

# Motion Capture Assisted Animation: Texturing and Synthesis

Katherine Pullen  
Stanford University

Christoph Bregler  
Stanford University

## Abstract

We discuss a method for creating animations that allows the animator to sketch an animation by setting a small number of keyframes on a fraction of the possible degrees of freedom. Motion capture data is then used to enhance the animation. Detail is added to degrees of freedom that were keyframed, a process we call texturing. Degrees of freedom that were not keyframed are synthesized. The method takes advantage of the fact that joint motions of an articulated figure are often correlated, so that given an incomplete data set, the missing degrees of freedom can be predicted from those that are present.

**CR Categories:** I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; J.5 [Arts and Humanities]: performing arts

**Keywords:** animation, motion capture, motion texture, motion synthesis

## 1 Introduction

As the availability of motion capture data has increased, there has been more and more interest in using it as a basis for creating computer animations when life-like motion is desired. However, there are still a number of difficulties to overcome concerning its use. As a result, most high quality animations are still created by keyframing by skilled animators.

Animators usually prefer to use keyframes because they allow precise control over the actions of the character. However, creating a life-like animation by this method is extremely labor intensive. If too few key frames are set, the motion may be lacking in the detail we are used to seeing in live motion (figure 1). The curves that are generated between key poses by computer are usually smooth splines or other forms of interpolation, which may not represent the way a live human or animal moves. The animator can put in as much detail as he or she wants, even to the point of specifying the position at every time, but more detail requires more time and effort. A second reason keyframing can be extremely labor intensive is that a typical model of an articulated figure has over 50 degrees of freedom, each of which must be painstakingly keyframed.

Motion capture data, on the other hand, provides all the detail and nuance of live motion for all the degrees of freedom of a character. However, it has the disadvantage of not providing for full

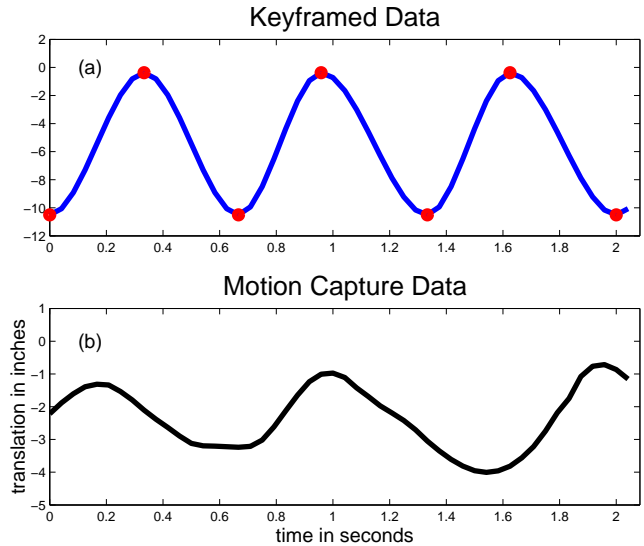


Figure 1: Comparison of keyframed data and motion capture data for root y translation for walking. (a) keyframed data, with keyframes indicated by red dots (b) motion capture data. In this example, the keyframed data has been created by setting the minimum possible number of keys to describe the motion. Notice that while it is very smooth and sinusoidal, the motion capture data shows irregularities and variations. These natural fluctuations are inherent to live motion. A professional animator would achieve such detail by setting more keys.

control over the motion. Motion capture sessions are labor intensive and costly, and if the actor does not do exactly what the animator had in mind or if plans change after the motion capture session, it can be difficult and time consuming to adapt the motion capture data to the desired application.

A more subtle problem with motion capture data is that it is not an intuitive way to begin constructing an animation. Animators are usually trained to use keyframes, and will often build an animation by first making a rough animation with few keyframes to sketch out the motion, and add complexity and detail on top of that. It is not easy or convenient for an animator to *start* creating an animation with a detailed motion he or she did not create and know every aspect of.

We propose a method for combining the strengths of keyframe animation with those of using motion capture data. The animator begins by creating a rough sketch of the scene he or she is creating by setting a small number of keyframes on a few degrees of freedom. Our method will use the information in motion capture data to add detail to the degrees of freedom that were animated if desired, a process we call adding “texture” to the motion. Degrees of freedom that were not keyframed at all are synthesized. The result is an animation that does exactly what the animator wants it to, but has the nuance of live motion.

## 2 Related Work

There has been a great deal of past research in a number of different areas that are related to our project. We divide this work into four main categories that are described below.

### 2.1 Variations in Animation

Many other researchers before us have made the observation that part of what gives a texture its distinctive look, be it in cloth or in motion, are variations within the texture. These variations are often referred to as noise, and one of the earliest papers to address this topic was in image texture synthesis, where random variability was added to textures with the Perlin-noise function [Perlin 1985]. These ideas were later applied to animations [Perlin and Goldberg 1996]. Other researchers have created motion of humans running using dynamical simulations [Hodgins et al. 1995] and applied hand crafted noise functions [Bodenheimer et al. 1999]. Statistical variations in motion were extracted directly from data by sampling kernel-based probability distributions in [Pullen and Bregler 2000]. Here we also create animations that exhibit natural variations, in this case because they are inherent to the fragments of motion capture data that we use in texturing and synthesis.

### 2.2 Signal Processing

There are a number of earlier studies in which researchers in both texture synthesis and motion studies have found it to be useful to look at their data in frequency space. In image texture synthesis, one of the earliest such approaches divided the data into multi-level Laplacian pyramids, and synthetic data were created by a histogram matching technique [Heeger and Bergen 1995]. This work was further developed by DeBonet [1997], in which the synthesis takes into account the fact that the higher frequency bands tend to be conditionally dependent upon the lower frequency bands.

