Example-Based Control of Human Motion

by

Eugene Hsu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2004

Author ....
Department of Electrical Engineering and Computer Science
:ay 7, 2004

Certified by..... ....
ın Popović
Assistant Professor
ıesis Supervisor

Accepted by...... [illegible] ....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Example-Based Control of Human Motion

by

Eugene Hsu

Submitted to the Department of Electrical Engineering and Computer Science
on May 7, 2004, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

In human motion control applications, the mapping between a control specification and an appropriate target motion often defies an explicit encoding. This thesis presents a method that allows such a mapping to be defined by example, given that the control specification is recorded motion. The method begins by building a database of semantically meaningful instances of the mapping, each of which is represented by synchronized segments of control and target motion. A dynamic programming algorithm can then be used to interpret an input control specification in terms of mapping instances. This interpretation induces a sequence of target segments from the database, which is concatenated to create the appropriate target motion. The method is evaluated on two examples of indirect control. In the first, it is used to synthesize a walking human character that follows a sampled trajectory. In the second, it is used generate a synthetic partner for a dancer whose motion is acquired through motion capture.

Thesis Supervisor: Jovan Popović
Title: Assistant Professor

Acknowledgments

I would first and foremost like to thank my advisor, Dr. Jovan Popović, for his guidance and support. Working with him has been a great pleasure, and I look forward to continuing my graduate studies under him. Sommer Gentry provided invaluable guidance on many aspects of this thesis. In particular, her technical abilities and expertise with Lindy Hop made the dance evaluation possible. I also greatly appreciate her assistance with writing.

The acquisition of motion capture data is a difficult and laborious process, and it could not have been done without the help of Jonathan Chu. His meticulous work greatly simplified this task. Dr. Tommi Jaakkola and Gregory Shaknarovich provided valuable assistance during the earliest stages of this research, when it was only a small class project. Discussions with Dr. Kevin Murphy allowed me to shape the research into what it became.

I would also like to thank my fellow students in the MIT CSAIL Computer Graphics Group for their valuable input on various incarnations of this work. I also owe our wonderful support staff a great deal, as Adel Hanna, Bryt Bradley, and Tom Buehler have gone above and beyond the call of duty to make this thesis possible. I not only consider my fellow group members coworkers, but friends as well. Along with my good friends from outside the group, Mario Christoudias, Dasha Lymar, and Jackie Yen, they have given me a great deal of support that has been immensely helpful during some difficult times.

I owe a great deal to my undergraduate research advisor, Dr. Zoran Popović, for helping me to discover my interest in computer animation. I would not be here without him.

Last but certainly not least, I would like to thank my parents for everything. I could not possibly express in words my appreciation for all that they have given me. So, a simple thank you will have to suffice.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Authoring human motion is difficult for computer animators, as humans are exceptionally sensitive to the slightest of errors. This process involves an animator providing a control specification which is mapped to a target motion by some means. In traditional keyframe animation, for instance, the keyframes are the control specification, and the target motion is achieved through spline interpolation.

Due to advances in data acquisition technology and computational power, techniques have been developed that allow desired target motion to be specified using a human performance. This is natural for traditional keyframe animators, who often use recorded or live human motion for reference. Motion capture is the most direct method to map performances to animated humans, as it is essentially an identity mapping. However, a generalization of this approach to allow for more indirect mappings creates an array of fantastic possibilities, such as mapping voice signals to facial motion [Bra99] or gestural actions to animated reactions [JP99].

Indirect mappings, however, must still be encoded in some way. Manually, this can be an exceptionally challenging task requiring detailed, domain-specific knowledge. Consider a partner dance scenario in which an animator wishes to control a follower using the captured motion of a leader. The mapping from leader to follower motion must minimally encode a significant amount of knowledge about the structure of the dance; this knowledge, unfortunately, would be out of reach to an animator who is not a skilled dancer. Indeed, it would still be difficult for a skilled dancer to state the precise mapping. Human dancers learn their skills by observation and practice; the objective of this work is to emulate this process on a computer for situations, such as partner dance when the control specification takes the form of one dancer's motion.

To learn indirect mappings, a memory-based approach is adopted which implicitly encodes the desired mapping with a database of semantically meaningful example instances. These instances store segments of synchronized control and target motion, which provide examples of how the mapping should be applied to input control motions. In partner dance, an instance might contain an example control motion of a leader pushing his or her partner forward. The corresponding example target motion would be that of the follower, taking a step backward in response.

A new input control motion can be interpreted as a sequence of rigidly transformed and temporally stretched control segments from the mapping database. Through the mapping instances, a given interpretation also corresponds to a sequence of target segments that can be assembled to form a target motion. Dynamic programming is used to select a sequence that balances the quality of interpretation with the continuity of the induced target motion. Various postprocessing techniques can be then be applied to smooth and adjust the desired target motion.

The approach is evaluated on two applications. In the first, its ability to map low-dimensional input to high-dimensional motion is exhibited by controlling walk motion from mouse trajectories. In the second, its ability to handle complex, stylized mappings is shown by controlling a dance follower using the motion of a dance leader.

Chapter 2

Background

Performance-driven animation, or computer puppetry, derives its broad appeal from its ability to map human performances automatically to animated characters [Stu98]. While these mappings can be as simple as a direct copy of joint angles, the ability to discover more complex mappings gives the approach a tremendous amount of power and flexibility. In online techniques [JP99], computational speed and instantaneous results are of paramount importance; offline techniques [Bra99] allow quality and global optimality to take precedence. The method in this thesis falls into the latter category.

Complex mappings often defy purely physical or mathematical encodings. As a result, many methods assume that mappings are described by parametric probabilistic models [Bra99, DB01, DYP03, JP99]. An advantage of these techniques is their ability to generalize to a variety of inputs. However, this comes at a price: statistical learning often necessitates large volumes of training data or severe restrictions on model complexity. For certain applications, this a worthwhile tradeoff, but for others, it can result in impractically long training times or loss of important detail. Our memory-based approach does not suffer from these disadvantages.

An important benefit of this design choice is the ability to use segments, rather than frames, as the primitive unit of motion. This allows for explicit preservation of higher-level motion semantics. Kim et al. demonstrate that a semantically guided segmentation of rhythmic motion allows for highly realistic motion synthesis, even using simple transition models [KPS03]. This work also uses partner dance for evaluation, but it does not address the problem of generating a follower *given* the motion of a leader.

