Learning the Grammar of Dance *

Joshua Stuart
Elizabeth Bradley

CU-CS-852-98

University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

* Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.

Learning the Grammar of Dance

Joshua Stuart and Elizabeth Bradley[1]

Department of Computer Science
University of Colorado
Boulder, Colorado, USA 80309-0430

`[stuartj,lizb]@cs.colorado.edu`

Department of Computer Science
Technical Report CU-CS 852-98

In review for the
International Conference on Machine Learning

Abstract

Human motion sequences that are generated by computer algorithms may contain abrupt transitions: places where consecutive body positions would require physically impossible or stylistically illegal moves. We use graph-theoretic methods to learn the "grammar" of joint movements in a given corpus and then apply memory-bounded A* search to the resulting transition graphs — using an influence diagram that captures the topology of the human body in order to reduce the search space — to find appropriate interpolation sequences. The application that motivated the development of these methods is an algorithm that uses the mathematical properties of chaos to generate variations on dance and martial arts sequences. Chaos's *sensitive dependence on initial conditions* introduces abrupt transitions in these variations, and the goal of the intelligent interpolation schemes described here is to smooth those transitions in a kinesiologically and stylistically consistent manner.

[1]Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.

that links the movement progression and the attractor geometry, as shown in figure 1. By definition, trajectories from different starting points[3] travel along the same attractor but *in a different order*. This property lets us use the mapping depicted in figure 1 (d) to create a variation: we simply follow a *new* trajectory around the attractor and invert the symbolic mapping, "playing" the body position for each cell the trajectory enters. Variations generated in this manner, whether musical or choreographic, are both aesthetically pleasing and strikingly reminiscent of the original sequences. The stretching and folding of the dynamics guarantee that the ordering of the pitches or movements in the variation is different from the original sequence; at the same time, the fixed geometry of the attractor ensures that a chaotic variation of Bach's Prelude in C Major or of a short Balanchine ballet sequence are related to the original piece in a sense reminiscent of the classic "variation on a theme." Broadly speaking, the chaotic variations resemble the originals with some shuffling of coherent subsequences. This subsequence-reordering feature is the source of the stylistic originality of our chaotic variation scheme, but the transitions at the subsequence boundaries can be quite jarring. Figure 2, for example, shows a short section of a chaotically generated variation on a short ballet adagio. Note the abrupt transition between the fifth and sixth moves of the variation.

The interpolation algorithms that are the topic of the following sections of this paper are designed to smooth these kinds of transitions in a manner that is both kinesiologically and stylistically consistent. These graph-theoretic methods "learn" the grammar of joint movements in a given corpus and then apply memory-bounded A* search — using an influence diagram that models the relationships of the joints in the human body in order to reduce the search space — to find an appropriate interpolation sequence that bridges the gap between two body positions. The search is complicated by the fact that joint positions cannot be interpolated in isolation: the movement patterns of the ankle, for instance, are strongly influenced by whether or not the foot is on the ground — information that is implicit in the positions of the pelvis, knees, etc. This requires that the expansion of nodes in the search be context dependent in a somewhat unusual way. These approaches, which were developed and evaluated in close collaboration with several expert dancers, are quite effective at capturing and enforcing the dynamics of a given group of movement sequences.

