Markerless Human Motion Transfer

German K.M. Cheung Simon Baker Jessica Hodgins Takeo Kanade
german+@cs.cmu.edu simonb@cs.cmu.edu jkh@cs.cmu.edu tk@cs.cmu.edu
Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213

Abstract

In this paper we develop a computer vision-based system to transfer human motion from one subject to another. Our system uses a network of eight calibrated and synchronized cameras. We first build detailed kinematic models of the subjects based on our algorithms for extracting shape from silhouette across time [6]. These models are then used to capture the motion (joint angles) of the subjects in new video sequences. Finally we describe an image-based rendering algorithm to render the captured motion applied to the articulated model of another person. Our rendering algorithm uses an ensemble of spatially and temporally distributed images to generate photo-realistic video of the transferred motion. We demonstrate the performance of the system by rendering throwing and kungfu motions on subjects who did not perform them.

1 Introduction

Generating animations of human characters with realistic motion and appearance is one of the most difficult problems in computer graphics but these characters are required for such applications as game development, virtual reality experiences and movie production. For example, the ability to create realistic videos of people performing new motions would allow more realistic video games. Such technology might also allow a movie director to change a scene in post-production without gathering the cast to re-shoot the scene.

In current approaches for transferring motion, a laser scanner is used to model the shape of one person and then an optical motion capture system is used to capture the motion of another person. The model and motion are then combined to create a rendering of the first person performing the action of the second person. In this paper, we suggest an alternative to this approach by describing a complete end-to-end markerless system to transfer articulated motion from one person to another and render it photo-realistically. Our system is based on computer vision techniques which use a set of simple cameras and *no markers* of any kind.

Figure 1 illustrates the overall structure of our markerless motion transfer system. Our system uses eight synchronized, calibrated and color balanced cameras evenly spaced around the perimeter of a capture area. The motion transfer process consists of three steps: (a) human kinematic modeling, (b) markerless motion capture and (c) image-based rendering of one subject moving with the motion performed by a different subject. The first two steps of the system are based on our recently published work in vision-based kinematic modeling and motion capture [5] which we briefly review in Sections 3 and 4. In Section 5, we describe in detail the image-based rendering algorithm that we use to generate photo-realistic videos of the articulated models and motion we acquired in the first two steps. We demonstrate the power of our approach by transferring a throwing motion and a kungfu motion from one subject to another (Figure 1).

2 Related Work

In the last decade, considerable research has been devoted to capturing motion or recognizing posture without using markers (see [14] for a survey). Among the markerless systems that use model-based approaches, the work by Sidenbladh et al. [15], that by Delamarre and Faugeras [8], that by Carranza et al. [3] and that by Mikic et al. [13] are most closely related to our tracking algorithm. Our markerless motion capture technique differs from their approaches in two ways: we use detailed and person-specific models to perform the tracking and we use both silhouette and color information as tracking cues.

The approach most closely related to our silhouette-based human kinematic model acquisition algorithm is Kakadiaris and Metaxas [11]. They used deformable templates to segment the 2D body parts in silhouette sequences from three orthogonal view-points and then combined the information into a 3D shape using Shape-From-Silhouette. Our approach of estimating the joint locations individually instead of all at once is partly inspired by their system.

Carranza et al. [3] rendered tracked human motion using a view-dependent texture mapping algorithm that is similar to our pixel rendering algorithm. However, they replay the captured motion on the same person rather than transfer the motion to another person. Moreover their texture mapping is a frame-based approach while our algorithm utilizes both spatial and temporal textures. Vedula et al. developed an image-based spatial and temporal view interpolation algorithm for non-rigid dynamic events [16, 17]. There are two major differences between our algorithm and the one in [16]. In our rendering algorithm, we assume the human model consists of articulated rigid body parts each with rigid motions, but Vedula et al. assumed the human and the motion were non-rigid. Moreover, Vedula et al. "interpolated" the motion spatially and temporally while we "extrapolate" the human body model with new motions. We adopt some of the efficiency optimization techniques by Vedula et al. [16] and the original view-dependent texture mapping idea by Debevec et al. [7].

