Video Based Motion Synthesis by Splicing and Morphing

Greg Mori, Alex Berg, Alyosha Efros, Ashley Eden,
and Jitendra Malik
University of California, Berkeley
Berkeley, CA 94720

June 2004

Abstract

In this paper we present a method for synthesizing videos of human motion by splicing together clips of input video. There are two main contributions in this work. The first is developing a method for "kinematically correct morphing" of images of human figure, which is used to splice together the clips in input video in a manner that produces smooth output sequences. The second contribution of this work is the application of activity recognition algorithms to our input data in order to automatically extract action labels, which allow us to control the synthesized video by issuing high-level action commands. We present results of synthetic sequences on two domains: ballet and tennis.

1 Introduction

In this paper we present a method for synthesizing videos of human motion. The synthetic videos are produced by splicing together clips of input video. There are two main challenges, allowing for high-level control of the figure (specifying actions to be performed), and making the figure look realistic.

Synthesizing videos of human motion directly from existing clips is an alluring goal. Ignoring the possible pitfalls, the prospect of controlling realistic looking characters performing accurately portrayed actions with properly deforming clothing is very appealing. Facade [7] was instrumental in popularizing image-based rendering techniques for stationary objects. The advantages of realistic appearance over that obtained through 3d modeling were obvious. Previous work by Schodl et al. [19] has shown impressive results in using image-based rendering to synthesize videos, particularly of natural phenomena such as waterfalls and fire. Articulated human figures have structure, of a type that is not present in these natural phenomena, which must be preserved in order to produce convincing animations. In this work we attempt to address this more difficult image-based rendering problem.

The problem of synthesizing novel motions in the realm of 3d motion capture data has been the focus of recent work [1, 12, 9]. These methods generate a motions given constraints such as position, orientation, keyframes, or paths, and are used to render 3d models of actors to produce novel videos. In this work we leverage these ideas as the basis for controlling the motions we synthesize.

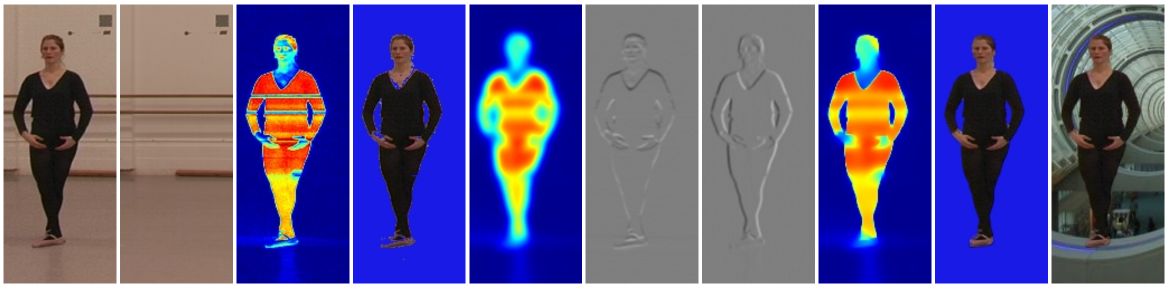
Our approach will use computer vision techniques to address the issues in video-based motion synthesis. The first contribution of this work is developing a method for "kinematically correct morphing" of images of human figure. The method relies on a vision-based technique used to automatically extract the 2d skeleton in a frame, which is used to deform the figure correctly. Another issue is the amount of video data required in order to make the synthesis work. In order to allow for high-level control, this data should be tagged with action labels. The second contribution of this work is the application of activity recognition algorithms to our input data in order to automatically extract these action labels.

The structure of the paper is as follows. We describe related work in Section 2. Preprocessing of the input video to extract sprites is described in Section 3. Next, we describe the details of our method: extracting the 2d skeletons (Section 4), the morphing and splicing (Section 5), building the motion graph (Section 6). Experimental results are provided in Section 7. We conclude in Section 8.

2 Related Work

There has been much previous work on motion synthesis using 3d models which can broadly be divided into the realms of physically based and motion capture based methods. Physically based methods [10, 21, 14] can generate motion without the use of motion data, but can't generate very realistic motions. Of the techniques using motion capture data, many of them (e.g. [1, 12, 17, 13]) generate a motion given constraints such as position, orientation, frame, or path, but are unable to synthesize based on a given sequence of actions. In order to trace a path of motions, one may follow a motion graph. Gleicher et al. [11, 9] make an explicit directed graph where each edge is a motion clip annotated with an action, but their method does not scale well with the number of different actions. Rose et al. [18] create a verb graph, interpolating between different adverb combinations of the same verb to create a new style of motion. Because the graph is hand-created, it is difficult to encode a large database, and the interpolation is not guarenteed to be realistic. Arikan et al. [2] did not create an explicit graph, but instead employed dynamic programming techniques to find the optimal path given possibly overlapping annotation constraints.

