Interactive Build-Up of Animation Sequences with Captured Motion Data

Sang Won Lee
Kwangwoon University
231@kw.ac.kr

Kang Hoon Lee
Kwangwoon University
kang@kw.ac.kr

Abstract

We present a new approach to interactive resequencing of captured motion data that allows the user to progressively extend his/her animations such that each extension can best fit user requirements with respect to space, time, and pose. In the preprocessing phase, captured frames are clustered into pose groups, and inter-connected into a transition graph. For each step in building an animation sequence at runtime, the user is allowed to browse through motion paths in the graph that are connectable to the existing sequence by specifying the desired locations and poses at path ends, and to select one of those paths to extend the animation. Instead of using graph search algorithms, our path browser unrolls the graph locally into spatial search trees and employs an efficient nearest-neighbor search method to quickly find the optimal paths that satisfy user inputs. We demonstrate the usefulness of our approach by creating animation examples in scenarios involving spatial, temporal, and postural requirements simultaneously.

Keywords: computer animation, motion capture, interactive editing

1 Introduction

Motion capture technology has opened new possibilities for enabling common people with no artistic skills to synthesize natural and realistic animation sequences of moving figures. Captured motion data can not only be used in its original form, but also be processed into various new shapes and sequences without recognizable loss of motion quality. Temporal resequencing methods can play back the original motion data in new orders by allowing transition between any two frames of similar poses. This process is often facilitated by representing motion data in a graph structure in which each node corresponds to a frame, and directed edges encode transitional relationships between frames. Then, any arbitrary paths on the graph can correspond to valid rearrangements of the original motion.

Graph representation reduces the task of animation authoring to a task of path selection, which is well-suited to automated computation. Given user-specified constraints on new animation sequences such as initial and final locations of characters, various graph search algorithms can be employed to find optimal paths under predefined optimality measures, and convert those paths into smoothly connected motions. However, such an optimization-based approach requires significant amount of computation, and may even fail if a large number of constraints are simultaneously given to describe the desired animation in detail. For example, if the user specifies a large number of way-points, each of which is associated with desired time and pose, any possible resequencing of the original motion could not satisfy all of these conditions. As a result, the user should iteratively adjust constraints and examine the corresponding results until satisfactory animation sequences are obtained.

Instead of constructing the entire sequence at one time, we present an alternative approach to

interactive motion resequencing that allows the user to progressively extend his/her animation sequence with input motion data. The input motion is preprocessed to obtain groups of similar poses and a graph representing motion transition (see Section 3). For each extension of an animation sequence, the user selects a pose from the pose groups, and connects the pose to the last pose of the existing sequence by one of the available motion paths in the graph (see Section 4). The user can quickly browse through a large number of connectable paths by specifying the desired location of the new pose, and examining the spatial trajectory and temporal interval of the best path whose end position is nearest to the input location. We maximize the responsiveness of this interaction cycle by unrolling the graph over the surrounding space and employing an efficient nearest-neighbor algorithm. Figure 1 shows the experimental user interface that supports our overall procedure in a seamless and intuitive way.

The main advantage of our approach is that the user can rapidly build up his/her intended animation sequences in a manner similar to assembling construction toys, while precisely controlling details in the space-time domain within the range of possible motion resequencing. In our experiments with boxing and breakdancing motion data, the usefulness of our approach is demonstrated by showing that target animations can be easily created through a small amount of interactive trial and error in the predefined scenarios involving spatial, temporal, and postural requirements (see Section 5). We expect that our approach would be highly effective for movie pre-visualization, dance choreography, and visual storytelling.

