Quick Transitions with Cached Multi-way Blends

Leslie Ikemoto
University of California, Berkeley

Okan Arikan
University of Texas, Austin

David Forsyth
University of Illinois, Urbana-Champaign

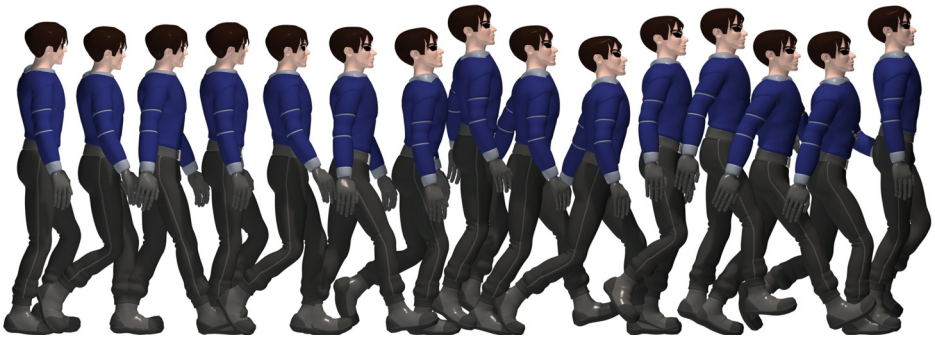

Figure 1: *This figure is a time-lapsed shot of a transition synthesized in real-time using our method. The character transitions in one second from walking to skipping in a seamless, natural way. We invite the reader to view this animation in the accompanying movie.*

Abstract

We describe a discriminative method for distinguishing natural-looking from unnatural-looking motion. Our method is based on physical and data-driven features of motion to which humans seem sensitive. We demonstrate that our technique is significantly more accurate than current alternatives.

We use this technique as the testing part of a hypothesize-and-test motion synthesis procedure. The mechanism we build using this procedure can quickly provide an application with a transition of user-specified duration from any frame in a motion collection to any other frame in the collection. During pre-processing, we search all possible 2-, 3-, and 4-way blends between representative samples of motion obtained using clustering. The blends are automatically evaluated, and the recipe (i.e., the representatives and the set of weighting functions) that created the best blend is cached.

At run-time, we build a transition between motions by matching a future window of the source motion to a representative, matching the past of the target motion to a representative, and then applying the blend recipe recovered from the cache to source and target motion. People seem sensitive to poor contact with the environment like sliding foot plants. We determine appropriate temporal and positional constraints for each foot plant using a novel technique, then apply an off-the-shelf inverse kinematics technique to enforce the constraints. This synthesis procedure yields good-looking transitions between distinct motions with very low online cost.

Keywords: Motion evaluation, motion synthesis, motion blending

I3D 2007, Seattle, Washington, April 30 – May 02, 2007.
© 2007 ACM 978-1-59593-628-8/07/0004 $5.00

1 Introduction

Many applications require realistic, high-quality character animation. Applications as diverse as simulation, movies, and video games depend upon natural-looking motion to increase the believability of virtual worlds.

Applications commonly demand sequences that transition between types of motion (such as skipping to running) or between particular frames in a motion collection. There may be several constraints on the transition, but typically the transition must look realistic and be of a particular duration.

Motion graphs [Molina-Tanco and Hilton 2000; Kovar et al. 2002a; Lee et al. 2002; Arikan and Forsyth 2002] are an effective motion synthesis technique, and can be used to generate such transitions. The major difficulty with doing so is that the motions may not be responsive, because the shortest path between two frames may be too long or look bad. This problem is common in practice (Section 8).

In this paper, we present a method that can generate transitions of a user-specified length from any frame in a motion collection to any other frame or type of motion. Our technique is a hypothesize-and-test method, which depends on an accurate scoring mechanism to automatically evaluate large numbers of transitions. We demonstrate that our scoring mechanism is more reliable at recognizing natural-looking transitions than current alternatives on our data sets. Although we cannot generate a natural-looking transition for every frame pair, we compute a score that indicates each transition's quality. Hence, applications know before display whether the transition will look natural or not.