In the segment modeling domain, the method in this thesis is most similar to that of Pullen and Bregler [PB02]. While Pullen and Bregler's method was shown to be an effective solution for the chosen application of texturing keyframed motion, its applicability to our problem is limited by several factors. First, their method assumes no spatial dependencies between the control (keyframed curves) and the target (textured motion). Second, there is no enforcement of motion continuity, other than a heuristic for consecutively observed segments. Our approach generates target motion segments that are amenable to simple blending. Finally, their method assumes that the input motion can be presegmented analogously to the examples, which is achieved in their work by observing sign changes in velocity. One could extend this approach for rhythmic motions using the automated approach of Kim et al. [KPS03]. In the general case, however, a control motion may not admit *any* intuitive presegmentation. One may wish, for instance, to generate walk motion from a constant-velocity trajectory. Our method requires no presegmentation; moreover, it produces a semantically guided segmentation as part of the optimization. In this context, the algorithm could be viewed as an extension of speech recognition methods that use connected word models [RJ93].

Arikan et al. describe an example-based approach to synthesizing human motion that satisfies sparse temporal annotation and pose constraints [AFO03]. Although their work differs from ours in intent, they also employ a dynamic programming algorithm that optimizes a weighted combination of interpretation and motion continuity. Our formulation differs in two subtle but important ways. First, our notion of continuity is dependent on the interpretation; that is, the continuity between two motion segments is undefined until a candidate interpretation specifies a coordinate frame for their comparison. Second, their objective function is defined over frames instead of segments. As a result, they must use coarse-to-fine iterations of their dynamic programming algorithm to gain the temporal consistency that is intrinsic to our segment-based approach.

Other related methods based on motion capture clip rearrangement include work by Kovar et al. [KGP02], Lee et al. [LCR$^+$02], and Arikan and Forsyth [AF02]. Although these do not aim to discover control by example, they have nevertheless provided inspiration for our work. An additional distinction is that these methods do not use continuous control from human performance and focus on sparser specifications such as keyframes and nontemporal paths. Our method is not designed to handle such control specifications and therefore should be viewed as an alternative to these approaches, rather than a replacement.

Many motion rearrangement techniques are derived from previous work in texture synthesis. Here, we consider our work most similar in intent to image analogies [HJO$^+$01]. This method, given an unfiltered and filtered version of the same image, applies an analogous filter to a new unfiltered image. Our method, given a set of synchronized control and target motions, applies an analogous mapping to a new input control motion. Image analogies was shown to be an elegant method with some unexpected applications like texture transfer, texture-by-numbers, and super-resolution. It is our hope that our method will have the same versatility for motion.

Our dance evaluation suggests an alternative view of our method as one of interaction modeling. In this domain, techniques have been developed that specify the mappings between character motions with explicit models of character interaction. Adaptive autonomous characters have used rules to exhibit complex flocking, herding, and locomotory behaviors [Rey87, TT94]. Approaches to explicit interaction modeling have included layered architectures [BG95], procedural descriptions [PG96], and even cognitive models [FTT99]. In this context, the work in this thesis might be viewed as a competency module that enhances the skills of characters to enable their participation in complex interactive performances.

Chapter 3

Database Construction

We begin by acquiring examples of synchronized control motions $\mathbf{A}$ and target motions $\mathbf{B}$. Each frame of motion is encoded by a point cloud. For human motion, we use skeletal joint positions, since this representation provides a more intuitive space than joint angle representations for comparing poses [KGP02]. Furthermore, point cloud representations allow for generalization to control motions without skeletal representations, such as mouse input.

The examples are divided into control segments $\mathbf{a}_1, \ldots, \mathbf{a}_N$ and target segments $\mathbf{b}_1, \ldots, \mathbf{b}_N$, where $\mathbf{a}_i$ and $\mathbf{b}_i$ are synchronized motions that together represent a primitive semantic instance of the mapping. Our dance motions are segmented into two-beat rhythm units, since they are a basic unit of interaction for the specific type of dance (Lindy Hop), as shown in Figure 3-1. Our walk motions, on the other hand, are segmented according to gait cycles. In both cases, we use manual transcription, since each example motion must only be segmented once. Methods exist to automate this process if desired. Dance motion could be segmented using motion beat analysis [KPS03]. More general motions could be segmented using annotation [AFO03] or curve clustering [CGMS03].

This database of mapping instances will be used by a dynamic programming algorithm to simultaneously interpret a new control motion and generate an appropriate target. The usage of semantic primitives restricts the search space to natural target motions. This, however, is an application-specific choice. Using segments that are too

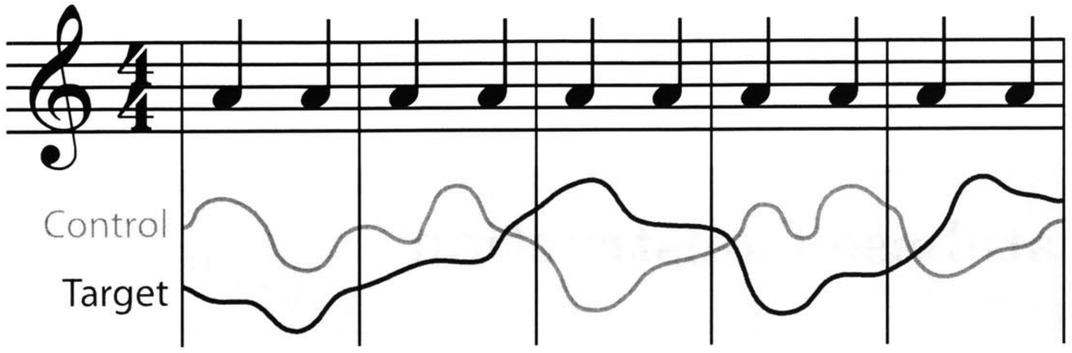

Figure 3-1: Segmentation of Lindy Hop motion into two-beat rhythm units.

short may result in motion that lacks coherence, while using segments that are too long will prevent the algorithm from generalizing to control motions that significantly differ from the examples. In general, the choice of the best segmentation is flexible.

Chapter 4

Algorithm Description

Given a control motion $\mathbf{x}$ with T frames, our goal is to generate an appropriate target motion. This is achieved by selecting a sequence of appropriate target segments from the database. To make the database motions more flexible, we allow each selected target segment to be spatially transformed and uniformly stretched in time. The proper selection of segments can be achieved using an efficient dynamic programming algorithm.

