Articulated Mesh Animation from Multi-view Silhouettes

Daniel Vlasic Ilya Baran Wojciech Matusik[†] Jovan Popović

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
[†]Adobe Systems Inc.

Abstract

Details in mesh animations are difficult to generate but they have great impact on visual quality. In this work, we demonstrate a practical software system for capturing such details from multi-view video recordings. Given a stream of synchronized video images that record a human performance from multiple viewpoints and an articulated template of the performer, our system captures the motion of both the skeleton and the shape. The output mesh animation is enhanced with the details observed in the image silhouettes. For example, a performance in casual loose-fitting clothes will generate mesh animations with flowing garment motions. We accomplish this with a fast pose tracking method followed by nonrigid deformation of the template to fit the silhouettes. The entire process takes less than sixteen seconds per frame and requires no markers or texture cues. Captured meshes are in full correspondence making them readily usable for editing operations including texturing, deformation transfer, and deformation model learning.

CR Categories: I.3.7 [Computer Graphics]: Three Dimensional Graphics and Realism—Animation

Keywords: deformation, motion capture

1 Introduction

Meshes are most often animated by hand using keyframing and procedural deformations. Adding detail with this approach is tedious because it requires setting more than a hundred 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 paper, we pursue a complementary direction to create detailed mesh animations through observation rather than simulation. Motion capture of moving meshes provides an alternative to efficient generation of detailed animations. The potential value of this approach is readily apparent in the success of motion capture for skeletal animation. There, observed motions of the skeleton are edited, transformed, interpolated, and recomposed to produce new high-quality animations. The relative simplicity of that approach has already taken root in the film industry where recorded motions are often used instead of more traditional animation techniques. What current motion capture lacks, however, is the ability to efficiently acquire both the internal skeleton and the external shape.

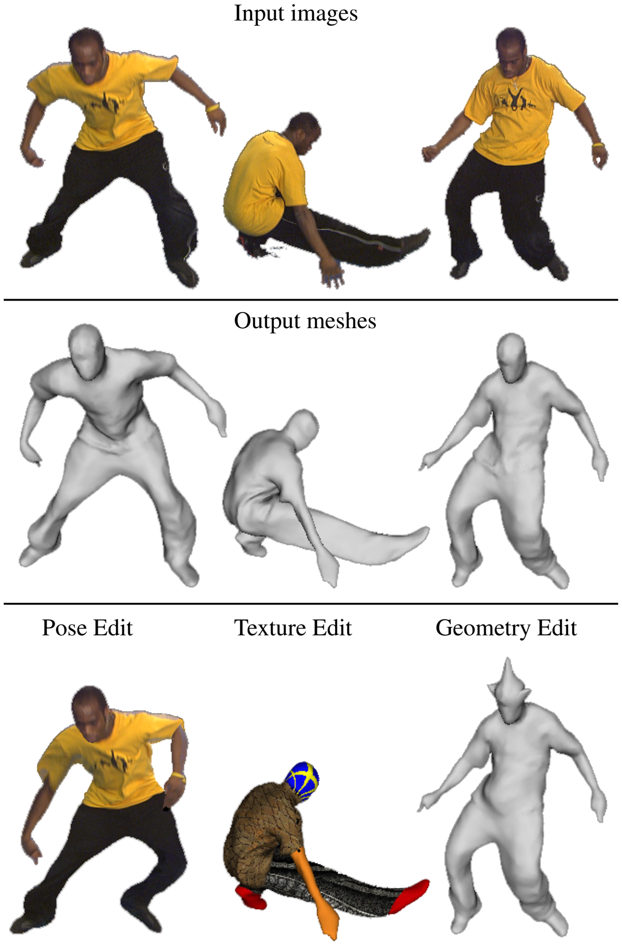


Figure 1: *Our methods for pose tracking and non-rigid shape matching make it possible to extract a mesh animation with full correspondence (middle row) from multi-view video data (top row), allowing easy editing of pose, appearance, and character geometry (bottom row).*

The ability to modify recorded shapes is also vitally important. Recorded mesh animations need to be edited, transformed, interpolated, and recomposed so that new high-quality meshes can be animated as easily as skeletons. This requires a method that maintains one-to-one vertex correspondence throughout the entire motion. This correspondence simplifies numerous applications including texture editing, deformation transfer, and posing.