Our work is similar in style to the Video Rewrite system [5]. In that work Bregler et al. use existing footage to create new footage of person moving their mouth to words not in the original video. The system automatically labels phonemes in training video using an HMM. Given a new speech track, it is labeled, and the triphone videos (sets of 3 phonemes plus their corresponding frames) from the training video that best match the new speech are stitched together.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 1: Background subtraction: (a) input frame, (b) background image, (c) residual image, (d) matting using a threshold on (c), (e) blurring residual with an isotropic Gaussian, (f,g) residual gradient images (x and y) obtained by background subtraction in gradient domain, (h) residual image smoothes with anisotropic diffusion using residual gradients for conduction, (i) matting using improved residual followed by simple alpha estimation, (j) compositing into a novel background.

3 Matting and Stabilization

The first step in any image-based motion synthesis approach is extracting the desired object (human figure in our case) from the input video. For each frame we must compute an α matte indicating which pixels belong to the object and which to the background ($\alpha \in [0, 1]$). This process, known variously as digital matting, layer extraction or motion segmentation, is underconstrained and very difficult in the general case. Even with various simplifying assumptions (e.g. stationary camera, rigid objects, non-opaqueness, parametric motion model, etc.) the problem is still generally unsolved. In practice, people either make use of a very controlled environment ("blue screen" techniques), or user input (e.g. the trimap matte approaches such as Video Matting [6]). Our goal, however, is to be able to process large amounts of video automatically, and in a variety of different environments, therefore neither approach would be feasible. On the other hand, we are mostly interested in videos taken from stationary cameras (such as webcams), so we can estimate the background image and reduce the problem to that of *background subtraction*.

3.1 Background Subtraction

The idea behind classic background subtraction is simple: first estimate the background (typically by taking a temporal median of all frames in the video), and then subtract it from each frame. Thresholding the residual will produce a binary matte, classifying each pixel as either foreground or background. However, in practice things aren't so rosy, as the thresholding step will inevitably produce mistakes. There are many sources of potential error, including shadows, temporal variations in background, transparency, motion blur, camera ringing, etc. However, the main problem with background subtraction is much more prosaic – it is simply that, in all real situations, there will *always* be some foreground pixels that just look like the background. In other words, the segmentation problem cannot be solved locally, just at the pixel level, without considering more global information. One such source of information can be found by considering the input video in the gradient domain, where the information is carried not by pixel intensities, but by the strength of edges between image regions. Weiss [20] has demonstrated that background subtraction in the gradient domain can be used to remove shadows. Here we propose to perform background subtraction in em both image and gradient domains, as these two sources of information are complimentary. We then combine the results of the two using a process similar to anisotropic diffusion.

Figure 1 illustrates our algorithm. Given an input frame (a) and the precomputed background image (b), the classic background subtraction is to compute the residual image (c) which is then thresholded to produce a matte (d). Note that (d) has many holes where the foreground and the background are too similar. One can attempt to "fix up" these holes by performing morphological operations on the binary matte or by blurring the residual image (as in (e)). Unfortunately, these *ad hoc* methods will introduce other problems, producing blurry boundaries and merging regions. Now let us look at background subtraction in the gradient domain. The background x and y gradient images $\hat{I}_x$ and $\hat{I}_y$ are computed by taking the median in time of the gradients I_x and I_y in each frame. Then, for a given frame i, we subtract out the background gradients: $I'_{x,y} = I^i_{x,y} - \hat{I}_{x,y}$, provided that we don't introduce any new edges ($|I^i_{x,y}| - |\hat{I}_{x,y}| > 0$). The resulting I'_x and I'_y for frame in (a) are shown on (f) and (g). These are only the gradients belonging to the figure, all other gradients in the image are irrelevant for our task. Note that the holes in (d) don't have all of their edges lie along the figure gradients, meaning that they have been influenced by the gradients in the background. Our goal is to remove this influence in the image domain using the foreground information we obtained in the gradient domain. We can do this by smoothing the original residual image in a way that preserves only the foreground edges. Anisotropic diffusion [16], a popular method for smoothing images along, but not across, boundaries, can be modified for our purposes: the smoothing proceeds on the residual image, while the gradients (conduction coefficients) are taken from the foreground gradients. The resulting residual (h) can be cleanly thresholded to produce a much better matte (i). In fact, thresholding at two different different values will automatically generate a trimap which one can use to statistically estimate very accurate values for opacity α allowing for compositing such as in (j).

Of course, this approach does not completely solve the background subtraction problem. We have only dealt with background edges but did nothing about the *internal* foreground edges. In a way, we have reduced an arbitrary stationary background problem to a constant-value background problem. Our method performs well when there are more edges in the background than in the foreground. In particular, all sequences in this work showed quite substantial improvement in performance.

3.2 Figure Centering

After finding the figure in each frame, we still need to extract it for further processing. The extracted sequence should have the figure always in the center. We do this by convolving the α matte with a vertically-elongated Gaussian (roughly at the scale of a person) and finding the maximum response whose coordinates become the new center.