2 Related Work

For the last decade, researchers in computer graphics have explored various ways to synthesize the motion of animated characters by using captured motion data. Recent techniques allow us to broaden the range of synthesizable motions by deforming, blending, and resequencing original motions [1, 2, 3]. Furthermore, these low-level motion editing operations are often

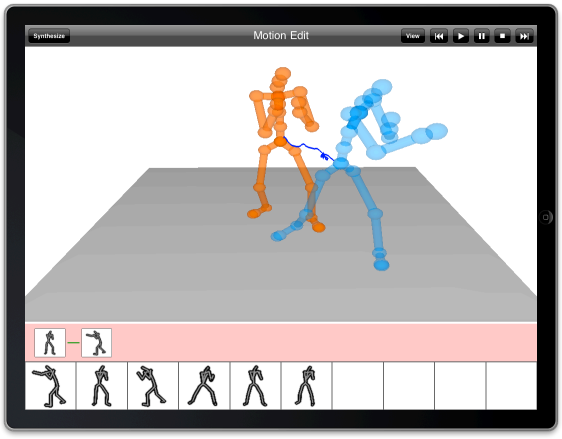

Figure 1: The user interface. For each extension of an animation sequence, the user selects a pose from the collection at the bottom of the screen, inserts the pose into the pink timeline, browses through connectable motion paths in the large panel, and accepts one of those paths by tapping the button at the top left.

combined with high-level descriptions of character behaviors such as navigation [4], interaction [5], and competition [6] to facilitate interactive or automatic generation of non-trivial animation sequences, involving a large number of characters and complex environments.

Temporal resequencing of motion data is one of the most important approaches in data-driven animation synthesis due to its capability of creating new narrative flows that are completely different from captured scenarios. Graph representation has been an effective means of encoding the transitional relationships between motion frames. Automatic construction of small-sized and richly connected graphs has been focused on to save memory space and improve computational efficiency [7, 8]. In addition, the original graph structure has been extended to increase the representational power with respect to human recognition [9] and motion synthesis [10]. Animation sequences of user-controlled or autonomous characters are often constructed by combining these graph structures with various search algorithms [11, 12]. Instead of using graph search techniques to obtain the entire animation sequence at once, we allow the

user to progressively compose his/her animation with the support of our path browser that searches for and displays locally connectable paths in response to the user-specified location and pose by using an efficient nearest-neighbor search method.

It has been an interesting topic in computer animation for novices to intuitively create their intended animation sequences of articulated characters. Davis et al. presented a sketch-based interface in which the user draws a series of stick figures in two-dimensional space to obtain articulated-body motion in three-dimensional space through pose estimation and key-frame interpolation [13]. Focusing mainly on locomotion such as walking and jumping, the sketch-based interface developed by Thorne et al. takes a curved path as its input instead of key poses and maps the style of each segment of the path to a predefined locomotion type to create a stylized moving sequence along the path [14]. As an alternative to the well-known key-frame interface, Igarashi et al. suggested a spatial key-framing in which key poses are arranged in the space domain, rather than in the typical time domain, so that the user can design interpolated motions by moving the location of a control cursor in the space [15]. With the same goal as in these previous methods, we provide the user with an intuitive interface in which only the iterations of selection and connection can construct a realistic animation sequences as intended.

3 Preprocessing Data

Given input motion data $\{\mathbf{x}_t | 1 \leq t \leq T\}$ in which T consecutive frames of articulated poses $\mathbf{x}$ are included, we preprocess the data to form K groups of similar poses and a graph representing allowable transitions among motion subsequences. Both processes require a common measure of the similarity between any given two poses $\mathbf{x}_i$ and $\mathbf{x}_j$, which is calculated by the distance function suggested by Lee et al. [4], with the positional difference intentionally removed from the original equation for supporting only the relative coordinate system, as follows.

$$\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j) = \sum_{m=1}^{M} \omega_m ||log(\mathbf{q}_{i,m}^{-1} \mathbf{q}_{j,m})||^2 \quad (1)$$

where $\mathbf{q}_m \in \mathbb{S}^3$ is the orientation of the m-th joint. The weight value ω_m controls the relative contribution of the m-th joint, which was fixed to a constant value for every joint in our experiments.