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 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.

1.1 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 [Ménier et al. 2006; Cheung et al. 2005; Theobalt et al. 2002; de Aguiar et al. 2004]. 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. [2003] capture some of those effects with silhouette-bounded needles radiating from each bone, without maintaining frame-to-frame correspondence. Park and Hodgins [2006] 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 [Scholz et al. 2005; White et al. 2007]. 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 [de Aguiar et al. 2007b; de Aguiar et al. 2007a]. These approaches are not as reliable on fast motions, especially when there are few textures cues (e.g., pants in Figure 1).

The multi-view stereo literature provides a powerful collection of tools for recovering a single static mesh [Seitz et al. 2006]. 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 [Hornung and Kobbelt 2006; Furukawa and Ponce 2006; Esteban and Schmitt 2004]. Faster carving methods sacrifice quality but capture meshes at speeds more practical for processing video streams [Rander et al. 1997; Starck and Hilton 2007; Goldlücke et al. 2007]. 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). Our approach is much faster and it generates mesh animations with frame-to-frame vertex correspondence.

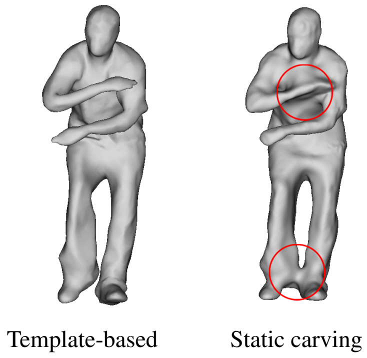


Figure 2: *Applying static carving techniques can result in topology issues, such as the connected feet and arm in this frame while template-based reconstruction ensures a consistent topology.*

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 [Carranza et al. 2003; Theobalt et al. 2007; Cheung et al. 2005; Anguelov et al. 2005; Corazza et al. 2006; Balan et al. 2007]. A template based on implicit surfaces [Plänkers and Fua 2001] can capture time-varying detail, but does not produce corresponded meshes. To learn a skinning model, Allen et al. [2002] 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 [2005] 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.

1.2 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 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

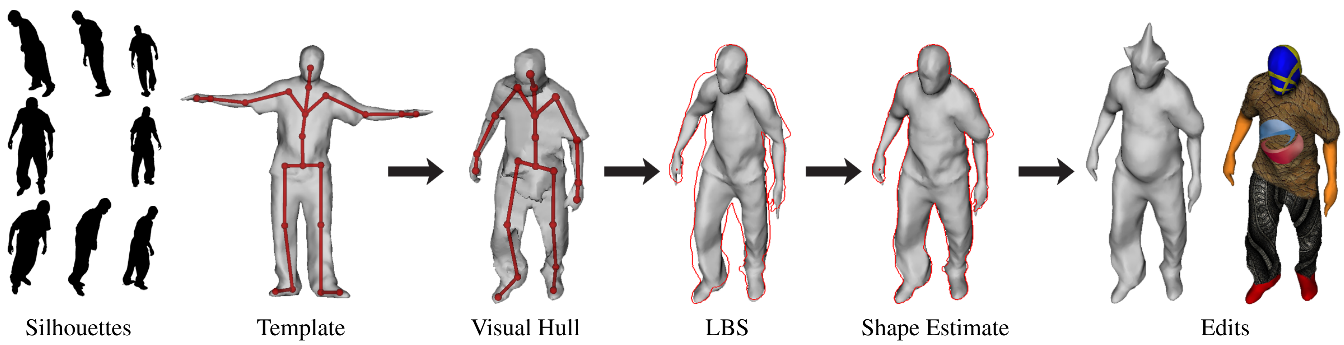


Figure 3: *From left to right, our system starts with a stream of silhouette videos and a rigged template mesh. At every frame, it fits the skeleton to the visual hull, deforms the template using linear blend skinning (LBS), and adjusts the deformed template to fit the silhouettes. The user can then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

and surface refinement. In the first stage (Section 2), 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 3), our system deforms the template according to the skeleton and adjusts its surface to better match the silhouettes.