2 Previous Work

Producing natural looking human motion is a topic of broad interest. Motion can be obtained using physical considerations [Witkin and Kass 1988; Sulejmanpašić and Popović 2005], statistical models [Li et al. 2002], or by measurement [Menache 1999]. Rearranging clips taken from a collection of motion can produce new, believable motions. These new motions are obtained by search on a *motion graph* — a graph where each node is a frame of motion, each directed edge represents the possibility that one frame can be placed after another, and each path represents an acceptable mo-

tion [Molina-Tanco and Hilton 2000; Kovar et al. 2002a; Lee et al. 2002; Arikan and Forsyth 2002]. The main question in motion graphs is how to insert edges that are not observed. Methods include: linking when frames are similar [Lee et al. 2002; Arikan and Forsyth 2002]; building short blends between similar frames [Kovar et al. 2002a]; or adding edges by hand [Gleicher et al. 2003].

Transitions present difficulties. [Wang and Bodenheimer 2004] show a careful choice of blend schedule and duration can significantly improve transitions between similar frames. Transitions between widely differing motions — from a walk to a stand, for example — are difficult to construct in the absence of observations, because the considerations that apply (such as avoiding awkward body configurations) are more than physical. [Wang and Bodenheimer 2003] describe some success with a metric for selecting transitions. Spacetime methods can be used to generate dynamically plausible constraints [Rose et al. 1996]. [Arikan et al. 2005] combine physical models with motion data to create transitions. Transitions are important, because one wants a motion graph of **small diameter** — it should be possible to get from one frame to another with a relatively short path. One trivial and unwise way to get a motion graph with a short diameter is by inserting edges between dissimilar frames (or in the most extreme case, between all frames). Transitioning between dissimilar frames will likely result in displeasing visual artifacts, where the character appears to move unnaturally. Currently, no method can automatically yield a motion graph with a short diameter that also produces natural human motion. We demonstrate some difficulties in Figure 4; [Gleicher et al. 2003] describe a method to engineer a motion graph with small diameter that also produces high quality motion by hand.

It is natural to **blend** motion clips to create transitions. This approach is known to be effective for similar motions, and a sensible choice of blending procedure can produce physically correct motion [Safonova and Hodgins 2005]. Sequences can be warped in time and space to create visually pleasing blends ([Kovar and Gleicher 2003]). [Wang and Bodenheimer 2004] demonstrates that the length of a linear interpolation is important. One may obtain a better blend by blending more than two sequences. The methods of [Bruderlin and Williams 1995; Wiley and Hahn 1997; Rose et al. 1998] generate visually pleasing multi-way blends for many blend sources, but produce unnatural-looking blends for others. Our method attacks the problem of determining which were successful. [Kovar and Gleicher 2004] describes how to find multiple motions that can be blended, and how to create parameterized blend spaces within these multiple examples. [Park et al. 2002; Park et al. 2004; Kwon and Shin 2005] show how to create a parameterized blend space for each of walking, running, and standing; their method can generate realistic transitions between these three types of motion.

Unlike previous papers that use multi-way blending, we do not restrict our blend space to contain only motions of a homogenous type. Instead, our method searches a very large number of blends for good results, and so we can find transitions between significantly different motions.

Our method relies on **automatic evaluation of motion**, so we require a reliable, automatic scoring technique. [Ikemoto and Forsyth 2004; Arikan et al. 2005] demonstrate that supervised classifiers can discriminate between natural and unnatural motions produced by their synthesis techniques. [Ren et al. 2005] demonstrate and evaluate several unsupervised techniques that discriminate natural-looking motion regardless of the synthesis technique used. In section 6, we demonstrate a novel motion evaluation method that improves on current practice.

Our blended motions may be subject to **footskate**. There are numerous footskate cleanup methods that involve some manual intervention: one marks footplants and plant locations, then uses inverse kinematics to ensure the constraints are met (e.g. [Bindiganavale and Badler 1998; Shin et al. 2001; Kovar et al. 2002b; Liu and Popovic 2002]). Because we must clean up a large collection of blends, we require a fully automatic method such as [Ikemoto et al. 2006] or [Le Callennec and Boulic 2006]. Our method is based on [Ikemoto et al. 2006].

3 Overview

Our goal is to create a compact motion synthesis mechanism that can meet transition demands in real-time with a natural-looking sequence of user-specified duration.

Interpolation is a popular technique for creating transitions. We can build a transition from frame s to t by interpolating between S (the motion clip that starts with s), T (the motion clip that ends with t), and one or more other motion clips (which we call *intermediaries*) once all of the clips have been time-aligned (Appendix A). Some multi-way blended transitions look natural and some do not. We need a method for deciding which intermediaries produce the most natural-looking blended transition.