4.1 Single Segment

Before developing our general algorithm, we address the simpler problem of *interpreting* the input as a single control segment from the database. We quantify the similarity of the input motion $\mathbf{x}$ and a control segment $\mathbf{a}_s$ with a distance function:

$$D(\mathbf{x}, \mathbf{a}_s^T) \equiv \left\|\mathbf{x} - \mathbf{M}(\mathbf{x}, \mathbf{a}_s^T)\mathbf{a}_s^T\right\|^2 . \tag{4.1}$$

Here, $\mathbf{a}_s^T$ represents the control segment $\mathbf{a}_s$, uniformly stretched in time to T frames, and $\mathbf{M}(\mathbf{x}, \mathbf{a}_s^T)$ is a rigid transformation that optimally aligns $\mathbf{x}$ and $\mathbf{a}_s^T$:

$$\mathbf{M}(\mathbf{x}, \mathbf{a}_s^T) \equiv \arg\min_{\mathbf{M}} \left\|\mathbf{x} - \mathbf{M}\mathbf{a}_s^T\right\|^2 . \tag{4.2}$$

This optimization is the solution to the Procrustes problem, which has several efficient numerical solutions [ELF97]. Since our example dance and walk motions only differ by ground translation and vertical rotation, our implementation uses a closed form solution [KGP02].

To compute the optimal interpretation, we determine the segment $\mathbf{a}_{s^*}$ that is most similar to the input motion:

$$s^* = \arg\min_{s} D(\mathbf{x}, \mathbf{a}_s^T). \tag{4.3}$$

The index s^* also identifies, by construction of the database, an appropriate target $\mathbf{b}_{s^*}$ to both the control segment $\mathbf{a}_{s^*}$ and the input motion $\mathbf{x}$. The stretch T completes the specification of the optimal interpretation, $\mathbf{M}(\mathbf{x}, \mathbf{a}_{s^*}^T)\mathbf{a}_{s^*}^T$, and the optimal target, $\mathbf{M}(\mathbf{x}, \mathbf{a}_{s^*}^T)\mathbf{b}_{s^*}^T$. This is illustrated in Figure 4-1.

The optimal target may not precisely satisfy desired physical or kinematic constraints. However, given a descriptive database, it can provide a good approximation which can be adjusted appropriately during post-processing.

In practice, we limit the allowed amount of uniform time stretch by a constant factor since the distance metric does not distinguish between motions of varying speed. A dancer that pushes his partner slowly, for instance, will elicit quite a different response if he pushes quickly. Limiting the amount of stretch also has the practical benefit of reducing the search space of the general algorithm.

4.2 Multiple Segments

In general, we must handle the case where the optimal control and target consist of a sequence of segments. We can specify this sequence analogously to the single segment case by the number of segments L^*, the segment indices $s_1^*, \ldots, s_L^*$, and the segment durations $d_1^*, \ldots, d_L^*$.

As in the single segment case, the distance metric D evaluates the interpretation quality of each segment in the sequence. However, the quality of the interpretation alone does not account for the continuity of the target motion, as shown in Figure 4-2. To offset this problem, we introduce a function which measures the continuity between segments $\mathbf{v}$ and $\mathbf{w}$:

$$C(\mathbf{v}, \mathbf{w}) = \|\omega(\mathbf{v}) - \alpha(\mathbf{w})\|^2. \tag{4.4}$$

Here, α and ω represent the head and tail functions, which respectively extract the positions of the first and last frame of a segment.

Given a sequence specification L, $s_1, \dots, s_L$, and $d_1, \dots, d_L$, we define a scoring function accounts both for the quality of interpretation and the continuity of the target:

$$\sum_{i=1}^{L} D(\mathbf{x}_i, \mathbf{a}_{s_i}^{d_i}) + k \sum_{i=1}^{L-1} C\left(\mathbf{M}_i \mathbf{b}_{s_i}^{d_i}, \mathbf{M}_{i+1} \mathbf{b}_{s_{i+1}}^{d_{i+1}}\right). \tag{4.5}$$

Here, $\mathbf{x}_i$ is the subinterval of the input that is implied by the segment durations $d_1, \dots, d_i$. These in turn induce the transformations $\mathbf{M}_i \equiv \mathbf{M}(\mathbf{x}_i, \mathbf{a}_{si}^{d_i})$. The user-specified constant k defines the balance of interpretation and continuity.

The optimal substructure property of the score function, as defined by the following recurrence, can be used to find a globally optimal solution using dynamic programming:

$$\begin{aligned} Q_{s,d}[t] &= \min_{r,c} Q_{r,c}[t-d] + D(\mathbf{x}_{d,t}, \mathbf{a}_s^d) \qquad (4.6) \\ &\qquad + k\, C(\mathbf{M}_{r,c,t-d}\mathbf{b}_r^c, \mathbf{M}_{s,d,t}\mathbf{b}_s^d) \\ Q_{s,d}[d] &= D(\mathbf{x}_{d,d}, \mathbf{a}_s^d). \qquad (4.7) \end{aligned}$$

Here, $\mathbf{x}_{d,t}$ represents the subsequence of input frames starting at frame $t-d$ and ending at frame t, which in turn induces the alignment matrix $\mathbf{M}_{s,d,t} \equiv \mathbf{M}(\mathbf{x}_{d,t}, \mathbf{a}_s^d)$. $Q_{s,d}[t]$ is defined as the score of the optimization on the subsequence $\mathbf{x}_{t,t}$, given that the last segment is indexed by s and stretched to duration d. By minimizing $Q_{s,d}[T]$ over all s and d, we can compute the score of the optimal sequence specification and recover it by backtracking. In the following section, we describe this process in more detail.

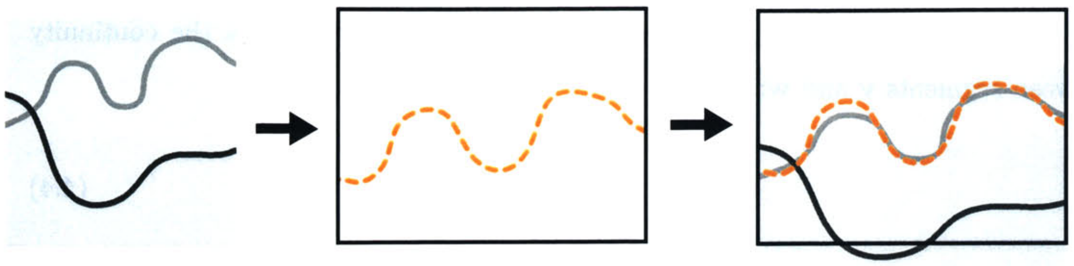

Figure 4-1: An example instance from the database is stretched and transformed to align the control segment with the input motion. The same stretch and transform can then be applied to the target segment.