3.1 Pose Clustering

Each iterative addition of short motion sequences starts by choosing a desired pose from an array of example poses. Displaying every pose in the input motion to the user makes the selection difficult because even a second of motion consists of thirty or more poses, and many of those are highly similar to each other. In order to summarize a large number of poses as a small set of representative poses, we sort poses into groups and exhibit only the central pose of each group. The *K-means* algorithm is employed to cluster poses by minimizing the following error function.

$$\mathrm{E} = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \mathrm{d}(\mathbf{x}_k^i, \mathbf{x}_k^c) \quad (2)$$

where K is the number of groups, N_k is the number of poses belonging to the k-th group, and $\mathbf{x}_k^c$ and $\mathbf{x}_k^i$ represent the central and i-th pose of the k-th group, respectively. In order to minimize this function, we initially choose arbitrary K poses as the centroids, and then alternately update the association of poses with groups and the centroids of groups until convergence. When updating the group membership, each pose is inserted into the closest group whose central pose is the most similar to the pose. Then, the centroid of each group is redetermined as the pose whose sum of the distances to other poses in the same group is the lowest. Once every pose has been grouped, each t-th pose is associated with the index of its group, which is denoted as $\mathrm{k}(t)$ in the remainder of this paper.

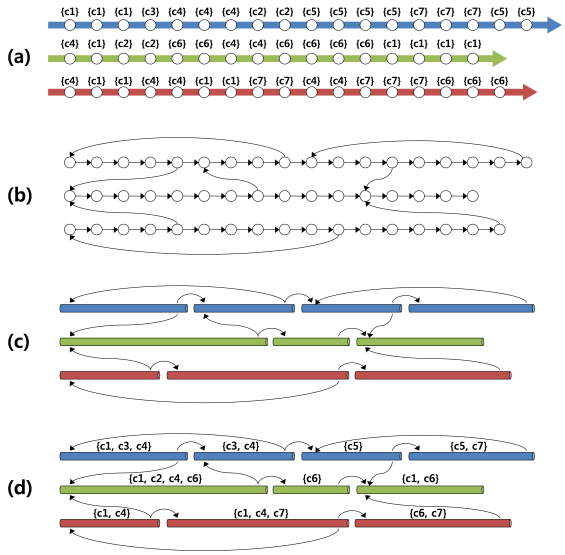

Figure 2: The process of our graph construction.

3.2 Graph Construction

We build a directed graph structure in which each node $\mathbf{n}$ corresponds to an interval $[s, e]$ of the input motion and each edge of an ordered pair of two nodes $(\mathbf{n}_i, \mathbf{n}_j)$ represents that the previous interval $[s_i, e_i]$ can be smoothly connected to the next interval $[s_j, e_j]$ via the similar poses at e_i and s_j. In order to accelerate the process of searching for motion paths at runtime, each node $\mathbf{n}$ is additionally annotated with a set of the pose groups included in its interval $\{\mathrm{k}(t)|s \leq t \leq e\}$. The size of this set is usually much less than the number of frames $(e - s + 1)$ because consecutive poses are highly probable to belong to the same group.

We first create nodes $\mathbf{n}_t$ of a single frame $[t, t]$ for every frame $1 \leq t \leq T$ (see Figure 2 (a)). The initial connectivity over the nodes is determined with the support of an affinity matrix $\mathbf{M}$ whose elements are the distances between every pair of frames $[\mathrm{d}(\mathbf{x}_i, \mathbf{x}_j)]_{i,j=1,\cdots,T}$ (see Figure 2 (b)). Specifically, any two nodes $\mathbf{n}_i$ and $\mathbf{n}_j$ are connected if $\mathbf{M}_{i,j}$ is a local minimum and less than an experimental threshold value. The threshold balances between the flexibility and quality of the resulting graph. In other words, a large threshold yields a richly connected but poor quality motion graph, and vice versa. To avoid dead ends, we run Tarjan's algorithm to find strongly connected components and leave only the largest component [16].