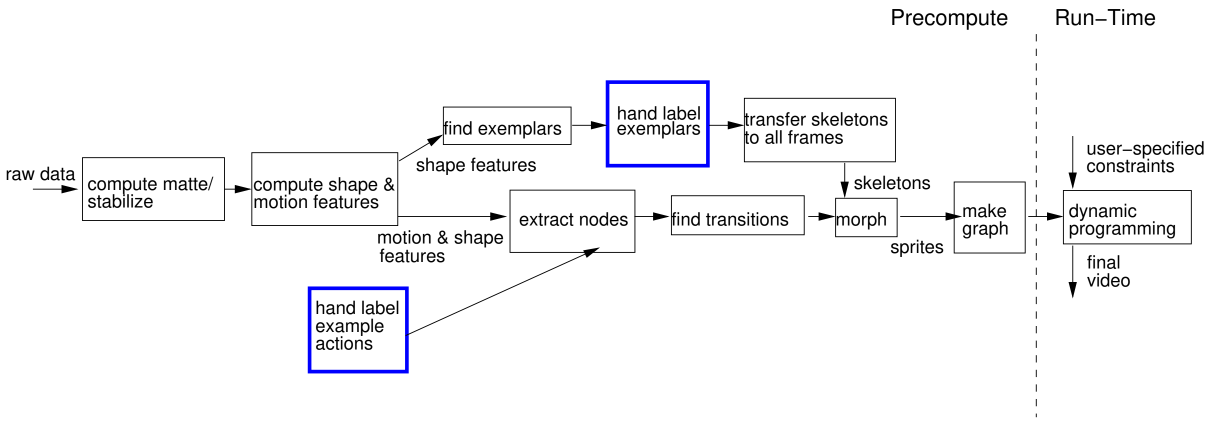


Figure 2: Flowchart

4 Finding Skeletons

In order to properly morph between similar images of the actor we need to know approximate locations for the body joints. The process we use for determining these locations is based on matching to *exemplars*. Given a large set of frames showing the actor in various poses a small set of exemplar frames is automatically extracted. The joints are then hand labeled in the exemplar frames. In order to find the locations of the joints in an input frame, the *closest* exemplar frame is found. The joints are then localized by finding the best matching locations for the closest exemplar's joints in the input frame.

The closest exemplar frame is found using a global description of the shape of the human figure in the entire frame, while the joints are localized using similar shape information which has been blurred in a novel manner. This procedure is similar in vein to our previous work on exemplars [15]. The main differences are that the shape feature used is more accurate (with an increased computational expense for matching) but only computed at the exemplar joint locations, no deformable model is used, and a single exemplar is chosen in advance of joint localization (to offset the increase in cost of matching).

In the following sections we first describe the shape descriptors used in selecting the closest exemplar and localizing the joint positions, and then develop a method for selecting a good set of exemplars that can be used to localize joint positions for a large collection of frames.

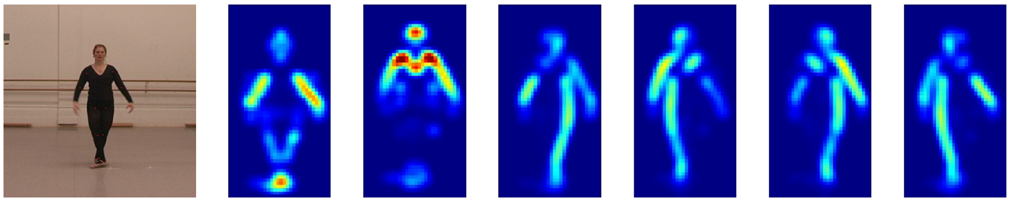
4.1 Shape Features

Finding feature correspondances between two images of an articulated object is a difficult task. Unless the two images are very similar, simple apperance-based methods such as corner features or optical flow (used in [19]) will not be effective, since apperance often changes dramatically over a small change in pose. Generally, it is the global shape, rather than local appearance, that can help us find correspondances between limb joints.

This global shape can be characterized with half-wave rectified, oriented filter responses. An example of these oriented filter responses is shown in Figure 3. An image is filtered with 3 oriented filters (horizontal, vertical, and 45 degrees). The outputs of these filters are split into positive only and negative only channels, resulting in 6 filter response images which will be used to construct our shape features.

The first, "whole body", shape feature, which is used in selecting the closest matching exemplar to a given input frame, is simply these 6 filter response images, blurred with a Gaussian. The exemplar frame with the highest correlation with the input frame in terms of this "whole body" shape feature is chosen.

For localizing the joints, we describe the shape at and around each of the chosen exemplar's joints and then search the input frame for the location where this shape feature matches best. Our second shape feature, which is used for this localization, is based on the description of Berg and Malik [3]. It takes these same 6 filter responses described above, crops them to fairly large area (about 1/3 of the image size) centered around the joint and applies a non-uniform *geometric blur*, a spatially varying smoothing filter that blurs the central region less than the periphery, producing a type of fish-eye lense effect. This descriptor is quite effective at finding approximate shape correspondances while ignoring local apperance changes, providing a rough idea of the overall body configuration. An example of the localization of body joints using this matching process is shown in Figure 4.