We use a hypothesize-and-test procedure that automatically searches over the intermediaries to find the most natural-looking transition available, scoring motions with a novel evaluation technique (Section 7). We demonstrate that this evaluation technique is more accurate for our datasets than current alternatives.

There are too many motion clips to search over, so we reduce the search space by clustering them (Section 5). We define the *representative* of a motion clip to be the medoid of the cluster to which the clip belongs. If the clustering is effective, the motion clip's representative will be similar to the clip. Searching over only the representatives greatly reduces the number of multi-way blends we need to evaluate, but still retains much of the variation we would encounter if we conducted the entire search.

4 Run-time mechanism

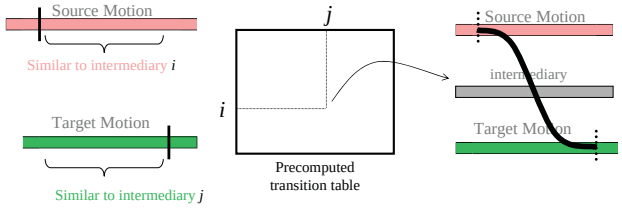

Figure 2: *To synthesize a frame-to-frame transition online, we find the representative i that is closest to the source motion and the representative j that is closest to the target motion. We use i and j as indices into a precomputed transition table that indicates which motion clips to blend with the source and target to create the transition. We call these motion clips intermediaries.*

To synthesize a frame-to-frame transition online, we find the representative that is most similar to the motion beginning at the source frame (S), and the representative that is most similar to the motion ending at the target frame (T). We then retrieve the intermediaries required for these two representatives. We perform the blend on S, T, and the cached intermediaries after time-warping the motion clips. Figure 2 illustrates our mechanism.

Before display, we clean up footskate automatically. We first decide when and where the feet should be planted using the method

described in Section 7.1. We then modify the character's degrees of freedom to meet the foot plant constraints using inverse kinematics (IK). Since IK may produce awkward poses if the target location is too far from the original position of the foot, we attempt to move the foot only if the target is less than a small distance away (we use 6-12 inches). Moving the feet may introduce discontinuities in the blended motion. We smooth these discontinuities over neighboring frames.

5 Clustering

We first split the motion dataset into overlapping clips with length equal to the user-specified transition duration, then use k-means clustering in spectral embedded space ([Pothen et al. 1990]) to separate the clips by similarity. We use the cluster medoids as the representatives.

Spectral embedding is a method developed in the computer vision community for producing features that tend to cluster data points consistently with an affinity matrix. To compute the affinity between two clips, we first align their center frames in the global coordinate system, then compute the sum of squared differences of the joint positions over the motion (similar to the method used in [Kovar et al. 2002a]). The affinity matrix is then used to embed all clips in a low-dimensional subspace by computing the principal components of the affinity matrix, then using the ones with largest variance to project each clip into a lower-dimensional subspace. We use 10 principal components. k-means clustering can then be used to cluster the clips. Setting k equal to 50 gave good results for our dataset.

Computing an affinity between all pairs of clips is expensive because the computation time is proportional to the number of motion clips squared, and the number of clips can be large. Using a dense affinity matrix to compute an embedding is also very expensive. We use the approximate spectral embedding method of [Fowlkes et al. 2004], which uses the Nystrom approximation to reduce the computational burden considerably. Briefly, the approximation computes affinities between only a small random subset of motion clips and all other clips. This partial affinity matrix is used to approximate an embedding.

6 Scoring Baseline Method

Hypothesize-and-test is a natural algorithm for constructing human motions. There are now many possible methods for producing hypotheses, but no wholly reliable scoring methods. Generalization — giving an accurate score to motions very different from the training motions — is a notoriously difficult problem. Ikemoto and Forsyth use a classifier to evaluate motions produced by a cut-and-paste method, and find the classifier significantly less accurate on novel motions [Ikemoto and Forsyth 2004]. The classifier is trained using both positive and negative examples.

There is some advantage to not using negative examples, which can be difficult to obtain. [Ren et al. 2005] fit an ensemble of generative models to positive examples; motion is scored by taking the lowest likelihood over all models to obtain a conservative score. While the combined generative model gives the best behavior in practice, their combined Hidden Markov model (ensemble of HMMs) is almost as accurate. There is no information on generalization behaviour. However, if negative examples are available, we expect that models trained discriminatively are likely to perform better, because they possess more information about the location of the boundary between good and bad motion.

Training set: We picked 400 transitions generated from multi-way blends of different source-target pairs and intermediaries at random. These were annotated as natural-looking or unnatural-looking motions by hand.