The runtime performance of our interactive path searching is inversely proportional to the number of nodes in the graph, because we unroll the graph nodes locally in the space for each new pose and search for the nearest node from the user-specified location. We reduce the number of nodes by merging adjacent nodes of only a single choice of transition to minimize memory space and runtime computation (see Figure 2 (c)). Specifically, if $\mathbf{n}_i$ and $\mathbf{n}_{i+h}$ in the initial graph have more than one or no incoming and outgoing edges, respectively, and each of their in-between nodes $\mathbf{n}_{i+1}, \cdots, \mathbf{n}_{i+h-1}$ has just one incoming edge from its previous node and one outgoing edge to its next node, we combine those h nodes into a single node inheriting only the incoming edges of $\mathbf{n}_i$ and the outgoing edges of $\mathbf{n}_{i+h}$. Finally, each combined node is annotated with the set of pose groups included in the merged interval (see Figure 2 (d)).

4 Interactive Sequencing

In our editing interface, the user progressively extends an animation sequence by repeatedly selecting and appending one of the representative poses to the animation such that each new pose is connected to its previous pose by one of the available motion paths in the given graph. Such a combinatorial approach intrinsically limits the range of possible motions, and does not allow poses to be placed at arbitrary positions and orientations, except for the starting pose.

In order to enable the user to adaptively compromise between his/her intention and the allowable range, we provide the user with a functionality of quickly browsing through a large number of connectable paths. Specifically, whenever the user points to a new location to be examined, our path browser immediately responds by displaying an optimal path whose end is the nearest to the input position. The responsiveness of this interaction cycle is maximized by unrolling the graph into a spatial tree for each new pose, and then running a nearest-neighbor search for each location query (instead of graph search).

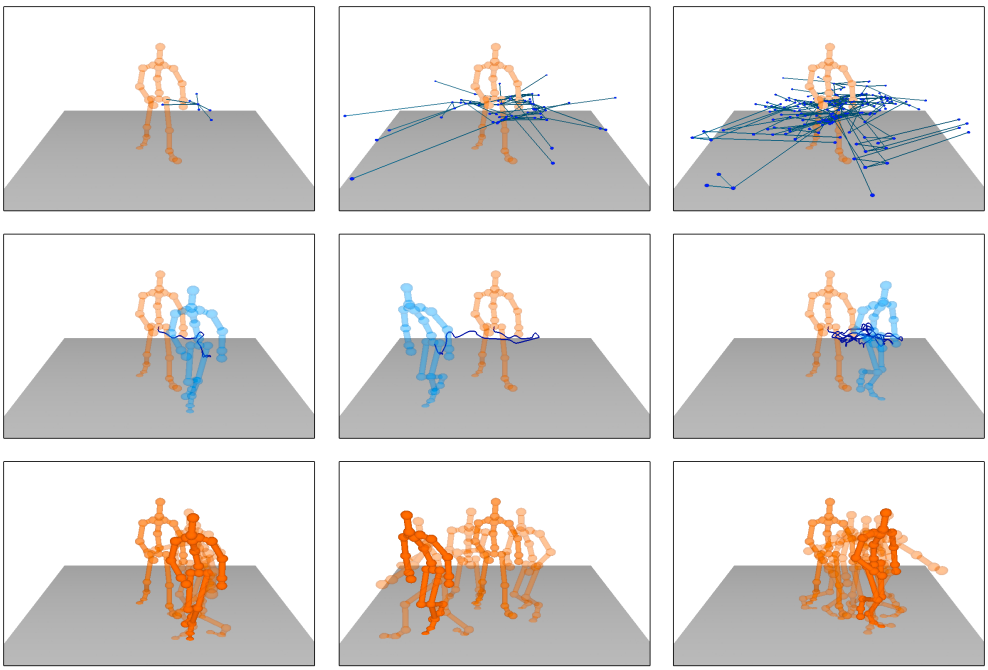

Figure 3: (top) Spatial search trees of depths 1, 3, and 5 from the left. The dots and lines represent the nodes and edges, respectively. (middle) Browsing through connectable motions with various locations. Our path browser takes the pose and location of the blue character as its input, and visualizes the trajectory of the optimal motion path as its output. (bottom) Accepting one of the available paths synthesize smoothly connected motions along the path.