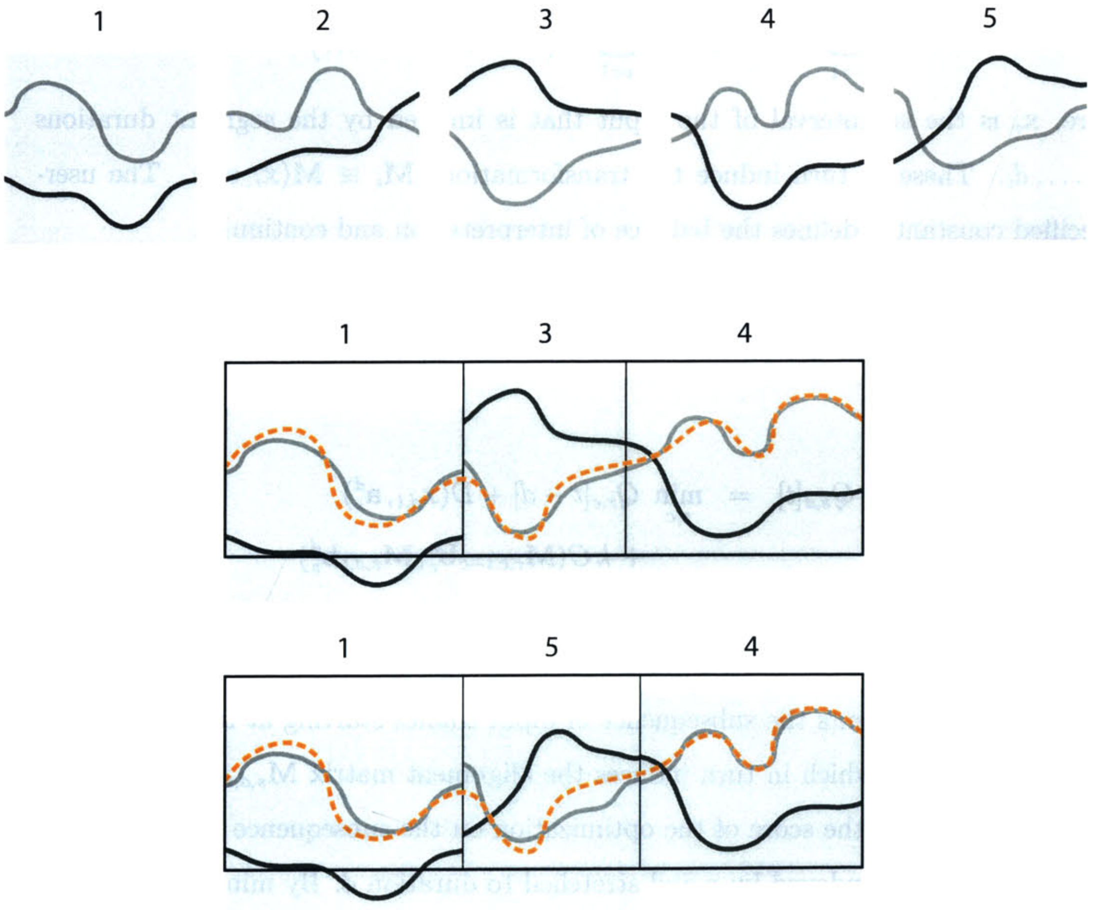

Figure 4-2: A good interpretation may not account for the continuity of the target (middle). Our scoring function strikes a balance between the two (bottom).

4.3 Implementation

To solve the recurrence efficiently, values of Q are stored in a two-dimensional array. Cells in this array are indexed by the time t on one axis and by all *legal* combinations of s and d on the other (recall from Section 4.1 that the amount of allowed stretch is limited). First, all legal values of $Q_{s,d}[d]$ are initialized according to the base case given in Equation 4.7, and all other array cells are set to infinity. The algorithm proceeds by iterating forward through time. At each time t, all non-infinite cells are located and scores are conditionally propagated forward in time according to Equation 4.6.

More specifically, suppose that we are currently processing the array cell $Q_{r,c}[t]$. For each legal combination of s and d, the candidate value z is computed:

$$z = Q_{r,c}[t] + D(\mathbf{x}_{d,t+d}, \mathbf{a}_s^d) + kC(\mathbf{M}_{r,c,t}\mathbf{b}_r^c, \mathbf{M}_{s,d,t+d}\mathbf{b}_s^d). \tag{4.8}$$

If the value in the array cell $Q_{s,d}[t+d]$ is greater than z, we set it to z and store a backpointer to cell $Q_{r,c}[t]$. By continuing this process, the entire array is filled. Since the indexing of each cell encodes a segment identifier and duration, the optimal sequence specification can be recovered by following backpointers from the best score at time T.

4.4 Efficiency

At each time t, $O(P)$ noninfinite cells are processed, where P is the number of legal combinations of s and d. Since processing an individual cell is an $O(P)$ operation, the total asymptotic time complexity of the algorithm is $O(P^2T)$. To increase its efficiency, we apply several heuristic optimizations. It is important to note that none of them will improve the quality of the results, only the speed at which the results are obtained. This is in contrast to the approach of Arikan et al. which relies on heuristics to obtain coherent motion [AFO03].

4.4.1 Beam Search

Rather than process all $O(P)$ noninfinite cells at each time t, we only process cells with scores less than $\min_{s,d} Q_{s,d}[t] + w$, where w is a user-specified constant. This technique is known as *beam search*, and w is known as the *beam width*. This is motivated by the fact that cells with worse scores are unlikely to be on the optimal backtracking path, and thus can be pruned from the search.

4.4.2 Clustering

In Chapter 3, we described the construction of a motion database by storing all instances derived from the examples. Since the time complexity of the algorithm scales quadratically with the database size, this leads to inefficiency when the number of instances is large. To resolve this issue, redundant instances are eliminated using *complete-linkage clustering* [DHS00]. In this algorithm, each instance begins in its own cluster. Larger clusters are formed iteratively by merging the two closest clusters, where the distance between clusters is defined as the maximum distance between any two instances in either cluster. We define the distance between two instances using Equation 4.1.