Baseline: We fit an ensemble of HMMs (one for the body, one for each limb, and one for each pair of limbs) to positive motion examples. Each example is represented by a feature vector containing joint position, velocity, and acceleration data over the motion. Motions are scored with their likelihood under this model.

7 Scoring Transitions

Several papers have observed that humans find footskate noticeable and objectionable [Kovar and Gleicher 2002; Arikan 2006]. While there are many effective techniques for fixing footskate, large amounts of footskate where the foot slides a great deal usually cannot be fixed in a natural-looking way. This motivates using an error metric based on footskate to determine whether a motion looks natural (Section 7.1).

Unnaturalness may also stem from other parts of the body. In a physically valid motion, the *zero moment point* (the point on the ground plane at which the moment of the ground reaction forces is zero) is always within or on the boundary of the *support polygon* (the convex hull of the points at which the character is contacting the ground). People seem to find motions which do not obey this constraint objectionable, as the physical inconsistency often appears as loss of balance (though it is important to note that the zero moment point constraint holds even if the character is falling) [Shin et al. 2003]. We can use the distance between the character's zero moment point and support polygon as another error metric, which we explain further in Section 7.2.

We use these two error metrics to automatically evaluate the naturalness of the transitions we generate between between two frames. The best transition is the one with the lowest combined error. We obtained good results by simply summing the foot plant error and the zero moment point error.

7.1 Foot plant error

To determine footskate error, we first need to decide when and where foot plants should occur in the blend. Then, we can measure how much the foot moves relative to where it should be planted. If the foot does not move very much, the IK module will only need to make small changes to the motion, which will probably look natural. However, if the foot moves a great deal when it is supposed to be planted, the motion may still look objectionable if IK is used because the model will make large changes to the motion. Hence, we can use the extent to which the foot moves as an approximate error metric for the naturalness of the motion.

Determining foot plants: We wish to identify when and where foot plants should occur in the blended motion. One solution is to identify the frames in the blended motion where one or both of the feet are close to the ground and roughly stationary. [Kovar and Gleicher 2002] used such a scheme to compute input constraints for their IK algorithm (though they note that this scheme required some manual clean-up to get good results). While this technique can be successful for fixing footskate, it may not work well for determining foot plant error because the error estimated will always be small.

An observation we make is that since the blended motion is a combination of known blend sources $\mathcal{B}$, we assume that the blended motion's foot plants are a combination of the blend sources' foot plants.

Given this observation, a possible solution is to blend the source

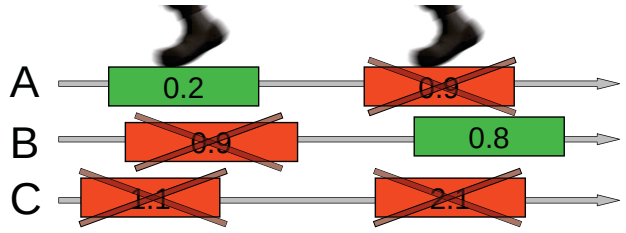

Figure 3: ***Choosing when and where foot plants should occur in the blended motion:*** *Consider three blend sources A, B, and C. We would like to choose foot plants for the left foot of the blended motion. The frames containing left foot plants in each of the source motions have been labeled. In this example, all three motions contain two left foot plants. For each left foot plant, we compute the displacement error between the plant's position and the position of the blended motion's left foot at the corresponding frames. Then we choose non-overlapping left footplants in order of lowest error. We first choose the first plant in A. This means we cannot choose the first plant in B or the first plant in C. The next valid footplant with lowest error is the second plant in B. We choose this one, which invalidates the rest of the footplants.*

foot plants using the same weighting scheme we used to blend the source motions. However, this technique will yield fractional values for the foot plants. These fractional values could be thresholded to obtain binary values, but it is unclear how to set the threshold to obtain good results.

Instead, we consider the entire set of source foot plants, and choose the non-overlapping combination that is closest to the trajectories of the character's feet in the blended motion.

More specifically, we first compute the position of each footplant for each motion in $\mathcal{B}$ relative to the blended character's torso coordinate frame. We now take each right footplant in each motion in $\mathcal{B}$. Over the period of the footplant, we average the absolute value of the displacement error between the trajectory of the right foot in the blended motion, and the location of the right footplant. We put each right footplant in a priority queue. We then follow the same procedure to put the left foot plants in a second priority queue.

