Quick Motion Transitions with Cached Multi-way Blends

Leslie Kanani Michiko Ikemoto
Okan Arikan
David Forsyth

Electrical Engineering and Computer Sciences
University of California at Berkeley

February 13, 2006

Acknowledgement

Quick Motion Transitions with Cached Multi-way Blends

Leslie Ikemoto*
University of California, Berkeley

Okan Arikan†
University of Texas, Austin

David Forsyth‡
University of California, Berkeley
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 1 second from walking to skipping in a seamless and natural way. We invite the reader to view this animation in the accompanying movie.

Abstract

We describe a method for responsive, high-quality synthesis of human motion. Our method can quickly provide a motion synthesizer with a one-second long, high-quality transition from any frame in a motion collection to any other frame in the collection.

We construct these transitions using 2-, 3- and 4-way blends. During pre-processing, we search all possible blends between representative samples of motion obtained using clustering. The blends are evaluated automatically with a novel motion evaluation procedure, which we demonstrate is significantly more accurate than current alternatives. The best blending recipe for each pair of representatives is then 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 and whatever stubs are required. This method yields good-looking transitions between distinct motions with very low online cost.

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. It is time-consuming and expensive to produce motion data in large quantities. Motion graphs ([Kovar et al. 2002a], [Lee et al. 2002], [Arikan and Forsyth 2002], [Molina-Tanco and Hilton 2000]) are a popular and effective way to organize re-use of sequences.

Applications commonly demand sequences that transition between particular frames, which may come from distinct types of motion (such as skipping to running). There are two major difficulties with using motion graphs to synthesize these transitions. First, the motions may not be responsive. A frame-to-frame transition synthesized by a motion graph may be too long or very bad. This prob-

lem is common in practice (Section 7). Second, the motions may lack variation, because few paths (sometimes only one) may exist between two frames. In this paper, we present a method that generates good quality, short transitions from any frame to any frame or to any type of motion. Also, these transitions can be made rich.

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 motion [Kovar et al. 2002a; Lee et al. 2002; Arikan and Forsyth 2002; Molina-Tanco and Hilton 2000]. 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 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. No current method of building a motion graph automatically can achieve this requirement (we demonstrate some difficulties below, see figure 3); [Gleicher et al. 2003] describe a method to engineer a motion graph with small diameter by hand.

It is natural to **blend** motion clips to create transitions. This approach is known to be effective for similar motions, and a sensi-

---

*e-mail: lesliei@eecs.berkeley.edu

†e-mail:okan@cs.utexas.edu

‡e-mail:daf@cs.uiuc.edu

ble 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 ([Rose et al. 1998], [Wiley and Hahn 1997], [Bruderlin and Williams 1995]). [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. [Kwon and Shin 2005; Park et al. 2002; Park et al. 2004] 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. ([Arikan et al. 2005], [Ikemoto and Forsyth 2004]) 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 5, we demonstrate a novel motion evaluation method that improves on current practice.

Our blended motions are 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. [Shin et al. 2001; Kovar et al. 2002b; Liu and Popovic 2002; Bindiganavale and Badler 1998]). Because we must clean up a large collection of blends, we use the fully automatic method of [Ikemoto et al. 2005].

3 Overview

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

Linear interpolation is often used to transition from a source motion S to a target motion T. The degrees of freedom at the end of S are interpolated with those at the beginning of T such that the weight on S decays while the weight on T grows. This technique is sometimes called motion blending. Its major strength is that it can meet transition demands quickly. Surprisingly, this simple, non-physically based technique produces natural looking transitions in many cases [Safonova and Hodgins 2005].

Two-way blending is most likely to produce unnatural transitions when S and T are dissimilar. We show that adding one or more intermediate motion stubs to the blend can make it look significantly more natural. If such intermediate motion stubs exist, then we can transition quickly and naturally from S to T even when they are unlike each other.