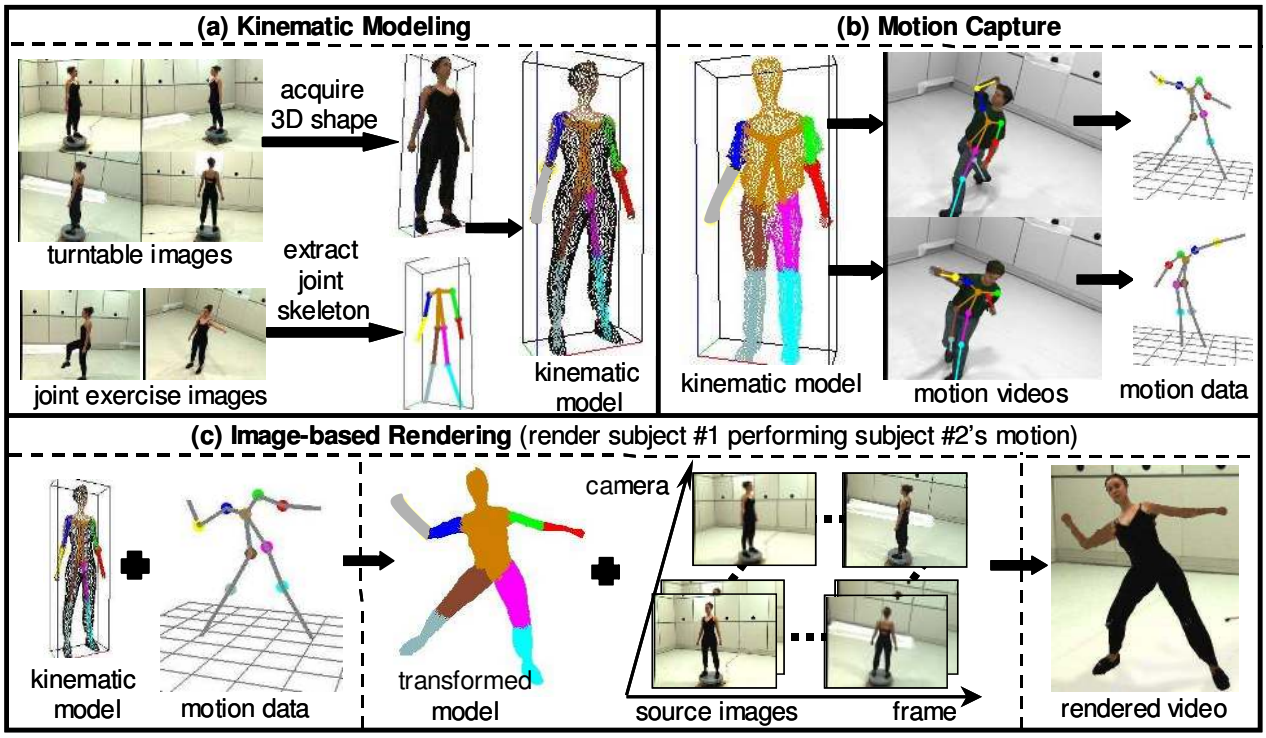

Figure 1. Our markerless motion transfer system consists of three parts: (a) kinematic modeling: extracting segmented 3D shape and joint locations, (b) markerless motion capture: articulated tracking of joint angles and (c) image-based rendering of the motion applied to the other subject.

3 Kinematic Modeling

The kinematic modeling process consists of two parts: shape acquisition and joint skeleton estimations (Figure 1(a)). To acquire the shape of the subject, we capture video of the subject standing on an uncalibrated turntable for 30 seconds. Using an algorithm called Shape-From-Silhouette Across Time [6], we recover the motion of the turntable automatically. The recovered motion is used to align the video images across both space and time and build a detailed voxel-based shape model of the subject using the traditional Shape-From-Silhouette algorithm [1].