We incorporate a similar approach, but applied to motion data. In animation, Unuma et al. [1995] use Fourier analysis to manipulate motion data by performing interpolation, extrapolation, and transitional tasks, as well as to alter the style. Bruderlin and Williams [1995] apply a number of different signal processing techniques to motion data to allow editing. Lee and Shin [2001] develop a multiresolution analysis method that guarantees coordinate invariance for use in motion editing operations such as smoothing, blending, and stitching. Our work relates to these animation papers in that we also use frequency bands as a useful feature of the data, but we use them to synthesize motion data.

### 2.3 Motion Editing

Many techniques have been proposed that start with existing motions, often obtained from motion capture data, and vary the motions to adapt to different constraints while preserving the style of the original motion. Witkin and Popovic [1995] developed a method in which the motion data is warped between keyframe-like constraints set by the animator. The spacetime constraints method originally created by Witkin and Kass [1988] was developed to allow the animator to specify constraints such as feet positions of a character, and then solve for these constraints by minimizing the difference from the original data [Gleicher 1997].

In further work, this method was applied to adapt a set of motion data to characters of different sizes [Gleicher 1998], and combined with a multiresolution approach for interactive control of the result [Lee and Shin 1999]. Physics were included in the method of Popovic and Witkin [1999], in which the editing is performed in a reduced dimensionality space.

We also are interested in adapting motion capture data to different situations. However, rather than starting with the live data, we start with a sketch created by the animator of what the final result should be, and fit the motion capture data onto that framework. As a result, it can be used to create motions substantially different from what was in the original data.

### 2.4 Style and synthesis

Numerous other projects besides ours have addressed the problem of synthesizing motions or altering pre-existing motions to have a particular style. A Markov chain Monte Carlo algorithm was used to sample multiple animations that satisfy constraints for the case of multi-body collisions of inanimate objects [Chenney and Forsyth 2000]. In work with similar goals to ours but applied to image-based graphics, other researchers [Schodl et al. 2000] develop the concept of a *video texture*, which enables a user to begin with a short video clip and then generate an infinite amount of similar looking video. Monte Carlo techniques are used to address the stochastic nature of the texture, and appropriate transitions are found in the motion to create a loop. The method was applied to example motions that contain both a repetitive and stochastic component, such as fire or a flag blowing in the wind.

In other projects, a common method of representing data has been to use mixtures of Gaussians and hidden Markov models. Bregler [1997] has used them to recognize full body motions in video sequences, and Brand [1999] has used them to synthesize facial animations from example sets of audio and video. Brand and Hertzmann [2000] have also used hidden Markov models along with an entropy minimization procedure to learn and synthesize motions with particular styles. Our approach differs in that we strive to keep as much of the information in the motion capture data intact as possible, by directly using fragments of real data rather than generalizing it with representations that may cause some of the fine detail to be lost. In other interesting work, Chi and her colleagues [Chi et al. 2000] presented work with similar goals to ours, in that they were seeking to create a method that allows animators to enhance the style of pre-existing motions in an intuitive manner. They made use of the principles of Laban Movement Analysis to create a new interface for applying particular movement qualities to the motion.

More recently, there have been a number of projects aimed toward allowing an animator to create new animations based on motion capture data. For example, in the work of Li et al. [2002], the data was divided into motion textons, each of which could be modelled by a linear dynamic system. Motions were synthesized by considering the likelihood of switching from one texton to the next. Other researchers developed a method for automatic motion generation at interactive rates [Arikan and Forsyth 2002]. Here the animator sets high level constraints and a random search algorithm is used to find appropriate pieces of motion data to fill in between. In closely related work, the concept of a *motion graph* is defined to enable one to control a character's locomotion [Kovar et al. 2002]. The motion graph contains original motion and automatically generated translations, and allows a user to have high level control over the motions of the character. In the work of [Lee et al. 2002], a new technique is developed for controlling a character in real time using several possible interfaces. The user can choose from a set of possible actions, sketch a path on the screen, or act out the motion in front of a video camera. Animations are created by searching through a motion data base using a clustering algorithm. Any of the above techniques would be more appropriate to use than ours in the case where the user has a large data base of motions and wants high level control over the actions of the character. Our project is geared more toward an animator who may have a limited set of data of a particular style, and who wants to have fine control over the motion using the familiar tools of keyframing.

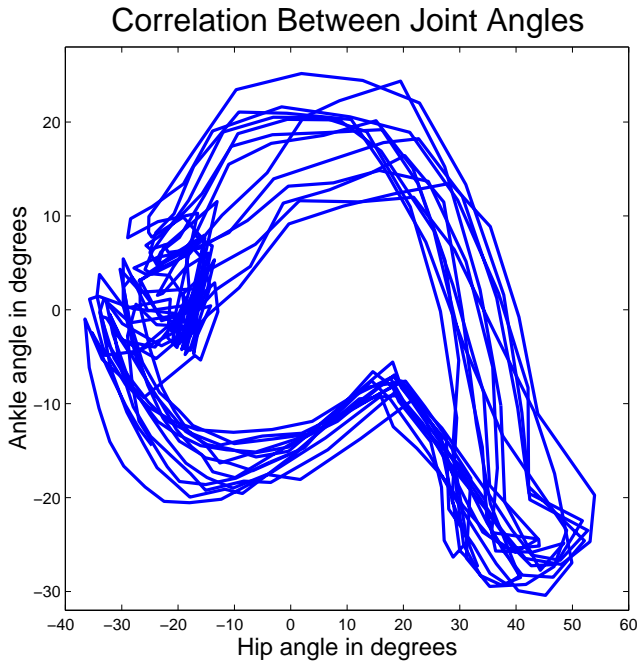


Figure 2: Correlation between joint angles. Shown is the ankle angle versus the hip angle for human walking data. The fact that this plot has a definite form demonstrates that the angles are related to each other.

