Video-based Characters – Creating New Human Performances from a Multi-view Video Database

Feng Xu[†] Yebin Liu[⋆] Carsten Stoll[⋆] James Tompkin[‡] Gaurav Bharaj[⋆]
Qionghai Dai[†] Hans-Peter Seidel[⋆] Jan Kautz[‡] Christian Theobalt[⋆]
[†]TNList, Tsinghua University, China [‡]University College London, UK [⋆]MPI Informatik, Germany

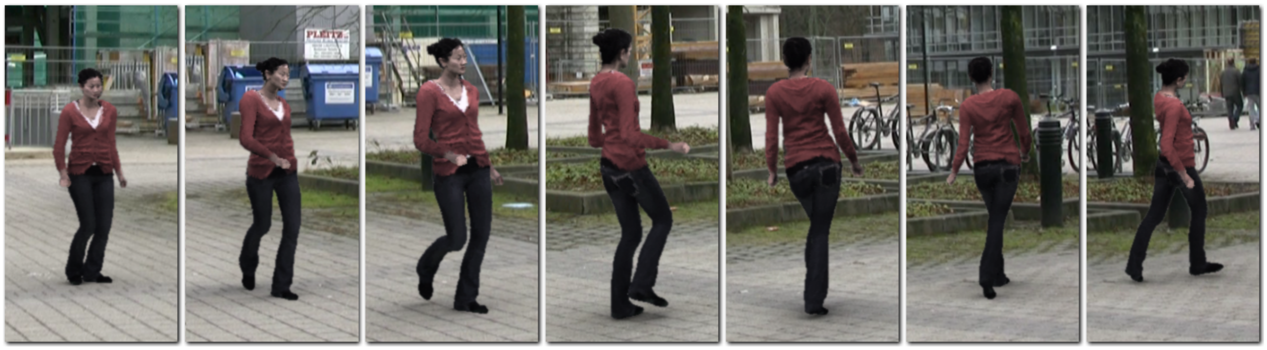
Figure 1: *An animation of an actor created with our method from a multi-view video database. The motion was designed by an animator and the camera was tracked from the background with a commercial camera tracker. In the composited scene of animation and background, the synthesized character and her spatio-temporal appearance look close to lifelike.*

Abstract

We present a method to synthesize plausible video sequences of humans according to user-defined body motions and viewpoints. We first capture a small database of multi-view video sequences of an actor performing various basic motions. This database needs to be captured only once and serves as the input to our synthesis algorithm. We then apply a marker-less model-based performance capture approach to the entire database to obtain pose and geometry of the actor in each database frame. To create novel video sequences of the actor from the database, a user animates a 3D human skeleton with novel motion and viewpoints. Our technique then synthesizes a realistic video sequence of the actor performing the specified motion based only on the initial database. The first key component of our approach is a new efficient retrieval strategy to find appropriate spatio-temporally coherent database frames from which to synthesize target video frames. The second key component is a warping-based texture synthesis approach that uses the retrieved most-similar database frames to synthesize spatio-temporally coherent target video frames. For instance, this enables us to easily create video sequences of actors performing dangerous stunts without them being placed in harm's way. We show through a variety of result videos and a user study that we can synthesize realistic videos of people, even if the target motions and camera views are different from the database content.

CR Categories: I.4.8 [Computer Graphics]: Scene Analysis—Time-varying imagery;

ACM Reference Format
Xu, F., Liu, Y., Stoll, C., Tompkin, J., Bharaj, G., Dai, Q., Seidel, H., Kautz, J., Theobalt, C. 2011.
Video-based Characters - Creating New Human Performances from a Multi-view Video Database.
ACM Trans. Graph. 30, 4, Article 32 (July 2011), 10 pages. DOI = 10.1145/1964921.1964927
http://doi.acm.org/10.1145/1964921.1964927.

1 Introduction

There is still a substantial quality gap between photo-realistic video sequences and fully animated human characters. In current video games, animation techniques are highly developed and intricate motions can be created. However, the realism of the rendered animation sequences still does not match a captured video. In contrast, acquired video sequences for movie productions are realistic because they are directly captured through high-quality cameras. However, all required motions and actions need to be performed by actors, and it is very difficult to make any kind of motion edits later on – even changing body appearance in videos with the same motion is already a challenge [Jain et al. 2010]. This means that actors need to repeat their performance many times until the desired motion/action is achieved to a sufficient quality. Consequently, generating photo-realistic sequences of human beings is highly desirable for both computer games and movie production. Video-texture methods first attempted to synthesize new video footage by recombining existing video clips [Schödl et al. 2000]. However, they do not allow modification of the camera viewpoint and they face a big challenge in resynthesizing videos showing plausible articulated body motion [Flagg et al. 2009]. To our knowledge, our proposed method is the first technique that enables the synthesis of photo-realistic videos of human characters performing user-defined motions, observed from user-defined viewpoints.

One of the main difficulties in synthesizing photo-realistic videos of animated characters lies in the creation of realistic textures. When people perform different kinds of motions, their appearance continually changes according to their motion. For example, highlights appear and disappear, folds and wrinkles form and move on clothes, and skin color varies with motion. As the appearance of human characters is affected by various physical conditions, simulating realistic texture is a difficult problem [Jimenez et al. 2010].

In our scheme, we develop an image-based method to overcome this difficulty. We build and search a multi-view multi-motion database for video frames with appropriate texture to synthesize frames of a novel target animation. We do not run any kind of simulation but rather perform retrieval and image-based warping to achieve realistic textures. Image-warping is guided by a detailed model of skeletal motion and dynamic shape that we capture for each multi-view sequence in the database with a state-of-the-art marker-free performance capture approach. Our re-animation approach also allows us to support novel viewpoints by jointly considering the pose and viewpoint. This guarantees sufficient view and pose information to synthesize target frames. Specifying a target animation is easy: In a standard animation tool, a new motion for the actor's skeleton is specified, applied to the body model, and rendered into the desired camera path. We then use a warping-based method to integrate information from multiple database frames to generate a rendering of the texture sequence of the target poses and views.

2 Related Work

We capitalize on recent performance capture approaches that reconstruct detailed dynamic shape and motion models of humans from multi-view video without using optical markers [de Aguiar et al. 2008; Vlasic et al. 2008; Bradley et al. 2008; Vlasic et al. 2009; Cagniart et al. 2010]. In particular, we use the joint skeleton and surface tracking approach of Gall et al. [2009] to automatically capture performance models of all multi-view sequences in the database without optical markers.

Several previous image-based approaches can render novel footage of humans from input video, but none can simultaneously alter the motion and the camera viewpoint and still achieve photo-realistic results. 3D video methods capture a dynamic scene with multiple video cameras (see Theobalt et al. [2007] for an overview), reconstruct [Zitnick et al. 2004; Matusik et al. 2000; Tung et al. 2009] or fit [Carranza et al. 2003; Ballan and Cortelazzo 2008] a geometry model, or use light field rendering to create novel viewpoints [Wilburn et al. 2005]. While these approaches enable rendering a moving actor from novel viewpoints, large changes in motion are not possible. Waschbüsch et al. [2006] presented a 3D video processing framework that extends 2D video operations such as segmentation and compositing to the 3D domain; editing the motion of a 3D video object is not possible. Weyrich et al. [2005] captured the reflectance field of a static object and used a geometric warping method to render the appearance of the object under new lighting after surface deformation. Scaling such an approach to videos of full humans would be a major engineering effort, and realistic surface deformations in the target animation would have to be entirely simulated. This is still a major problem in itself. Recently, Stoll et al. [2010] modified animation models of humans captured from multi-view video with estimated physics-based cloth models. However, they cannot generate realistic textures for the new animations.