The advantage of complete-linkage clustering over other methods (such as k-means) is that it explicitly limits the distance of any two instances in a cluster by a user-defined threshold. After clusters are formed, a representative instance is chosen at random from each cluster to remain in the database, and all other instances are discarded. An additional benefit of this process is that it helps beam search; since clustering reduces ambiguity in interpretation, a larger proportion of search paths can be pruned.

4.4.3 Downsampling

High sampling rates are common for systems such as motion capture, but they are generally unnecessary for interpreting the input control motion. By downsampling motions by a user-chosen constant, we can effectively reduce the length of the input sequence. However, the resulting optimal sequence specification will also be at the lower frame rate, and it is generally desirable to have it at the frame rate of the original input. Simple upsampling often introduces slight but undesirable temporal errors. To remedy this, we run a highly constrained version of our dynamic programming algorithm that only adjusts the durations appropriately. Constraints can be easily encoded by making appropriate cells in the Q array illegal. For instance, we can force the result to contain a certain target segment $\mathbf{b}_s$ at some time t by disallowing any processing on cells $Q_{r,c}[u]$, where $r \neq s$ and $u - c \leq t \leq u$.

Chapter 5

Postprocessing

As described in Section 4, the output of our optimization is a specification of an appropriate target motion in terms of target segments in a database. More specifically, it provides a sequence of target segment indices $s_1^*, \ldots, s_L$ and durations $d_1^*, \ldots, d_L^*$. The corresponding target segments can be copied out from the database, stretched, transformed by the induced matrices $\mathbf{M}_1^*, \ldots, \mathbf{M}_L^*$, and concatenated. The results is a moving point cloud that approximates the desired result. Of course, the same selections, stretches, and transformations can just as easily be applied to the source motions that generated the point cloud.

From the perspective of motion synthesis, the main problem with our approach is that the raw result will generally contain some kinematic errors. In our dance example, footplant and handhold constraints are never explicitly enforced by our algorithm. For such constraints, existing methods can be applied to postprocess the data [KSG02], but such methods often require some amount of manual constraint annotation. Similarly to other motion capture clip rearrangement techniques, we can propagate constraints by example. In other words, each example instance can be annotated with constraints which can be transferred to the target motion. This is demonstrated by our propagation of handhold constraints, as shown in Figure 5-1.

We do not aim to introduce novel solutions for motion blending or constraint satisfaction. Instead, our goal is to provide motion that is amenable to postprocessing with these approaches. To demonstrate our method's capabilities in this regard, we

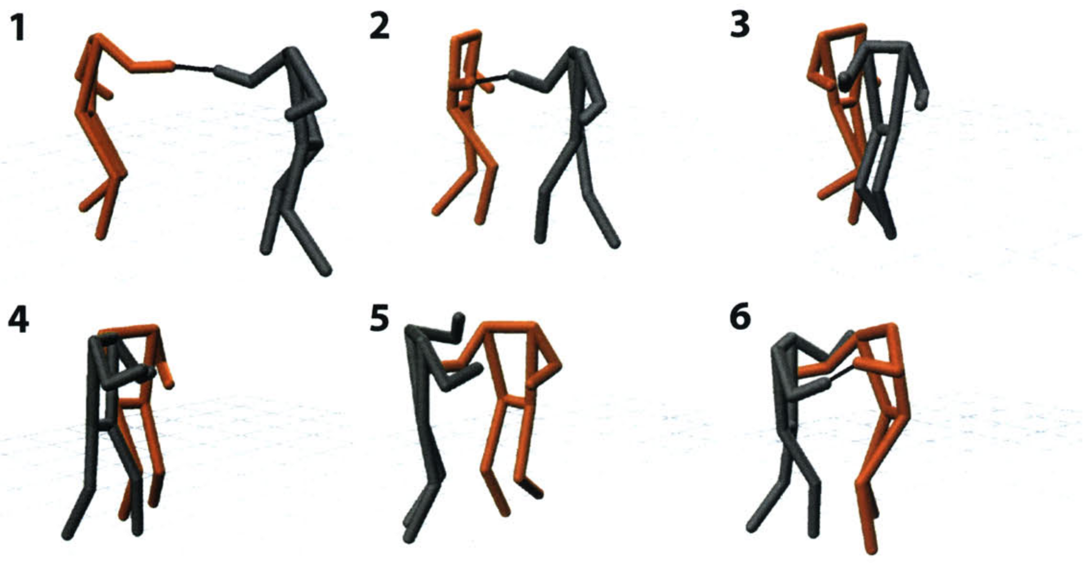

Figure 5-1: A handhold constraint, indicated by the line connecting the characters, was propagated from annotated examples to this generated motion. Here, the leader begins in an open crosshand stance and pulls the follower in (1,2). The follower releases handhold and performs an inside turn toward the leader (3,4). Nearing completion of the spin, the follower prepares to catch the leader's hand and enter embrace (5), and handhold is reestablished in closed stance (6). This sequence spans two beats.

show that it can generate realistic and compelling motion, even with extremely simple postprocessing. Our results, shown in the following section and in our accompanying video, are filtered with a basic smoothing operation that linearly adjusts motion curves to match across segment boundaries.

Chapter 6

Results and Evaluation

We evaluate our technique with two examples. In the first, we animate a realistic walking human from time-sampled mouse movement. Walk motions, however, do not show the full ability of our technique to discover complex mappings. To better demonstrate this aspect, we apply our method to a partner dance called Lindy Hop. Specifically, we use the complex motion of the dance leader to drive the motion of the follower.

In the following sections, all human motions were acquired in a motion capture studio and standard commercial tools were used to estimate joint positions [Vic03]. For the point cloud representation of body motion, we used only the positions of the hands and feet, as we found that these end-effectors were sufficient to evaluate interpretation and continuity in both evaluations. To generate the motion, we applied the resulting sequence specification to the source motion and used basic smoothing.

All timings were performed on a workstation with dual 2.4 Ghz Intel Xeon processors. Where applicable, we state the clock times for the dynamic programming algorithm (Section 4.3), upsampling (Section 4.4), and postprocessing (Section 5). The continuity constant, defined in Section 4.2, and the stretch limit were chosen experimentally.