To estimate the joint skeleton of the subject, we ask the subject to exercise each of his or her body joints one at a time. By extending the Shape-From-Silhouette Across Time algorithm to articulated objects [5], we recover the joint position and segmentation of each joint of the subject from the captured video. The joint information is then merged with the shape model obtained from the footage captured with the subject on the turntable. The resulting kinematic model consists of nine body parts: torso, right/left upper/lower arms, right/left upper/lower legs connected by eight joints. The head is modeled as rigidly connected to the torso as are the hands to the lower arms and the feet to the lower legs. The shoulder and hip joints have three degrees of freedom (DOF) each while there is one DOF for each of the elbow and knee joints. Including translation and rotation of the torso base, there are a total of 22 DOF in the model. The kinematic models of subjects #1 and #2 are shown in Figures 1(a) and (b) respectively. Further details of the kinematic modeling process can be found in [4].

4 Markerless Motion Capture

Once the kinematic models have been acquired, they are used to recover the motion data (joint angles) of each subject. We incorporate joint constraints into the Shape-From-Silhouette Across Time algorithm for articulated objects [5] and align the kinematic model with respect to both the silhouette and color video images. The alignment is done hierarchically: first fit the torso base and then fit each limb independently. No markers are required. Figure 1(b) shows two frames of the tracked motion of subject #2 miming a throw motion. Details of our motion tracking algorithms can be found in [4, 5].

5 Image-Based Rendering

The last step of our markerless motion transfer system is to render one subject performing the motion of the other subject. While we can apply the captured motion of one subject to the textured voxel model of the other subject and render the model directly, the resulting video is not photo-realistic due to the lack of detailed texture.

Instead we develop an image-based algorithm for rendering articulated objects using an ensemble of spatially and temporally distributed images. The data available in this

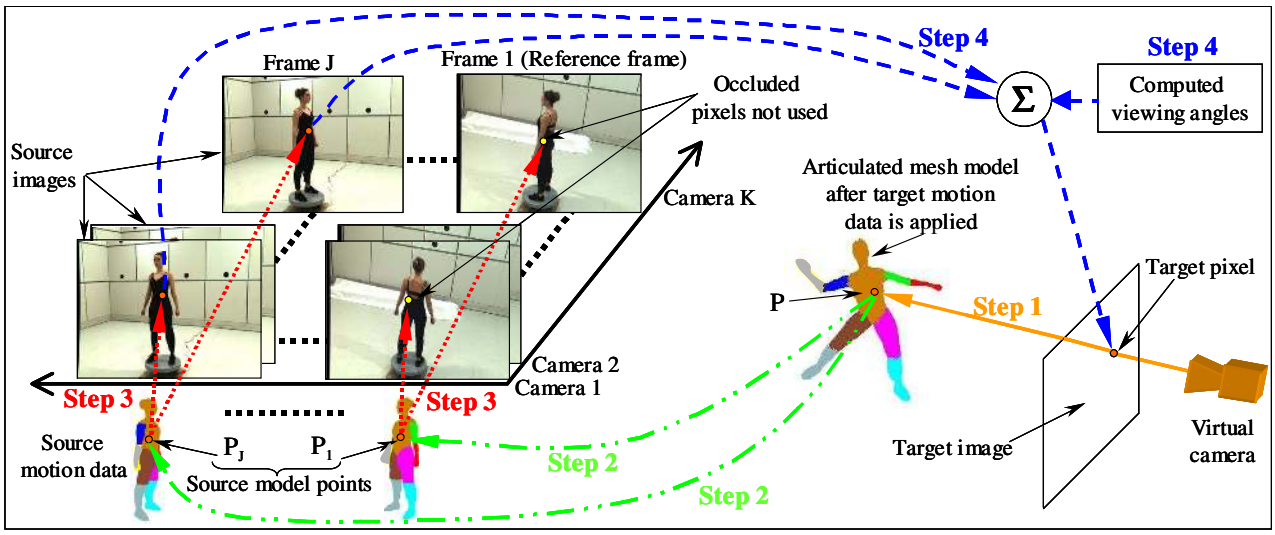

Figure 2. The pixel rendering process of our Image-Based Rendering Algorithm for articulated objects.