Some previous work tries to create novel images or videos in which certain appearance aspects of a person are modified. Leyvand et al. [2008] proposed a learning based method to change the face shape of an input image. Kemelmacher-Shlizerman et al. [2010] transferred source facial expressions to a target character by finding similar frames in a video of the target's face. Zhou et al. [2010] reshaped a human body in one image based on user interaction. Jain et al. [2010] edited body shape in video sequences. All these approaches deliver realistic novel image or video footage, but none produce new camera views *and* new motions different from the input. In Hornung et al.'s [2007] work, a person in a single picture is animated by fitting to it a kinematic skeleton, and warping the image based on motion capture data. Viewpoint and texture cannot be changed. Cobzas et al. [2002] synthesized non-rigid motions of a moving arm by learning dynamic texture from multiple images. However, full body motion and varying viewpoints are not considered.

The video textures concept [Schödl et al. 2000] is also related to our work. The original idea is to analyze video sequences for possible transitions between any two frames. A new video can be assembled by rearranging the video frames and passing through transitions in a random order. As an extension, Schödl and Essa [2002] built a motion graph of input video sequences and use repeated sub-sequence replacement to support the control of transitions by users. Applying video textures to synthesize new videos of moving humans is challenging since identifying correct transitions without any knowledge of 3D pose and shape is difficult. Celly and Zordan [2004] took on that challenge by identifying transition regions between a restricted set of filmed human motions and performing purely image-based warping at transitions. Mori et al. [2004] followed a similar strategy but manually marked skeletons on images to support the creation of correct transitions. Flagg et al. [2009] used marker-based motion capture on the video footage to build a motion graph structure for synthesizing new footage. All these methods use some form of internal motion graph, i.e., they create new motion by re-ordering entire sub-sequences of specific input motions such as walking, jogging, or running. In other words, no novel motions can be synthesized and viewpoint changes are not feasible. The work by Starck et al. [2005] extends some of the above ideas to 3D video sequences, i.e., it creates a motion graph based on shape-from-silhouette meshes and textures reconstructed from multi-view video. For a few basic motion types, 3D blends are pre-computed. This enables creating new animations by concatenating the basic motions through the transitions. Similarly, Huang et al. [2009] built a motion graph to transition between 3D meshes reconstructed from a handful of multi-view video sequences showing basic motions. New 3D mesh sequences can be concatenated from the original mesh sequences. However, new sequences are restricted to the space of basic motions in the database, and texture synthesis is not considered. In contrast, our approach can synthesize new motions and textures even if none of the poses and camera views are part of the database.

When creating textures for the frames of a target animation we need to warp textures from the input video frames both across pose and viewpoints. One way of synthesizing textures in novel views or poses would be to use some form of projective texturing and blending, e.g., [Debevec et al. 1996; Narayanan et al. 1998; Buehler et al. 2001; Carranza et al. 2003; Cheung 2003; Hornung and Kobbelt 2009]. However, these approaches are known to produce texture ghosting when there are even the slightest of inaccuracies in scene geometry. For our work this is even more of an issue since we perform texture transfer across view and model pose. An alternative to texturing approaches are view synthesis methods that warp the source images into the target view using image-based correspondences, sometimes supported by a 3D scene model [Einarsson et al. 2006; Stich et al. 2008]. Warping approaches often prevent ghosting artifacts even if none or only approximate scene geometry is at hand. Ballan et al. [2010] used a billboard to transition from one video sequence to another at some optimized temporal frame using a warping approach without considering human geometry. Thus, we use an image-based warping method guided by our reconstructed performance models to warp textures across pose and viewpoint.

3 System Overview

Input to our system is a user-defined query: a skeletal animation seen from a user-defined camera, corresponding to a rigged surface mesh of an actor. The posed mesh in each animation time step is used to retrieve appropriate images from our multi-view video database of the captured actor in order to synthesize a realistic, animated output video (Fig. 2).

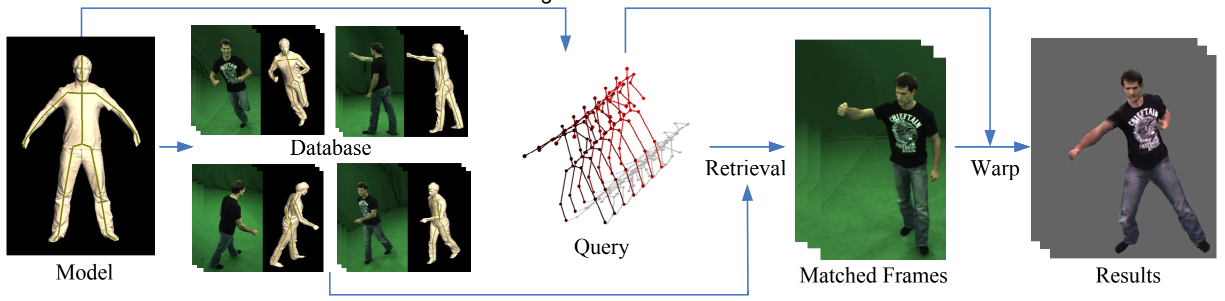

Figure 2: *Overview. The system maintains a database of multi-view video sequences of an actor performing simple motions. Skeleton motion and surface geometry were captured for each sequence using marker-less performance capture. The user is given a skeleton and surface mesh model of an actor. By specifying skeletal motion and camera settings, he can create a query animation. Our algorithm synthesizes a photo-realistic video of the actor performing the user-designed animation by selecting appropriate poses and their respective images from the database and warping these images to the target view.*

Database To guarantee sufficient images for synthesizing novel pose and viewpoint, we capture multi-view images of a character performing various basic motions. We call these *database sequences*, containing *database frames*. We then perform a skeleton-based performance capture algorithm [Gall et al. 2009] on the database sequences to obtain a 3D bone skeleton and a deformed mesh surface for each database frame. The obtained skeletons and meshes form the *database skeletons* and *database meshes*. The database sequences only need to be acquired and tracked once to synthesize novel target animations of the same character.

Query The skeleton and rigged surface mesh of the actor, or *body model*, is presented to the user for creating an animation. The user can also animate the camera to synthesize video sequences from novel viewpoints. For creation of a target animation, henceforth called a *query*, the user can employ any standard 3D animation package that supports character animation. Once the query is created, we deform the mesh surface of the character according to the skeleton. After that, for each query frame we have a query skeleton, a query camera and a query mesh. The same mesh and skeleton models of the actor are used to specify the queries and to capture the database performances.

Retrieval In this step we find appropriate images for synthesizing each target frame in our final output sequence (a so-called *target sequence*, frames of which are called *target frames*). We define a similarity to measure the distance between a database frame and a query frame from joint and camera positions. We find database frames matching the whole query sequence with this similarity, and call them *source frames*.

Warping-based Synthesis We synthesize the target sequence with the retrieval results. In this step, image-based warping adjusts the pose of the source character and the viewpoint of the source camera to satisfy the query. This is necessary as our database frames are not exactly the same as our target frames. We use vertex correspondence between the query mesh and the database mesh to guide the warping. By using the projected vertex correspondences in the image domain, the proposed warping method warps source images to obtain the final target sequence.

4 Multi-view Database and Motion Query

We first capture an actor performing multiple basic motions in our multi-view capture system. Our system has 12 synchronized and calibrated video cameras that capture the actor from different viewpoints at 45 fps with a frame resolution of 1296 $\times$ 972 pixels. Our basic motions are walking, running, jumping, kicking, punching, rolling, twisting and waving sequences. The database could be extended by adding more complex motions; however, we found that these basic motion types were enough to realistically synthesize a range of complex input animations. We found it unnecessary to design database motions such that the range of poses is maximized. To acquire sufficient view-dependent information, each basic motion is performed several times facing different directions.

After acquiring multi-view video sequences, skeleton-based marker-less performance capture [Gall et al. 2009] is applied to all captured sequences. This method requires an initial mesh surface of the captured character and a corresponding skeleton. The initial mesh surface is obtained by a static full-body laser scan. The resolution of the mesh surface is low, containing only 5000 vertices for one human character. The underlying skeleton of the mesh is created by manually marking joint positions. We use a skeleton with 16 joints as shown in Fig. 2. The skinning weight between each mesh vertex and each skeleton joint is calculated using the method of [Baran and Popovic 2007]. Motion tracking provides a skeleton and a mesh surface for the actor for each database frame. The skeletons, models, and captured multi-view sequences will be used in the following steps. The same skeleton and surface model are provided to the user to create new queries.