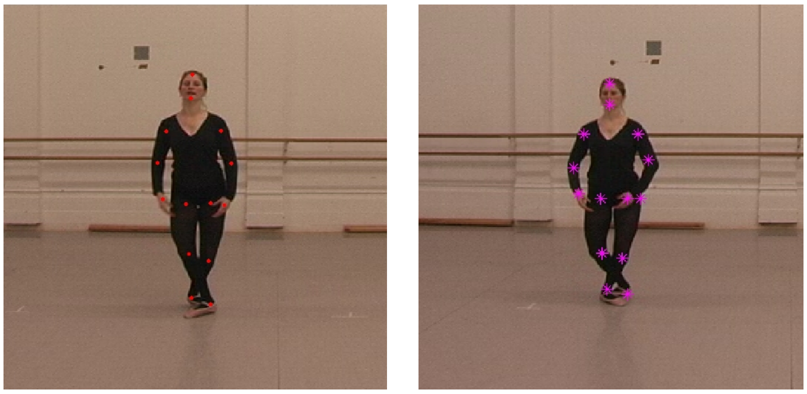
(a) (b)

Figure 3: (a) Exemplar frame with (b) the half-wave rectified filter responses used in constructing shape feature. The filter responses show 3 orientations of filter (horizontal, vertical, 45 degrees), with 2 images for each: half-wave rectified to contain positive only or negative only signal.

4.2 Selecting a Set of Exemplars

The exemplars should be a set of frames such that any frame for which a skeleton is required is similar enough to one of the exemplars, so that the exemplar can be warped to that frame. We would also like to ensure that the number of exemplars used is small, as user-interaction to supply the joint locations is required for each one.

We automatically select exemplars with a simple greedy algorithm. Each input frame of video is compared to the current set of exemplars using the "whole body" shape feature. If the frame is further than a fixed minimum distance from all of the



(a) (b)

Figure 4: (a) Closest matching exemplar image, with labeled joint locations. (b) Novel image, with each joint position localized using geometric blur of shape feature around joint from exemplar image.

exemplars, then it is added as an exemplar. With the minimum distance used in experiments a relatively small set of exemplars is selected. As a point of reference, in each of the genres we consider in this work we use fewer than 100 exemplars for approximately 20000 or 30000 frames. The exemplars extracted for the ballet dataset are shown in Figure 5.

5 Splicing and Morphing

The final synthesized videos that we produce will be obtained by splicing together clips of the original video. In order to make the transitions between the spliced clips appear natural we ensure that the trajectories of the actor's joints are smooth across the splice points. This requires procedures for: (i) locating the actor's joints, (ii) defining smoothly varying joint positions across the splice points, (iii) morphing the images of an actor to conform to target joint positions.

Simple procedures for splicing, such as a cross-fade of frames around the splice points, would lead to noticeable artifacts. The goal is to infuse the splicing procedure with knowledge about human figures to avoid the unnatural effects of naive techniques.

5.1 Morphing Articulated Figures

In this section we develop the notion of a "kinematically correct" morph of an image of a human figure. Given joint positions on a human figure in the image plane and a set of target joint positions, we morph the image such that the joint positions are moved to the desired taget positions. Figure 6 shows an example of this morphing, and a comparison to other naive techniques.

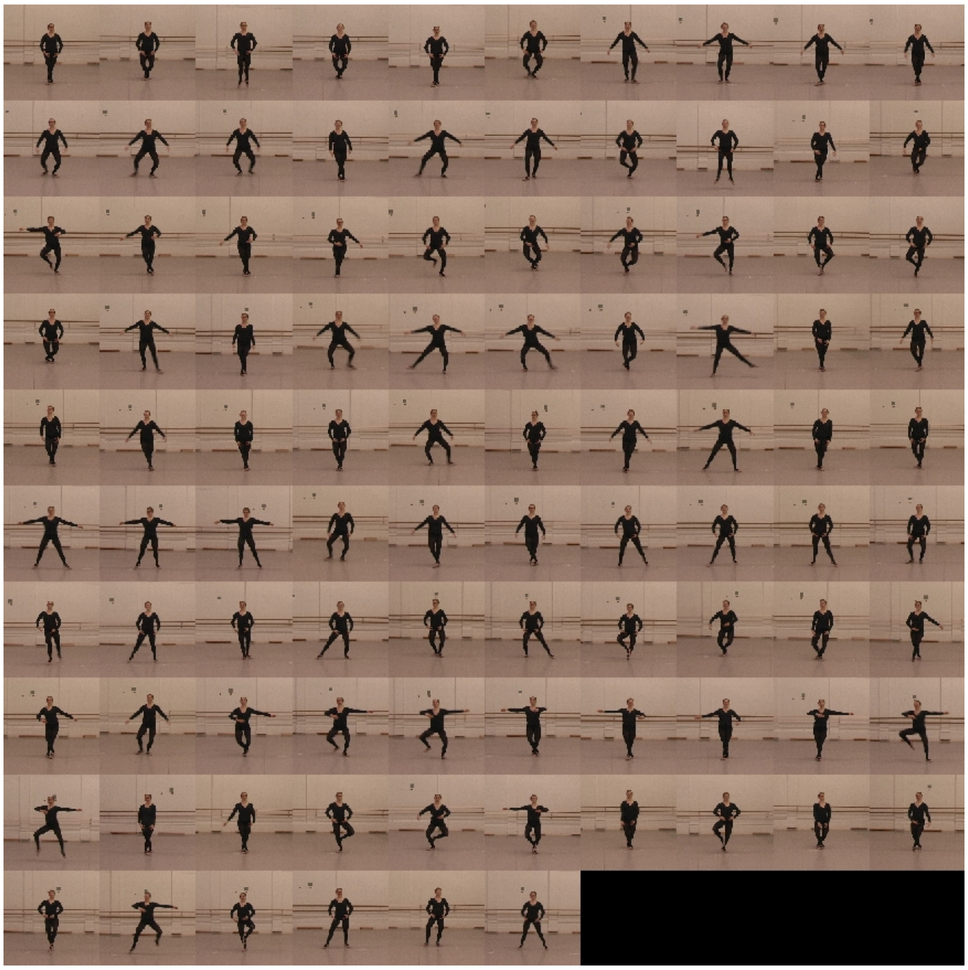
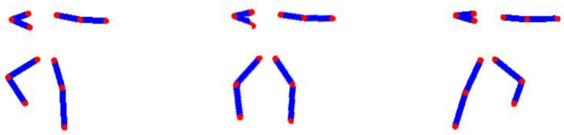