4.1 Search Tree

We unroll the graph into the space from the last node for the current animation sequence within a depth d, while creating a new tree node for each unrolled graph node. Each tree node inherits the original information of its corresponding graph node including the motion interval and the indices of pose groups. Additionally, it is associated with the two-dimensional position and orientation at which the character would be placed and the time when the character would arrive. Our spatial search tree is similar to the embedded graph by Reitsma and Pollard [17]. However, our tree is repeatedly rebulit whenever the animation is extended and covers a relatively small region surrounding the last pose with a small d, about 5 in our experiments, because we use the tree for fast browsing of possible animation build-up rather than to evaluate motion graphs.

Because the number of tree nodes grows exponentially with respect to the depth d, brute-force linear search for the nearest-neighbor nodes is impractical (see Figure 3 (top)). Instead, we use a well-known search algorithm in which a precomputed data structure called the cover tree accelerates the search process in response to the nearest neighbor query significantly [18]. In our implementation, the cover tree is constructed once for each new search tree, which takes a few seconds of computation for tens of thousands of tree nodes. Once the precomputation has been done, each location query can be responded to in an instant (see Table 1).

4.2 Path Browsing

Given a two-dimensional location $\mathbf{p}$ of a new pose belonging to the k-th group, we first find n closest tree nodes whose Euclidean distances to $\mathbf{p}$ are within a radius r by using the cover tree algorithm. In our experiments, we chose n and r to be 5 and about half of the character's height, respectively. Then, only the nodes associated with the k-th pose group are picked out as candidates satisfying the user requirement. If there remains only one or no candidates, we sim-

ply accept the candidate as the user selection or ignore the user input, respectively. Otherwise, given more than one candidate node, we choose the temporally closest node, to which it takes the shortest time from the last pose of the existing animation, by comparing the times associated with candidate nodes.

Once one of the tree nodes has been identified as the user selection, we trace the tree back to the root node while constructing the motion path in reverse order such that the motion intervals from the root node to the selected node are threaded together as in $\langle [s_1, e_1], [s_2, e_2], \cdots, [s_{d'}, e_{d'}] \rangle$, where the first and $d'(\leq d)$-th intervals are associated with the root and selected node, respectively. We need to carefully adjust the first and last (d'-th) intervals to their sub-intervals before motion synthesis. For the first interval, the start frame s_1 should be updated as the next frame of the last pose of the existing animation to satisfy continuity. The end frame $e_{d'}$ of the last interval should be altered to belong to the user-intended k-th pose group. Note that the last interval $[s_{d'}, e_{d'}]$ may include such poses at several different frames. We determine $e_{d'}$ to be the first frame belonging to the k-th group in favor of the shortest extension without redundancy.

The adjusted motion intervals are then stitched together to form a long, smoothly connected animation sequence by transforming each interval to be aligned with the end of its previous interval, and blending poses around each pair of transition frames e_i and s_{i+1}. As feedback on the user input, we visualize the spatial trajectory and temporal duration of this new sequence (see Figure 3 (middle) and (bottom)). If the user accepts the result, we proceed to the next round of animation build-up by appending the new sequence to the existing animation and reconstructing the search tree from the last node for the new sequence. Otherwise, the user continues examining other connectable paths by moving the input location until a satisfactory path is obtained.

5 Experimental Results

We implemented our interactive editing system on a second generation iPad to provide the user with an easy rapid interaction using the iPad's multi-touch screen. The user is allowed to select poses, add poses in the time line, browse motion paths by moving new poses in the space, and accept motion paths to extend his/her animation sequence, using only tap and swipe gestures. In order to demonstrate the effectiveness of our approach, we preprocessed captured data of boxing and breakdancing into motion graphs and pose clusters, and interactively built up new animation sequences under the predefined scenarios involving spatial, temporal, and postural requirements simultaneously. Please refer to our accompanying video for visually examining the process of building up animation sequences in detail.