[3]within the basin of attraction, of course

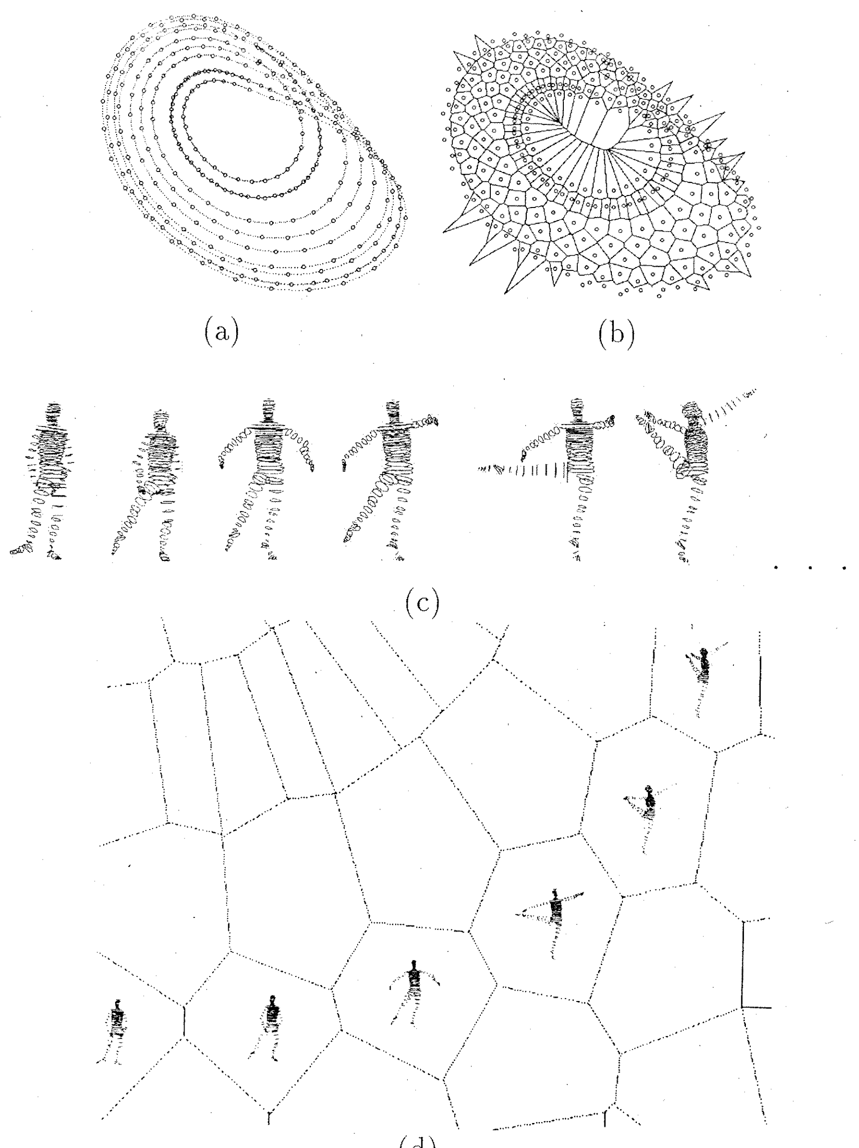

Figure 1: A chaotic mapping that links a short ballet adagio and the chaotic Rössler attractor. A Voronoi diagram algorithm is used to divide the region covered by the trajectory shown in part (a) into cells, yielding the tiling shown in part (b). The order in which the original trajectory traverses those cells defines the temporal order of the *cell itinerary* that corresponds to that trajectory. Successive body positions from the predefined movement sequence (c) are mapped to successive cells in that itinerary, linking the structure of the movement sequence and the attractor geometry. A small section of the overall mapping as shown in part (d).

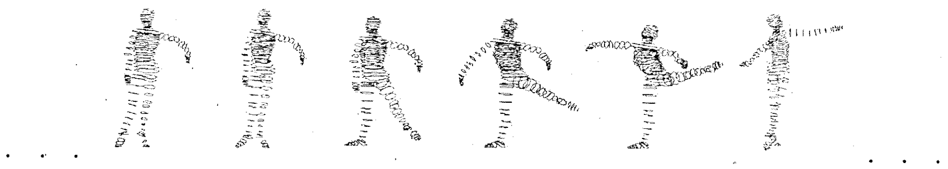

Figure 2: Part of a variation on a short ballet sequence, generated using the chaotic shuffling procedure diagrammed in the previous figure. Note the abrupt transition between the fifth and sixth frames. The goal of the interpolation schemes described in this paper is to smooth such transitions in a kinesiologically and stylistically consistent fashion.

2 Corpus-based interpolation algorithms for movement sequences

The interpolation schemes described in this section use corpora of human movement — one composed of ten Balanchine ballets, for instance, if one is working with dances of that particular genre[4] — to select a movement sequence that would naturally occur between any two positions. The basic algorithms involved are fairly straightforward, but the application requires some unusual tactics and variations. We first examine the corpus, capturing typical progressions of joint positions in a set of transition graphs. Then, given a pair of body positions, we use a variant of the A^* algorithm (and a somewhat-unusual scoring function that reflects the topology of the body) to search these graphs for interpolation subsequences. A typical interpolation sequence might, for instance, first move the shoulder from its position in the fifth frame of figure 2 to its position in the sixth frame according to the rules for *shoulder* movement that are implicit in the corpus, then repeat for the elbow, and so on. Such sequences can be inserted in order to "smooth" abrupt transitions. Our original approach[3] was much more coarse-grained; the atomic unit was *a full body position* — a representation that does not scale well with corpus size because the number of unique body positions is so large. The methods described in this paper construct the body positions in the interpolation sequence in a joint-

[4]The composition of the corpus will, of course, affect the nature of the interpolation; smoothing abrupt transitions in ballet pieces using an interpolation scheme that is mathematically rooted in a karate corpus will negate the very aesthetic resemblance that the core of the algorithm is designed to preserve. On the other hand, this might be an interesting source of innovation, whereby one could mathematically mix two or more styles.

wise manner *on the fly*; this approach is finer grained and avoids the storage problems of the previous approach.