### 3 Methods

In human and animal motion, there are many correlations between joint actions. These correlations are especially clear for a repetitive motion like walking. For example as the right foot steps forward, the left arm swings forward, or when the hip angle has a certain value, the knee angle is most likely to fall within a certain range. We can see those correlations graphically with a plot such as that shown in figure 2, where we plot the ankle angle as a function of hip angle for some human walking data. The fact that the plot has a specific shape, a skewed horseshoe shape in this case, indicates that there is a relationship between the angles.

These relationships hold true for more complex motions as well, but may be more local in time, specific to a particular action within a motion data set. In our method we take advantage of these relationships to synthesize degrees of freedom that have not been animated. Similarly, we can add detail to a degree of freedom that has been animated by synthesizing only the higher frequency bands, a process we refer to as texturing.

The animator must provide the following information: (1) which joint angles should be textured (2) which joint angles should be synthesized (3) which joint angles should be used to drive the motion in each case. For example, suppose an animator sketches out a walk by animating only the legs and wants to synthesize the upper body motions. A good choice for the angles to drive the animation would be the hip and knee x angles (where we define the x axis as horizontal, perpendicular to the direction of walking) because they define the walking motion. These data are broken into fragments, and used to find fragments of the motion capture data with hip and knee x angles similar to what has been created by keyframing. The corresponding fragments of motion capture data for the upper body motion can then be used to animate the upper body of the computer character.

To achieve this task, we require a method to determine what con-

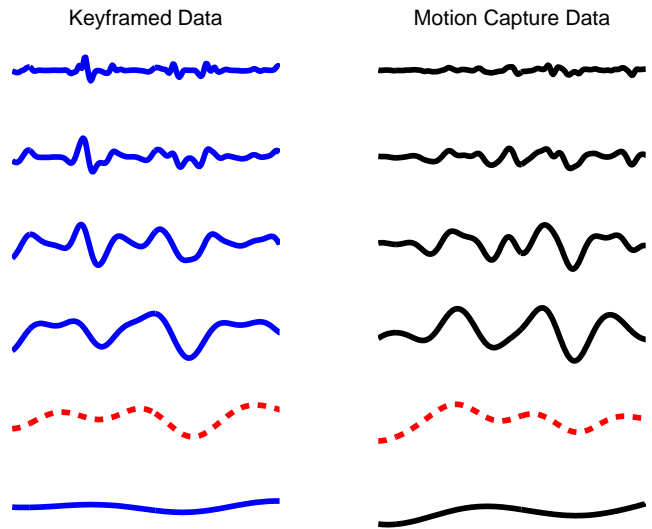


Figure 3: Frequency analysis. Shown are bands 2-7 of a Laplacian Pyramid decomposition of the left hip angle for dance motions from both keyframing and motion capture. One band, shown with a red dashed line, is chosen for the matching step

stitutes a matching region of data. The problem is complicated by the fact that the keyframed data may be of a different time scale from the real data. In addition, the ends of the fragments we choose must join together smoothly to avoid high frequency glitches in the motion. We address these issues in our method, which we divide into the following steps: (1) frequency analysis (2) matching (3) path finding and (4) joining.

In the following explanation, we will use the example of using the left hip and left knee x angles to synthesize upper body motions. These degrees of freedom will be referred to as the matching angles. Also note that we define “keyframed data” as the data at every time point that has been generated in the animation after setting the keyframes. An example of such data is in figure 1a.

#### 3.1 Frequency Analysis

In order to separate different aspects of the motion, the first step is to divide the data (both keyframed and motion capture) into frequency bands (figure 3). For a joint that has already been animated, we may only want to alter the mid to high frequency range, leaving the overall motion intact. For a degree of freedom that has not been animated, we may wish to synthesize all of the frequency bands.

#### 3.2 Matching

Matching is at the heart of our method. It is the process by which fragments of data from the keyframed animation are compared to fragments of motion capture data to find similar regions. To begin this step, a low frequency band of one of the joints is chosen, in our example the left hip angle. The results are not highly dependent upon which frequency band is chosen, as long as it is low enough to provide information about the overall motion. For example in figure 3 we illustrate choosing band 6 of the Laplacian pyramid, but choosing band 4 or 5 also yields good results. Band 7 is too low, as can be seen by the lack of structure in the curve, and band 3 is too high, as it does not reflect the overall motion well enough.

We find the locations in time where the first derivative of the chosen band of one of the matching angles changes sign. The real and keyframed data of all of the matching angles of that band (the

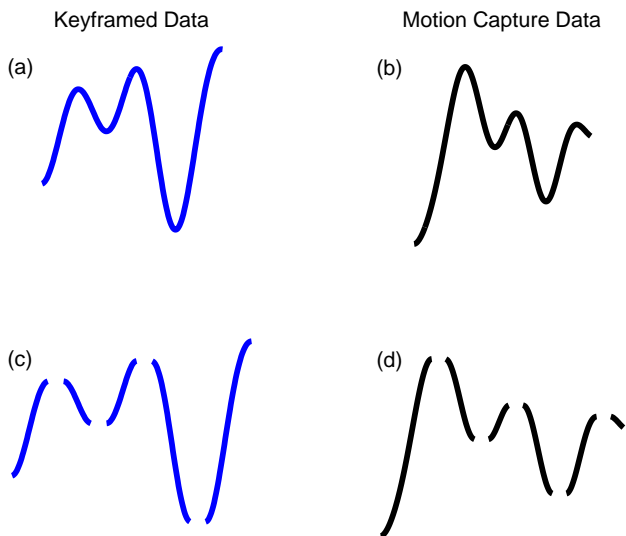


Figure 4: Breaking data into fragments. The bands of the keyframed data and motion capture data shown with red dashed lines in figure 3 are broken into fragments where the sign of the first derivative changes. (a) keyframed data. (b) motion capture data. (c) keyframed data broken into fragments. (d) motion capture data broken into fragments.