collection of images increases the photo-realism of the resulting video. We explain our algorithm using the example of rendering subject #1 performing the throwing motion of subject #2. We assume the following data is available: (1) the articulated kinematic model of subject #1 which we built in Section 3, (2) the throwing motion (hereafter referred to as the target motion) we captured in Section 4, and (3) a set of JK (from K cameras each with J frames) source images of subject #1 performing some kind of motion with captured motion data (see Figure 1(c)). The first frame of this set of source images is called the reference frame.

As a pre-processing step, the articulated voxel model is converted into a smooth *mesh* model using the Marching Cubes Algorithm [12] and a mesh smoothing technique [10]. The triangles of the mesh-based model can be skinned to eliminate discontinuities at the joints (see Figure 5).

After the mesh model has been transformed with the target motion data, it can be rendered from any virtual camera viewpoint to produce the target image. The pixels in the target image are taken from a set of source images of the subject on the turntable as shown in Figure 2. The color of a target pixel is determined using the following four steps:

Step 1: Intersect rays with the articulated model.

A viewing ray is cast from the (virtual) camera center through the target pixel and intersected with the transformed mesh model of the person in the 3D space. If the ray does not intersect the model, the target pixel is assigned the appropriate background color. Otherwise, the body part Z where the ray first intersects the mesh model and the point of intersection, P, are found (Step 1 in Figure 2).

The most straightforward way to find the intersection point P is to intersect the viewing ray with all the faces of the articulated mesh model and choose the intersection that is closest to the camera. However, in practice this approach is slow as the model contains thousands of faces. Instead, we employ an idea from hardware acceleration called the item buffer [16, 18]. Each mesh face is assigned a distinct RGB color as its identity (ID) number (with 24 bit colors, up to 16M faces can be assigned distinct ID numbers). After the model is transformed by the target motion data, the mesh faces are rendered from the virtual camera viewpoint using their ID colors. The triangular face that is intersected by the ray through a target pixel can easily be found by reading the ID color of the same pixel in the rendered ID picture. Once the correct mesh face is found, it is intersected with the viewing ray to locate the target model point P.

Step 2: Compute the source model points P_j.

The position of P in the source reference frame, P_1, can be computed from the inverse target motion transformation equations of part Z. Once P_1 is known, its positions at all the other source frames, represented by $\{P_j;\ j = 2, \cdots, J\}$ are calculated using the given source motion data (Step 2 in Figure 2). We refer to P_j as the source model points of the target pixel.

Because skinning weights are used to smooth the motion of the model near the joints, some of the mesh faces are stretched after the target motion data is applied (Figure 3). We compensate for this stretching when calculating P_1 (the source model point at the reference frame) from P. Let V^1, V^2, V^3 be the vertices of the intersecting mesh face after applying the target motion data, and V_1^1, V_1^2, V_1^3 be the corresponding vertices of the same face at the reference frame. Note that V_1^1, V_1^2, V_1^3 are known and V^1, V^2, V^3 can be calculated using the motion weights. Now because P lies on and inside the triangular patch formed by V^1, V^2, V^3, we have

$$P = a^1 V^1 + a^2 V^2 + a^3 V^3 \,, \tag{1}$$

where a^1, a^2, a^3 are constants between 0 and 1. Now we apply the same constants to the corresponding vertices V_1^1, V_1^2, V_2^3 in the reference frame as

$$P_1 = a^1 V_1^1 + a^2 V_1^2 + a^3 V_1^3 \,. \tag{2}$$

By substituting Equation (1) into Equation (2), P_1 is calculated by

$$P_1 = \begin{bmatrix} V_1^1 & V_1^2 & V_1^3 \end{bmatrix} \begin{bmatrix} V^1 & V^2 & V^3 \end{bmatrix}^{-1} P \,. \tag{3}$$