2 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.

2.1 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:

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

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 3) 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.

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.

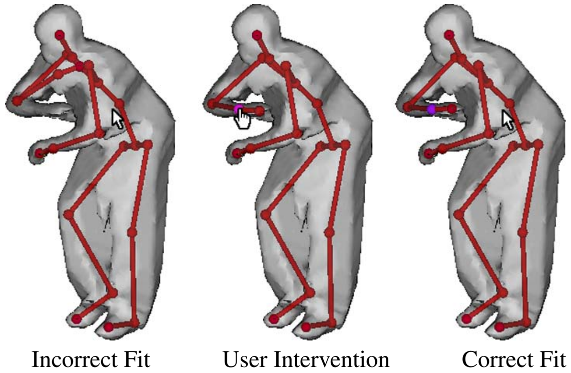


Figure 4: *An incorrectly fit skeleton is fixed by the user dragging the wrist joint. Our system repositions the hand joint automatically to fit the visual hull.*

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, respectively, 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 4.2, our method is not overly sensitive to their precise values.

2.2 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 [Erol et al. 2005].

To compute d_i on the image plane, we adapt a vector distance transform algorithm [Danielsson 1980]. 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.

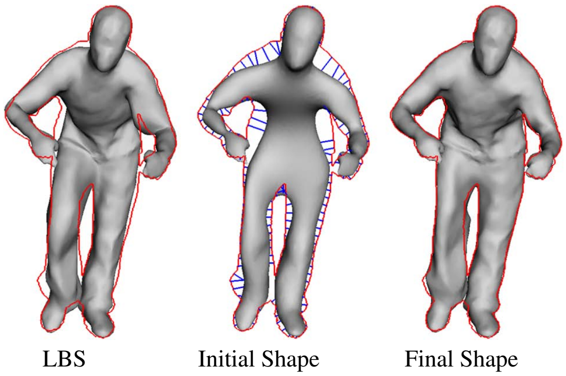


Figure 5: *The template deformed with LBS suffers from artifacts around joints and does not fit the silhouettes. Our method smooths it and constrains it to the contours, gradually reintroducing detail. The result after several iterations has the detail of the original template, fits the silhouettes,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

2.3 Processing

Because our objective function is nonlinear in joint angles, we use SNOPT [Gill et al. 2002] 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 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.

3 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 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 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.

3.1 Laplacian Coordinates

We convert our template mesh to Laplacian coordinates [Alexa 2003; Lipman et al. 2004] in order to represent the iteratively

Algorithm 1 Shape Estimation

```
 1: $k \leftarrow$ number of frames
 2: for $t = 1$ to $k$ do
 3:     $(\mathbf{C}_{\text{LBS}}, \mathbf{P}_{\text{LBS}}) \leftarrow$ nearly-rigid vertices
 4:     $\mathbf{V}_t \leftarrow$ solve Equation 1
 5: end for
 6: $\mathbf{V}_{1..k} \leftarrow \text{TemporalFilter}(\mathbf{V}_{1..k})$
 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:         $(\mathbf{C}_{\text{SIL}}, \mathbf{P}_{\text{SIL}}) \leftarrow$ silhouette constraints using current $\mathbf{V}_t$
11:         $\mathbf{V}_t \leftarrow$ solve Equation 2
12:     end for
13:     $\mathbf{V}_{1..k} \leftarrow \text{TemporalFilter}(\mathbf{V}_{1..k})$
14: end for
```

deforming shape. Let $\mathbf{V}_{\text{T}}$ be the $n \times 3$ matrix storing the template mesh vertex coordinates. Then Laplacian coordinates $\mathbf{L}$ are generated with:

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

where $\mathbf{\Delta}$ is the $n \times n$ mesh Laplacian matrix (cotangent weights work best [Meyer et al. 2003]). 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 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.