A query sequence is a user defined sequence that specifies the target video sequence using the same skeleton and mesh model as are in the database (Fig. 3, top row). It comprises a query skeleton sequence which defines the pose of the character at all time instances and a query camera sequence which specifies the camera parameters. The skeleton motion is used to animate the surface mesh from the query camera view. Animation tools allow the whole animation sequence to be defined, including key frame-based skeletal animation and camera motion. Alternatively, motion retargeting techniques [Gleicher 1998] can apply motion capture data from existing databases to the given skeleton, and match-moving software can extract camera paths from background videos, e.g., Voodoo[1].

5 Retrieval

In this step we take the query sequence and perform a database retrieval to find source frames for synthesis. The retrieved source frames need to contain enough information to synthesize the target frame. Our retrieval method accounts for pose and view similarity, and temporal consistency.

We base our retrieval method on the following observations: Rendering the target animation with a static texture creates implausible results as the natural expected motion and appearance changes on the surface of the character are not reproduced, see Sec. 7. It is crucial that the target animation features a spatio-temporally coherent

[1] `http://www.digilab.uni-hannover.de/docs/manual.html`

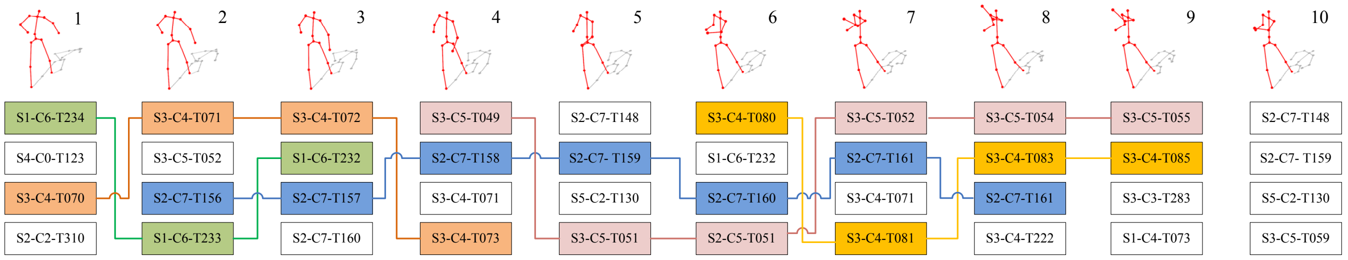

Figure 3: *Illustration of finding matching candidate sequences using $\ell = 2$. This figure shows 4 candidate frames for each query skeleton. Here* SX-CY-TZ *denotes the image frame from sequence X, camera view Y and time index Z. Candidate sequences are color-coded. No sequence of suitable length can be found for query frame* 10.

time-varying textural appearance that shows plausible dynamics for the given pose sequence and as little spatial and temporal aliasing as possible. Thus, the first goal is to find a plausible texture for each query pose. It is clear that for most poses and camera views in the query sequence we will not find an exact matching frame in the database. However, in most cases the appearance of the surface in a database frame will be very similar to the desired target if the camera angle and posture in the database are close to the query. The appearance is also influenced by lighting differences, but most of these can be equalized through color-based post-processing, as described in Sec. 6. The second goal is to favor temporally coherent texture patterns from the database rather than rapidly switching between database frames that are not temporally adjacent. This leads to naturally evolving and temporally coherent surface detail motion. The two goals are often conflicting, and the retrieval and synthesis strategy should find a balance that leads to high-fidelity output. In designing the best retrieval strategy we can capitalize on additional qualitative observations, such as that texture patterns usually change more rapidly if the motion in the scene is very fast. In the following paragraphs we describe how we translate the above concepts into a practical retrieval algorithm.

We need a *frame-to-frame* distance metric that measures the similarity between a query frame and a database frame (Sec. 5.1). To guarantee temporal consistency in the output, we prefer to find sub-sequences in the database in which every frame is sequentially matched to a query sub-sequence. We call this step *sequence-to-sequence matching* (Sec. 5.2). Matching only individual frames would often lead to temporal flickering artifacts in the result.

Sequence-to-sequence matching can only be established when there are some sub-sequences in the database similar to the query. However, as we use a database which contains only a handful of basic motion sequences to synthesize all possible motions, this similarity is difficult to guarantee. Therefore, we also allow the algorithm to fall back to frame-to-sequence matching, i.e., matching successive query frames to a single common database frame if no appropriate sequence-to-sequence match can be found (Sec. 5.3). The face region undergoes special processing, as we explain in Sec. 5.4.

5.1 Frame-to-frame Distance Metric

Let us first define a frame-to-frame distance metric which we will then extend to find sequence-to-sequence matches. Our distance definition is based on pose and camera differences. Both the tracked database skeletons and the query skeletons are first registered to the same world coordinate system by translating all root joints to the same 3D position and rotating all skeletons to face to the same direction. The camera is moved using the same global transformation as the skeleton in each frame to preserve its relative position and orientation to the skeleton. Given two frames, $\mathcal{F}_q$ from the query and $\mathcal{F}_d$ from the database, we first measure the camera difference:

$$D_{\text{cam}}(\mathbf{C}_q, \mathbf{C}_d) = \sqrt{(\mathbf{C}_q - \mathbf{C}_d)^2} \tag{1}$$

between the registered camera centers $\mathbf{C}$ in the registered query and database frame. Considering that all cameras are facing the actor at similar distances, this is a reasonable measurement. We also measure the skeletal distance:

$$D_{\text{skel}}(\mathbf{S}_q, \mathbf{S}_d) = \sqrt{\sum_{j=1}^{16} \frac{(\mathbf{S}_q^j - \mathbf{S}_d^j)^2}{\sigma_j^2}} \tag{2}$$

where $\mathbf{S}_q^j$ is the position of the j-th joint of the skeleton in $\mathcal{F}_q$ (and $\mathbf{S}_d^j$ for $\mathcal{F}_d$) and σ_j is the variance of the position of joint j in the database. We weight the individual joint position distances according to the variance in 3D pose of the respective joint in the database. Without such a weighting, joints that naturally undergo larger motions (such as joints towards the tips of end effectors like the wrist or ankle joints) would dominate the distance measure when compared to joints that are closer to the torso. We finally combine the two distance measures:

$$D(\mathcal{F}_q, \mathcal{F}_d) = D_{\text{skel}}(\mathbf{S}_q, \mathbf{S}_d) + \lambda \cdot D_{\text{cam}}(\mathbf{C}_q, \mathbf{C}_d) \tag{3}$$

where λ weights the camera and joint-similarity contributions. In practice, $D_{\text{skel}}(\mathbf{S}_q, \mathbf{S}_d)$ and $D_{\text{cam}}(\mathbf{C}_q, \mathbf{C}_d)$ are normalized, and we set $\lambda = 2$ which we determined through experiments. Using this distance measure, we search the database and collect the 100 best-matching frames $\mathcal{F}_d^c$ $(c = 1 \ldots 100)$ as candidates for synthesizing the texture in each query image.

Ranking Candidates Our final aim is to synthesize a realistic image for each query $\mathcal{F}_q$, so we wish the chosen database frames to contain as much information as possible. The measure $D(\cdot,\cdot)$ finds similar poses and views reliably and very quickly, which is crucial to ensure that overall retrieval run times are reasonable. However, it is based on the assumption that pose and view similarity corresponds to the amount of information available in a candidate frame. While this assumption generally holds, the measure is not based on candidate frame information content and does not allow for an accurate ranking of candidates $\mathcal{F}_d^c$. Therefore, we have derived a different measure to rank candidate frames $\mathcal{F}_d^c$. As described in Sec. 4, we have a 3D model and camera parameters for each frame in both the query and the database sequences. We use as a measure a visibility score $V(\mathcal{F}_q, \mathcal{F}_d^c)$, defined as the number of mesh facets that are visible from both the view of the query and the view of the tracked mesh of a database frame. While it may be possible to use this visibility score to find the 100 best candidate frames as well (instead of ranking them), it would be inefficient as only similar poses and views are likely to contain reasonable matches.