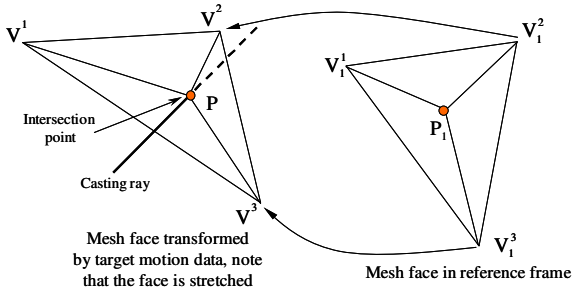

Figure 3. Step 2 of the pixel rendering process: every mesh face is stretched because of the different skinning weights of the vertices. This stretching has to be compensated for when calculating P_1 from P.

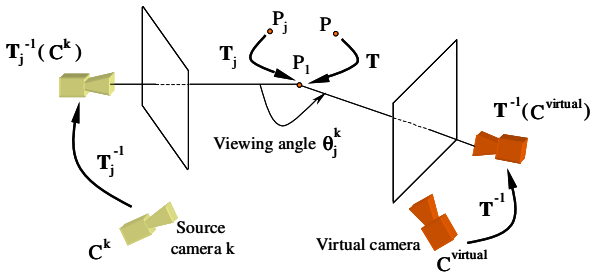

Figure 4. Step 4 of the pixel rendering process: computing the viewing angle between the virtual camera, the k^{th} source camera at the j^{th} frame and the target model point P.

Step 3: Project P_j onto source images to get pixel colors.

After computing the source model points P_j, they are projected into the source color images to get a total of JK source pixels that *could* be used as the color of the target pixel (Step 3 in Figure 2). However, among these JK pixels, only those which come from frames where P_j is *visible* in camera k are valid. We test the visibility of P_j against the k^{th} camera at frame j using a 2D z-buffer approach as discussed in [9, 16]. We first generate a 2D depth image of the mesh model at frame j of camera k using the z-buffer graphic hardware. The visibility of any point is then determined by first projecting the point into the depth image to get a depth value and then comparing this depth value with the distance of the point from the camera (details of the approach can be found in [9, 16]). This 2D z-buffer approach is faster than directly testing the visibility against the 3D shape model because each depth test only requires one 3D to 2D projection and one scalar (depth) comparison.

Step 4: Average the visible source pixel colors according to the viewing angles.

The final color of the target pixel is the weighted average of the *valid* source pixel colors using the viewing angle between the virtual camera and the source camera as the weight (Step 4 in Figure 2). The viewing angles are calculated in the reference frame. Figure 4 shows the viewing angle for the source pixel color from the k^{th} camera at the j^{th} frame. Both the virtual camera $C^{virtual}$ and the k^{th} source camera C^k are transformed to the reference frame. Let the 6D rigid transformation between P (the target frame) and P_1 (the reference frame) be $\boldsymbol{T}$ and that between P_j (the j^{th} source frame) and P_1 be $\boldsymbol{T}_j$, i.e. $P_1 = \boldsymbol{T}(P)$ and $P_1 = \boldsymbol{T}_j(P_j)$ (Note that both $\boldsymbol{T}_j$ and $\boldsymbol{T}$ can be found from the target and the source motion data). The position of $C^{virtual}$ and C^k in the reference frame are then given by $\boldsymbol{T}^{-1}(C^{virtual})$ and $\boldsymbol{T}_j^{-1}(C^k)$ respectively. The viewing angle θ_j^k is then calculated using the following equation:

$$\cos\theta_j^k = \frac{(\boldsymbol{T}_j^{-1}(C^k) - P_1) \cdot (\boldsymbol{T}^{-1}(C^{virtual}) - P_1)}{\|\boldsymbol{T}_j^{-1}(C^k) - P_1\| \|(\boldsymbol{T}^{-1}(C^{virtual}) - P_1\|} . \quad (4)$$

Because the reliability of the source pixel is inversely related to the viewing angle we compute weights (see [7, 16]) for each color pixel as $\frac{1}{1-\cos\theta_j^k}$. The *valid* source pixel colors are then sorted according to their weights and the pixel colors corresponding to the n largest weights are averaged to compute the target pixel color. The value of n is chosen experimentally.