Figure 5: Exemplars for ballet dataset. These 96 exemplar frames were automatically extracted from the input video sequence. Human body joint locations were manually marked on these exemplar frames, and then used to automatically detect joint locations in the rest of the video sequence.

We perform this morphing in the 2d image plane. The model we use is a “cardboard person” model (Figure 7) consisting of a torso region and eight half-limbs (upper and lower arms and legs). Each half-limb has 2 degrees of freedom. Joints are allowed to rotate in 2d, and each half-limb may be scaled in length. Joint angles for elbows and knees are measured with respect to the adjacent hands/feet and shoulders/hips. Joint angles for shoulders and hips are measured with respect to the adjacent elbow/knee, and the shoulder/hip on the same side of the body. Note that there are singularities in this representation, for example when a standing human is viewed from above, however, they do not appear in the sequences we are interested in.

(a) (b) (c)



(d) (e) (f)

(g)

Figure 6: Morphing articulated figures. (a,c) Original frames. (b) Synthetic morphed frame halfway (in body parameters) between original frames. (d,f) Skeletons from original frames. (e) Target skeleton for morphed frame (b). (g) Cross-fade between original frames.

Given an image I of a human figure with joint positions $\phi(I)$, and target joint positions $\phi(T)$, we construct a morphed image with the human in I in configuration specified by $\phi(T)$. We start by assigning foreground pixels in I as belonging to a particular half-limb or the torso. This is accomplished by measuring shortest distance to bone-line of limbs. The torso warp is then modeled as a thin plate spline [4] deformation using the shoulders and hips as control points. This warp is applied to all foreground pixels belonging to the torso. Next, each limb is deformed to match the parameters in $\phi(T)$. The translation of the shoulder joint under the thin plate spline torso warp is applied to all pixels on the limb. Then, the correct rotation and scaling is applied to all foreground pixels in the upper half-limb. Lower half-limbs are deformed in first by the transformation of the upper limb, and then their own rotation and scaling.

After all foreground pixels have been deformed, interpolation is used to get values of pixels in the morphed image.

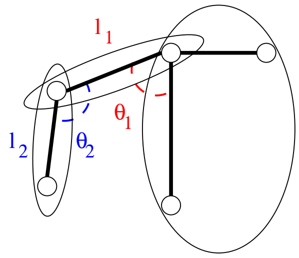


Figure 7: Torso and one limb of the cardboard person model. A human figure is modelled as a torso region along with 8 half-limbs (upper and lower arms and legs). Each half-limb has 2 degrees of freedom, allowing for variation in 2d angle and in length.

5.2 Sequence Splicing

With the procedure for kinematically correct warping in place, all that remains is to define the target joint positions for morphed frames that will give rise to a smooth change in the actor's joints. Skeletons are extracted for the frames on either side of the splice point. We then perform a linear blend of the parameters of our cardboard person model across the splice point. These blended parameter values become the target joint positions for use in the morphing procedure.