5.2 Sequence-to-sequence Matching

After collecting the best-matching frames for the whole query sequence, we try to extend the matches to generate sequence-to-sequence correspondence as follows: For each candidate of the first query frame, we check whether its temporally successive frame in the database is also a candidate of the second query frame. If so, we link the later temporal candidate to the former candidate. This process is continued to form a candidate sequence until no temporally consistent candidate can be found. In this case, we start a new candidate sequence at the frame where the sequence is interrupted. We repeat this for each candidate frame, building all possible candidate sequences $\mathcal{S}_i$ which may cover different parts of the query sequence and may have different lengths (see Fig. 3). The candidate sequences can start at any point in the animation.

We now select the best candidate sequences by considering three different factors. First, sequences should be as long as possible, since short sequences cannot guarantee temporal consistency. Second, the frames in a candidate sequence should be highly ranked (among the best matching frames for each query frame), as the ranking order corresponds to the similarity between the target frame and the database frames. Third, the amount of motion in a sequence should be similar to the query sequence. This ensures that the detailed surface motion (i.e., folds and wrinkles) does not vary infrequently because a slow motion from the database was used to synthesize a fast motion query sequence.

To describe the amount of motion, we define a motion energy between consecutive frames $\mathcal{F}^I$ and $\mathcal{F}^{I+1}$ as:

$$E_{\text{motion}}\left(\mathcal{F}^I, \mathcal{F}^{I+1}\right) = D_{\text{skel}}(\mathbf{S}_I, \mathbf{S}_{I+1}), \tag{4}$$

where $\mathbf{S}_I$ represents the skeleton corresponding to frame $\mathcal{F}^I$. The sum of motion energies in all frames in a sequence defines the amount of motion in the sequence.

The combined score $\Lambda(\cdot)$ for a candidate sequence $\mathcal{S}_i$, is:

$$\Lambda(\mathcal{S}_i) = \exp\left(-\alpha_{\text{dis}} \cdot E_{\text{dis}}(\mathcal{S}_i) - \alpha_{\text{rank}} \cdot Rank(\mathcal{S}_i) - 1/L(\mathcal{S}_i)\right), \tag{5}$$

$$E_{\text{dis}}(\mathcal{S}_i) = \sum_{I \in \mathcal{S}_i} E_{\text{motion}}(\mathcal{F}_d^I, \mathcal{F}_d^{I+1}) - \sum_{I \in Q_{\mathcal{S}_i}} E_{\text{motion}}(\mathcal{F}_q^I, \mathcal{F}_q^{I+1}), \tag{6}$$

$$Rank(\mathcal{S}_i) = \frac{1}{L(\mathcal{S}_i)} \sum_{I \in \mathcal{S}_i} Rank(\mathcal{F}_d^I), \tag{7}$$

where $L(\mathcal{S}_i)$ is the number of frames in candidate sequence $\mathcal{S}_i$, α_{dis} and α_{rank} weight the three factors and $Q_{\mathcal{S}_i}$ is the query subsequence corresponding to the candidate sequence $\mathcal{S}_i$. We usually set $\alpha_{E_{\text{dis}}} = \min_{I \in Q_{\mathcal{S}_i}} 1/E_{\text{motion}}(\mathcal{F}_q^I, \mathcal{F}_q^{I+1})$ and $\alpha_{\text{rank}} = \frac{1}{200}$ (200 is twice the number of candidates for a single query). This achieves good performance in our experience. A candidate sequence will not be used when its sequence length is smaller than a threshold ℓ (with $\ell = 10$ for all our results) or its corresponding query sequence is covered by another sequence element with a higher score. According to this mechanism, long sequence elements composed of high ranking candidates and with similar amounts of motion to the query will be chosen as the final matching result $\mathcal{F}_d^{match(I)}$ for frames $\mathcal{F}_q^I$.

5.3 Frame-to-sequence Matching

After the sequence-to-sequence matching there may still be some query frames with no matches, because all candidate sequences were less than ℓ frames long. For example, the 10th query in Fig. 3 is a frame with no match. If the query sequence motion and viewpoint is quite different from the database then the number of unmatched query frames may be large. In this step, we ensure consistent matches are found from the 100 candidate frames. We could reduce the length ℓ to guarantee no unmatched query frames. However, this would introduce many short sequences to the result and so add temporal jitter (see Sec. 7 **validation II**).

For an unmatched query sub-sequence, $\mathcal{F}_q^{U_1}$ to $\mathcal{F}_q^{U_2}$, we could simply find the top ranked candidate $\mathcal{F}_d^{best(U)}$ for each query frame $\mathcal{F}_q^U$ ($U \in [U_1, U_2]$) as $\mathcal{F}_d^{match(U)}$. However, top candidates may change frequently in a sub-sequence and therefore cause considerable temporal jittering in the final synthesized sequence (see user study, Sec. 7). Considering that consecutive frames in a query sequence are likely to change slowly, we instead use the single best database frame for as many consecutive query frames as possible. Here, we trade reduced texture motion (moving cloth folds) in the synthesized result for reduced temporal jitter.

Given that for frame $\mathcal{F}_q^J$ (initially $J = U_1$) we have found the best matching database frame $\mathcal{F}_d^{\text{best}(J)}$, we must decide how many consecutive query frames $\mathcal{F}_q^{J+n}$ can use this database frame match. We apply insight as before: The query sequence parts with fast motion will require more database frames to represent them. Conversely, slow motions will require fewer frames. With this observation, we keep the best database frame $\mathcal{F}_d^{\text{best}(J)}$ until the visibility score between the current query frame $\mathcal{F}_q^{J+n}$ and the database frame falls below a threshold:

$$V(\mathcal{F}_q^{J+n}, \mathcal{F}_d^{\text{best}(J)}) < \theta_{\text{vis}}. \tag{8}$$

This threshold is based on accumulated motion energy over consecutive frames:

$$\theta_{\text{vis}} = 0.9 \cdot V(\mathcal{F}_q^{J+n}, \mathcal{F}_d^{\text{best}(J+n)}) \cdot \exp\left(\sum_{I=J}^{J+n-1} \alpha_{\text{mot}} \cdot E_{\text{motion}}\left(\mathcal{F}_q^I, \mathcal{F}_q^{I+1}\right)\right),$$

We require an increasingly large visibility score as the number of frames increases (we set $\alpha_{\text{mot}} = \alpha_{\text{dis}}/200$). This ensures that for fast motions we switch to the current best candidate $\mathcal{F}_d^{\text{best}(J+n)}$ quickly and a transition occurs.

5.4 Retrieval for Face Region

The described pipeline retrieves database entries for the whole body except for the head. The head is processed separately by a simplified version of the retrieval pipeline. The complete pipeline retrieves enough information to reconstruct complete spatio-temporally coherent target views (see Sec. 6), which is essential for the body region. The quality criteria for the head is slightly different. Since human perception is tuned to recognize faces, any form of visual discontinuity in the rendered target face would be highly disturbing (such as a visible seam across the face). Spatial coherence is significantly more relevant here, and other forms of small errors such as missing texture are less visually objectionable. We do not use the full retrieval pipeline because even subtle errors in tracked head poses in the database lead to starkly visible discontinuities when several source images are combined to form the target face. Instead, in each query frame we find the single best database frame $\mathcal{F}_d^{\text{head}}$ from which to copy over the entire head texture. Since we want to copy the face region from a database frame where the face is completely visible, we choose the frame for which the visibility score $\mathcal{V}$ between target face facets and database face facets is highest. Similar to Sec. 5.3, we use the same $\mathcal{F}_d^{\text{head}}$ for as many subsequent query frames as possible.