2.1 Posture Representation

We represent a human body posture by specifying the position of each of the 23 main joints with a *quaternion*, a standard representation in rigid-body mechanics that dates back to Hamilton[10]. A quaternion $q = (r, \vec{u})$ consists of an axis of rotation $\vec{u}$ and a scalar r that specifies the angle of rotation about $\vec{u}$. Thus, a body-position symbol is quite complicated: 23 descriptors (`pelvis, right-wrist`, etc.), 92 numbers (four for each joint), and a variety of information about the position and orientation of the center of mass.

Joint orientations are, in reality, continuous variables, but computational complexity requires they be discretized in our algorithms. Specifically, each joint λ can take on a finite number M^λ of allowed orientations[5]. Formally, we define Q^λ as the set of *allowed* orientations for joint λ and then replace the *actual* orientation of the joint with the closest quaternion in Q^λ. We can express a body position $\vec{b}$ as a discretized vector $\vec{s}$ by setting each component quaternion s_λ equal to the quaternion in Q^λ that is closest to b_λ: $s_\lambda = r$ such that $||b_\lambda - r|| \leq ||b_\lambda - q||$ for all $q, r \in Q^\lambda$ where $||x - y||$ is the Euclidean distance[6] between the quaternions x and y. We can find r in $\log(M^\lambda)$ time using K-D trees[9] to represent the Q^λ sets. This procedure is analogous to "snapping" objects to a grid in computer drawing applications.

2.2 Representation of a movement corpus

2.2.1 Joint orientation graphs

A transition graph is a weighted-directed graph that captures the transition probabilities in a symbol sequence. In general, each vertex v in such a graph represents a symbol and each weighted edge (v, u) reflects the probability that the symbol associated with vertex u follows the symbol associated with vertex v. For the purposes of analyzing a human movement corpus, we build one

[5]In practice, $M^\lambda < 400$; discretization issues are addressed in more detail in section 3.

[6]One of the main features of quaternions is that they can be treated as 4-vectors in the standard norm and transformation operations.

transition graph for each joint, using the corpus to identify orentations that the joint assumes and to estimate the corresponding transition probabilities. Vertices in this kind of graph represent particular discretized joint orientations, and edges correspond to the movement of the joint from one orientation to another.

The transition graph construction procedure is fairly straightforward. We first transform every body position in the corpus to a discretized position, as described in the previous section, so that a consecutive pair of body positions $(\vec{a}, \vec{b})$, each consisting of 23 continuous-valued quaternions, becomes the discretized pair $(\vec{s}, \vec{t})$ (where $\vec{s}$, $\vec{t}$ consist of 23 *discretized* quaternions). We then build a transition graph G^λ for each joint λ that contains M^λ vertices, each of which corresponds to exactly one quaternion in Q^λ. For convenience, we will refer to vertices in G^λ by the corresponding quaternions in Q^λ. Recalling that each component in s_λ and t_λ is a quaternion in Q^λ and therefore corresponds to a unique vertex in G^λ, we record the fact that joint λ is allowed to move from a_λ to b_λ by introducing an edge in G^λ from vertex s_λ to vertex t_λ. We assign a weight to this edge that models the "unlikeliness" with which such a transition occurs in the corpus. This measure of unlikeliness is related to $P(q \rightarrow r)$, the probability that joint λ moves from the quaternion $q \in Q^\lambda$ to quaternion $r \in Q^\lambda$, per the following expression for the weight of edge $(q, r) \in G^\lambda$:

$$\begin{aligned} w^\lambda_{q,r} &= -log(P(q \rightarrow r)) = -log(P(r|q)) \\ &\approx log(C(q)) - log(C(q,r)) \end{aligned}$$

where $C(q)$ is the number of times joint λ assumed an orientation approximated by q and $C(q, r)$ is a conditional count: given that λ assumed an orientation approximated by q, $C(q, r)$ is the number of times the joint moved to an orientation approximated by r. Larger weights correspond to transitions that are less likely to occur[7].