Offline, we obtain representative samples of motion by clustering. We cluster all one-second long clips in a motion collection (Section 6). We treat each cluster medoid as a representative sample or *motion stub*. To synthesize a frame-to-frame transition online, we find a stub that is close to the source frame and a stub that is similar to the target. We then retrieve a cached blending recipe for these two clusters, and perform the blend recipe between the source, target, and any required intermediate motion stubs.

The blending recipes are cached during preprocessing. We generate all 2-, 3-, and 4-way transitions possible using our blending recipe and stubs (Section 4). Using an accurate scoring mechanism (Section 5), we can search this large set of transitions automatically to find a recipe that produces natural-looking transitions. 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.

4 Multi-way blending

In this section, we describe how to create a multi-way blended transition of a particular duration between a source motion and a target motion. [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.

Note that when we form the distance matrix to compute an alignment, we can choose any cells on the border as the starting and ending points. We find that picking pairs of local minima as the start and end points can yield good candidate alignments. We compute an alignment for all such pairs, then pick the one with minimum cost. Specifically, we consider all vertical local minima (i.e., all cells with a value less than their upstairs and downstairs neighbors) in the first column and all horizontal local minima (i.e., cells with a value less than their righthand and lefthand neighbors) in the first row as potential starting points. All vertical local minima in the final column and horizontal local minima in the last row are candidate end points.

Call the set of motions to blend $\mathscr{B}$ and the ith motion in the set B_i. To form a multi-way blend, we time-align all of the motions in $\mathscr{B}$, 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 obey 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 $\mathscr{B}$, 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 rotation of each joint expressed as a quaternion. We blend each joint's rotation using multi-way spherical linear interpolation

(SLERP). 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 tth frame of the blend is:

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

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.

4.1 Fixing footskate

Footskate, where a character's foot slides on the floor when it should be planted firmly, manifests itself in almost any attempt to synthesize motion. There are three steps to fixing footskate:

1. Label the frames in which footplants should occur. We can do automatically this with a classifier [Ikemoto et al. 2005].

2. Locate a target location for each foot in the labeled frames. We discuss our method in this section.

3. Modify the character's degrees of freedom so that the foot reaches the target location. This can be done with inverse kinematics (IK).

Recall that the set of motions to blend is called $\mathscr{B}$. We have already labeled the frames in our database containing footplants using [Ikemoto et al. 2005], so we know which frames in $\mathscr{B}$ contain footplants. Now we wish to identify footplants and synthesize footplant positions for the blended motion. We will do so by comparing the blended motion with each of the sources of the blend. If the blended motion is close in space to a source with a footplant at that time, this is evidence that the blended motion should have a footplant.

First, we compute the position of each footplant for each motion in $\mathscr{B}$ relative to the blended character's torso coordinate frame. We now take each right (resp. left) footplant in each motion in $\mathscr{B}$. Over the period of the footplant, we average the absolute value of the displacement error between the trajectory of the right (resp. left) foot in the blended motion, and the location of the right (resp. left) footplant. We put each right (resp. left) footplant in a priority queue, keeping right and left in separate queues.

We now dequeue the right footplant with lowest error. We label the corresponding frames in the blend to indicate that they contain a footplant. However, we attempt to fix the position of the foot only if the error is less than a small threshold (we use 6 inches). We use IK to fix the foot to that footplant's location. 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 if the error is small enough, fix the foot. We stop once the queue is empty. We do the same for the left footplants. Moving the feet may introduce discontinuities in the blended motion. We smooth these discontinuities over neighboring frames. Figure 2 illustrates this algorithm. Notice that blend frames may be labeled as containing a footplant, but not have a fixed foot, because the error might be too large. The foot is only fixed to the footplant's location if the error is lower than the threshold. This allows us to compute a score for the motion by summing the displacement error to the closest blend source foot position over *all* frames labeled as footplants.

