker-free kinematic skeleton estimation from sequences of volume data. In *Symposium on Virtual Reality Software and Technology*, pages 57–64, November 2004.

[140] Christian Theobalt, Marcus Magnor, Pascal Schüler, and Hans-Peter Seidel. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. In *Pacific Conference on Computer Graphics and Applications*, pages 96–103, October 2002.

[141] Michael E. Tipping and Christopher M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61(3):611–622, 1999.

[142] L. Torresani, D. Yang, E. Alexander, and C. Bregler. Tracking and modeling non-rigid objects with rank constraints. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 493–450, 2001.

[143] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, September 1966.

[144] Raquel Urtasun, David J. Fleet, and Pascal Fua. Temporal motion models for monocular and multiview 3d human body tracking. *Computer Vision and Image Understanding*, 104(2):157–177, 2006.

[145] Nicholas Michael Vallidis. *WHISPER: a spread spectrum approach to occlusion in acoustic tracking.* PhD thesis, University of North Carolina at Chapel Hill, 2002.

[146] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *European Conference on Computer Vision (ECCV)*, pages 447–460, May 2002.

[147] M. Alex O. Vasilescu and Demetri Terzopoulos. Tensortextures: multilinear image-based rendering. *ACM Transactions on Graphics*, 23(3):336–342, August 2004.

[148] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[149] Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popović. Practical motion capture in everyday surroundings. *ACM Transactions on Graphics*, 26(3):35:1–35:9, 2007.

[150] Daniel Vlasic, Ilya Baran, Wojciech Matusik, and Jovan Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3):97:1–97:9, 2008.

[151] Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. *ACM Transactions on Graphics*, 24(3):426–433, 2005.

[152] Daniel Vlasic, Pieter Peers, Ilya Baran, Paul Debevec, Jovan Popović, Szymon Rusinkiewicz, and Wojciech Matusik. Dynamic shape capture using multi-view photometric stereo. *ACM Transactions on Graphics*, 28(5):174, 2009.

[153] G. Vogiatzis, C. Hernandez, and R. Cipolla. Reconstruction in the round using photometric normals and silhouettes. In *2006 Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 1847–1854, June 2006.

[154] G. Vogiatzis, P. H. S. Torr, and R. Cipolla. Multi-view stereo via volumetric graph-cuts. In *Computer Vision and Pattern Recognition*, pages 391–398, 2005.

[155] 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 Transactions on Graphics*, 28(2):1–15, 2009.

[156] Michael Wand, Philipp Jenke, Qixing Huang, Martin Bokeloh, Leonidas Guibas, and Andreas Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. Symposium on Geometry Processing*, pages 49–58, July 2007.

[157] Robert Y. Wang, Kari Pulli, and Jovan Popović. Real-time enveloping with rotational regression. *ACM Transactions on Graphics*, 26(3):73:1–73:9, July 2007.

[158] Yang Wang, Xiaolei Huang, Chan-Su Lee, Song Zhang, Zhiguo Li, Dimitris Samaras, Dimitris Metaxas, Ahmed Elgammal, and Peisen Huang. High resolution acquisition, learning and transfer of dynamic 3-d facial expressions. *Computer Graphics Forum*, 23(3):677–686, September 2004.

[159] Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *Personal Communications*, 4(5):42–47, October 1997.

[160] Jamie A. Ward, Paul Lukowicz, and Gerhard Tröster. Gesture spotting using wrist worn microphone and 3-axis accelerometer. In *Joint Conference on Smart Objects and Ambient Intelligence*, pages 99–104, 2005.

[161] Keith Waters. A muscle model for animating three-dimensional facial expression. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, pages 17–24, 1987.

[162] G. Welch and E. Foxlin. Motion tracking: no silver bullet, but a respectable arsenal. *Computer Graphics and Applications*, 22(6):24–38, November/December 2002.

[163] Greg Welch and Gary Bishop. Scaat: Incremental tracking with incomplete information. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 333–344, August 1997.

[164] Tim Weyrich, Wojciech Matusik, Hanspeter Pfister, Bernd Bickel, Craig Donner, Chien Tu, Janet McAndless, Jinho Lee, Addy Ngan, Henrik Wann Jensen, and Markus Gross. Analysis of human faces using a measurement-based skin reflectance model. *ACM Transactions on Graphics*, 25(3):1013–1024, 2006.

[165] Ryan White, Keenan Crane, and D. A. Forsyth. Capturing and animating occluded cloth. *ACM Transactions on Graphics*, 26(3):34:1–34:8, July 2007.

[166] Lance Williams. Performance-driven facial animation. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, pages 235–242, 1990.

[167] H. J. Woltring. New possibilities for human motion studies by real-time light spot position measurement. *Biotelemetry*, 1(3):132–146, 1974.

[168] R. J. Woodham. Photometric stereo: A reflectance map technique for determining surface orientation from image intensity. In *Proc. SPIE's 22nd Annual Technical Symposium*, volume 155, August 1978.

[169] Yasuyoshi Yokokohji, Yuki Kitaoka, and Tsuneo Yoshikawa. Motion capture from demonstrator's viewpoint and its application to robot teaching. *Journal of Robotic Systems*, 22(2):87–97, 2005.

[170] Hao Zhang, Alla Sheffer, Daniel Cohen-Or, Qingnan Zhou, Oliver van Kaick, and Andrea Tagliasacchi. Deformation-driven shape correspondence. *Proc. Symposium on Geometry Processing*, 27(5):1431–1439, July 2008.

[171] Li Zhang, Noah Snavely, Brian Curless, and Steven M. Seitz. Spacetime faces: high resolution capture for modeling and animation. *ACM Transactions on Graphics*, 23(3):548–558, August 2004.

[172] S. Zhang and P. Huang. High-resolution real-time three-dimensional shape measurement. *Optical Engineering*, 45(12), 2006.