We now dequeue the right footplant with lowest error. We label the corresponding frames in the blend to indicate that they contain a footplant and where the footplant should be located. We then dequeue another right footplant. If the footplant does not overlap the first, we again label the corresponding frames to indicate that they contain a footplant and where the footplant should be located. We stop once the queue is empty. We do the same for the left footplants. Figure 3 illustrates this algorithm.

Computing footskate error: Now that we have determined when and where footplants should occur, we can compute footskate error. We compute the displacement between the foot's actual position and the target foot plant location at every frame labeled as a foot plant. The displacement indicates how much the IK module will move the foot. IK can generally produce natural-looking motion for small displacements but not for large ones. Also, a large displacement indicates that the blended transition is very dissimilar from the blending sources, which means that it probably looks unnatural. We use the average displacement as our measure of footskate error.

7.2 Zero moment point error

At every point at which the character contacts the ground, the character is subject to ground reaction forces. The zero moment point (ZMP) is the point on the ground plane where the moment of these forces is zero. In a physically valid motion, the ZMP lies within or on the boundary of the support polygon. This concept was first introduced by [Vukobratovic and Juricic 1969] and has been used previously in robotics and graphics (e.g., [Tak et al. 2000; Shin et al. 2003]).

We compute the distance between the ZMP and the support polygon at every frame in a transition we wish to evaluate, then use the maximum distance as our error metric.

We calculate the ZMP at each frame using the equations from Shin, Kovar, and Gleicher [Shin et al. 2003]. These equations depend on knowing the mass of each segment of the character's body. As [Shin et al. 2003] suggest, we compute the character's mass distribution using optimization with an average mass distribution (measured by [Winter 2005]) as the starting point. The optimization yields a mass distribution close to the starting point that gave plausible ZMP locations for our original motion collection. We believe that this is because our motion collection is large and contains varied types of motion.

8 Results

The accompanying movie contains several comparisons of the 2-, 3-, and 4-way blends we synthesize using our method. All transitions we generated were one-second long (or 60 frames in our motion collection, which was captured at 60 Hz). We can generate high-quality transitions which are normally considered difficult, such as a transition from a walk to a skip, and a run to a stand. These examples are included in the video. The video shows our results side-by-side with transitions obtained using the baseline HMM-based classifier described in Section 6 and by traversing a motion graph. These examples demonstrate that the HMM-based classifier can inaccurately choose visually displeasing transitions, and that a transition obtained by traversing a motion graph may be too long or may have visual artifacts.

Our motion synthesis mechanism (Section 4) operates in real-time. We randomly sampled 200 source frame/target frame pairs. The average time the synthesizer required to create a one second long blend and cleanup footskate in the sequence was 0.012 seconds. Amortized over 60 frames, this cost is negligible. We also randomly sampled 200 source frame/target annotation pairs. The average time the synthesizer required to create the blend and cleanup footskate was 0.014 seconds, which means the extra time to search for a good target cluster is very small.

Our mechanism is also compact. We can compute the upperbound on the size of our transition mechanism. A maximum size entry (i.e., intermediaries and timewarp) contains a 4-way blend and the worst case timewarp, which is 120 frames long. Such an entry requires 1.9 KB. There are 50 clusters, so there are 50 members in the representative set. Thus, there are 50×50 entries in the transition mechanism, so the upperbound on the size of the matrix is less than 5 MB. Storing the differences between all one-second clips in the database to all cluster medoids consumes 200 bytes per frame (4 bytes per float $\times$ 50 cluster medoids).

Figure 4 contains a scatter plot comparing the performance of motion graphs to our synthesizer. Each point represents a random source frame/target frame pair (there are 500 total). The horizontal axis is the number of frames required to make the transition in a motion graph. The vertical axis is the cost our scoring mechanism estimated for making a 60-frame transition in our synthesizer. This figure demonstrates that our synthesizer can synthesize many transitions that motion graphs cannot. We believe that the two mechanisms can nicely complement each other. Our plot is separated into

quadrants. The bottom lefthand quadrant contains the frame pairs for which the motion graph can synthesize a motion that is less than 1 second long. For these demands, it is best to use a motion graph, but our synthesizer still produces good results. The upper lefthand quadrant contains the pairs for which our synthesizer cannot generate a natural looking transition, but a motion graph can easily generate one, so for these demands also, it is best to use a motion graph. The bottom righthand quadrant contains the frame pairs for which the transition synthesized by the motion graph is long, but we can produce a natural 1 second long transition. These demands are best served with our synthesizer. The upper righthand quadrant contains frame pairs for which the motion graph and our synthesizer cannot produce a good transition.