6 Warping-based Video Synthesis

We now want to synthesize a realistic image for each of the target frames. After retrieval (for the body), we have the best matching frame $\mathcal{F}_d^{\text{match}}$ from the database (and the corresponding camera view and pose / surface mesh as seen from that camera) for every query frame $\mathcal{F}_q$ of the target sequence (and the query mesh and query camera at every time step). In addition, we also employ the frames $\mathcal{F}_d^k$ ($k \in \mathbf{C}_d \setminus \{\text{match}\}$, $\mathbf{C}_d$ being the set of database camera views) from all other camera views onto the *same* database pose. For every query frame, we also have the single best matching frame $\mathcal{F}_d^{\text{head}}$ for synthesizing the head texture in the target view. For every query

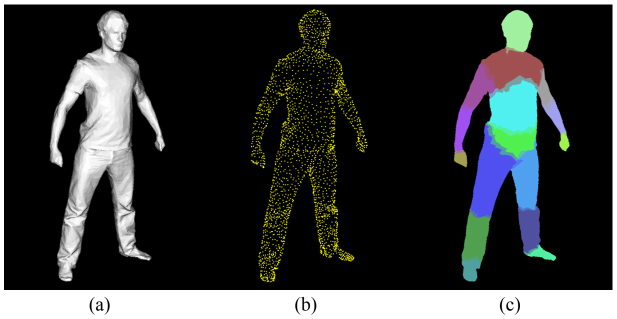

Figure 4: *(a) Surface mesh of a person used in query and database. (b) Vertices of the model that are used as MLS warping constraints. (c) Vertices are labeled according to the body part upon which they lie (color=label). In MLS warping each vertex only affects the appearance of the respective body part, except for the boundary regions between body parts where warps are blended.*

frame, we also synthesize an initial guess for the appearance of the target frame by projectively texturing the query mesh with a static surface texture taken from *all* camera views at the first time step in the database, henceforth called $\mathcal{F}_d^{\text{static}}$. The final texture for the query frame is then synthesized by model-guided warping of texture information from the best matching database frames to the target. In practice, we first synthesize the target view for the head, and subsequently for the rest of the body (Fig. 5).

For the head, only a single source image $\mathcal{F}_d^{\text{head}}$ is warped. In the case of the body, frame $\mathcal{F}_d^{\text{match}}$ contains most of the texture information needed to synthesize the target texture. However, due to slight pose and camera differences between database and target, we warp additional image data $\mathcal{F}_d^{\text{k}}$ to the target and blend with the existing result (Fig. 5). If any holes remain we blend with $\mathcal{F}_d^{\text{static}}$. Image-based warping with this data creates complete target textures even with visibility differences, completely invisible regions and geometry inaccuracies in the query and database meshes.

We capitalize on the fact that we are given the same surface mesh in both the query and database poses (i.e., the same connectivity but different poses). We use vertex correspondences between image projections of the same vertex in the source and target to determine the warp. Using these correspondences as guidance, we use moving least square (MLS) [Schaefer et al. 2006] to obtain corresponding pixels in the source frame for all pixels in the target frame; i.e., to calculate motions of pixels between source and target frame.

MLS warping weights every pixel neighborhood in an image equally, and thus through a local coherence assumption creates smoothly interpolated warps even if actual warping constraints are sparser than the pixel grid (see Fig. 4(b)). In some regions, such as at occlusion boundaries between different parts of the body (e.g., an arm in front of the torso) this smooth interpolation may be unwanted. However, if two neighboring body parts are actually connected, the warp should be smooth. The problem is that the original MLS method is agnostic to scene geometry. Therefore, we develop a strategy to enable smooth warping where it is wanted and to prevent it across occlusion edges. First, we group human body pixels into 16 segments based on the body part information of the query 3D mesh (see Fig. 4(c)). The motions of pixels in each body part are computed only from vertex correspondences in the same body part. For every pixel lying on the boundary between connected body parts A and B, we compute two warp hypothesis: one under the assumption that it belongs to A, and one that it belongs to B. The final warp is computed by weighted blending according to the distance to the body part boundary. Finally, we warp the source frame to the target. There are specific implementation details which need to be addressed during the warping, and these are described in the following paragraphs.

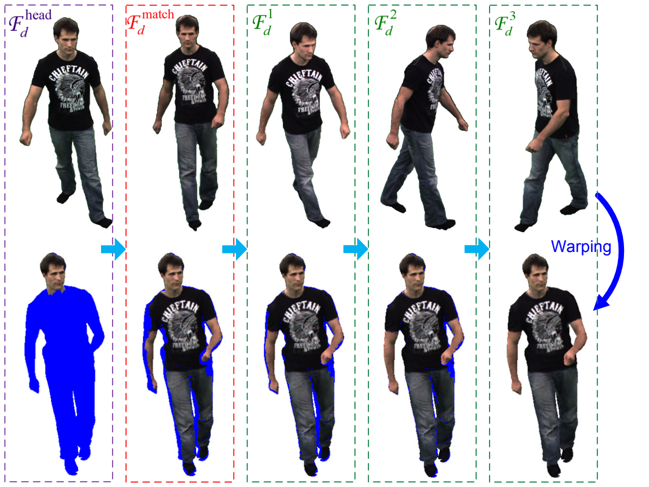

Figure 5: *Step-by-step warping-based synthesis of the result using MLS. First row: source frames used in warping ($\mathcal{F}_d^{\text{match}}$ and $\mathcal{F}_d^{k}$). Second row: results after warping the respective frame (above) to the target. Holes in the target view (blue) are continuously filled in.*

Missing Regions The best-matching database frame $\mathcal{F}_d^{\text{match}}$ does not usually contain all the information required to synthesize the target frame, and missing regions may occur in the target image after warping. As a consequence, $\mathcal{F}_d^{\text{k}}$ are also used in our warping process. The warping algorithm performed on these frames is the same as the warping of $\mathcal{F}_d^{\text{match}}$. However, only missing pixels in the target frames are considered for warping. A complete target image is obtained after spatial blending of the warping results from different images. This procedure is shown in Fig. 5. Any pixel to which the query mesh projects that is still untextured after all warps is colored from the statically textured result $\mathcal{F}_d^{\text{static}}$ using Poisson image blending for compositing.

Ghosting Due to small inaccuracies in the tracked 3D mesh, its projection will never align exactly with the database frames, and pixels may be matched with incorrect body parts. These mismatches will lead to ghosting artifacts (see Fig. 6(b)).

We detect the occurrence of these artifacts using depth information. Ghosting in the target frame is likely to originate from areas around depth discontinuities in the database frame. In these regions, occlusion boundaries may not be correctly represented by the 3D geometry, e.g., pixels on an arm may be treated as pixels on the torso and warped to torso pixels in the target frame. Based on this observation, we first define occlusion boundaries in database frames by analyzing depth discontinuities. We then ignore pixels near occlusion boundaries in the warping procedure. Detecting depth discontinuities alone is not sufficient since the depth difference may be very small if the arm is close to the body, but ghosting may still occur. Thus, we ignore pixels if they are close to a depth *and* body part label discontinuity. Consequently, all pixels used are far from occlusion boundaries and have correct body part labels, and ghosting artifacts are eliminated from our final results.

Temporal Jumps Noticeable temporal jumps in appearance can occur when consecutive target images are synthesized from very different database images. While we ensure that this happens as little as possible (see Sec. 5), it can still occur. We overcome this problem by temporally blending between consecutive sequences $\mathcal{S}_r$, $\mathcal{S}_s$. Namely, each target frame is synthesized twice from two

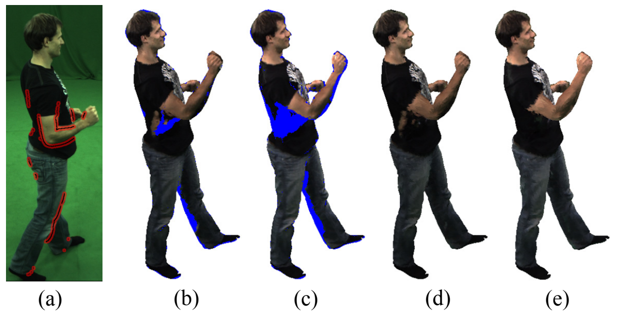

Figure 6: *Ghosting is minimized by ignoring pixels near depth discontinuities in the database frame. (a) the best matching database frame $\mathcal{F}_d^{\text{match}}$ – ignored pixels are shown in red. (b,c) are the warping results from $\mathcal{F}_d^{\text{match}}$ without and with the deghosting method. (d,e) are the final results corresponding to (b) and (c).*