In concrete terms, suppose sequences of frames $S = \{s_1, s_2, ..., s_n\}$ and $R = \{r_1, r_2, ..., r_k\}$, with s_n and r_1 determined to corresponding frames, are to be spliced together. Then a new sequence Q will be created, consisting of frames:

$$Q = \{s_1, s_2, ..., s_{n-w-1}, m_{-w}, ..., m_w, r_{w+1}, ..., r_k\}$$

The m_i are the morphed frames created using the procedure described above. For values of i less than or equal to zero, a frame from S is used as the source of the morphing, otherwise, a frame from R. The target parameters $\phi(m_i)$ for the morphed frames are given by the linear blending:

$$\begin{aligned}
\phi(m_i) &= \phi(q) - sgn(i) \cdot \frac{w - |i|}{2w} \cdot (\phi(r_1) - \phi(s_n)) \\
\text{where } q &= \begin{cases} s_{n+i} & \text{if } i \leq 0 \\ r_i & \text{otherwise} \end{cases}
\end{aligned}$$

The parameter w controls the width of the morphing window. In our experiments w was set to 2. The linear blending process is illustrated in Figure 8.

6 Motion Graph

We organize frames in a *motion graph* similar to that used by Arikan and Forsyth. In our graph, a node n_i is a contiguous sequence of frames from the original video consisting largely of a single action a_i. There is a directed edge $e =< f_i, f_j, c_{ij} >$

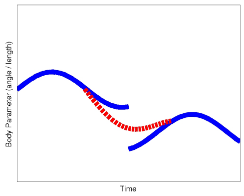


Figure 8: Linear interpolation of limb parameters. Blue curves denote single joint parameter values over time for two sequences to be spliced together. Dashed red curve shows linearly blended values used as target parameters for morphing.

between nodes n_i and n_j if it is possible to splice together the two sequences. Each edge stores f_i and f_j, the frames at which the splicing occurs, and c_{ij}, the cost of making this transition. The following sections describe how we construct this motion graph.

Before delving further into the details of constructing the graph, it is worth noting a couple of high level points. First is that the graphs we construct will be sparsely connected. Unlike motion capture, we do not have access to 3d joint positions. Realistic morphing of 2d images is more difficult and requires a pair of images to be rather similar in order to succeed. Transitions are rarer, and as such the graph representation with blocks of frames connected by an explicit set of transition points is a useful abstraction.

Second, empirically we found defining a “single action” a_i as a conjunction of “half-actions”, such as “move left - to - begin forehand” in tennis, to be advantageous. At first glance this distinction appears rather arbitrary, but there is an intuitive explanation why this yields better results.

Consider two possibilities for the action following “move left”: “forehand” and “move right”. The portion of motion at the end of the “move left” will be rather different in the two cases. In the former, the player will likely be turning his upper body and releasing the grip of his left hand on the racket in preparation for the coming forehand. In the latter, the player’s final step to the left will be quite different as he gathers his weight onto his right foot and prepares to push himself in the opposite direction.

However, the “middle” of actions, such as forehand strokes, are more likely to be similar irrespective of the preceeding action. Defining our nodes to explicitly contain the transitional periods between semantic actions, and searching for splice points in the middle of actions leads to more realistic synthetic videos. Note that with A unary action labels, taking conjunctions of actions leads to at worst A^2 different node types in our graph. However, in practice there will not be that many since the majority of these conjunctions are not possible.

6.1 Finding nodes

We find nodes for the motion graph by searching our footage for clips corresponding to a single action. Initially, the user specifies a set of actions that he is interested in modeling by providing example clips for each. The clips need not be of the same person and the labelling is very easy: one only needs to specify the start and end frames of the clip and the action label. Now, we need to automatically find similar actions in the novel footage. This is an action recognition problem and here we use the method of [8] for describing and comparing actions based on motion features. These motion features describe the coarse motion over a given spatio-temporal extent and is, in fact, very similar to the shape feature used earlier, except that frame-to-frame optical flow is used as the basic measurement instead of image gradients.

Figure 9 illustrates the search process. We start by computing a frame-to-frame similarity matrix between each of our hand labeled sequences and the block of the footage we wish to search for clips. Entry (i, j) in each similarity matrix W_k contains the correlation of motion features computed on frame i of the hand labeled sequence and frame j of the unclassified footage. This similarity matrix is blurred by filtering with the *blurry I* (see [8] for details). Next, we perform a version of a Hough transform. We tranform the matrix W_k into a vector H_k:

$$H_k(j) = \max_{i=\frac{-N}{2},\ldots,\frac{N}{2}} W_k(N/2 + i, i + j)$$

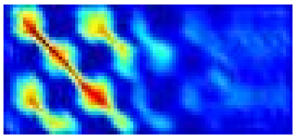
Each $H_k(j)$ represents the maximum value on the line of slope -1 passing through $(\frac{N}{2}, j)$, where N is the length of the labeled sequence. Intuitively, the entries in W_k "vote" for the center of the labeled sequence in the novel sequence.

We perform this voting procedure comparing all of the labeled sequences against the novel sequence to obtain a collection of vectors H_k. Nodes are extracted in a greedy fashion using these vectors. At each step we construct a node centered around the position of maximum value in the set of vectors. The node inherits the action label and temporal extent of the matched sequence – i.e. whose H_k contained that maximal value. New nodes must not overlap more than 30% with previous nodes, and must have motion similarity above a fixed threshold.