Figure 3 shows an example: a transition graph for the pelvis that was constructed from a corpus of 38 short ballet sequences. This corpus included 919 positions, and the patterns in their progressions, as represented by the topology of the graph, are obviously quite complex. In the interests of clarity, edge weights and isolated vertices have been omitted from this figure.

[7]Given this formulation, saying that two vertices are disconnected is synonymous with saying that two are connected by an edge with infinite weight.

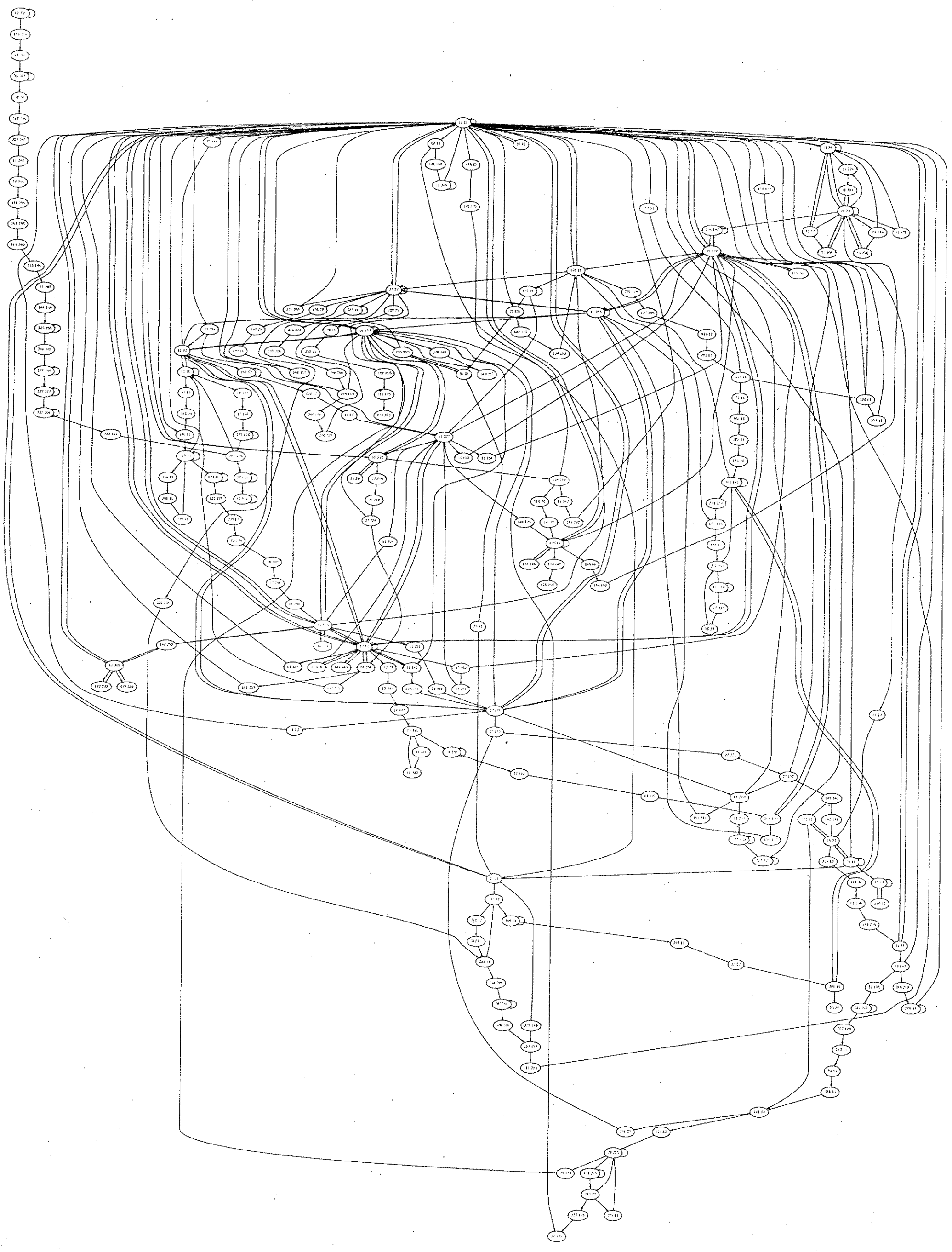

Figure 3: This transition graph represents the movement patterns of the pelvis in a small corpus of 38 short ballet sequences. The numbers in each state identify the discretized position of the joint.