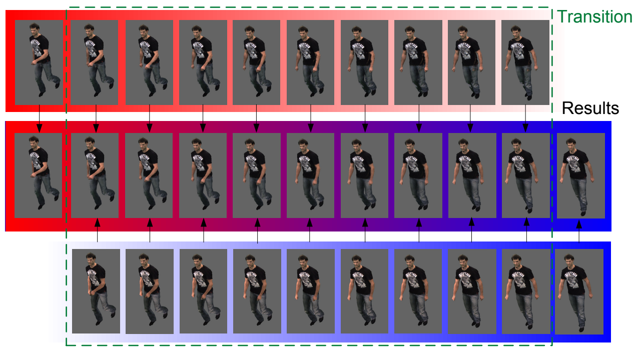

Figure 7: *We avoid temporal appearance jumps by smoothly blending between two consecutive sequences $\mathcal{S}_r$ (top) and $\mathcal{S}_s$ (bottom). Red and blue are used to color-code respective blending weights used for the final result in the middle.*

different candidate sequences. A smooth transition is achieved by blending the two results (after spatial alignment with optical flow), weighted by their distances to the centers of the two sequences (see Fig. 7). The time duration of the blend can be adjusted to suit each sequence (see Sec. 7). Temporal blending is only triggered for segments of the target animation for which candidate sequences in the database were found.

Lighting Adjustment When compositing a target image, database frames are warped and combined into a single view that may exhibit noticeable lighting changes. To compensate, we color adjust warped frames to the already existing target texture by means of a linear color transform. Since the new texture to be warped will usually have some overlap with the existing target texture, this transform can be estimated from overlapping regions.

Color adjustment also needs to be applied over time. During a transition from one target frame to another, whenever a transition happens to a new best matching database sequence a similar color adjustment is computed to match the new database sequence to the color of the previous target frame.

7 Results

We have evaluated our approach with three actor databases: a male wearing a black t-shirt with a logo pattern and jeans (**s1**, Fig. 8 top), a male subject wearing a blue t-shirt with a strongly textured logo and normal pants (**s2**, Fig. 8 bottom), and a female subject wearing a red sweater and normal pants (**s3**, Fig. 8 middle). For each subject we recorded a database of 10 multi-view video sequences, each one showing a basic motion such as walking, running, punching (one hand punch, two hand punch, up-down punch, left-right punch), jumping (one foot and two feet) or waving. Please refer to the supplemental video for a visualization. The database sequences are between 400 and 1000 frames long. Skeleton and mesh motion for each sequence were captured with the camera system and the algorithm described in Sec. 4. Also, as described in that section, we create a skeleton model and a laser scan of each subject and rig the skeleton to the latter.

We use 8 different skeleton motion sequences as queries. Six of these were captured with the marker-less capture system (but are not part of the multi-view database): **KungFu**, **Fight1**, **Fight2**, **March**, **Run1**, and **Run2**. We also use two skeletal motion sequences that we downloaded from an online motion capture repository[2]: **MJDance** and **Catwalk**. These sequences are between 600 and 1000 frames long. All motions are quite different from the contents of the databases. Using these data, we created 13 new video animations by applying the query motions to the different subjects, and by specifying different camera paths. In the paper and the video we will refer to a synthesized sequence by the name **s_m_c**, meaning subject **s** performing motion **m** seen from camera **c**. We used Autodesk Maya$^{\text{TM}}$ to set up cameras and preview the queries, and also used the built-in retargeting tool if the motion data came from a different subject than the one in the target animation. The size of the synthesized characters in our sequences varied from 80×220 to 400×600 pixels.

[2] `http://mocap.cs.cmu.edu/`

The figures in the paper and the accompanying video show a variety of synthesized animations. In addition to the synthesized character, we also render synthetic shadows in the background using the 3D model from the target animation and 8 light sources.

In all results, the appearance of the final video result looks plausible since the motions *and* the dynamic textural appearance are synthesized in a realistic and spatio-temporally coherent way. In sequences **s2_March_c9** and **s2_MJWalk_c6** one can see that plausible coherent patterns of folds are created on the clothes, and the dynamics of the texture patterns reflects the dynamics of the overall character motion.

We also successfully synthesized the same target motion from two different camera views (**s3_Catwalk_c7** and **s3_Catwalk_c8**, as well as **s1_Run1_c2** and **s1_Run1_c3**) showing the flexibility and freedom in animation synthesis that our approach provides. Similarly, we can generate in high quality the same target motion from the same camera view for two different characters (**s1_Kungfu_c5** and **s3_Kungfu_c5**).

The video and Fig. 1 also show a composite of the sequence **s3_Catwalk_c11** with a real background. Camera **c11** was tracked with a commercial camera tracker and shadows of the person on the ground were rendered in Maya. The final result looks convincing and demonstrates that it is feasible to insert video-based animations with user-designed motions into real videos.

Our approach also succeeds if the frame rates of target animation and database videos are not the same, but equal frame rates alleviates some problems. Tracking accuracies of the database sequences may vary, and thus some parameters of the method may need adjustment: for instance, the number of pixels around depth discontinuities that are ignored for ghosting removal (4 for **s2** and 8 for other subjects); also, the temporal window in which transitions are blended varies between 6 and 10 frames.

User Study We performed a user study with 32 participants to validate the visual fidelity of our synthesized video animations.

Each of the participants was shown a web page containing four videos: one (video B) was a real video recorded in our studio, while the other three videos were generated all from the same camera view with our system. To generate these animations, a test sequence, **s1_Run2_i2**, was recorded in our studio, which was not part of the database used for synthesis. We then use the captured performance of that sequence from one of the input camera views, **i2**, as query to synthesize new results. For the synthetic sequences we created synthetic shadows and composited the results with a background frame of the empty studio to match the background of the real video. Video C was synthesized by applying a static projective texture to the model ($\mathcal{F}_d^{\text{static}}$ from Sec. 6). Video A uses the best matching database frame (according to the visibility score from Sec. 5.1) plus six closest camera views from the same database sequence at the same time step to synthesize each target frame independently. Video D shows the result of our full pipeline.

In the first experiment, participants were asked to evaluate how realistic and convincing the character in each video looks with respect to its motion and appearance on a scale from 1 (very realistic) to 5 (not realistic at all). All participants correctly identified video B showing the real footage as the most realistic character with average score 1.4. Our result was ranked as second best (video D: 3.1), and is clearly preferred over the static texture (video C: 4.15) and the best match per frame of animation (video D: 4.12). This result is statistically significant with a one-way ANOVA p-value < 0.01. A similar result was achieved in our second experiment, where we asked the subjects to rank the videos according to realism (1=most realistic, 4=least realistic). All subjects ranked the real sequence (video B) highest. 66% of the subjects ranked our result (video D) as the second most realistic character.

Validation The user study confirms that all components of the retrieval and rendering pipelines are crucial to the success of our method. Not surprisingly, the user study shows that using a static texture for creating the final video is not considered plausible, as any variation of surface appearance characteristic of real videos is missing entirely. Using the best matching database frames at each query frame independently does not take spatio-temporal coherence into account and leads to objectionable high-frequency brightness variations and incoherent texture patterns. Thus, retrieval of matching sequences is an important feature.

A crucial rendering component is temporal blending at sequence transitions, as a comparison on sequence **s1_Run2_i2** (labeled **validation I** in the second supplementary video) shows. We show our full pipeline versus the pipeline without temporal blending. The video without blending contains objectionable jumps.

Another comparison on **s1_Run2_i2** (labeled **validation II** in second supplemental video) shows that simply lowering parameter ℓ (see Sec. 5.3) to minimize the number of frames not matched to a candidate sequence from the database is not advisable. As opposed to our method, the latter strategy creates too many temporal artifacts. This is why we resorted to the best frame matching from Sec. 5.3 only as a last resort after as-long-as-possible sequences from the database were matched. In all our experiments, the ratio of the target frames from frame-to-sequence matching to the target frames from sequence-to-sequence matching ranged from 1:9.99 to 1:2.39.