3.2 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 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:

$$\underset{\mathbf{V}}{\arg\min} \left(\|\mathbf{\Delta V}\|^2 + w_C \|\mathbf{C}_{\text{LBS}}\mathbf{V} - \mathbf{P}_{\text{LBS}}\|^2 \right), \tag{1}$$

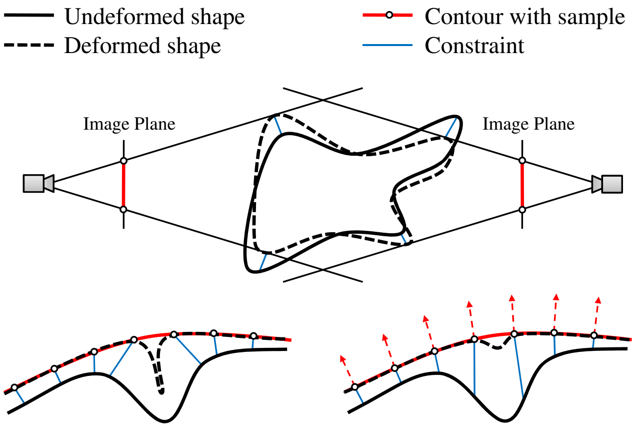


Figure 6: *We deform the template to fit the silhouettes by constraining at most one surface point to each contour sample ray (top). Adjusting the template by pulling the closest surface point toward each contour sample (bottom left) exacerbates folding effects, while preferring the points in the direction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

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 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

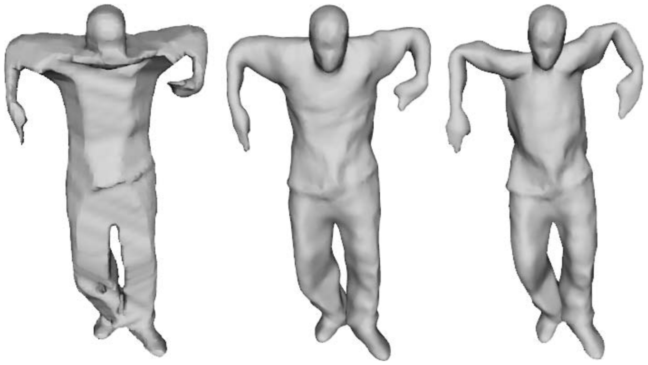
$$\underset{\mathbf{V}}{\arg\min} \left(\|\mathbf{\Delta V} - w_L \mathbf{L}'\|^2 + w_C \|\mathbf{C}_{\text{SIL}}\mathbf{V} - \mathbf{P}_{\text{SIL}}\|^2 \right), \tag{2}$$

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 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 6 (top) illustrates both of these cases. To prevent the surface from folding in on itself (Figure 6, bottom), we distort the distance to the shape by preferring



Visual hull Our reconstruction LBS

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

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 $5\mathrm{cm}$ (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. Figure 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 sampling the contours causes the estimated shapes in neighboring frames to exhibit "texture sliding" [Anuar and Guskov 2004] 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, we use $\sigma = 0.07\mathrm{m}$. 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 filtering between shape estimation iterations greatly reduces them.

4 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 8 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.

4.1 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 [2007]. 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. [2005].

Sequence	Total Frames	Fixed Frames	Total Time (minutes)	Avg. Seconds per Frame
Flashkick	250	4	57	13.7
Headstand	250	7	61	14.6
Kickup	220	1	53	14.5
Lock	250	13	64	15.4
Walkpose	66	0	16	14.5
Handstand	175	2	40	13.7
Bouncing	175	4	40	13.7
Crane	175	0	32	11.0
Jumping	150	2	32	12.8
Swing	150	5	33	13.2
Samba	175	0	36	12.3

Table 1: *Number of frames requiring user intervention, along with total and per frame processing times (including user interaction) for our test sequences. We captured the bottom six sequences, while the other five were provided by Starck and Hilton.*

The template mesh may be obtained by various means, such as 3D scanning [Levoy et al. 2000], static multi-view stereo of a single frame [Seitz et al. 2006], or manual modeling (as in [Park and Hodgins 2006]). For the Starck and Hilton data set, we use a good frame obtained by their multi-view stereo method [2007] as the template. For our data sets we use a 3D scan of the performer (see Figure 8) 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 3, left). We attached each mesh to its skeleton with linear blend skinning using weights generated with Pinocchio [Baran and Popović 2007].