left hip and left knee x angles, in our example) are broken into fragments at those locations (figure 4). Note that in the figures we illustrate the process for just one of the matching angles, the hip, but actually the process is applied to all of the matching angles simultaneously. We also match the first derivative of the chosen band of each of these angles. Including the first derivatives in the matching helps choose fragments of real data that are more closely matched not only in value but in dynamics to the keyframed data. Note that the sign change of the first derivative of only *one* of the angles is used to determine where to break all of the data corresponding to the matching angles into fragments, so that all are broken at the same locations.

All of the fragments of keyframed data in the chosen frequency band and their first derivatives are stepped through one by one, and for each we ask which fragment of real data is most similar (figure 5). To achieve this comparison, we stretch or compress the real data fragments in time by linearly resampling them to make them the same length as the keyframed fragment. In the motion capture data, there are often unnatural poses held for relatively long periods of time for calibration purposes. To avoid choosing these fragments, any real fragment that was originally more than 4 times as long as the fragment of keyframed data being matched is rejected. We find the sum of squared differences between the keyframed fragment being matched and each of the real data fragments, and keep the  $K$  closest matches. As we save fragments of the matching angles, we also save the corresponding fragments of original motion capture data (not just the frequency band being matched) for all of the angles to be synthesized or textured (figure 6).

At this point, it is sometimes beneficial to include a simple scale factor. Let  $A$  be the  $m \times n$  matrix of values in the keyframed data being matched, where  $m$  is the number of matching angles and  $n$  is the length of those fragments. Let  $M$  be the  $m \times n$  matrix of one of the  $K$  choices of matching fragments. Then to scale the data, we look for the scale factor  $s$  that minimizes  $\|Ms - A\|$ . The factor  $s$  is then multiplied by all of the data being synthesized. In practice such a scale factor is useful only in a limited set of cases, because it assumes a linear relationship between the magnitude of the match-

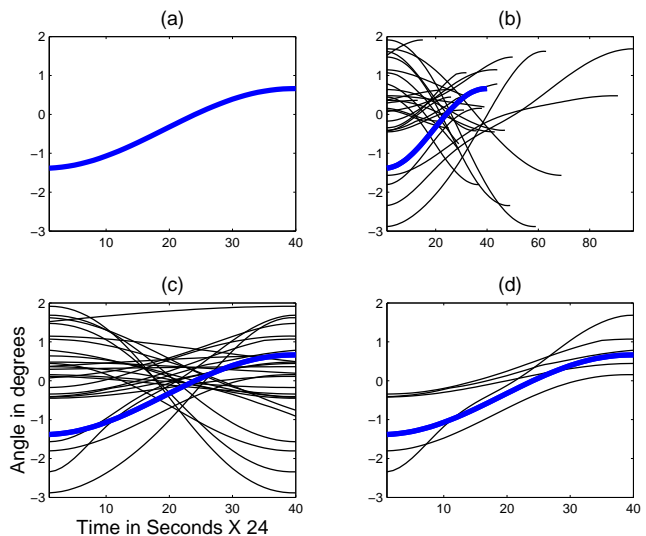


Figure 5: Matching. Each keyframed fragment is compared to all of the motion capture fragments, and the  $K$  closest matches are kept. Shown is the process of matching the first fragment shown in figure 4 (c). (a) The keyframed fragment to be matched. (b) The keyframed fragment, shown in a thick blue line, compared to all of the motion capture fragments, shown in thin black lines. (c) Same as (b), but the motion captured fragments have been stretched or compressed to be the same length as the keyframed fragment. (d) Same as (c), but only the 5 closest matches are shown.

ing angles and the magnitude of the rest of the angles, which is not usually likely to be true. However, it can improve the resulting animations for cases in which the keyframed data is similar to the motion capture data, and the action is fairly constrained, such as walking.

More fragments than just the closest match are saved because there is more to consider than just how close the data fragment is to the original. We must take into consideration which fragments come before and after. We would like to encourage the use of consecutive chunks of data as described in the next section.

### 3.3 Path finding

Now that we have the  $K$  closest matches for each fragment, we must choose a path through the possible choices to create a single data set. The resulting animation is usually more pleasing if there are sections in time where fragments that were consecutive in the data are used consecutively to create the path. As a result, our algorithm considers the neighbors of each fragment, and searches for paths that maximize the use of consecutive fragments.

For each join between fragments, we create a cost matrix, the  $ij$ th component of which gives the cost for joining fragment  $i$  with fragment  $j$ . A score of zero is given if the fragments were consecutive in the original data, and one if they were not. We find all of the possible combinations of fragments that go through the points of zero cost.