6.1 Walk

We acquired 2 minutes of motion captured walk footage at 30 Hz. The subject was directed to walk within the capture area with random changes in direction and speed. We artificially constructed a synchronized example control motion by projecting the positions of the hip joints onto the floor and normalizing their distance. As stated previously, the target motions were represented by end-effector positions.

The walk footage was transcribed manually according to the gait cycle. More specifically, a segmentation point was manually placed at each footplant. From this process, we created 200 segments, which was reduced to 70 using clustering. For our tests, we downsampled these motions to 10 Hz and allowed each segment to be stretched ± 0.2 seconds.

Our first evaluation involved creating control motions from new walk motions that were not in the database. As before, we projected the hip joints onto the ground and normalized their distance. We ran our algorithm on these control motions and compared our results to the original source motions. Experimentally, we found that larger values of the continuity constant were more effective. For short walks, the generated motion was highly realistic. The frequency of the generated gait cycle nearly matched the frequency of the source, but phase differed. In more concrete terms, the generated motion might choose to start on the left foot, whereas the original source motion might start on the right. This was expected, as the control signals did not encode any phase information.

For longer walk motions, however, we were surprised to discover that the generated motions often kept in nearly perfect phase with the source. The reason for this was that the subject preferred to make sharp turns with the same footwork pattern. These served as synchronizing signals for the dynamic programming algorithm which were propagated throughout the generated gait cycle due to the global optimization.

For timing tests, we used a 57 second control motion. We first ran the algorithm without the beam search optimization. The dynamic programming algorithm took 12.5 seconds, upsampling from 10 Hz to 30 Hz took 0.4 seconds, and postprocessing

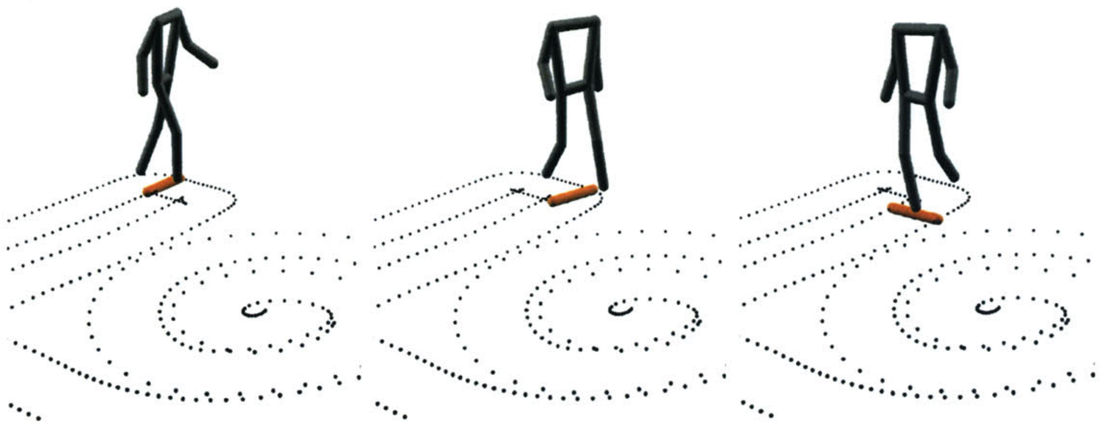
Figure 6-1: A synthetic character walks along a trajectory from mouse input. The spacing of points indicates the speed.

took 1.1 seconds. With the beam search optimization on, we were able to reduce the clock time of the algorithm to 1.2 seconds (47 seconds of input processed per second of clock time) while retaining visually perfect results. The upsampling and postprocessing times remained the same. We ran the algorithm on shorter and longer inputs and experimentally confirmed the asymptotic linear dependency of running time on input length, described in Section 4.4.

In our second evaluation, we built an interface that allowed users to draw paths using mouse input, as shown in Figure 6-1. The position of the mouse pointer was sampled at 30 Hz, and Frenet frames were used to generate a control motion. For a wide variety of user inputs, our method was capable of generating highly realistic walking motion. Since the timing of the path was important, we found that users required minor training to understand the concept of *performing* a path instead of *drawing* it. It was often tempting, for instance, to rapidly move the mouse to draw a straight line. This would correspond to a impossibly fast run, well beyond the capabilities of a human. To resolve these issues, our interface allows a user to overlay the playback of an existing motion on the drawing canvas to get a sense of speed. Furthermore, it provides options to smooth the trajectory spatially and temporally. The speed of the algorithm allows for rapid feedback.

6.2 Dance

Our choice of partner dance as a demonstration was primarily motivated by the complexity of its style and mappings. From a small segmented set of example instances, we generate a follower's motion to accompany a leader's motion. Generating partner dance motion would be a difficult trial for both physical methods, which would yield underdetermined systems; and statistical methods, which would typically require a very large database in place of our small segmented one. Swing dance also allows for a more principled evaluation of our results than most types of motion, since the performance of the algorithm at generating valid mappings can be evaluated independently of style considerations or subjective judgments of motion quality.

Lindy Hop is a subgenre of swing dance that, at a basic level, can be described as a state machine. A dance couple moves between four basic stances: *open* ($\bar{\circ}$), *closed* ($\bar{\bullet}$), *open crosshand* ($\underline{\circ}$), and *closed crosshand* ($\underline{\bullet}$). Open and closed refer to whether the couple is apart or in embrace, respectively. Crosshand refers to the case when the leader and follower hold right hands (we could also refer to it as a handshake).

Basic Lindy Hop motions switch between these four stances by means of transitions: an *inside turn* ($\curvearrowleft$), when the follower spins towards the leader, an *outside turn* ($\curvearrowright$), when the follower spins away from the leader, and a simple *step* ($\rightarrow$). At the end of each transition, the dancers may also change their handhold to instantly transition between crosshand states ($\underline{\circ}$, $\underline{\bullet}$) and non-crosshand states ($\bar{\circ}$, $\bar{\bullet}$). Figure 5-1 shows a couple transitioning from open crosshand stance to closed stance using an outside turn: $\underline{\circ}\curvearrowright\bar{\bullet}$. Each of these transitions occurs over four beats of music, which are assembled from two-beat segments; this was our motivation for performing two-beat segmentation, as described in Section 3. Figure 5-1 shows only the last two beats of a four-beat transition that starts with a two-beat rocking motion.