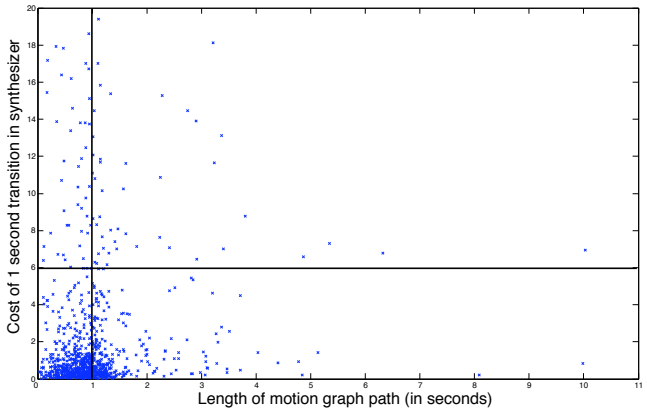

Figure 4: ***Comparison to motion graphs.*** *This scatter plot compares the performance of a motion graph to our synthesizer. Each point represents a random source frame/target frame pair (there are 1000 total). The x-axis represents the length of the shortest transition between this pair of frames using motion graph, and y represents the cost of the 1-second long transition between the frames using our synthesizer. A cost under the horizontal line generally corresponds to a natural-looking motion. Note that there are many points in the bottom righthand quadrant (302 points), indicating that there are many frame pairs for which our synthesizer can generate a natural-looking, 1-second long transition while the shortest transition in a motion graph would take longer than 1 second. (Some of the motion graph paths are in fact quite long, reaching 10 seconds in one case.) Note also that there are few points in the upper righthand quadrant (57 points), which means there are few frame pairs for which both our synthesizer and a motion graph cannot produce a good transition. There are 581 points in the lower left hand quadrant (where both our method and motion graphs perform well), and 60 points in the upper left (where motion graphs outperform our method). These results indicate that our technique and motion graphs nicely complement each other, and that together there are few transition demands the combination cannot meet.*

We compare our scoring mechanism to the state-of-the-art HMM baseline described in Section 6. Figure 5 demonstrates that our method outperforms the baseline. This is consistent with the general belief that natural behavior at the feet and whether the character appears in balance are important tests for the goodness of a motion.

Because our scoring mechanism is reliable, if a natural-looking blended transition exists in our search space, we are likely to find and cache it. Therefore, our aggressive hypothesize-and-test strategy can produce natural-looking transitions for many demands. For some demands, there may not be a natural-looking transition in our search space. This problem could be alleviated by conducting more search, but in general, transitioning to a very different activity over a short time period is difficult to synthesize convincingly (and may in fact be hard for a human to perform).

Our method can be easily extended to synthesize transitions between a source frame and a target annotation by simply picking a target frame with the desired annotation. We demonstrate an application that uses this extension in the video.

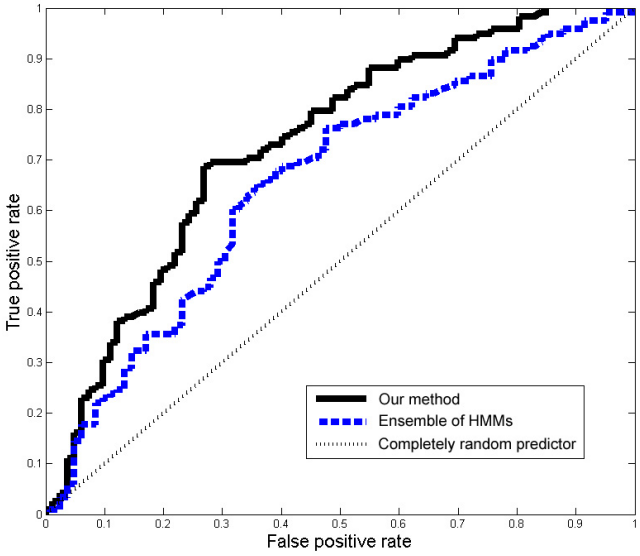

Figure 5: *We compare the performance of our method for scoring the naturalness of motion with HMM's. The most natural way to compare is to classify good from bad motion using a threshold, then plot the receiver operating curves (ROC). A completely random classifier's ROC would lie along the black dotted line from bottom left to top right (sometimes called the no-discrimination line), and a perfect classifier's ROC would be a point in the upper left corner. The area between a classifier's ROC and the no-discrimination line is commonly used as an indicator of the performance of the classifier, where a larger area indicates a more discriminative classifier. We used a test set of 200 multi-way blended transitions chosen at random which were not in the training set. A user labeled each as natural (118 total in our test set) or unnatural (82 total). We compared each classifier's labels against the user's labels. As the figure shows, our method clearly outperforms the HMM's.*