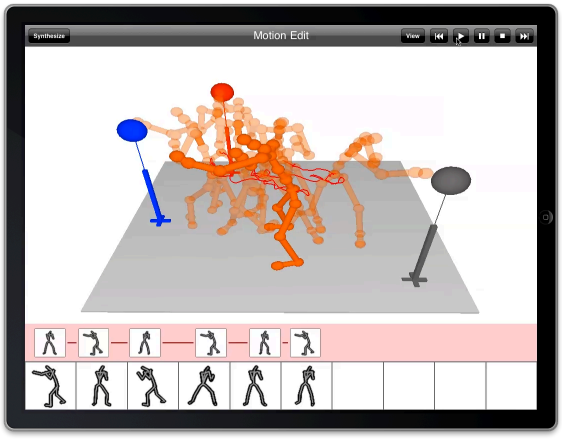

Figure 4: Boxing.

Boxing data comprises a broad variety of combinations of footwork and punches performed by a professional boxer. We arranged three punch targets of different colors at arbitrary locations, and specified an animation scenario requiring the character to approach and throw punches at those targets in a specific order (see Figure 4). To confirm whether each target was hit or not, we changed its color to gray if its distance from the character's hand became less than a threshold distance, about half of the character's arm length. The user was able to quickly link available motions together to satisfy the given requirements in a few minutes, while flexibly controlling the complexity and length of the intermediate motions between each pair of successive targets as intended.

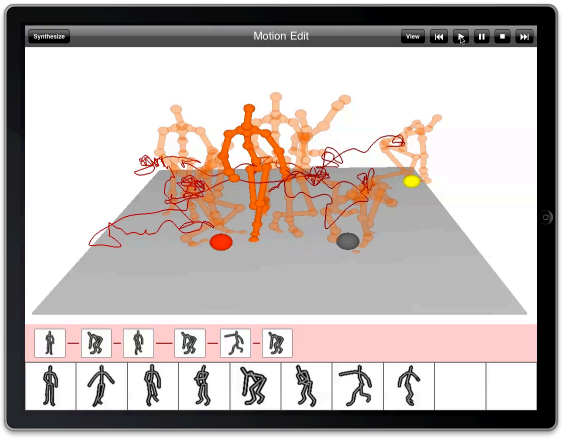

Figure 5: Breakdancing.

Breakdancing data captures a free-style performance of a professional dancer, which includes various combinations of footwork and upper body gestures. As in the boxing animation, we placed three targets of different colors at arbitrary locations on the floor, and required the character to bend its body to touch its toes at the target locations as if trying to pick up the targets (see Figure 5). Because the variation of the poses were more wide and dynamic than with the boxing, the frequency of allowable transitions between similar poses became lower, producing the longer motion intervals associated with graph nodes in average. As a result, the motion paths available in our search trees were generally very complicated and sparsely distributed in the space, which makes precise control over animation sequences challenging. However, the user successfully achieved the goal through a few number of trials and errors with the help of our location-based browsing interface.

6 Conclusion

We have presented a new approach to animation authoring using graph representation of captured motion data in which the user is able to progressively extend his/her animation sequences in specific scenarios. In comparison with the existing approaches that mainly focus on automatic optimization over every available motion path, our approach delegates the global composition of animations to the user by allowing the user to browse through locally connectable motion paths to the existing sequences and to choose the best paths to extend the animations. Whenever our path browser receives the desired location and pose, it efficiently finds a locally optimal path ending with the pose at the nearest location and displays its spatial trajectory and temporal duration, which enables the user to quickly and flexibly choose among a diverse array of options to meet spatial, temporal, and postural requirements.