This technique is easiest to explain using an example, which is diagrammed in figure 7. Suppose we had 4 fragments of synthetic data to match, and saved 3 nearest matches. In the illustration we show that for fragment 1 of the keyframed data, the best matches were to fragments 4, 1, and 3 of the real data, and for fragment 2 of the keyframed data the closest matches were to fragments 5, 7, and 2 of the real data, and so on. We have drawn lines between fragments to indicate paths of zero cost. Here there are three best

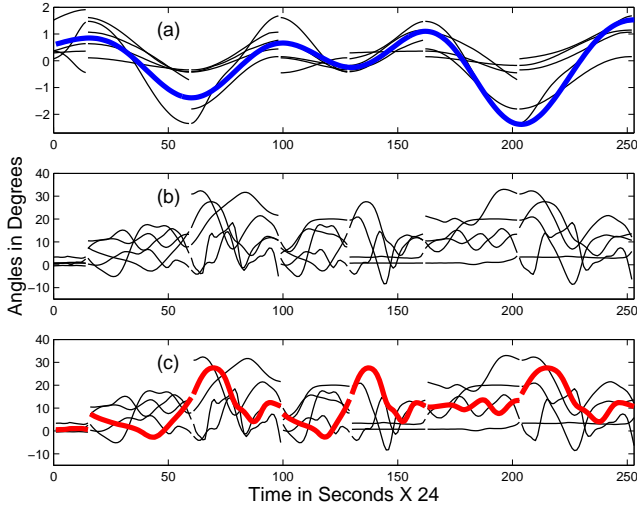


Figure 6: Matching and synthesis. (a) The five closest matches for a series of fragments of keyframed data is shown. The keyframed data is shown with a thick blue line, the matching motion capture fragments are shown with thin black lines. (b) An example of one of the angles being synthesized is shown, the lowest spine joint angle rotation about the x axis. The five fragments for each section come from the spine motion capture data from the same location in time as the matching hip angle fragments shown in plot (a). (c) An example of a possible path through the chosen spine angle fragments is shown with a thick red line.

choices. One is fragment 4, 5, 6, and 2 from the real data. In this case we choose fragment 2 of the real data to match the fourth fragment of keyframed data rather than 8 or 5 because it was originally the closest match. A second possible path would be 4, 5, 4, and 5, and a third would be 1, 2, 4, 5. All three would yield two instances of zero cost. An example of an actual path taken through fragments chosen by matching is shown in figure 6c.

Note that for  $z$  instances of zero cost, there can be no greater than  $z$  paths to consider, and in fact will usually be far less because the instances can be linked up. In our example (figure 7) there were four instances of zero cost, but only three possible paths that minimize the cost. The  $P$  best paths (where  $P$  is a parameter set by the animator) are saved for the animator to look at. All are valid choices and it is an artistic decision which is best.

In practice we found that saving roughly 1/10 the total number of fragments produced good results. Saving too many matches resulted in motions that were not coordinated with the rest of the body, and saving too few did not allow for sufficient temporal coherence when seeking a path through the fragments.

### 3.4 Joining

Now that we have the best possible paths, the ends may still not quite line up in cases where the fragments were not originally consecutive. For example, in figure 6c we show an example of data after matching and choosing the paths. To take care of these discontinuities, we join the ends together by the following process.

For fragment  $i$ , we define the new endpoints as follows. The new first point will be the mean between the first point of fragment  $i$  and the last point of fragment  $i - 1$ . (Note that there is overlap between the ends of the fragments; if the last point of fragment  $i$  is placed at time  $t$ , the first point of fragment  $i + 1$  is also at time  $t$ .) The new last point of fragment  $i$  will be the mean between the last point of fragment  $i$  and the first point of fragment  $i + 1$ .

Keyframed Fragment	1	2	3	4
Matching Data Fragments	4 — 5 1 — 7 3 — 2	5 — 9 7 — 4 2 — 6	9 — 4 4 — 6	2 — 8 8 — 5
Cost Matrices	$\begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$	

Figure 7: Choosing a path by maximizing the instances of consecutive fragments. In the table we show a hypothetical example of a case where four keyframed fragments were matched, and the  $K = 3$  closest matches of motion capture fragments were kept for each keyframed fragment. The matches at the tops of the columns are the closest of the 3 matches. Blue lines are drawn between fragments that were consecutive in the motion capture data, and the cost matrices between each set of possible matches are shown below.

The next step is to skew the fragment to pass through the new endpoints. To achieve this warping, we define two lines, one that passes through the old endpoints, and one that passes through the new endpoints. We subtract the line that passes through the old endpoints and add the line that passes through the new endpoints to yield the shifted fragment. The process is diagrammed in figure 8.

In order to further smooth any remaining discontinuity, a quadratic function is fit to the join region from  $N$  points away from the joint point to within 2 points of the joint point, where  $N$  is a parameter. A smaller value of  $N$  keeps the data from being altered too greatly from what was in the motion capture data, and a larger value more effectively blends between different fragments. In practice we found a  $N$  from 5-20 to be effective, corresponding to 0.2 - 0.8 seconds. The resulting quadratic is blended with the original joined data using a sine squared function as follows.

Define the blend function  $f$  as

$$f = (\cos \frac{\pi t}{2N} + 1)^2 \quad (1)$$

where  $N$  is half the length of the shortest of the fragments that are to be joined and  $t$  is the time, shifted to be zero at the join region. If we define  $q$  as the quadratic function we obtained from the fit, and  $m$  as the data after matching, then the data  $s$  after smoothing is

$$s(t) = f(t)q(t) + (1 - f(t))m(t). \quad (2)$$

An example of this process is shown in figure 9.

## 4 Experiments

We tested our method on several different situations, three of which are described below. All of these examples are presented on the accompanying video tape.