5 Scoring transitions

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
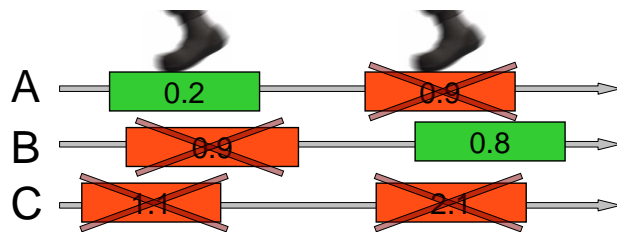

Figure 2: **Choosing positions for the feet in the blended motion:** Consider three blend sources A, B, and C. We would like to choose footplants for the left foot of the blended motion. The frames containing left footplants in each of the source motions have been labeled. In this example, all three motions contain two left footplants. For each left footplant, 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.

— 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 both inaccurate and 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 (HMM) 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 different source-target pairs and blending strategies at random. These were annotated as natural-looking or unnatural motions by hand.

Baselines: We must compare a large number of blends, and require good motion scoring to do so. Accordingly, we have evaluated several possible methods. First, 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. Second, we apply logistic regression to a feature vector containing the magnitude of the maximum absolute acceleration of each joint (for logistic regression, see, for example, [Hastie et al. 2001]). This feature is justified by the observation that humans seem very sensitive to large changes in acceleration. However, this feature understates the significance of natural looking feet. We build a feature that emphasizes footstrike labelling error by aligning footstrike annotations of source, target and intermediate motions, and counting the number of frames where footstrike annotations do not agree. Our third classifier uses logistic regression applied to this feature. Neither classifier is individually as reliable as our fourth, which involves taking the minimum of the two logistic regression scores. We use this classifier and the HMM as baselines.

Our method: We use a novel scoring scheme based on a measure

of foot strike error. The final step of the blending procedure corrects footskate by determining an appropriate strike/no-strike annotation, and, in the case of strikes, determining where the foot should be planted. A fair measure of the quality of the blend is the extent to which the foot's actual position differs from where it should be planted. We score a motion by averaging the magnitude of the foot displacements over the whole transition. The best transition is the one with the lowest score. If there are no footstrikes in the blended motion (e.g., jumping motions), then we assign a score using the logistic regression technique explained previously.

Comparisons: Our method outperforms both the minimum of two logistic regressions and the HMM (Figure 4). This is consistent with the general belief that natural behavior at the feet is an important test for the goodness of a motion.

6 Clustering

We use spectral clustering [Shi and Malik 2000] to cluster every one-second clip of motion in our database. This clustering method requires a matrix that stores the affinity of every datapoint to every other datapoint. To compute the affinity between two one-second long motions, we first align their center frames in the global coordinate system. We then compute the sum of squared differences of the joint positions over the motion. We use the fourth-root of the sum of these differences as the affinity.

It is difficult to fix the number of clusters required *a priori*, so we use an iterative clustering scheme. Starting with a small number of clusters (in our case 20), we clustered the database and visually checked the results. Once we reached 50 clusters, our clusters appeared to contain similar motions. Since we perform clustering several times, it is important to make it as efficient as possible. We use the Nystrom approximation [Fowlkes et al. 2004] to speed up clustering considerably.

7 Results

The accompanying movie contains several comparisons of the 2-, 3-, and 4-way blends we synthesize using our method. 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 on the video.