9 Conclusions & Future Work

We have demonstrated a fast, compact mechanism that can synthesize transitions of a user-specified duration between arbitrary frames in a motion dataset. Our technique searches over and automatically evaluates many candidate blends to find the best ones. If two frames differ significantly, there may not be a good transition between them. In this case, we synthesize the best blend possible. However, there is no reason to believe that our blending algorithm is optimal. Another blending algorithm might produce natural-looking blends where ours fails. Because we have an effective scoring mechanism, we could expand the search to incorporate different blending weights and strategies. This is an area of future work.

The error metrics we use to evaluate the naturalness of motion are only valid if there are foot plants in the motion. For ballistic motion, another error metric, such as checking that conservation of angular momentum is maintained, may work well.

A third area of future work is to explore the potential to produce transitions that vary significantly from one another. Currently, a transition between particular frames is fixed. However, replacing intermediaries with other motions in the same cluster could produce

a rich source of variation.

For our motion collection, we found that our scoring mechanism performed markedly better than HMMs. Improving hypothesize-and-search motion synthesis methods crucially depends on improved scoring methods. We expect further advances in motion scoring to be a significant area of future work.

A Multi-way blending

In this appendix, we describe how we create a multi-way blended transition of a particular duration between a source motion, a target motion, and intermediaries. Multi-way blending is not new (see for example [Rose et al. 1998; Wiley and Hahn 1997; Bruderlin and Williams 1995; Kovar and Gleicher 2004]) and has been shown to produce high-quality motion for many blend sources. We describe our time-warping and weighting techniques for completeness.

[Kovar and Gleicher 2003] demonstrate that aligning motion sequences in time so that similar frames correspond is an important precursor to blending. They show how to use dynamic timewarping to time-align two motions by constructing a distance matrix. Each row in the distance matrix corresponds to a frame in the source sequence, and each column corresponds to a frame in the target sequence. Each cell corresponds to a possible frame correspondence, and stores the error between the source and target frame. Paths through the matrix are possible time alignments. Dynamic programming can find a minimum cost path.

Call the set of motions to blend $\mathcal{B}$ and the i^{th} motion in the set B_i. To form a multi-way blend, we time-align all of the motions in $\mathcal{B}$, then blend. Because we search large pools of possible blends, we may need to blend very dissimilar motions. This means that we may not be able to follow the dynamic timewarp constraints recommended in [Kovar and Gleicher 2003] and [Kovar and Gleicher 2004]. In these cases, we simply use the time-alignment that traverses the diagonal of the distance matrix. Timewarping squashes and stretches time slightly, and so the blends that we produce may not be exactly 1 second (60 frames) long. We allow time to stretch at maximum by a factor of 2, but in practice 99% of our cached transitions are fewer than 80 frames in length.

Now that we have computed a time alignment for the motions in $\mathcal{B}$, we can interpolate them. To do so, we need to devise blend weights. We create a weighting function W, where $W(t)$ gives us the weights for the frames we are interpolating. Our blending weights are Bezier functions. We chose Bezier functions because they seem to produce visually good blends, and because they extend linear blending. For example, a quadratic Bezier function is a linear interpolation of two linear interpolations. Therefore, choosing a quadratic Bezier function for a three-way blend is equivalent to interpolating B_0 and B_1, then B_1 and B_2, and then the two resulting blends. However, there are several plausible possibilities for weighting functions. It is an easy extension to our method to search over alternative weighting functions.

Once we have obtained the time alignment and the weighting functions, we can interpolate the motions. Each frame contains the 2D position of the root, the rotation of the root about the vertical axis, and the 3D position of each joint. We interpolate each joint's position using W. We cannot simply interpolate the root's position because doing so can cause the root's path to collapse on itself [Kovar and Gleicher 2003]. Instead, we interpolate the differential movement of the root of each B_i. The position of the root at the t^{th} frame of the blend is:

$$p(t-1) + \sum_{i=0}^{N} W(i,t)(p_{B_i}(t) - p_{B_i}(t-1))$$

where p is the position of the blended root, p_{B_i} is the position of B_i's root, $W(i,t)$ is the current blend weight on B_i, and N is the total number of motions in B. The position of the blended root at time 0 is taken from the first frame in the source motion.