### 4.1 Walking

A short animation of two characters walking toward each other, slowing to a stop, stomping, and crouching was created using

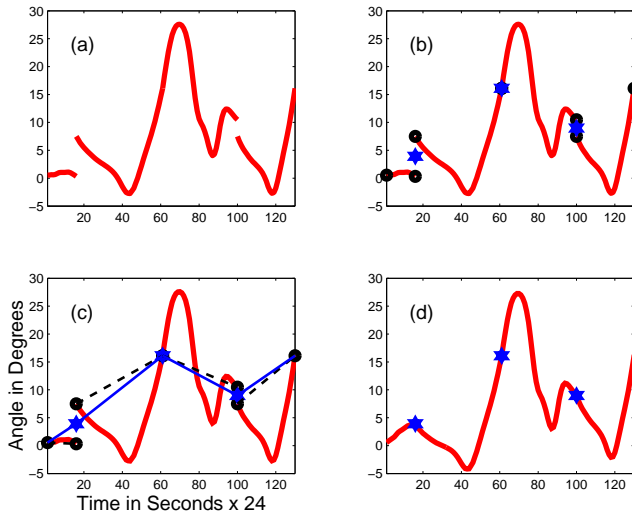


Figure 8: Joining the ends of selected fragments. (a) Four fragments of spine angle data that were chosen in the matching step are shown. Note this graph is a close up view of the first part of the path illustrated in figure 6c. There are significant discontinuities between the first and second fragments, as well as between the third and fourth. (b) The original endpoints of the fragments are marked with black circles, the new endpoints are marked with blue stars. The second and third fragments were consecutive in the motion capture data, so the new and old endpoints are the same. (c) For each fragment, the line between the old endpoints (black dashes) and the line between the new endpoints (blue solid line) are shown. (d) For each fragment, the line between the old endpoints is subtracted, and the line between the new endpoints is added, to yield the curve of joined fragments. The new endpoints are again marked with blue stars.

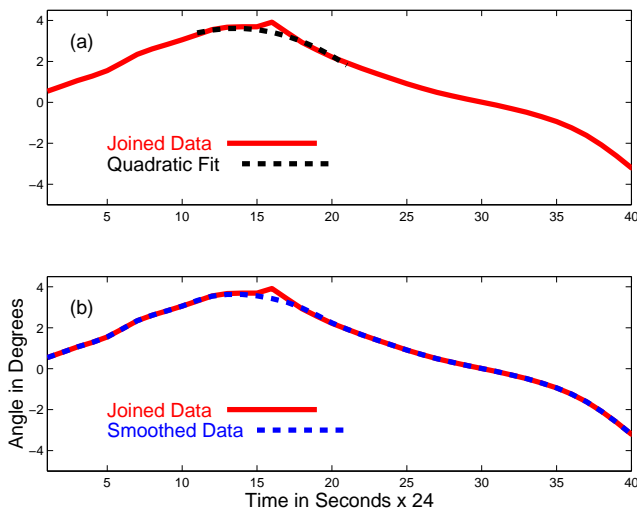


Figure 9: Smoothing at the join point. A close up of the join between fragments 1 and 2 from figure 8 is shown with a red solid line. (a) The quadratic fit using the points on either side of the join point (as described in the text) is shown with a black dashed line. (b) The data after blending with the quadratic fit is shown with a blue dashed line.

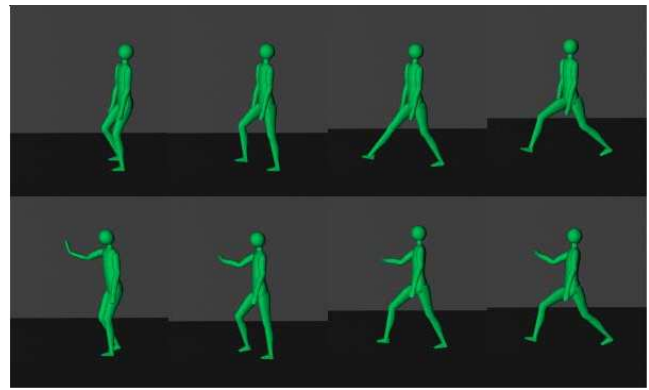


Figure 10: Example frames from the walking animations. On the top row are some frames from the keyframed sketch, and on the bottom row are the corresponding frames after enhancement.

keyframes. Keyframes were set only on the positions of the root (not the rotations) and feet. Inverse kinematics were used on the feet at the ankle joint, as is customary in keyframe animation of articulated figures. The joint angles for the hips and knees were read out afterwards for use in texturing and synthesis.

Each character’s motion was enhanced using a different motion capture data set of walking motion. The two data sets each consisted of walking with roughly a single step size, but each exhibited a very different style of walking. One was a relatively “normal” walk, but rather bouncy, and the other was of a person imitating a drag queen and was quite stylized, containing unusual arm and head gestures. The length of each data set was 440 time points at 24 frames per second, or about 18 seconds worth of data. A Laplacian pyramid was used for the frequency analysis. The 4th highest band was used for matching. For texturing, bands 2-3 were synthesized, and for synthesis, all bands 2 and lower. The very highest frequency band tended to add only undesirable noise to the motion.

The upper body degrees of freedom could successfully be synthesized using a number of different combinations for the matching angles. For example, both hip x angles; the left hip x and left knee x angle; or the right hip x and right knee x all gave good results. The most pleasing results were obtained by using data from the left hip and left knee x angles during the stomp (the character stomps his left foot) and data from both hips for the rest of the animation. Scaling after matching also improved the results in this case, for example when the character slows down and comes to a stop, scaling caused the motion of the body and arm motions to reduce in coordination with the legs.

The method does not directly incorporate hard constraints, so we used the following method to maintain the feet contact with the floor. First the pelvis and upper body motions were synthesized. Since altering the pelvis degrees of freedom causes large scale motions of the body, inverse kinematic constraints were subsequently applied to keep the feet in place on the floor. This new motion was used for texturing the lower body motion during times the feet were not in contact with the floor.

The motion of the characters was much more life-like after enhancement. The upper body moved in a realistic way and responded appropriately to the varying step sizes and the stomp, even though these motions were not explicit in the motion capture data. In addition, the style of walking for each character clearly came from the data set used for the enhancement. Some example frames are shown in figure 10.