4.2 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.

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 1 shows the number of frames that needed user constraints in our test sequences.

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.

4.3 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 9). A geometry edit to the mesh can be propagated through all or part of the motion using deformation transfer (see Figure 10) [Sumner and Popović 2004] or Kircher and Garland's method [2006]. It is also possible to train a better skinning model than LBS such as [Wang et al. 2007] and use it with new skeletal data or to generate new poses (Figure 1, bottom left), using deformation transfer to apply details that the skinning model does not capture.

5 Conclusion

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 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.

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.

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.

Because future research can greatly benefit from a large volume of data, we have built a studio that will enable us to capture more performances, including multiple people interacting, animals, and props. We hope that we can support further improvement of mesh capture and processing techniques by acquiring a large data set and sharing it with other researchers.

Acknowledgments

First of all, we wish to express our gratitude to Jonathan Starck and Adrian Hilton for making this project possible by sharing their painstakingly collected data sets. Their actions have inspired us to do the same with our data sets. Furthermore, Jonathan's immediate response to our last-minute data request helped us demonstrate that our method generalizes beyond a single subject. We also thank the members of the Ergonomics Team at the US Army Natick Soldier Research, Development and Engineering Center for creating the static templates by scanning some of our performers. Robert Wang processed our data with his rotational regression algorithm and deformation transfer. We thank Peter Sand for helping us set up a multi-view studio to acquire and compute clean silhouette data. We are grateful to Tom Buehler for editing our video and to Emily Whiting for narrating it. Thanks to Tilke Judd for performing the dance sequences. We also thank other members of the MIT Graphics Group for providing valuable feedback on the paper. This work was supported by the Singapore-MIT Gambit Game Lab, the National Science Foundation (CCF-0541227 and a Graduate Research Fellowship), Adobe Systems, Pixar Animation Studios and software donations from Autodesk and Adobe.

References

ALEXA, M. 2003. Differential coordinates for local mesh morphing and deformation. *The Visual Computer 19*, 2, 105–114.

ALLEN, B., CURLESS, B., AND POPOVIĆ, Z. 2002. Articulated body deformation from range scan data. *ACM Transactions on Graphics 21*, 3 (July), 612–619.

ANGUELOV, D., SRINIVASAN, P., KOLLER, D., THRUN, S., RODGERS, J., AND DAVIS, J. 2005. Scape: shape completion and animation of people. *ACM Transactions on Graphics 24*, 3 (Aug.), 408–416.

ANUAR, N., AND GUSKOV, I. 2004. Extracting animated meshes with adaptive motion estimation. In *Workshop on Vision, Modeling, and Visualization*, 63–71.

BALAN, A. O., SIGAL, L., BLACK, M. J., DAVIS, J. E., AND HAUSSECKER, H. W. 2007. Detailed human shape and pose from images. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*.

BARAN, I., AND POPOVIĆ, J. 2007. Automatic rigging and animation of 3d characters. *ACM Transactions on Graphics 26*, 3 (July), 72:1–72:8.

CARRANZA, J., THEOBALT, C., MAGNOR, M. A., AND SEIDEL, H.-P. 2003. Free-viewpoint video of human actors. *ACM Transactions on Graphics 22*, 3 (July), 569–577.

CHEUNG, K. M., BAKER, S., AND KANADE, T. 2005. Shape-from-silhouette across time part II: Applications to human modeling and markerless motion tracking. *International Journal of Computer Vision 63*, 3 (July), 225–245.

CORAZZA, S., MÜNDERMANN, L., CHAUDHARI, A., DEMATTIO, T., COBELLI, C., AND ANDRIACCHI, T. P. 2006. A markerless motion capture system to study musculoskeletal biomechanics: Visual hull and simulated annealing approach. *Annals Biomed. Eng. 34*, 6 (July), 1019–1029.