Depending on the source images used, there may be some target pixels whose pixel color cannot be determined because the 3D source model points P_j of a target pixel were not visible in any camera while the subject was on the turntable. To fill in the color of a missing target pixel (or hole), the average color of its neighboring pixels is used. In the averaging process, only colors of those neighbors from the same body part as the missing pixel are used to prevent color diffusion between body parts.

Shadows and background are also added to the rendered image to increase photo-realism. In particular, we use the planar surface shadowing technique in [2].

6 Results

We compare the quality of the rendered images using three different methods: render the textured voxel model directly, render the textured mesh model directly and render using our algorithm. We also explore the effect of the number of source pixels, n, on the quality of the rendered target pictures. We use subject #1 and 240 source images (8 cameras each with 30 frames) from the sequence where subject #1 stood on the turn-table.

The top and middle rows of Figure 5 shows subject #1 performing a motion (taken from a motion database) rendered from two different camera viewpoints. The bottom row of the figure displays the face region from the top row at a higher resolution for better comparison. Figures 5(a) and (b) show the results of directly rendering the colored voxel and texture-mapped mesh models. Figures 5(c), (d) and (e) show rendering results using the algorithm described in this paper with the target pixel color computed by averaging 1, 5 and 9 source pixels.

The images obtained using our rendering algorithm are sharper than those obtained from the direct rendering of