2.2.2 Coordinating joint movements

Joint transition graphs represent the behavior of individual joints *independent of one another*. This information alone cannot capture the physical constraints that govern the interrelationships of the joints in the body. For example, if the shoulder is in its resting position with the palm facing the thigh, the elbow can bend nearly 180 degrees, but if the shoulder is turned 90 degrees on its long axis (until the palm faces backwards), the elbow can only bend about five degrees before the hand collides with the leg. Joints are not influenced by *every* other joint; the position of the wrist, for instance, strongly affects the position of the fingers but has little effect on the toes. Briefly put, we need a simple and efficient model of human joint coordination in order to construct sensible interpolation sequences.

In order to reduce the search space, we use a simplified model that explicitly represents the relationships of the joints to one another — a type of *influence diagram*[12] that reflects the structure and physics of the human body. As shown in figure 4 (b), the nodes (joints) in the tree only affect their immediate children. The pelvis is the root of this tree; three branches lead from this root

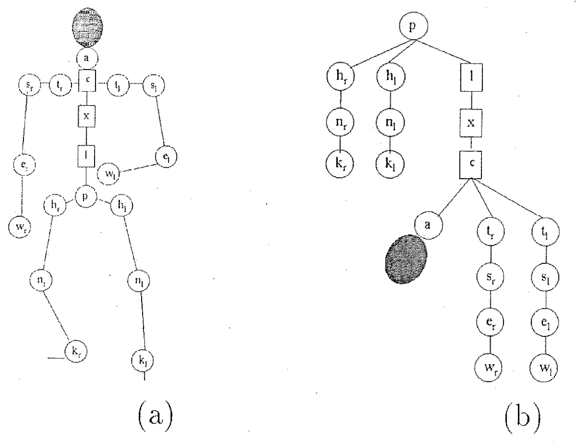

Figure 4: An influence diagram that explicitly represents the coordination of joints of the human body. Part (a) depicts the body and part (b) shows the inter-joint dependencies induced by gravity and topology: for instance, the position of the pelvis influences the positions of both hips h_r and h_l and the lumbar spine l, but the right and left ankles k_r and k_l do not directly influence one another.

to nodes corresponding to the right thigh/hip joint, the left thigh/hip joint, and a joint representing the lower spine[8]. Each hip joint is the parent node to a knee, and so on. We assign a conditional probability distribution, estimated from the corpus, to every (parent,child) pair in the tree. For every combination of states that a parent λ and its child μ can assume, the distributions estimate the probability that joint μ is in orientation r given that joint λ is in orientation q, for every pair of discretized of quaternions $q \in Q^\lambda, r \in Q^\mu$.

2.3 Joint position interpolation algorithms

Given a pair of discretized body postures $(\vec{s}, \vec{t})$ and a set of joint orientation transition graphs constructed from a corpus of movement as described in the previous section, we can use a memory-bounded A* search strategy[14] to find an interpolation subsequence that moves smoothly between $\vec{s}$ and $\vec{t}$. In general, A* finds a path from an initial state to a goal state by progressively generating successors of the current state in the search. The algorithm places successor states on a priority queue, sorted according to a score that estimates the cost of finding a goal state. In the next iteration, the state with the best score is drawn from the priority queue, its successor states are computed and added to the queue, and the procedure is repeated until a goal state is found or until the queue is empty.

The states in this search space are *body states* — 23-vectors of discretized quaternions that represent full body positions. To generate successors of a body-state $\vec{s}$, we first use the transition graphs to find successors for each *joint-state* s_λ independently, and then take all combinations across the joints (cross-product) to obtain the list of body-state successors. The successors of the joint-state, s_λ are those vertices in G^λ that are connected to s_λ by an edge directed away from s_λ.