Skilled Lindy Hop dancers use a greater variety of moves, ranging from more complex transitions such as double outside turns to complex aerial maneuvers. We did not include the entire range of motions. Instead, we constructed a smaller database with seven basic 8-beat dance patterns that every Lindy Hop dancer knows (first

Database Patterns		Test Patterns			
1	$\bar{\circ}\rightarrow\bar{\bullet}\rightarrow\bar{\circ}$	1	$\bar{\circ}\rightarrow\bar{\bullet}\curvearrowleft\underline{\circ}$	8	$\underline{\circ}\curvearrowleft\bar{\bullet}\rightarrow\bar{\bullet}$
2	$\bar{\circ}\rightarrow\underline{\bullet}\curvearrowright\underline{\circ}$	2	$\bar{\bullet}\rightarrow\bar{\bullet}\curvearrowleft\bar{\circ}$	9	$\underline{\circ}\curvearrowleft\bar{\bullet}\curvearrowright\bar{\circ}$
3	$\bar{\circ}\rightarrow\bar{\bullet}\rightarrow\bar{\bullet}$	3	$\bar{\bullet}\rightarrow\bar{\bullet}\rightarrow\bar{\bullet}$	10	$\underline{\circ}\curvearrowleft\bar{\bullet}\rightarrow\underline{\circ}$
4	$\bar{\circ}\rightarrow\bar{\bullet}\curvearrowleft\bar{\circ}$	4	$\bar{\bullet}\rightarrow\bar{\bullet}\curvearrowright\bar{\circ}$	11	$\bar{\circ}\rightarrow\bar{\bullet}\curvearrowright\underline{\circ}$
5	$\bar{\circ}\rightarrow\bar{\bullet}\curvearrowright\bar{\circ}$	5	$\bar{\bullet}\rightarrow\bar{\bullet}\curvearrowleft\bar{\circ}$	12	$\underline{\circ}\curvearrowleft\bar{\bullet}\curvearrowleft\underline{\circ}$
6	$\bar{\bullet}\rightarrow\bar{\bullet}\rightarrow\bar{\circ}$	6	$\bar{\bullet}\rightarrow\bar{\bullet}\rightarrow\underline{\circ}$	13	$\underline{\circ}\curvearrowleft\bar{\bullet}\curvearrowleft\bar{\circ}$
7	$\underline{\circ}\curvearrowleft\bar{\bullet}\rightarrow\bar{\circ}$	7	$\bar{\bullet}\rightarrow\underline{\bullet}\curvearrowright\underline{\circ}$	14	$\bar{\circ}\rightarrow\bar{\bullet}\rightarrow\underline{\circ}$

Table 6.1: A notational description of the dance patterns stored in the database and the novel test patterns performed in our three test dances. Our technique adapts by rearranging the segments in the database to recreate the patterns it has not seen before.

column of Table 6.1). We constructed the motion database from a set of 12 short dances, each containing the seven basic 8-beat patterns, giving a total of 5 minutes of motion. These dances were segmented into 364 two-beat mapping instances, with lengths varying from approximately 0.6 seconds to 1 second due to different music.

For our evaluations, we captured three longer test dances (approximately 2-3 minutes each) in which the dancers were instructed to improvise with the transition and stances included in the database. Their improvisation led to dances which included thirteen new 8-beat patterns not found in the database (shown in the last column of Table 6.1) as well as some repeats of patterns in the database. These test dances spanned a tempo range from about 120 beats per minute to about 190 beats per minute. We used the motion of the leader to control a synthetic follower, which was then compared with the actual follower.

Visually, the results exhibited the fluidity, grace, and style of the original dancer. Some footskate and handhold violations are visible because we wanted to show the output in its almost raw form with smoothing applied just for visual coherence. In a direct comparison with the actual follower motions, we found that the synthetic follower matched very well in closed stances. In open stances, the follower was much more free to include stylistic variations, so the generated motions often differed visually from the actual motions. Additionally, the synthesized dancers were almost always in perfect rhythm with the leader.

Our algorithm ably recreated the semantics of the leader to follower mapping,

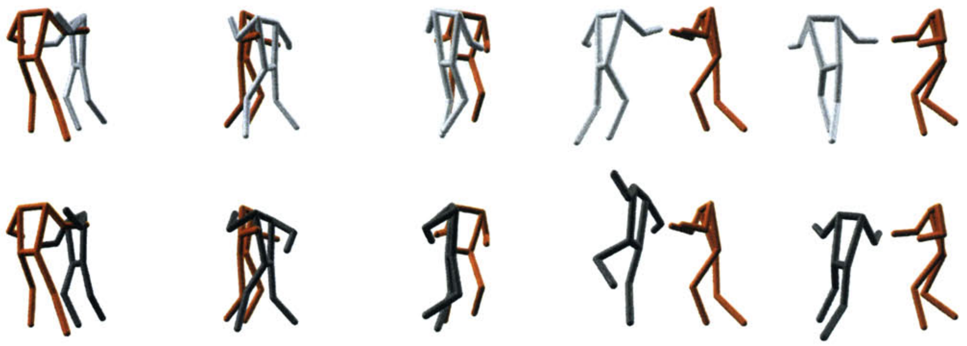

Figure 6-2: On the top, a clip of an actual dance is displayed. Here, the leader performs a regular handhold change during a step transition. This transition never occurs in our motion database. In response to the same motion cue, our algorithm generates a leaping outside turn, as show on the bottom. This is one of five two-beat segments (out of 380 two-beat segments in our three test dances), where the algorithm differs in its selection of response from an experienced dance follower. In other instances of this regular handhold change during a step transition in the test data, the algorithm correctly sequences motions to discover this novel vocabulary element.

even for novel patterns. When the algorithm encountered a pattern that was not in the database (one of 14 such patterns shown in table), it was able to correctly reconstruct the novel sequence by rearranging the two-beat segments. Of the 91 patterns (21 unique) in our three test dances the synthetic dancer matched the pattern of the actual dancer in all but 5 cases, one of which is shown in Figure 6-2. When the algorithm did differ from the real dancer in the composition of the pattern, the leader and follower still executed a valid Lindy Hop pattern. In these misinterpreted instances, the leader's motion is quite similar across two different follower patterns. To disambiguate these, we might add information to the control signal, such as force-plate readings, or we might accept these rare mismatches because they are in fact valid mappings. Furthermore, all 5 mismatched patterns differed by a single two-beat segment, so of $91 \times 4 = 364$ two-beat segments in the test dances, the technique misinterpreted the signal in 5 cases for an error rate of less than 2%.