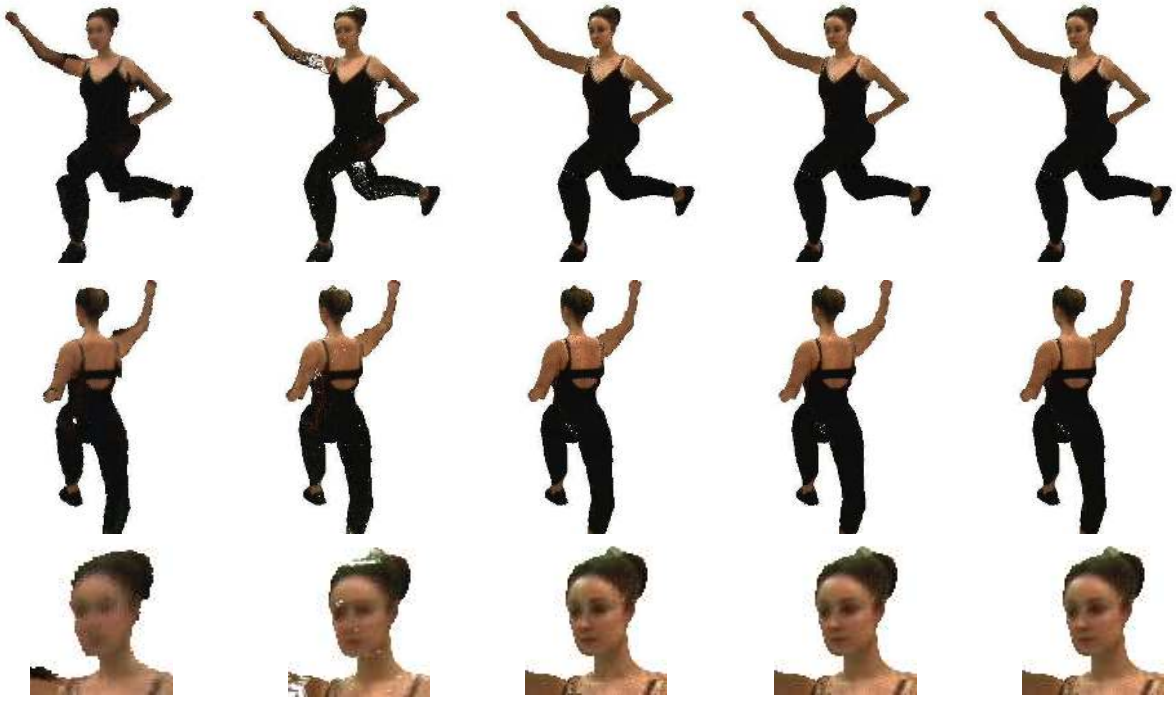

Figure 5. Images obtained by (a) direct rendering of the colored voxel model, (b) direct rendering of the texture-mapped mesh model, (c) using our algorithm with each target pixel color computed by averaging one source pixel, (d) five source pixels and (e) nine source pixels. The top and the middle rows show subject #1 performing a motion (taken from a motion database) rendered from two different viewpoints. The bottom row displays the face region from the top row at a higher resolution for better comparison.

either the voxel or mesh models, especially in highly textured areas such as the face. The voxel model also suffers from discontinuities at the joints (see the knee joints in Figure 5(a)). The white patches in Figure 5(b) represents the part of the mesh model where no texture can be obtained from the source sequence (hole filling was not used here). The last three columns of Figure 5 show that the images rendered using different numbers of source pixels are visually very similar to each other. If the cameras are color-balanced, our rendering algorithm is not particularly sensitive to the number of source pixels used.

Figure 6(a) shows subject #1 performing a throwing motion with the corresponding frames from the original throwing sequence of subject #2. For comparison, the camera viewpoint of the rendered sequence is that of the first camera of the source sequence. Our system can also render the subject from novel and moving viewpoints. An example is included in the supplementary video clip[1]. Figure 6(b) shows another example of transferring a kungfu motion from subject #3 to subject #2. The complete rendered sequence in Figure 6(a) is included in the supplementary video clip.

On a 750MHz computer, Step 1 of our rendering algorithm takes approximately 2 seconds per image (640 x 480 pixels), Step 2 takes about 9 seconds and Steps 3 and 4 take about 4 minutes each (the timing of Step 4 includes the time required to read the source images from memory).

[1]The supplementary video clip can be found at http://www.cs.cmu.edu/~german/Research/MotionTransfer/MT.mpg

7 Discussion

Three factors contribute to the visual artifacts in the images rendered using our algorithm. First, the captured motion was not re-targeted from subject #2 to subject #1, causing the feet of subject #1 to slide in the rendered video. Second, although the cameras are color balanced, self shadows in the source images produce unevenly lit pixels in our final rendered video. This artifact is caused by the ceiling lights in our setup and could be reduced by careful placements of lighting. Third, because the position of the 3D source model point in each frame is calculated using the motion data of the source image sequence, any tracking errors in the source motion data cause the algorithm to pick the wrong source pixel color. To reduce artifacts caused by this problem, it is important to make sure the motion of the person in the source sequence is tracked correctly.

One limitation of our system is that both the motion tracking and rendering algorithms are appearance based. To apply our algorithms to the same subject in a different set of clothing, we have to update the texture of the kinematic model and capture new video sequences of the subject in the new clothing as source images for rendering.