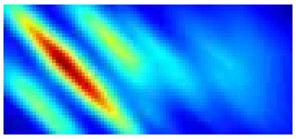
6.2 Finding Edges

In order to obtain smooth synthesized sequences, we must only place edges in our graph between very similar frames. In addition, we only allow incoming edges to go to the first third, and outgoing edges to leave the latter third of the frames in a node. This restriction ensures that a simple query procedure will allow for reasonable control over the animation.

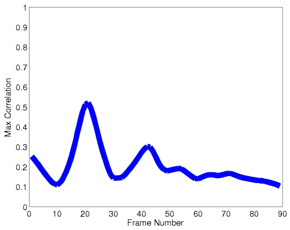
We use the motion and shape descriptors to determine whether a transition exists between a pair of frames. Since shape and motion need only be consistent locally when splicing together clips, we use a small 5-by-5 identity matrix for the filtering kernel. If thresholds (0.93 for shape, 0.4 for motion in both experiments) for similarity are exceeded, an edge is added between a pair of nodes. The cost on an edge is set as the negative of the shape similarity between the two frames it connects.



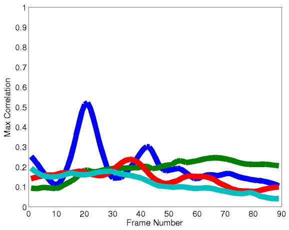
(a)



(b)



(c)



(d)

Figure 9: Classification of sequences. (a) Frame-to-frame similarity between novel sequence (horizontal axis) and one labeled sequence (vertical axis). Red values denote high similarity. (b) Blurred similarity matrix. (c) Hough transform of single similarity matrix, plot of j vs. $H_k(j)$. $H_k(j)$ is maximum values along lines with slope -1 through center row of similarity matrix, "votes" for center of labeled sequence in novel sequence. (d) Hough transforms of comparisons to collection of labeled sequences. Classification scheme chooses set of nodes from this collection via non-maximum suppression.

6.3 Querying the Motion Graph

We have constructed a motion graph where the nodes are single action clips, and the edges are transitions between them. Now, we can create a novel video corresponding to a particular sequence of actions (such as "backhand stroke", "move left", "lob stroke" in a graph of tennis actions) by finding a path in the graph passing through the correct types of nodes[1].

Given a user-specified sequence of actions to perform, we find the sequence with minimum cost transtions satisfying the desired sequence of actions. Since all constraints are local, we can find this best sequence using dynamic programming. The search takes $O(n^2m)$ time, where n is the number of nodes in the graph and m is the length (in number of actions) of the desired sequence.

This motion graph allows for intuitive synthesis of new sequences, but lacks some control over the resulting sequence. Previous work [1, 11] has shown how, given sufficient annotations, one could perform more complicated queries in such a graph.

[1] Defining nodes to be pairs of half-actions does not add any complexity to this process.

(a)	(b)
5^{th} position	
changement	
glissade right	
glissade left	
pas de chat right	stand
pas de chat left	left
pas de bouree right	right
pas de bouree left	forehand
pique right	backhand
pique left	slice
jete	lob
	serve

Figure 10: Action labels for (a) ballet and (b) tennis datasets.

7 Experiments

We applied our method to two domains, ballet and tennis. For each domain we filmed an amateur performing a few repetitions of loosely scripted actions. Figure 10 gives the list of the unary action types. In the ballet dataset we chose 11 actions (leading to 47 half-action pairs), each of which was performed 5-10 times. A total of 27148 frames of ballet video were recorded. For the tennis dataset we chose 8 actions (leading to 19 half-action pairs). Each of these actions was performed numerous times, particularly the forehand/backhand swings, and the movement actions. A total of 18845 frames of tennis video were recorded.

In the ballet dataset 453 nodes and 8022 edges between them were extracted using the algorithms in Section 6. In the tennis dataset 347 nodes and 1111 edges between them were extracted.

Synthetic sequences were generated by performing action-level queries on the motion graph. Sample frames from these synthetic sequences can be seen in Figures 11 and 12.

8 Conclusion

In this paper we have presented a method for creating synthetic videos of human motion by splicing together clips of existing footage. The key components of our method are using activity recognition to select appropriate clips of motion to render and an algorithm for kinematically correct morphing of human figures which is used to splice the clips together.

Our method produces reasonable quality videos, but noticeable artifacts do remain. The background subtraction algorithm has limitations. These could be remedied with user interaction, or the use of a “blue screen” in a studio environment. The largest problem is that the goal of smooth transitions between clips of original footage across

Figure 11: Example frames from synthetic ballet video. Frames 1,2 and 8 contain unmorphed actor from original video, simply translated across the frame. The next 5 frames show an example of a transition. These morphed frames smoothly vary across the the splice point, frame 5.