Acknowledgements

We would like to thank the Berkeley graphics group for their helpful comments and suggestions. This work was generously supported by ONR N00014-01-1-0890, and benefited from motion capture data donated by Sony Computer Entertainment America.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *SIGGRAPH*.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Pushing people around. In *SCA*.

ARIKAN, O. 2006. Compression of motion capture databases. In *SIGGRAPH 2006*.

BINDIGANAVALE, R., AND BADLER, N. I. 1998. Motion abstraction and mapping with spatial constraints. In *CAPTECH 1998*, Springer-Verlag, London, UK, 70–82.

BRUDERLIN, A., AND WILLIAMS, L. 1995. Motion signal processing. In *SIGGRAPH*.

FOWLKES, C., BELONGIE, S., CHUNG, F., AND MALIK, J. 2004. Spectral grouping using the nystrom method. In *PAMI*.

GLEICHER, M., SHIN, H. J., KOVAR, L., AND JEPSEN, A. 2003. Snap-together motion: assembling run-time animations. In *Symposium on Interactive 3D graphics*.

IKEMOTO, L., AND FORSYTH, D. A. 2004. Enriching a motion collection by transplanting limbs. In *SCA*.

IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2006. Knowing when to put your foot down. In *I3D*.

KOVAR, L., AND GLEICHER, M. 2002. Footskate cleanup for motion capture editing. In *SCA*.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In *SCA*.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *SIGGRAPH*.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *SIGGRAPH*.

KOVAR, L., SCHREINER, J., AND GLEICHER, M. 2002. Footskate cleanup for motion capture editing. In *SCA*.

KWON, T., AND SHIN, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *SCA*.

LE CALLENNEC, B., AND BOULIC, R. 2006. Robust Kinematic Constraint Detection for Motion Data. In *SCA*.

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. In *SIGGRAPH*.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: a two-level statistical model for character motion synthesis. In *Computer graphics and interactive techniques*.

LIU, C. K., AND POPOVIC, Z. 2002. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH*.

MENACHE, A. 1999. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan-Kaufmann.

MOLINA-TANCO, L., AND HILTON, A. 2000. Realistic synthesis of novel human movements from a database of motion capture examples. In *Workshop on Human Motion*.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. In *SCA*.

PARK, S. I., SHIN, H. J., KIM, T. H., AND SHIN, S. Y. 2004. On-line motion blending for real-time locomotion generation: Research articles. *Comput. Animat. Virtual Worlds*.

POTHEN, A., SIMON, H., AND LIOU, K. 1990. Partitioning sparse matrices with eigenvectors of graphs. In *SIAM Journal of Matrix Anal. Appl.*

REN, L., PATRICK, A., EFROS, A. A., HODGINS, J. K., AND REHG, J. M. 2005. A data-driven approach to quantifying natural human motion. *SIGGRAPH*.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH*.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*.

SAFONOVA, A., AND HODGINS, J. K. 2005. Analyzing the physical correctness of interpolated human motion. In *SCA*.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*.

SHIN, H. J., KOVAR, L., AND GLEICHER, M. 2003. Physical touch-up of human motions. In *Pacific Conference on Computer Graphics and Applications*.

SULEJMANPAŠIĆ, A., AND POPOVIĆ, J. 2005. Adaptation of performed ballistic motion. *ACM Trans. Graph.*.

TAK, S., YOUNG SONG, O., AND KO, H.-S. 2000. Motion balance filtering. *Eurographics*.

VUKOBRATOVIC, M., AND JURICIC, D. 1969. Contributions to the synthesis of biped gait. In *IEEE Transactions on Biomedical Engineering*.

WANG, J., AND BODENHEIMER, B. 2003. An evaluation of a cost metric for selecting transitions between motion segments. In *SCA*.

WANG, J., AND BODENHEIMER, B. 2004. Computing the duration of motion transitions: an empirical approach. In *SCA*.

WILEY, D. J., AND HAHN, J. K. 1997. Interpolation synthesis for articulated figure motion. In *Virtual Reality Annual International Symposium*.

WINTER, D. 2005. *Biomechanics and Motor Control of Human Movement, Third edition*. John Wiley and Sons.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *SIGGRAPH*.