Fig. 6 shows that deghosting is key to achieving plausible results. It effectively removes artifacts stemming from subtle geometry-to-image misalignments that would otherwise spoil the impression.

Timings We evaluated our method on an Intel Core 2 with 2.66 GHz. The skeleton is manually embedded in the mesh for each character which takes around 5 minutes. This is the only manual processing step in our pipeline (apart from generating query animations and viewpoints), and it only needs to be performed once for each actor. The database tracking takes around 10s per multi-view video time step and needs to be performed only once for each actor database. Given a database and a target animation, synthesizing the output video takes between 40s and 50s per frame. This includes neighbor search (about 10s per frame), and warping+composition (about 30s per frame). Per-frame run times are mostly dependent on the resolution of the person in the target frame, as well as the number of frames for which temporal blending is triggered. All steps are performed automatically without any user intervention. Manual intervention is possible to inspect results and rerun computations with changed parameters if desired.

7.1 Discussion

Even though we cannot guarantee that our final results look exactly like the real person performing a specified motion, our results are plausible. Our textures are spatio-temporally coherent, exhibit little aliasing, and reproduce plausible time-varying appearance patterns that match the dynamics of the target motion. This enable us to convey a very realistic impression of a real person.

Our approach is subject to limitations. To a certain extent, the quality of the final results depends on the performance capture accuracy of the database. As a specific example, even subtle inaccuracies in tracked head pose sometimes lead to artifacts in the final warping result. These artifacts could be mitigated through an improved pose tracking of the head, or through optional manual correction of the pose in the database.

Currently, we do not demonstrate results with people wearing loose apparel, but we believe this is feasible. The performance capture approach could track such database sequences, but we would need to identify cloth on the body model and simulate its motion when generating the query. The latter is a challenging problem in itself. [Stoll et al. 2010] show an approach to identify cloth in 3D performances and simulate new motions, and combining it with our approach may be an avenue worth taking.

Our method is geared towards minimizing objectionable spatial and temporal artifacts. However, they cannot be entirely prevented. Sometimes the result videos have subtle brightness variations in some regions, which are due to lighting differences between database frames. Our strategy to minimize these artifacts is to create uniform lighting in the database recording environment. In future, more advanced lighting adjustment strategies could be researched. Some subtle implausible texture shifting is due to non-optimal skinning of the model. In professional productions, it is very common that skinning weights are adjusted for every animation sequence individually. For time reasons we did not do this, and some artifacts may be lessened by this.

While we can synthesize previously unseen motions, the synthesized animations may contain artifacts if the query motion is too dissimilar to all database motions. In this case, one should augment the database with additional sequences closer to the query. However, our examples suggest that with a database of moderate size, a decent range of cameras and motions can be generated.

Finally, we purposefully decided to use an image-based MLS warping scheme to synthesize the final video rather than a projective texturing variant. A purely image-based warping scheme conceptually provides more flexibility. For instance, with MLS warping it is possible to warp pixel information to the target from areas close to the projected model boundaries, even if they fall outside of the projected model in the database frames. This way, inaccuracies in the shape model will be less noticeable in the target image. Also,

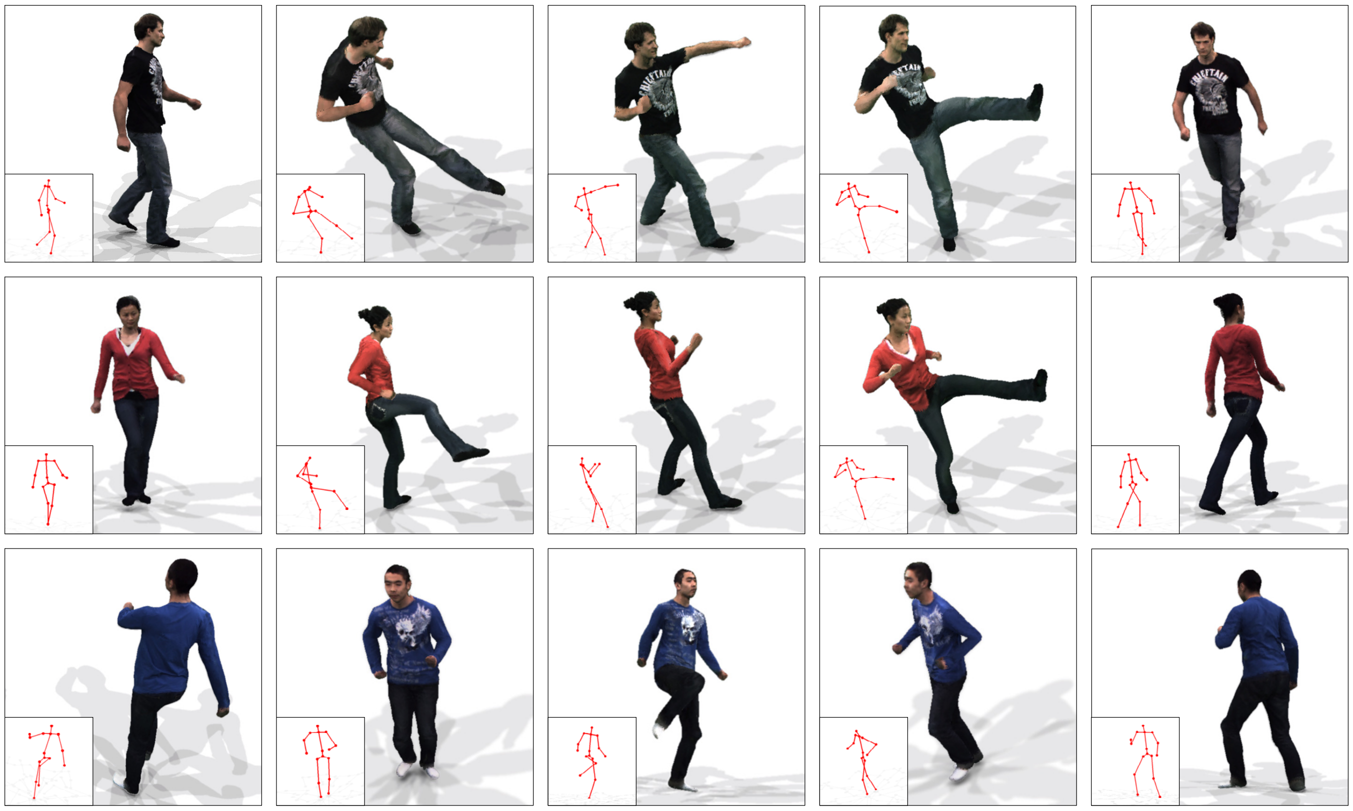

Figure 8: *Individual frames from several of our result videos for actors* **s1** *(top),* **s3** *(middle), and* **s2** *(bottom). The inlays show the respective query skeleton poses used to synthesize the target frame.*

the warping scheme gives us more control over how many warping constraints to use in target texture generation and we expect the result to be smoother in the target for a large range of triangle mesh resolutions. However, in principle it is feasible to implement source-to-target frame warping with projective texturing.

8 Conclusion

We have presented a new data-driven approach for image-based synthesis of realistic video animations containing user-defined human motions seen from user-defined camera views. One key component of our approach is a retrieval algorithm to find suitable images for synthesis of target textures from a multi-view video database of simple motions. Our approach exploits skeleton and shape knowledge for each input sequence, obtained via marker-less performance capture. The retrieval component is designed to assemble spatio-temporally coherent texture patterns in the target that are plausible depictions of the actor performing the user-designed motion. An image-base warping approach composites the textural appearance in the target frame and is designed to minimize temporal and spatial appearance artifacts, as well as lighting mismatches and ghosting. We have demonstrated the success of our method with numerous examples and validated it with a user study.