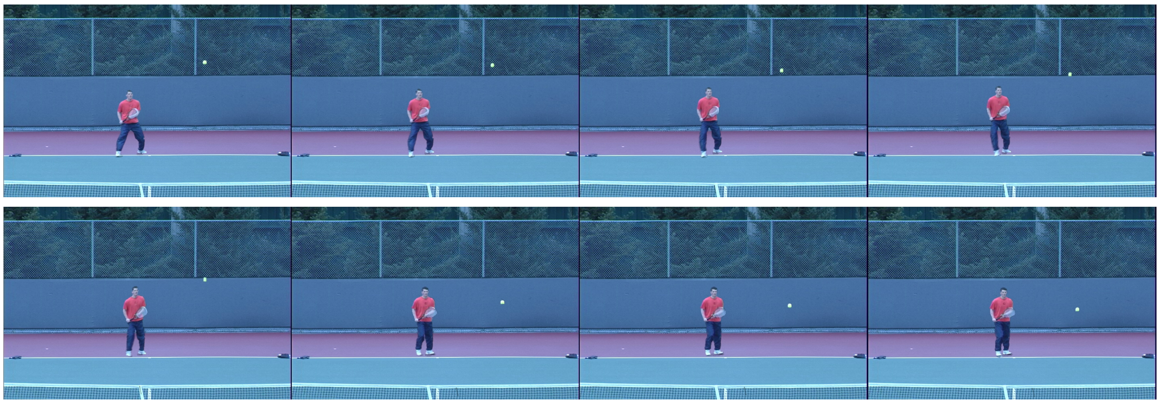


Figure 12: Example frames from synthetic tennis video. As before, frames 1,2 and 8 contain unmorphed actor from original video, simply translated across the frame. The next 5 frames show an example of a transition. These morphed frames smoothly vary across the the splice point, frame 5.

splice points remains elusive. A number of poor quality transitions exist in the video results, particularly in the ballet sequence when the figure is stationary. Transition points (edges in the motion graph) are placed at locations where the human figures are similar in terms of shape and motion. However, these are not necessarily the locations at which the splicing and morphing algorithms will produce a convincing output video. In particular, if the human figure is stationary the viewer is much less forgiving of small artifacts and jumps in the video. Moreover, the morphing algorithm is much more successful when the human figure is in certain body configurations. When the limbs are crossed or otherwise occluding each other, the morphing is much more difficult. The current measure used to determine transition points should be improved in future work. Actual information about the body joint locations and overall motion of the human figure should be used in more elaborate manner.

References

[1] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of SIGGRAPH 2002*, pages 483–490. ACM Press, 2002.

[2] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *ACM Trans. Graph.*, 22(3):402–408, 2003.

[3] A. Berg and J. Malik. Geometric blur for template matching. In *Proc. IEEE Comput. Soc. Conf. Comput. Vision and Pattern Recogn.*, pages 607–614, 2001.

[4] F. L. Bookstein. Principal warps: thin-plate splines and decomposition of deformations. *IEEE Trans. PAMI*, 11(6):567–585, June 1989.

[5] Christoph Bregler, Michele Covell, and Malcolm Slaney. Video rewrite: driving visual speech with audio. In *Proceedings of SIGGRAPH '97*, pages 353–360. ACM Press/Addison-Wesley Publishing Co., 1997.

[6] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H. Salesin, and Richard Szeliski. Video matting of complex scenes. In *Proceedings of SIGGRAPH 2002*, pages 243–248. ACM Press, 2002.

[7] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Proceedings of SIGGRAPH 96*, pages 11–20, 1996.

[8] A.A. Efros, A.C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *Proc. 9th Int. Conf. Computer Vision*, volume 2, pages 726–733, 2003.

[9] Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: assembling run-time animations. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 181–188. ACM Press, 2003.

[10] Jessica K. Hodgins, Wayne L. Wooten, David C. Brogan, and James F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH '95*, pages 71–78. ACM Press, 1995.

[11] Lucas Kovar, Michael Gleicher, and Frederic Pighin. Motion graphs. In *Proceedings of SIGGRAPH 2002*, pages 473–482. ACM Press, 2002.

[12] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of SIGGRAPH 2002*, pages 491–500. ACM Press, 2002.

[13] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of SIGGRAPH 2002*, pages 465–472. ACM Press, 2002.

[14] C. Karen Liu and Zoran Popović. Synthesis of complex dynamic character motion from simple animations. In *Proceedings of SIGGRAPH 2002*, pages 408–416. ACM Press, 2002.

[15] G. Mori and J. Malik. Estimating human body configurations using shape context matching. In *European Conference on Computer Vision LNCS 2352*, volume 3, pages 666–680, 2002.

[16] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *Proc. Int. Conf. Computer Vision*, pages 52–7, Osaka, Japan, Dec 1990.

[17] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: texturing and synthesis. In *Proceedings of SIGGRAPH 2002*, pages 501–508. ACM Press, 2002.

[18] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.

[19] Arno Schodl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *Proceedings of SIGGRAPH '00*, pages 489–498, 2000.

[20] Yair Weiss. Deriving intrinsic images from image sequences. In *Proc. 8th Int. Conf. Computer Vision*, pages 68–75, 2001.

[21] Andrew Witkin and Michael Kass. Spacetime constraints. In *Proceedings of SIGGRAPH '88*, pages 159–168. ACM Press, 1988.