Our motion synthesis mechanism operates in real-time. We randomly sampled 200 source frame/target frame pairs. The average time the synthesizer required to create a 1 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 the transition matrix. A maximum size entry contains a 4-way blend and the worst case timewarp, which is 120 frames long. Such an entry requires 1.9 KB. There are 50×50 entries in the matrix, 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 3 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.
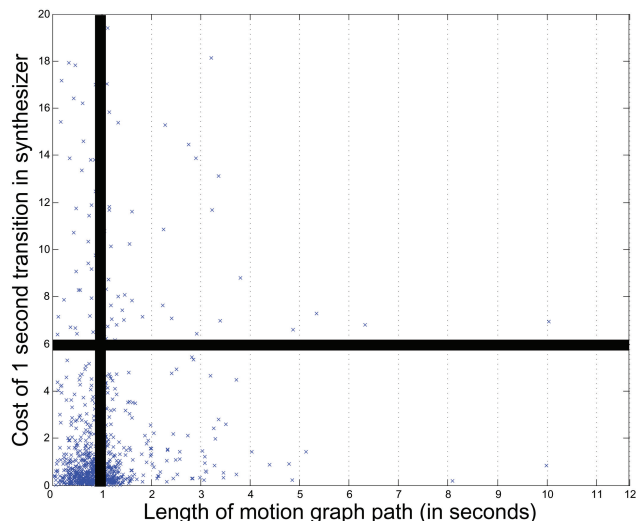
Figure 3: **Comparison to motion graphs.** This scatter plot compares the performance of a motion graph to our synthesizer. Each point represents a random source frame/target frame pair (there are 500 total). The *x*-axis represents the length of a transition between the frames synthesized using the motion graph, and *y* represents the cost of a 1-second long transition between the frames using our synthesizer. A cost under 6 generally corresponds to a natural-looking motion. Note that some of the motion graph-generated transitions are quite long (there even some that are 10 seconds long). 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.

We compare our scoring mechanism to three other baselines described in 5. Figure 4 demonstrates that our scoring method outperforms all 3 baselines.

8 Conclusions & Future Work

We have demonstrated a fast, compact mechanism that can synthesize 1-second long transitions 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 1-second transition between them. In this case, we synthesize the best blend possible. However, this problem may be alleviated by expanding the search to incorporate different blending weights and recipes. This is an area of future work.

Another 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 intermediate motion stubs with other motions in the cluster could produce a rich source of variation.
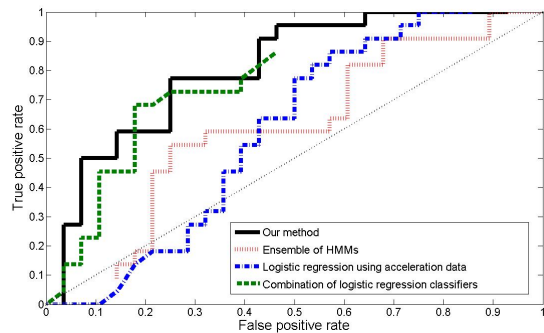
Figure 4: We compare the performance of four methods for scoring motion (HMM's, logistic regression on acceleration features, the minimum of that and a logistic regression on footskate features, and our footskate score). Each produces a number with the same semantics (smaller is better), but somewhat different scales. The most natural way to compare is to classify good from bad motion using a threshold, then plot the receiver operating curves (ROC). A very good ROC would mean that good motions tended to have low scores and bad motions high scores, and these scores would be well separated. We use a test set of 50 labelled sequences chosen at random (28 bad and 22 good). As the figure shows, both HMM's and logistic regression on acceleration alone are clearly inferior to the other alternatives. Our footskate score has a somewhat better ROC than the combined logistic regression measure. In practice, this small improvement in the ROC seems to produce significantly better search results, most likely because the probability that the best score found is a good motion increases sharply with small improvements in ROC.

References

ARIKAN, O., AND FORSYTH, D. A. 2002. Interactive motion generation from examples. In *SIGGRAPH 2002*, ACM Press, New York, NY, USA, 483–490.

ARIKAN, O., FORSYTH, D. A., AND O'BRIEN, J. F. 2005. Pushing people around. In *SCA 2005*, ACM Press, New York, NY, USA, 59–66.

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 1995*, ACM Press, New York, NY, USA, 97–104.

FOWLKES, C., BELONGIE, S., CHUNG, F., AND MALIK, J. 2004. Spectral grouping using the nystrom method. 214–225.

GLEICHER, M., SHIN, H. J., KOVAR, L., AND JEPSEN, A. 2003. Snap-together motion: assembling run-time animations. In *Symposium on Interactive 3D graphics*, ACM Press, New York, NY, USA, 181–188.

HASTIE, T., TIBSHIRANI, R., AND FRIEDMAN, J. 2001. *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer Verlag.