For all our evaluations and timing tests, we reduced the size of the database from 364 to 168 with clustering, downsampled to 7.5 Hz, and allowed a segment stretch of ± 0.15 seconds. We cite our efficiency figures for generating, from leader motion only, a particular 150 second dance motion. Without beam search, the dynamic programming algorithm ran for 78 seconds, 2 seconds were spent on upsampling, and 26 seconds were spent on postprocessing. With beam search enabled with modest parameters, we were able to drive the runtime of the dynamic programming to 10 seconds while maintaining excellent visual and semantic results. As with our walk motion evaluation, we found that clock times scaled linearly with the length of the input.

Chapter 7

Conclusion

This thesis has presented a method for example-based performance control of human motion. The dynamic programming algorithm uses segments of motion along with an objective function that accounts for both the quality of control interpretation and the continuity of the target motion to generate visually and semantically correct motions. The semantic accuracy of the generated motion was evaluated in the setting of partner dance, where the follower's motion is generated from the leader's motion. The algorithm generated semantically correct partner motion even from test sequences of leader motions that did not appear in the training set.

The dynamic programming algorithm performs a global optimization, which precludes the local decisions that are required for online applications. However, as demonstrated in the evaluations, it can compute results significantly faster than input motion can be recorded, thus making it suitable for rapid-feedback motion authoring applications. This implies that segmental approaches like ours hold great promise for real-time performance-driven animation, and consider it a promising area of future research.

To preserve spatial dependencies in mappings, we apply rigid transformations to optimally align control segments with input control motions. Target segments inherit these transformations. This approach is effective for our applications or whenever the control signal indicates appropriate spatial and temporal cues. It is also possible to select other transformations for applications outside the domain of human motion control. For instance, allowing arbitrary homogeneous transformations in two dimensions might form an alternative segmental solution to the curve analogies problem [HOCS02]. Eliminating transformations entirely might also be appropriate for applications such as synthesis of facial motion from speech signals [Bra99].

In the process of generating target motion, the dynamic programming algorithm performs a semantically guided segmentation of the input control motion. The entire process, however, relies on the availability of semantically segmented examples. For our evaluations, we were able to perform this segmentation manually by tapping a key in response to the rhythm of music or the gait pattern of a walk cycle. While specific methods exist to automate this segmentation for the cases of dance and walk, a more general method is desirable. For this, we could begin with a few manually segmented examples and grow the set of example instances by iterative application of our algorithm. This approach would be similar in spirit to the semiautomatic SVM-based annotation approach of Arikan et al. [AFO03].

The annotation propagation we describe above suggests that our method could be used for interpretation rather than control. Paralleling our automatic annotation of handholds, it is possible to annotate any new control motion given a set of labeled example instances. This could be used to transcribe the motion into a symbolic representation, such as the one used in this paper, or even Laban notation [Hut73]. Such a representation could then be analyzed or summarized using natural language processing techniques.

Bibliography

[AF02] Okan Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Transactions on Graphics*, 21(3):483–490, July 2002.

[AFO03] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *ACM Transactions on Graphics*, 22(3):402–408, July 2003.

[BG95] Bruce M. Blumberg and Tinsley A. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. In *Computer Graphics (Proceedings of SIGGRAPH 95)*, Annual Conference Series, pages 47–54. ACM SIGGRAPH, August 1995.

[Bra99] Matthew Brand. Voice puppetry. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 21–28, August 1999.

[CGMS03] Darya Chudova, Scott Gaffney, Eric Mjolsness, and Padhraic Smyth. Translation-invariant mixture models for curve clustering. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 79–88. ACM Press, 2003.

[DB01] F. De la Torre and M. Black. Dynamic coupled component analysis. *Computer Vision and Pattern Recognition*, pages 643–650, 2001.

[DHS00] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* John Wily & Sons, Inc., New York, second edition, 2000.

[DYP03] Mira Dontcheva, Gary Yngve, and Zoran Popović. Layered acting for character animation. *ACM Transactions on Graphics*, 22(3):409–416, July 2003.

[ELF97] D. W. Eggert, A. Lorusso, and R. B. Fisher. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Machine Vision and Applications*, 9:272–290, 1997.

[FTT99] John Funge, Xiaoyuan Tu, and Demetri Terzopoulos. Cognitive modeling: Knowledge, reasoning and planning for intelligent characters. In *Proceedings of SIGGRAPH 99*, Computer Graphics Proceedings, Annual Conference Series, pages 29–38, August 1999.

[HJO$^+$01] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 327–340, August 2001.

[HOCS02] Aaron Hertzmann, Nuria Oliver, Brian Curless, and Steven M. Seitz. Curve analogies. In *Rendering Techniques 2002: 13th Eurographics Workshop on Rendering*, pages 233–246, June 2002.

[Hut73] Ann Hutchinson. *Labanotation: The System of Analyzing and Recording Movement.* Routledge, New York, third edition, 1973.

[JP99] Tony Jebara and Alex Pentland. Action reaction learning: Automatic visual analysis and synthesis of interactive behaviour. In *ICVS*, pages 273–292, 1999.

[KGP02] Lucas Kovar, Michael Gleicher, and Frédéric Pighin. Motion graphs. *ACM Transactions on Graphics*, 21(3):473–482, July 2002.

[KPS03] Tae-hoon Kim, Sang Il Park, and Sung Yong Shin. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 22(3):392–401, July 2003.

[KSG02] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 97–104, July 2002.

[LCR+02] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, July 2002.

[PB02] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: Texturing and synthesis. *ACM Transactions on Graphics*, 21(3):501–508, July 2002.

[PG96] Ken Perlin and Athomas Goldberg. Improv: A system for scripting interactive actors in virtual worlds. In *Computer Graphics (Proceedings of SIGGRAPH 96)*, Annual Conference Series, pages 205–216. ACM SIGGRAPH, August 1996.

[Rey87] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics (Proceedings of SIGGRAPH 87)*, volume 21, pages 25–34, July 1987.

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of Speech Recognition.* Prentice Hall, New Jersey, 1993.

[Stu98] David J. Sturman. Computer puppetry. *IEEE Computer Graphics and Applications*, 18(1):38–45, 1998.

[TT94] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *Proceedings of SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 43–50, July 1994.

[Vic03] Vicon. *Vicon iQ Reference Manual.* Vicon Motion Systems Inc., Lake Forest, CA, 2003.