The score assigned to a body-state $\vec{u}$ has two parts: (1) the cost of the path from the initial state $\vec{s}$ to $\vec{u}$ and (2) an estimate of the distance between $\vec{u}$ and the goal state $\vec{t}$. The cost of the path starting at $\vec{s}$ and ending at $\vec{u}$ is simply the sum of the costs of the transitions taken in the path. Furthermore, since each body movement is composed of a group of joint movements, we can compute the cost of one body-state transition by summing the weights over the

[8]The sacrum and the five lumbar vertebrae are lumped together. This compromise sacrifices back suppleness for lowered complexity.

edges traversed by the joints. To make this concrete, suppose we are trying to find an interpolating path between the body states $\vec{s}$ and $\vec{t}$. At some point in the search, we reach the body-state $\vec{u}$ and must assign the path from $\vec{s}$ to $\vec{u}$ a score. If we write the path from $\vec{s}$ to $\vec{u}$ as $\vec{s} \rightsquigarrow \vec{u} = (\vec{s} = \vec{x}^1, \vec{x}^2, \cdots, \vec{x}^{z-1}, \vec{u} = \vec{x}^z)$, we can express the cost of such a path as

$$g(\vec{s} \rightsquigarrow \vec{u}) = \sum_{i=1}^{z-1} \sum_{\lambda} w^{\lambda}_{x^i_\lambda, x^{i+1}_\lambda}$$

The heuristic part of the score, $h(\vec{u})$, estimates how far $\vec{u}$ is from the goal state $\vec{t}$. $h(\vec{u})$ is calculated by summing the weights of the shortest paths from u_λ to t_λ, $u_\lambda, t_\lambda \in G^\lambda$ over all the joints. We obtain these shortest path weights using Dijkstra's single-source shortest path algorithm[8], implemented as described in [5]. The final score assigned to body-state $\vec{u}$ is then $f(\vec{s} \rightsquigarrow \vec{u}) = g(\vec{s} \rightsquigarrow \vec{u}) + h(\vec{u})$.

At the time of this writing, we have only tested a greedy search strategy that ignores the cost of paths and scores nodes in the search based solely on the estimated distance between them and the goal (i.e., $f(\vec{s} \rightsquigarrow \vec{u}) = h(\vec{u})$). In the following section, we describe the implications of this strategy and suggest how different A* scoring functions are likely to affect the interpolation sequences. We are working on incorporating more information about the position, velocity, and acceleration of the center of mass, so the momentum of the body is conserved as it passes through the interpolated sections of the movement. Finally, we are also in the process of testing how different influence diagram topologies affect the interpolation algorithm's ability to select good postures during the search. (For example, to model the symmetry of the body, we could combine left and right counterparts into one node.)

3 Results and evaluation

Figure 5 shows the results of applying the learning and search algorithms in the previous section — with a greedy search strategy: an A* score $f(\vec{s} \rightsquigarrow \vec{u}) = h(\vec{u})$ that only factors in the distance to the goal — to smooth the abrupt transition between the fifth and sixth frames of figure 2. The starting and ending body postures (top left and top right in figure 5) are quite different; note the facing of the dancer and the weight distribution on the feet, for example. The eight-move interpolation sequence shown linking these two positions fills the gap in a very natural way. Its first move, for instance, is to lower the