IKEMOTO, L., AND FORSYTH, D. A. 2004. Enriching a motion collection by transplanting limbs. In *SCA 2004*, ACM Press, New York, NY, USA, 99–108.

IKEMOTO, L., ARIKAN, O., AND FORSYTH, D. 2005. Knowing when to put your foot down. In *I3D: Symposium on Interactive 3D Graphics and Games (to appear)*.

KOVAR, L., AND GLEICHER, M. 2003. Flexible automatic motion blending with registration curves. In *Proceedings of the Symposium on Computer animation*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 214–224.

KOVAR, L., AND GLEICHER, M. 2004. Automated extraction and parameterization of motions in large data sets. *ACM Trans. Graph. 23*, 3, 559–568.

KOVAR, L., GLEICHER, M., AND PIGHIN, F. 2002. Motion graphs. In *SIGGRAPH 2002*, ACM Press, New York, NY, USA, 473–482.

KOVAR, L., SCHREINER, J., AND GLEICHER, M. 2002. Footskate cleanup for motion capture editing. In *SCA 2002*, ACM Press, New York, NY, USA, 97–104.

KWON, T., AND SHIN, S. Y. 2005. Motion modeling for on-line locomotion synthesis. In *SCA 2005*, ACM Press, New York, NY, USA, 29–38.

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 2002*, ACM Press, New York, NY, USA, 491–500.

LI, Y., WANG, T., AND SHUM, H.-Y. 2002. Motion texture: a two-level statistical model for character motion synthesis. In *29th annual conference on Computer graphics and interactive techniques*, ACM Press, 465–472.

LIU, C. K., AND POPOVIC, Z. 2002. Synthesis of complex dynamic character motion from simple animations. In *SIGGRAPH 2002*, ACM Press, New York, NY, USA, 408–416.

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 (HUMO'00)*, 137–142.

PARK, S. I., SHIN, H. J., AND SHIN, S. Y. 2002. On-line locomotion generation based on motion blending. In *SCA 2002*, ACM Press, New York, NY, USA, 105–111.

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 15*, 3-4, 125–138.

REN, L., PATRICK, A., EFROS, A. A., HODGINS, J. K., AND REHG, J. M. 2005. A data-driven approach to quantifying natural human motion. *ACM Transactions on Graphics (SIGGRAPH 2005) 24*, 3 (Aug.), 1090–1097.

ROSE, C., GUENTER, B., BODENHEIMER, B., AND COHEN, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *SIGGRAPH 1996*, ACM Press, New York, NY, USA, 147–154.

ROSE, C., COHEN, M. F., AND BODENHEIMER, B. 1998. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl. 18*, 5, 32–40.

SAFONOVA, A., AND HODGINS, J. K. 2005. Analyzing the physical correctness of interpolated human motion. In *Proceedings of the 2005 Symposium on Computer animation*, ACM Press, New York, NY, USA, 171–180.

SHI, J., AND MALIK, J. 2000. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 8, 888–905.

SHIN, H. J., LEE, J., SHIN, S. Y., AND GLEICHER, M. 2001. Computer puppetry: An importance-based approach. *ACM Trans. Graph. 20*, 2, 67–94.

SULEJMANPAŠIĆ, A., AND POPOVIĆ, J. 2005. Adaptation of performed ballistic motion. *ACM Trans. Graph. 24*, 1, 165–179.

WANG, J., AND BODENHEIMER, B. 2003. An evaluation of a cost metric for selecting transitions between motion segments. In *SCA 2003*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 232–238.

WANG, J., AND BODENHEIMER, B. 2004. Computing the duration of motion transitions: an empirical approach. In *SCA 2004*, ACM Press, New York, NY, USA, 335–344.

WILEY, D. J., AND HAHN, J. K. 1997. Interpolation synthesis for articulated figure motion. In *Virtual Reality Annual International Symposium (VRAIS '97)*.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *SIGGRAPH 1988*, ACM Press, New York, NY, USA, 159–168.