DANIELSSON, P.-E. 1980. Euclidean distance mapping. *Computer Graphics and Image Processing 14*, 227–248.

DE AGUIAR, E., THEOBALT, C., MAGNOR, M., THEISEL, H., AND SEIDEL, H.-P. 2004. M3: marker-free model reconstruction and motion tracking from 3d voxel data. In *12th Pacific Conference on Computer Graphics and Applications*, 101–110.

DE AGUIAR, E., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2005. Reconstructing human shape and motion from multi-view video. In *2nd European Conference on Visual Media Production (CVMP)*, 42–49.

DE AGUIAR, E., THEOBALT, C., STOLL, C., AND SEIDEL, H.-P. 2007. Marker-less deformable mesh tracking for human shape and motion capture. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*.

DE AGUIAR, E., THEOBALT, C., STOLL, C., AND SEIDEL, H.-P. 2007. Rapid animation of laser-scanned humans. In *IEEE Virtual Reality 2007*, IEEE, Charlotte, USA, 223–226.

EROL, A., BEBIS, G., BOYLE, R. D., AND NICOLESCU, M. 2005. Visual hull construction using adaptive sampling. In *Proc. of IEEE Workshop on Application of Computer Vision*, 234–241.

ESTEBAN, C. H., AND SCHMITT, F. 2004. Silhouette and stereo fusion for 3D object modeling. 367–392.

FURUKAWA, Y., AND PONCE, J. 2006. Carved visual hulls for image-based modeling. In *Proc. European Conf. Computer Vision*, I: 564–577.

GILL, P. E., MURRAY, W., AND SAUNDERS, M. A. 2002. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization 12*, 4, 979–1006.

GOLDLÜCKE, B., IHRKE, I., LINZ, C., AND MAGNOR, M. 2007. Weighted minimal hypersurface reconstruction. *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI) 29*, 7 (July), 1194–1208.

HORNUNG, A., AND KOBBELT, L. 2006. Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding. In *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 503–510.

KIRCHER, S., AND GARLAND, M. 2006. Editing arbitrarily deforming surface animations. *ACM Transactions on Graphics 25*, 3 (July), 1098–1107.

LEVOY, M., PULLI, K., CURLESS, B., RUSINKIEWICZ, S., KOLLER, D., PEREIRA, L., GINZTON, M., ANDERSON, S., DAVIS, J., GINSBERG, J., SHADE, J., AND FULK, D. 2000. The digital michelangelo project: 3d scanning of large statues. In *Proceedings of ACM SIGGRAPH 2000*, Computer Graphics Proceedings, Annual Conference Series, 131–144.

LIPMAN, Y., SORKINE, O., COHEN-OR, D., LEVIN, D., RÖSSL, C., AND SEIDEL, H.-P. 2004. Differential coordinates for interactive mesh editing. In *Proceedings of Shape Modeling International*, IEEE Computer Society Press, 181–190.

MÉNIER, C., BOYER, E., AND RAFFIN, B. 2006. 3D skeleton-based body pose recovery. In *Proc. Int. Symp. on 3D Data Processing, Visualization and Transmission*, 389–396.

MEYER, M., DESBRUN, M., SCHRÖDER, P., AND BARR, A. H. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and Mathematics III*. Springer-Verlag, Heidelberg, 35–57.

PARK, S. I., AND HODGINS, J. K. 2006. Capturing and animating skin deformation in human motion. *ACM Transactions on Graphics 25*, 3 (July), 881–889.

PLÄNKERS, R., AND FUA, P. 2001. Articulated soft objects for video-based body modeling. In *ICCV*, 394–401.

RANDER, P. W., NARAYANAN, P., AND KANADE, T. 1997. Virtualized reality: Constructing time-varying virtual worlds from real world events. In *IEEE Visualization '97*, 277–284.

SAND, P., MCMILLAN, L., AND POPOVIĆ, J. 2003. Continuous capture of skin deformation. *ACM Transactions on Graphics 22*, 3 (July), 578–586.