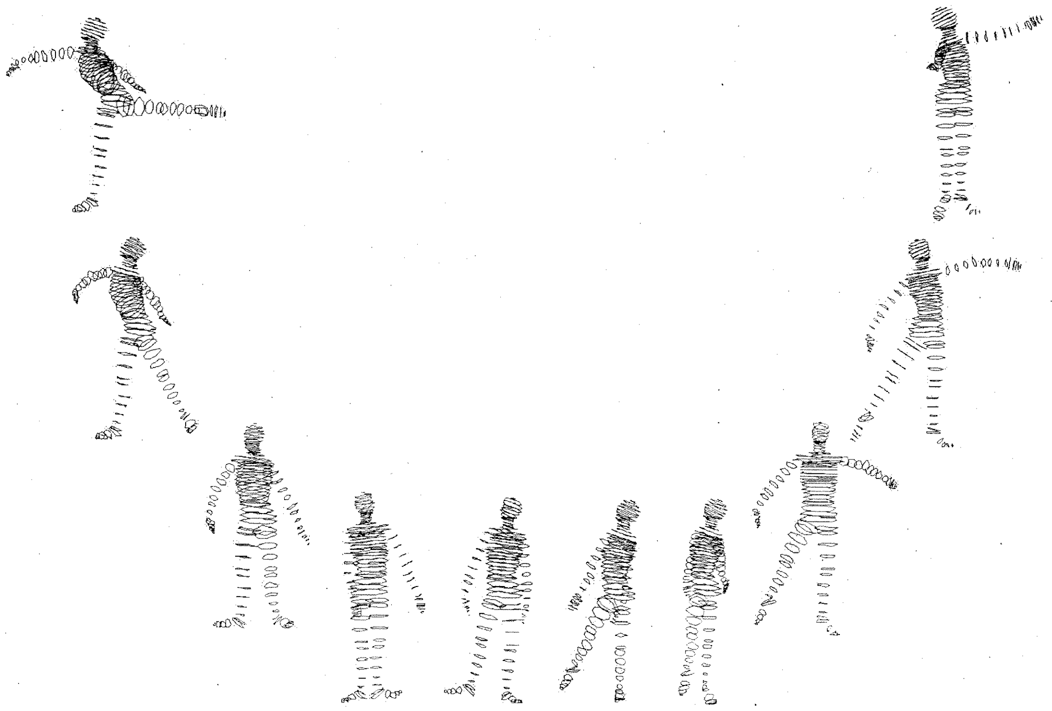

Figure 5: An interpolation sequence computed by the corpus-based techniques described in the previous section. The starting and ending positions passed as input to the interpolation procedure — the fifth and sixth frames of figure 2 — are shown at the top left and top right, respectively; the eight frames in the semicircle below them were computed by the interpolator.

left leg, a natural strategy if one is going to change one's facing and end up on two feet. The following move is a simple weight shift (fourth and fifth frames), in preparation for a lift of the right leg. This lift, which is not strictly necessary to move from the fifth frame to the tenth, is an innovation that the program inserted because of the observed patterns in the corpus: rarely do ballet dancers spin with *both* feet on the ground. Perhaps the most interesting thing about this interpolation sequence, from a balletic standpoint, is the relévé[9] that the interpolation procedure inserted between frames seven and ten. There are many relévés in the corpus, but none whose associated upper body positions even remotely resemble the one in this sequence; our interpolation algorithm has invented a physically *and stylistically* appropriate way to move the dancer between the specified positions. This sequence contains a variety of other stylistically consistent innovations; consider, for example, the uplifted chest and chin in the second and ninth frames — posture elements that are quintessential ballet style. Again, recall that these postures were not simply pasted in verbatim from the corpus; they were synthesized *piece by piece* by the transition graphs via influence-diagram directed A* search, and their fit to the genre is strong evidence of the success of the methods described in the previous section.

The greedy search strategy of the algorithms is reflected by several "inefficiencies" in this sequence — places where the dancer appears to be headed towards the goal state, but then moves away. For example, one of the interpolation goals is to change the facing almost 180 degrees. By the fourth frame, the dancer has turned to the right, but in the fifth frame s/he turns back to the left again, which is part of what necessitates the relévé sequence discussed in the previous paragraph. We are in the process of testing different search strategies and analyzing the results; instead of choosing the state that is closest to the goal, for instance, we are incorporating the path weights up to the current point in the solution as part of the scoring function. This should allow the search algorithm to find shorter, more-direct sequences. Finally, note that some search strategies — e.g., always taking the highest-probability branch — can be a significant source of cliché.

The original ballet sequence from which this snapshot is drawn contained 68 frames, and the chaotic shuffling scheme introduced eight abrupt transitions into the variation. In all eight cases, our interpolation scheme was successful in

[9]A relévé consists of lifting up on one's toes before transferring one's weight, and it is a stylistically required component of a direction shift.

smoothing these gaps. The interpolating subsequences so constructed, which range in length from two to 60 frames, include a variety of stylistically consistent and often innovative sequences; among other things, the interpolation routine used relévés, pliés and fifth-position rests in highly appropriate ways — and all with no hard coding. Many of the longer interpolation sequences were somewhat stilted: in one, for example, the dancer spent 30+ frames performing small arm movements. This is almost certainly a side effect of the small size of the corpus. In a large, rich corpus, the search algorithms would have more leeway; in the existing corpora, however, the paucity of edges constrains them to very narrow (and long) search paths that translate to idiosyncratic sequences like arm-waving.

The whole procedure is quite rapid; the 60-frame interpolation sequence, for instance, required[10] 30 seconds on an HP9000/735 workstation running HP-UX v10.20. The chaotic shuffling procedure is also fast: for a 1000-position movement sequence, the chaotic shuffling procedure required 18 seconds on the same workstation, while a 9000-move sequence required 156 seconds.