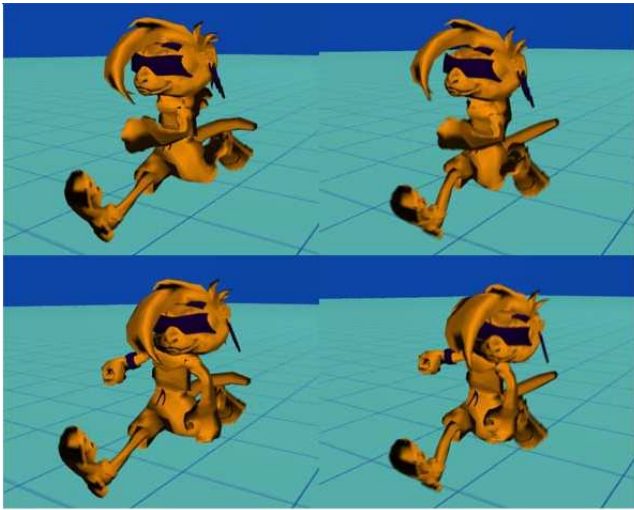


Figure 11: Example frames from animations of the otter character. On the top row are some frames from the original keyframed animation, while on the bottom are the corresponding frames after texturing.

## 4.2 Otter Character

Although we have focussed on the idea of filling in missing degrees of freedom by synthesis or adding detail by texturing, the method can also be used to alter the style of an existing animation that already has a large amount of detail in it.

To test this possibility, we used an otter character that had been animated by keyframe animation to run. Using the motion capture sets of walking described above, we could affect the style of the character's run by texturing the upper body motions, using the hip and knee angles as the matching angles. The effect was particularly noticeable when using the drag queen walk for texturing, the otter character picked up some of the head bobs and asymmetrical arm usage of the motion capture data. Some example frames are shown in figure 11.

## 4.3 Modern Dance

In order to investigate a wider range of motions than those related to walking or running, we turned to modern dance. Unlike other forms of dance such as ballet or other classical forms, modern does not have a set vocabulary of motions, and yet it uses the whole body at its full range of motion. Thus it provides a situation where the correlations between joints will exist only extremely locally in time, and a stringent test of our method.

A modern dance phrase was animated by sketching only the lower body and root motions with keyframes. Motion capture data of several phrases of modern dance was collected, and a total of 1097 time points (24 frames per second) from 4 phrases was used. The upper body was synthesized, and the lower body textured. The same method for maintaining feet contact with the floor that was described above for the walking experiments was used here. The frequency analysis was the same as for the walking, except that the 6th highest band was used for matching. A lower frequency band was used because the large motions in the dance data set tended to happen over longer times than the steps in walking.

The results were quite successful here, especially for synthesis of the upper body motions. The motions were full and well coordinated with the lower body, and looked like something the dancer who performed for the motion capture session *could* have done, but

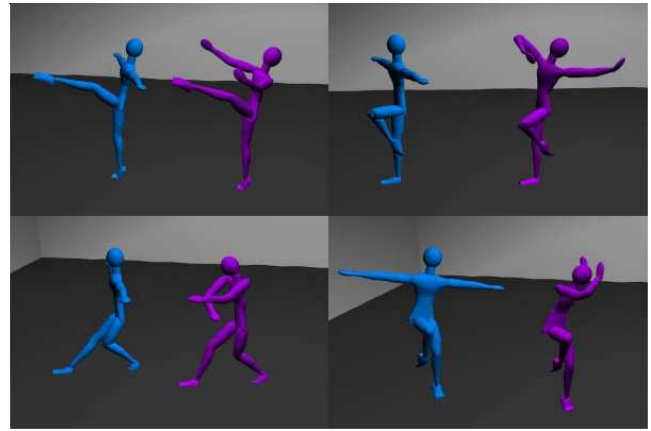


Figure 12: Example frames from the dance animations. The blue character, on the left in each image, represents the keyframed sketch. The purple character, on the right in each image, shows the motion after enhancement.

did not actually do. Some example frames are shown in figure 12. The best results were obtained by using all of the hip and knee angles as the matching angles, but some good animations could also be created using fewer degrees of freedom. In these experiments, the effects of choosing different paths through the matched data became especially noticeable. Because of the wide variation within the data, different paths yielded significantly different upper body motions, all of which were well coordinated with the lower body.

## 5 Conclusion and Future Work

Presently the two main methods by which the motions for computer animations are created are by using keyframes and by using motion capture. The method of keyframes is labor intensive, but has the advantage of allowing the animator precise control over the actions of the character. Motion capture data has the advantage of providing a complete data set with all the detail of live motion, but the animator does not have full control over the result. In this work we present a method that combines the advantages of both methods, by allowing the animator to control an initial rough animation with keyframes, and then fill in missing degrees of freedom and detail using the information in motion capture data. The results are particularly striking for the case of synthesis. One can create an animation of only the lower body, and given some motion capture data, automatically create life-like upper body motions that are coordinated with the lower body.

One drawback of the method as it currently stands is that it does not directly incorporate hard constraints. As a result the texturing cannot be applied to cases where the feet are meant to remain in contact with the floor, unless it were combined with an inverse kinematic solver in the animation package being used. Currently we are working to remedy this deficiency.

Another active area of research is to determine a more fundamental method for breaking the data into fragments. In this work we used the sign change of the derivative of one of the joint angles used for matching, because it is simple to detect and often represents a change from one movement idea to another. The exact choice of where to break the data into fragments is not as important as it may seem. What *is* important is that both the keyframed and real data are broken at analogous locations, which is clearly the case with our method. The method could be made more efficient by detecting more fundamental units of movement that may yield larger fragments. However, due to the complexity of human motion, this

problem is a challenging one, and an ongoing topic of research.