SCHOLZ, V., STICH, T., KECKEISEN, M., WACKER, M., AND MAGNOR, M. 2005. Garment motion capture using color-coded patterns. *Computer Graphics Forum (Proc. Eurographics EG'05) 24*, 3 (Aug.), 439–448.

SEITZ, S. M., CURLESS, B., DIEBEL, J., SCHARSTEIN, D., AND SZELISKI, R. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, 519–528.

STARCK, J., AND HILTON, A. 2007. Surface capture for performance based animation. *IEEE Computer Graphics and Applications 27(3)*, 21–31.

SUMNER, R. W., AND POPOVIĆ, J. 2004. Deformation transfer for triangle meshes. *ACM Transactions on Graphics 23*, 3 (Aug.), 399–405.

SVOBODA, T., MARTINEC, D., AND PAJDLA, T. 2005. A convenient multi-camera self-calibration for virtual environments. *PRESENCE: Teleoperators and Virtual Environments 14*, 4 (August), 407–422.

THEOBALT, C., MAGNOR, M., SCHULER, P., AND SEIDEL, H.-P. 2002. Combining 2d feature tracking and volume reconstruction for online video-based human motion capture. In *10th Pacific Conference on Computer Graphics and Applications*, 96–103.

THEOBALT, C., AHMED, N., LENSCH, H., MAGNOR, M., AND SEIDEL, H.-P. 2007. Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics 13*, 4 (July/Aug.), 663–674.

WANG, R. Y., PULLI, K., AND POPOVIĆ, J. 2007. Real-time enveloping with rotational regression. *ACM Transactions on Graphics 26*, 3 (July), 73:1–73:9.

WHITE, R., CRANE, K., AND FORSYTH, D. A. 2007. Capturing and animating occluded cloth. *ACM Transactions on Graphics 26*, 3 (July), 34:1–34:8.

