Quick Transitions Using Multi-way Blends

Leslie Ikemoto*
University of California Berkeley

Okan Arikan†
University of Texas Austin

David Forsyth‡
UC Berkeley / UI 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.

Many applications require character animation that is both natural-looking and controllable. These two requirements are often in opposition. For example, motion graphs (e.g., [Kovar et al. 2002]) are a popular technique that can create realistic-looking transitions between frames of motion. However, the technique is not very controllable because: (1) it may take a long time (proportional to the square of the number of frames for a dense graph using Djikstra's algorithm) to find the shortest transition between the character's current frame of motion and a desired frame; and (2) the shortest transition may be too long. This sketch describes an alternative transition mechanism that can generate a transition of user-specified length between any two frames of motion.

A typical motion dataset may contain many versions of the same motion, so we first find a set of motion clips that represents the dataset well. We split the motions in the dataset into overlapping clips (with length equal to the user-specified transition length), then cluster them. We use the cluster medoids as the representative set.

Now we wish to find a natural-looking transition from every member of the representative set to every other member. We propose transitions to a classifier which decides whether the motion is realistic-looking or not. A transition involves blending from one motion to another (two-way blends), or blending from one motion to another using intermediate motions (multi-way blends). We search over all combinations of two, three, and four way blends between the representatives.

We classify motions using a weighted combination of perceptually important motion features. The features we use measure the footskate and zero moment point error. Our classifier outperforms other state-of-the-art classifiers (e.g., [Ren et al. 2005]) on our motion databases.

We store the blending schedule (i.e., the frames of the source motions used at each tick) for the blend the classifier liked best. These blending schedules form a transition matrix which we can use to transition between any two frames in the time the user specified.

At run-time, the application can demand a transition at any time from the character's current frame to a desired frame. We find the representative that is closest to the clip of motion starting at the current frame (with user-specified duration), and the representative closest to the clip ending at the desired frame. We then look up the blending schedule for these representatives and employ it, using the current and desired clips in place of the representatives.

*e-mail: lesliei@cs.berkeley.edu
†e-mail:okan@cs.utexas.edu
‡e-mail:daf@cs.uiuc.edu

The scatter plot in figure 2 compares the performance of a motion graph to our mechanism (with transition length set to one second). This figure demonstrates that our mechanism and motion graphs can nicely complement each other. Our plot is separated into quadrants. The bottom left quadrant contains the 558 frame pairs for which the motion graph can synthesize a motion less than one second long. For these demands, it is best to use a motion graph, but our mechanism still produces good results. The upper left quadrant contains the 57 pairs for which our synthesizer cannot generate a natural looking transition, but a motion graph can easily generate one. For these demands also, it is best to use a motion graph. The bottom right quadrant contains the 325 frame pairs for which the transition synthesized by the motion graph is long, but we can produce a natural one second long transition. These demands are best served with our mechanism. The upper right hand quadrant contains the 60 frame pairs for which the motion graph and our synthesizer cannot produce a good transition.

We believe that the two mechanisms can nicely complement each other. The classifier's score for each transition is a proxy for the quality of the transition. An application can use this score to decide whether to use our mechanism or to use a motion graph to generate a transition. As the accompanying video shows, we have created a transition mechanism for character animation that is controllable, yet still produces natural-looking motion for many demands.

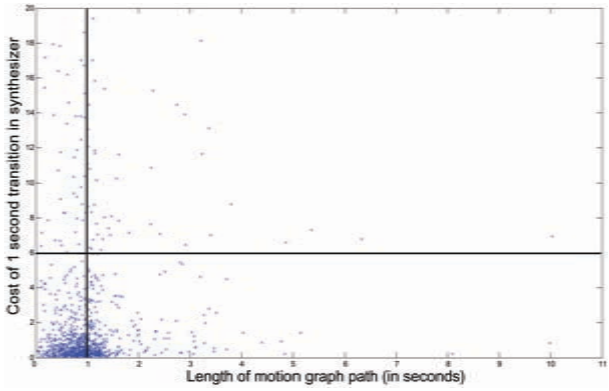

Figure 2: This scatter plot compares the performance of a motion graph to our synthesizer. Each point represents a random source frame/target pair (there are 1000 total). The *x*-axis represents the length of a transition between the frames synthesized using the motion graph, and *y* represents the cost of a one-second long transition between the frames using our synthesizer. A score below the horizontal line generally corresponds to a natural looking motion. Note that some of the motion graph generated transitions are quite long (some even reaching ten seconds).

References

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

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 2005*.