One of the major drawbacks of our approach is that an animation sequence can only be extended or shrunken at the end. The insertion or deletion in the middle of a sequence would significantly transform the latter part of the original animation, possibly breaking the flow with the surrounding environment or the planned scenario. Also, as an intrinsic limitation of the combinatorial approach, it is often impossible to precisely fulfill the spatial and temporal requirements, such as hitting a target at a specific time, within the range of possible resequencing. For more precise control, we regard it as a promising future direction to incorporate well-known techniques for deforming motion paths into our framework [19].

Acknowledgements

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (No. 2012-0002241).

References

[1] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48, 1999.

[2] Lucas Kovar and Michael Gleicher. Automated extraction and parameterization of

Motion data	Tree \depth	1	3	5	7	9
Boxing	# of nodes	8	47	389	4442	46340
	construction (s)	0.000559	0.004731	0.041086	0.634487	8.901384
	each query (ms)	0.163	0.191	0.3	0.416	0.599
Breakdancing	# of nodes	8	50	367	2591	18652
	construction (s)	0.000582	0.002936	0.027238	0.246508	2.338281
	each query (ms)	0.117	0.152	0.22	0.316	0.403

Table 1: The runtime performance of our interactive path browsing with respect to the increasing depth of the spatial search tree. The 'construction' corresponds to the time taken in seconds to build both the search tree and the cover tree. The 'each query' means how much time it takes in milli seconds to find the nearest tree node for a given location of a new pose.

motions in large data sets. *ACM Transactions on Graphics*, 23(3):559–568, 2004.

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

[4] 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, 2002.

[5] Hubert P. H. Shum, Taku Komura, Masashi Shiraishi, and Shuntaro Yamazaki. Interaction patches for multi-character animation. *ACM Transactions on Graphics*, 27(5), 2008.

[6] Kevin Wampler, Erik Andersen, Evan Herbst, Yongjoon Lee, and Zoran Popović. Character animation in two-player adversarial games. *ACM Transactions on Graphics*, 29, 2010.

[7] Leslie Ikemoto, Okan Arikan, and David Forsyth. Quick transitions with cached multi-way blends. pages 145–151, 2007.

[8] Liming Zhao, Aline Normoyle, Sanjeev Khanna, and Alla Safonova. Automatic construction of a minimum size motion graph. In *Proceedings of the 2009 Symposium on Computer Animation*, pages 27–35, 2009.

[9] Philippe Beaudoin, Stelian Coros, Michiel van de, and Panne Pierre Poulin. Motion-motif graphs. In *Proceedings of the 2004 Symposium on Computer Animation*, pages 117–126, 2008.

[10] Hyun Joon Shin and Hyun Seok Oh. Fat graphs: Constructing an interactive character with continuous controls. In *Proceedings of the 2006 Symposium on Computer Animation*, pages 291–298, 2006.

[11] Manfred Lau and James Kuffner. Precomputed search trees: Planning for interactive goal-driven animation. In *Proceedings of the 2006 Symposium on Computer Animation*, pages 299–308, 2006.

[12] Alla Safonova and Jessica Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics*, 26(3), 2007.

[13] James Davis, Maneesh Agrawala, Zoran Popović, and David Salesin. A sketching interface for articulated animation. In *Proceedings of the 2007 Symposium on Computer Animation*, pages 320–328, 2007.

[14] Matthew Thorne, David Burke, and Michael van de Panne. Motion doodles: An interface for sketching character motion. *ACM Transactions on Graphics*, 23(3), 2004.

[15] Takeo Igarashi, Tomer Moscovich, and John F. Hughes. Spatial keyframing for performance-driven animation. In *Proceedings of the 2005 Symposium on Computer Animation*, pages 107–115, 2005.

[16] Tarjan R. Depth-first search and linear

graph algorithms. *SIAM Journal of Computing*, 1:146–160, 1972.

[17] Paul S. A. Reitsma and Nancy S. Pollard. Evaluating motion graphs for character animation. *ACM Transactions on Graphics*, 26(4):18, 2007.

[18] Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*, pages 97–104, 2006.

[19] Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. Synchronized multi-character motion editing. *ACM Transactions on Graphics*, 28(3), 2009.