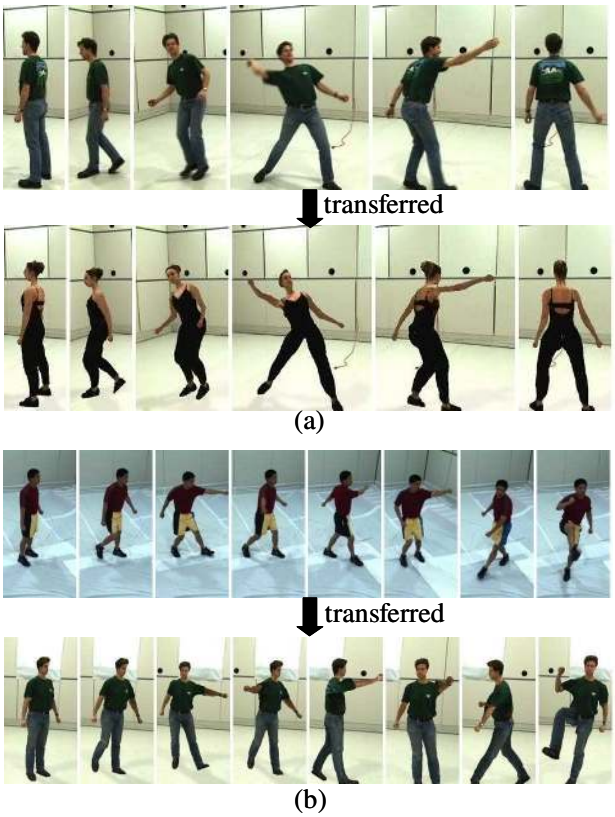

Figure 6. Selected frames showing (a) a throwing motion transferred from subject #2 to subject #1, (b) a kungfu motion transferred from subject #3 to subject #2.

Moreover, because we model each separate body part as rigid, our system is not able to capture and render subtle surface deformation effects caused by movement of the muscle and clothing. One of our future directions of research is to incorporate deformable models to capture and animate muscle movement and skin deformation.

In this paper, we have shown that using vision-based algorithms, we can create accurate human kinematic models, track motion in video sequences and generate realistic video of motion transferred from one person to another person without using markers of any kind. Our system provides an inexpensive and simple alternative to marker-based systems for human motion transfer.

References

[1] N. Ahuja and J. Veenstra. Generating octrees from object silhouettes in orthographic views. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 11(2):137–149, February 1989.

[2] J. Blinn. Me and my (fake) shadow. *IEEE Computer Graphics and Applications*, 8(1):82–86, January 1988.

[3] J. Carranza, C. Theobalt, M. Magnor, and H. Seidel. Free-viewpoint video of human actors. *ACM Transactions on Graphics*, 22(2):569–577, 2003.

[4] G. Cheung. *Visual Hull Construction, Alignment and Refinement for Human Kinematic Modeling, Motion Tracking and Rendering*. PhD thesis, Carnegie Mellon University, 2003.

[5] G. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette for articulated objects and its use for human body kinematics estimation and motion capture. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, MI, June 2003.

[6] G. Cheung, S. Baker, and T. Kanade. Visual hull alignment and refinement across time: A 3D reconstruction algorithm combining shape-frame-silhouette with stereo. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Madison, MI, June 2003.

[7] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics Annual Conference Series (SIGGRAPH'96)*, pages 11–20, 1996.

[8] Q. Delamarre and O. Faugeras. 3D articulated models and multi-view tracking with silhouettes. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[9] J. Foley, A. Van Dam, S. Feiner, and J. Hughes. *Computer Graphics: Principle and Practice*. Addison Wesley, second edition, 1992.

[10] A. Johnson and M. Hebert. Control of polygonal mesh resolution for 3D computer vision. Technical Report CMU-RI-TR-96-20, Carnegie Mellon University, Pittsburgh, PA, April 1997.

[11] I. Kakadiaris and D. Metaxas. 3D human body model acquisition from multiple views. In *Proceedings of International Conference on Computer Vision (ICCV'95)*, pages 618–623, Cambridge MA, June 1995.

[12] W. Lorensen and H. Cline. A high resolution 3D surface reconstruction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[13] I. Mikic, M. Trivedi, E. Hunter, and P. Cosman. Human body model acquisition and tracking using voxel data. *International Journal on Computer Vision*, 53(3):199–223, July 2003.

[14] T. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding: CVIU*, 81(3):231–268, 2001.

[15] H. Sidenbladh, F. DeLaTorre, and M. Black. A framework for modeling the appearance of 3D articulated figures. In *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition (ICAFGR'00)*, March 2000.

[16] S. Vedula, S. Baker, and T. Kanade. Spatio-temporal view interpolation. In *Proceedings of the 13th Eurographics Workshop on Rendering*, June 2002.

[17] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade. Three-dimensional scene flow. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, Corfu, Greece, September 1999.

[18] H. Weghorst, G. Hooper, and D. Greenberg. Improved computational methods for ray tracking. *ACM Transactions on Graphics*, 3(1):52–69, 1984.