On the surface, another drawback of the method may appear to be the need to choose a particular frequency band for the matching step. However, the choice is not difficult to make, and in fact the results are not highly dependent upon the choice. Any low frequency band that provides information about the overall motion will provide good results. The resulting animations sometimes vary from one another depending upon which frequency band is chosen as slightly different regions of data are matched, but more often they are quite similar. If too high a band is chosen, however, the resulting animation has an uncoordinated look to it, as the overall motion is not accurately represented in the matching step.

Similarly, another drawback of the method may appear to be that the animator must specify which degrees of freedom to use as the matching angles. However, if one has spent some time keyframing a character, it is quite easy in practice to specify this information. The most simplistic approach is to simply use all of the degrees of freedom that the animator has sketched out the motion for with keyframes. In many cases, however, fewer degrees of freedom can be specified, and equally good results can be obtained. If the motion has less variation, such as walking, the results will still be pleasing if fewer angles are chosen as the matching angles. In fact it is fascinating how correlated the motions of the human body are. Given only the data in two angles, such as the hip and knee x angles of one leg, one can synthesize the rest of the body. However, for a motion with more variation in it, such as dancing, it is better to include more angles, to ensure good choices during matching. If fewer joints are used for matching in this case, some of the resulting paths may still be good results, but others may appear somewhat uncoordinated with the full body.

In fact, the goal of this project was not to create a completely automatic method, but to give the animator another tool for incorporating the information in motion capture data into his or her creations. Different choices of the matching angles can yield different results and provide the animator with different possibilities to use in the final animation. Another source of different motions comes from examining different paths through the best matches. The animator has the option of looking at several possibilities and making an artistic decision which is best. Ultimately we hope that methods such as this one will further allow animators to take advantage of the benefits of motion capture data without sacrificing the control they are used to having when keyframing.

## 6 Acknowledgments

Special thanks to Rearden Steel studios for providing the motion capture data, and to Electronic Arts for providing the otter model.

## References

- ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. *Proc. SIGGRAPH 2002*.
- BODENHEIMER, B., SHLEYFMAN, A., AND HODGINS, J. 1999. The effects of noise on the perception of animated human running. *Computer Animation and Simulation '99, Eurographics Animation Workshop* (September), 53–63.
- BONET, J. S. D. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. *proc. SIGGRAPH 1997*, 361–368.
- BRAND, M., AND HERTZMANN, A. 2000. Style machines. *proc. SIGGRAPH 2000*, 183–192.
- BRAND, M. 1999. Voice puppetry. *Proc. SIGGRAPH 1999*, 21–28.

- BREGLER, C. 1997. Learning and recognizing human dynamics in video sequences. *Proc. CVPR*, 569–574.
- BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. *proc. SIGGRAPH 1995*, 97–104.
- CHENNEY, S., AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. *proc. SIGGRAPH 2000*, 219–228.
- CHI, D., COSTA, M., ZHAO, L., AND BADLER, N. 2000. The emote model for effort and shape. *proc. SIGGRAPH 2000*, 173–182.
- GLEICHER, M. 1997. motion editing with spacetime constraints. *1997 Symposium on Interactive 3D graphics*, 139–148.
- GLEICHER, M. 1998. Retargeting motion to new characters. *proc. SIGGRAPH 1998*, 33–42.
- HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. *proc. SIGGRAPH 1995*, 229–238.
- HODGINS, J., WOOTEN, W. L., BROGAN, D. C., AND O'BRIEN, J. F. 1995. Animating human athletics. *proc. SIGGRAPH 1995*, 229–238.
- KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. *Proc. SIGGRAPH 2002*.
- LEE, J., AND SHIN, S. Y. 1999. A hierarchical approach to interactive motion editing for human-like figures. *proc. SIGGRAPH 1999*, 39–48.
- LEE, J., AND SHIN, S. Y. 2001. A coordinate-invariant approach to multiresolution motion analysis. *Graphical Models* 63, 2, 87–105.
- LEE, J., CHAI, J., REITSMA, P. S. A., HODGINS, J. K., AND POLLARD, N. S. 2002. Interactive control of avatars animated with human motion data. *Proc. SIGGRAPH 2002*.
- LI, Y., WANG, T., AND SHUM, H. 2002. Motion texture: A two-level statistical model for character motion synthesis. *Proc. SIGGRAPH 2002*.
- PERLIN, K., AND GOLDBERG, A. 1996. Improv: a system for scripting interactive actors in virtual reality worlds. *proc. SIGGRAPH 1996*, 205–216.
- PERLIN, K. 1985. An image synthesizer. *Computer Graphics* 19, 3, 287–296.
- POPOVIC, Z., AND WITKIN, A. 1999. Physically based motion transformation. *proc. SIGGRAPH 1999*, 159–168.
- PULLEN, K., AND BREGLER, C. 2000. Animating by multi-level sampling. *IEEE Computer Animation Conference*, 36–42.
- SCHODL, A., SZELISKI, R., SALESIN, D. H., AND ESSA, I. 2000. Video textures. *Proc. SIGGRAPH 00*, 489–498.
- UNUMA, M., ANJYO, K., AND TEKEUCHI, R. 1995. Fourier principles for emotion-based human figure animation. *proc. SIGGRAPH 1995*, 91–96.
- WITKIN, A., AND KASS, M. 1988. Spacetime constraints. *Computer Graphics* 22, 159–168.
- WITKIN, A., AND POPOVIC, Z. 1995. Motion warping. *proc. SIGGRAPH 1995*, 105–108.