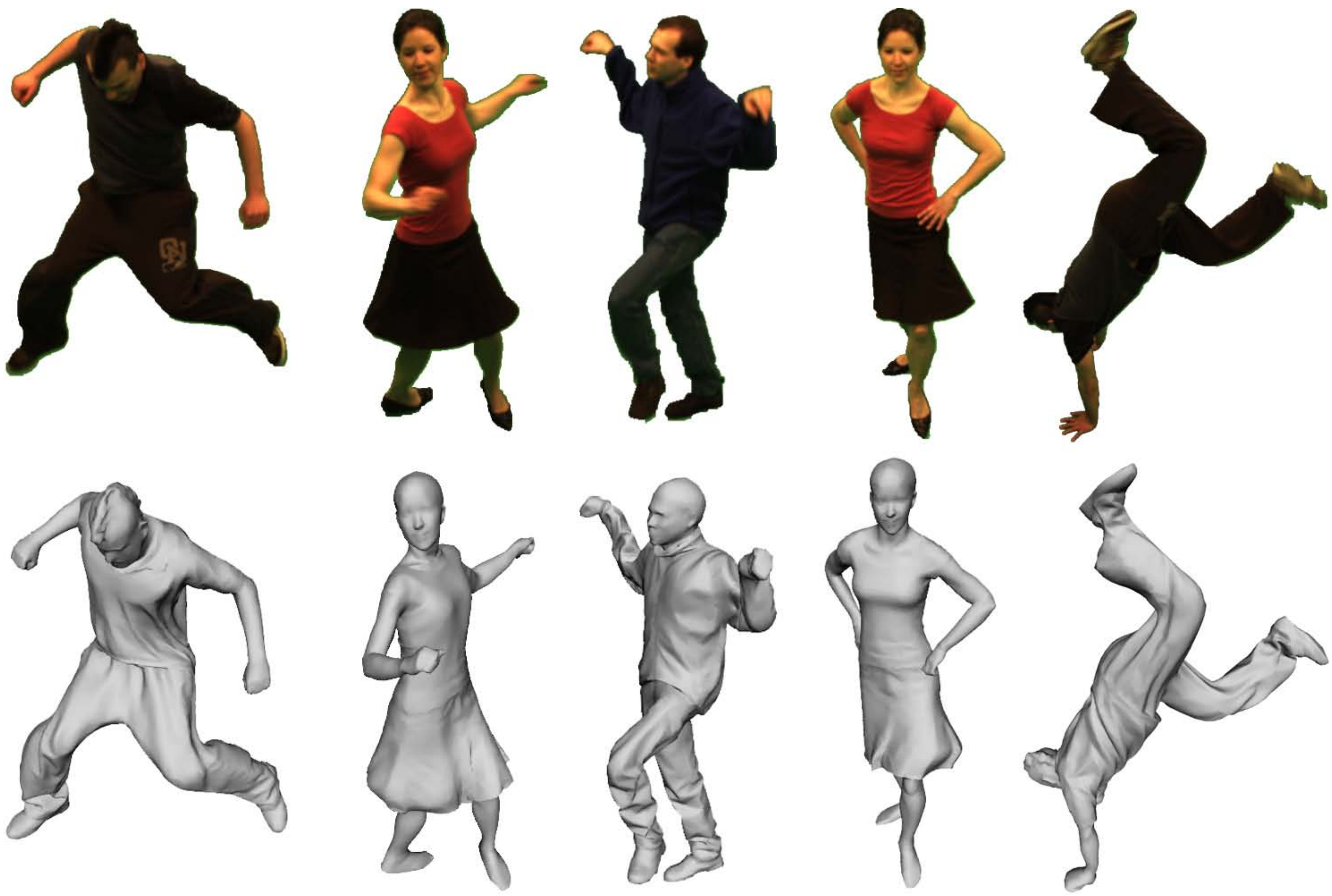


Figure 8: *Background-subtracted video frames (top) and the corresponding recovered shapes (bottom) for three subjects in five sequences.*

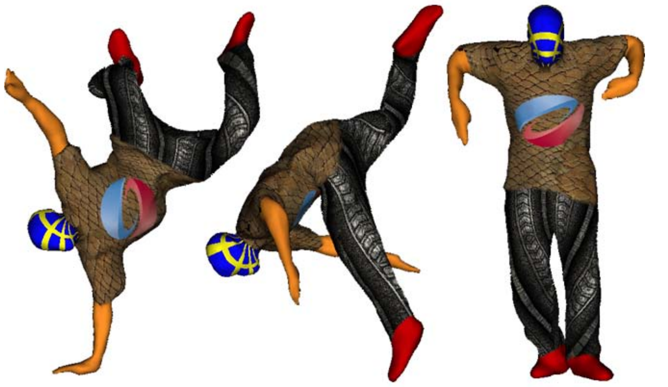


Figure 9: *Frame-to-frame correspondence allows a static texture to be easily applied to the entire motion.*

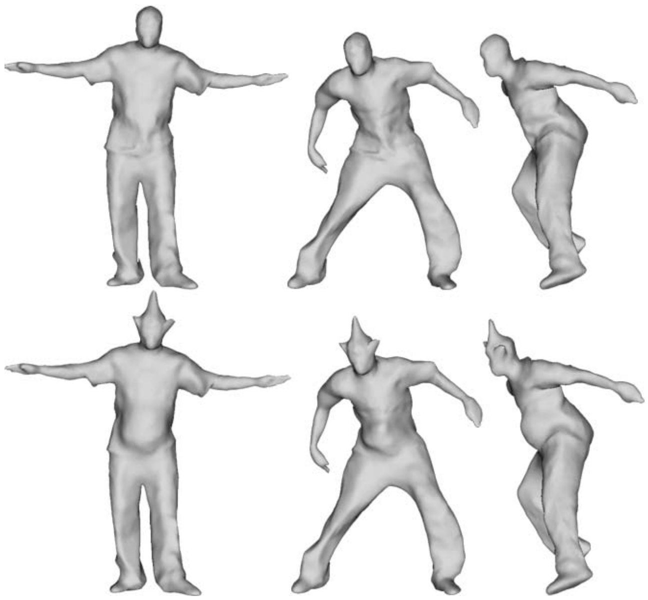


Figure 10: *Top row: the template mesh and two frames of animation. Bottom row: the template mesh geometry is edited and the change is propagated to the two frames using deformation transfer.*