Deriving a successful discretization of joint states was unexpectedly difficult. Simply discretizing the quaternion variable values — that is, classifying all positions between, say, `(right-wrist, 1, 1, 0, 1)` and `(right-wrist, 1, 1, 0.2, 1)` as an equivalence class and representing them in the algorithms as a single posture — produced visibly awkward animations. The human visual perception system appears to be very sensitive to small variations in quaternion coefficients: any change in a single coefficient seems to violate the "motif" of the motion. The same problem arose when we attempted a physically more-realistic discretization by transforming quaternion data to Euler angles and then discretizing θ, ϕ, and ψ instead. The solution on which we eventually settled uses a discretization library that was created by hand by an expert dancer.

4 Conclusion

By applying techniques from graph theory, artificial intelligence, and statistics to a corpus of movement sequences from a particular genre, the interpolation methods described here — which were developed in close cooperation with

[10]This will obviously depend on the positions involved.

several dancers – smooth awkward body-posture transitions in a physically *and stylistically* coherent fashion. Evaluation of these results is necessarily somewhat subjective. We have shown animations of these results to hundreds of people, including dozens of dancers and martial artists. The consensus is that the variations not only resemble the original pieces, but also are in some sense pleasing to the eye. They are both different from the originals and faithful to the dynamics of the genre; there are no jarring transitions or out-of-character moves. This is a non-trivial accomplishment. A previous attempt to use mathematics to generate choreographic variations — a subsequence randomization scheme introduced by the now well-known choreographer Merce Cunningham in the 1960s — met with a strongly negative reception in the dance world, *primarily because of the awkwardness at the transition points*[11].

Many of the techniques used here, as well as others on which we are currently working, were inspired by solutions to similar problems that arise in molecular biology (e.g., DNA sequence analysis) and computational linguistics (e.g., learning a grammar from a corpus and then using it to construct meaningful sentences). For example, one can view the transition graphs in section 2.2.1 and figure 3 as first-order Markov chains, where a single chain represents the probabilistic behavior of each joint in the body. These ideas and techniques presented in this paper can be extended to other domains where the genre of sequence is important, such as text[12]. Finally, the implementation allows for arbitrary body topologies, so this scheme is by no means limited to *human* motion sequences — though one would, of course, have to adapt the quaternion-based symbol set and the influence diagram to the topology of the limbs and joints that are involved.

Acknowledgements: Stephen Schroeder discretized each joint's position by hand, and Eric Schell and Meenakshy Chakravorty helped set up the website and write the associated animation code. The authors would like to thank David Capps and Nadia Rojasadame of the University of Colorado Department of Theater and Dance and Dan Jurafsky of the University of Colorado Department of Linguistics for helpful conversations, Margo Seltzer of Harvard University for suggesting and demonstrating the karate sequences, and Diana Dabby of and Apollo Hogan for numerous fruitful and thought-provoking suggestions.

[11]Since that time, *aleatory* choreography — wherein randomization schemes are used to shuffle sequences — "has by now become one of the important currencies of dance composition approaches."[2]

[12]For instance, we ran our code on a collection of Shakespearean sonnets, with some interesting results that will be covered in a forthcoming paper.

References

[1] http://www.cs.colorado.edu/~lizb/chaotic-dance.html.

[2] David Capps, University of Colorado Department of Theater and Dance, personal communication.

[3] E. Bradley and J. Stuart. Using chaos to generate choreographic variations. In *Proceedings of the Fourth Experimental Chaos Conference*, 1997.

[4] E. Bradley and J. Stuart. Using chaos to generate variations on movement sequences. *Chaos*, 1998. In review.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990. pp 527-531.

[6] D. S. Dabby. Musical variations from a chaotic mapping. *Chaos*, 6:95–107, 1996.

[7] D. S. Dabby. A chaotic mapping for musical and image variation. In *Proceedings of the Fourth Experimental Chaos Conference*, 1997.

[8] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[9] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

[10] H. Goldstein. *Classical Mechanics.* Addison Wesley, Reading MA, 1980.

[11] J. K. Hodgins, W. L. Wooten, D. C. Brogan, and J. F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH*, 1995.

[12] R. M. Oliver and J. Q. Smith, editors. *Influence Diagrams, Belief Nets and Decision Analysis.* Wiley, 1990.

[13] C. M. Software. http://fas.sfu.ca/lifeforms.html.

[14] P. H. Winston. *Artificial Intelligence.* Addison Wesley, Redwood City CA, 1992. Third Edition.

[15] W. L. Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing.* PhD thesis, Georgia Institute of Technology, 1998.