Acknowledgements

This work was completed when the first author was a visitor at MPI Informatik. Financial support came from the Intel Visual Computing Institute, Saarbrücken, Germany. Support in China came from the National Basic Research Project (No.2010CB731800), NSFC grants No. 61035002 & 60933006, and the 863 Program No. 2009AA01Z327. Support in the UK came from the EngD VEIV center at UCL and from BBC R&D.

References

BALLAN, L., AND CORTELAZZO, G. M. 2008. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. In *3DPVT*.

BALLAN, L., BROSTOW, G. J., PUWEIN, J., AND POLLEFEYS, M. 2010. Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM TOG (Proc. SIGGRAPH)*, 1–11.

BARAN, I., AND POPOVIC, J. 2007. Automatic rigging and animation of 3d characters. *ACM TOG (SIGGRAPH) 26*, 3, 72.

BRADLEY, D., POPA, T., SHEFFER, A., HEIDRICH, W., AND BOUBEKEUR, T. 2008. Markerless garment capture. *ACM TOG (Proc. SIGGRAPH) 27*, 3, 99.

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *SIGGRAPH*, 425–432.

CAGNIART, C., BOYER, E., AND ILIC, S. 2010. Free-form mesh tracking: a patch-based approach. In *Proc. IEEE CVPR*, 1–8.

CARRANZA, J., THEOBALT, C., MAGNOR, M., AND SEIDEL, H.-P. 2003. Free-viewpoint video of human actors. In *ACM TOG (Proc. SIGGRAPH)*.

CELLY, B., AND ZORDAN, V. 2004. Animated people textures. In *Proc. of CASA*, 331–338.

CHEUNG, G. 2003. *Visual Hull Construction, Alignment and Refinement for Human Kinematic Modeling, Motion Capture and Rendering*. PhD thesis, Carnegie Mellon University.

COBZAS, D., YEREX, K., AND JAGERSAND, M. 2002. Dynamic textures for image-based rendering of fine-scale 3d structure and animation of non-rigid motion. In *In Eurographics*, 1067–7055.

DE AGUIAR, E., STOLL, C., THEOBALT, C., AHMED, N., SEIDEL, H.-P., AND THRUN, S. 2008. Performance capture from sparse multi-view video. *ACM TOG (SIGGRAPH) 27*, 1–10.

DEBEVEC, P. E., TAYLOR, C. J., AND MALIK, J. 1996. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *SIGGRAPH*, 11–20.

EINARSSON, P., CHABERT, C.-F., JONES, A., MA, W.-C., LAMOND, B., IM HAWKINS, BOLAS, M., SYLWAN, S., AND DEBEVEC, P. 2006. Relighting human locomotion with flowed reflectance fields. In *Proc. EGSR*, 183–194.

FLAGG, M., NAKAZAWA, A., ZHANG, Q., KANG, S. B., RYU, Y. K., ESSA, I., AND REHG, J. M. 2009. Human video textures. In *Proc. of I3D*, 199–206.

GALL, J., STOLL, C., AGUIAR, E., THEOBALT, C., ROSENHAHN, B., AND SEIDEL, H.-P. 2009. Motion capture using joint skeleton tracking and surface estimation. In *Proc. IEEE CVPR*, 1746–1753.

GLEICHER, M. 1998. Retargetting motion to new characters. In *SIGGRAPH '98*, 33–42.

HORNUNG, A., AND KOBBELT, L. 2009. Interactive pixel-accurate free viewpoint rendering from images with silhouette aware sampling. *Comput. Graph. Forum 28*, 8, 2090–2103.

HORNUNG, A., DEKKERS, E., AND KOBBELT, L. 2007. Character animation from 2d pictures and 3d motion data. *ACM TOG 26*, 1, 1:1–1:9.

HUANG, P., HILTON, A., AND STARCK, J. 2009. Human motion synthesis from 3d video. In *Proc. CVPR*, 1478 –1485.

JAIN, A., THORMÄHLEN, T., SEIDEL, H.-P., AND THEOBALT, C. 2010. Moviereshape: tracking and reshaping of humans in videos. *ACM TOG (Proc. SIGGRAPH Asia) 29*, 148:1–148:10.

JIMENEZ, J., SCULLY, T., BARBOSA, N., DONNER, C., ALVAREZ, X., VIEIRA, T., MATTS, P., ORVALHO, V., GUTIERREZ, D., AND WEYRICH, T. 2010. A practical appearance model for dynamic facial color. *ACM TOG (Proc. SIGGRAPH Asia) 29*, 141:1–141:10.

KEMELMACHER-SHLIZERMAN, I., SANKAR, A., SHECHTMAN, E., AND SEITZ, S. M. 2010. Being john malkovich. In *Proc. of ECCV*, 341 – 353.

LEYVAND, T., COHEN-OR, D., DROR, G., AND LISCHINSKI, D. 2008. Data-driven enhancement of facial attractiveness. *ACM TOG (Proc. SIGGRAPH) 27*, 3, 38:1–38:9.

MATUSIK, W., BUEHLER, C., RASKAR, R., GORTLER, S. J., AND MCMILLAN, L. 2000. Image-based visual hulls. SIGGRAPH '00, 369–374.

MORI, G., BERG, A., EFROS, A., EDEN, A., AND MALIK, J. 2004. Video based motion synthesis by splicing and morphing. *UC Berkeley Technical Reports, No. UCB/CSD-4-1337*.

NARAYANAN, P. J., RANDER, P., AND KANADE, T. 1998. Constructing virtual worlds using dense stereo. In *Proc. of ICCV*, 3 – 10.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. D. 2006. Image deformation using moving least squares. *ACM TOG (Proc. SIGGRAPH) 25*, 3, 533–540.

SCHÖDL, A., AND ESSA, I. 2002. Controlled animation of video sprites. In *Proc. of SCA*, 121–127.

SCHÖDL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. In *SIGGRAPH*, 489–498.

STARCK, J., MILLER, G., AND HILTON, A. 2005. Video-based character animation. In *Proc. of SCA*, 49–58.

STICH, T., LINZ, C., ALBUQUERQUE, G., AND MAGNOR, M. 2008. View and Time Interpolation in Image Space. *Computer Graphics Forum (Proc. Pacific Graphics) 27*, 7.

STOLL, C., GALL, J., DE AGUIAR, E., THRUN, S., AND THEOBALT, C. 2010. Video-based reconstruction of animatable human characters. *ACM TOG (Proc. SIGGRAPH Asia) 29*, 139:1–139:10.

THEOBALT, C., WUERMLIN, S., DE AGUIAR, E., AND NIEDERBERGER, C. 2007. New trends in 3d video. In *Eurographics Courses*.

TUNG, T., NOBUHARA, S., AND MATSUYAMA, T. 2009. Complete multi-view reconstruction of dynamic scenes from probabilistic fusion of narrow and wide baseline stereo. In *Proc. IEEE ICCV*, 1709 –1716.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM TOG (Proc. SIGGRAPH '08)*.

VLASIC, D., PEERS, P., BARAN, I., DEBEVEC, P., POPOVIĆ, J., RUSINKIEWICZ, S., AND MATUSIK, W. 2009. Dynamic shape capture using multi-view photometric stereo. In *ACM TOG (Proc. SIGGRAPH Asia '09)*.

WASCHBÜSCH, M., WÜRMLIN, S., AND GROSS, M. 2006. Interactive 3d video editing. *Vis. Comput. 22*, 631–641.

WEYRICH, T., PFISTER, H., AND GROSS, M. 2005. Rendering deformable surface reflectance fields. *IEEE TVCG 11*, 48–58.

WILBURN, B., JOSHI, N., VAISH, V., TALVALA, E.-V., ANTUNEZ, E., BARTH, A., ADAMS, A., HOROWITZ, M., AND LEVOY, M. 2005. High performance imaging using large camera arrays. *ACM TOG (Proc. SIGGRAPH) 24*, 765–776.

ZHOU, S., FU, H., LIU, L., COHEN-OR, D., AND HAN, X. 2010. Parametric reshaping of human bodies in images. *ACM TOG (Proc. SIGGRAPH) 29*, 4, 126:1–126:10.

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S. A. J., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. *ACM TOG (Proc. SIGGRAPH) 23*, 3, 600–608.