7-2009

Achieving Good Connectivity in Motion Graphs

Liming Zhao
University of Pennsylvania, liming@seas.upenn.edu

Alla Safonova
University of Pennsylvania, alla@cis.upenn.edu

Recommended Citation
Zhao, L., & Safonova, A. (2009). Achieving Good Connectivity in Motion Graphs. *Graphical Models, 71* (4), 139-152.
http://dx.doi.org/10.1016/j.gmod.2009.04.001

This articles is part of a Special Issue of ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2008.

Achieving Good Connectivity in Motion Graphs

Abstract
Motion graphs have been widely successful in the synthesis of human motions. However, the quality of the generated motions depends heavily on the connectivity of the graphs and the quality of transitions in them. Achieving both of these criteria simultaneously though is difficult. Good connectivity requires transitions between less similar poses, while good motion quality requires transitions only between very similar poses. This paper introduces a new method for building motion graphs. The method first builds a set of interpolated motion clips, which contains many more similar poses than the original data set. The method then constructs a well-connected motion graph (wcMG), by using as little of the interpolated motion clip frames as necessary to provide good connectivity and only smooth transitions. Based on experiments, wcMGs outperform standard motion graphs across different measures, generate good quality motions, allow for high responsiveness in interactive control applications, and do not even require post-processing of the synthesized motions.

Keywords
motion graph, motion synthesism interpolation, motion capture, human animation

Disciplines
Computer Sciences | Engineering | Graphics and Human Computer Interfaces

Comments
This articles is part of a Special Issue of ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2008.

Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2008)
M. Gross and D. James (Editors)

Achieving Good Connectivity in Motion Graphs

Liming Zhao and Alla Safonova

University of Pennsylvania

Abstract

Motion graphs provide users with a simple yet powerful way to synthesize human motions. While motion graph-based synthesis has been widely successful, the quality of the generated motion depends largely on the connectivity of the graph and the quality of transitions in it. However, achieving both of these criteria simultaneously in motion graphs is difficult. Good connectivity requires transitions between less similar poses, while good motion quality results only when transitions happen between very similar poses. This paper introduces a new method for building motion graphs. The method first builds a set of interpolated motion clips, which contain many more similar poses than the original dataset. Using this set, the method then constructs a motion graph and reduces its size by minimizing the number of interpolated poses present in the graph. The outcome of the algorithm is a motion graph called a well-connected *motion graph with very good connectivity and only smooth transitions. Our experimental results show that well-connected motion graphs outperform standard motion graphs across a number of measures, result in very good motion quality, allow for high responsiveness when used for interactive control, and even do not require post-processing of the synthesized motions.*

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation

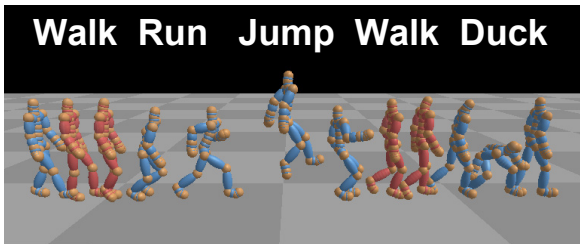


Figure 1: *Using graph wcMG to synthesize a smooth motion consisting of 4 different behaviors (walking, running, jumping and ducking). Poses shown in blue belong to the original dataset, poses in red are interpolated poses introduced during the construction of graph wcMG.*

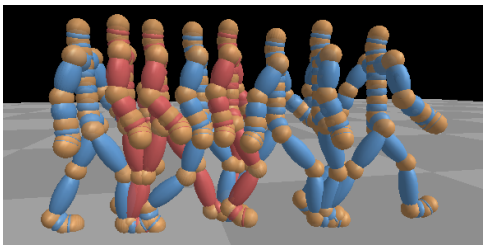


Figure 2: *Using graph wcMG to synthesize smooth transitions between walks with different step lengths. Color coding is the same as in Figure 1.*

1. Introduction

Automatic methods for synthesizing human motion are useful in a variety of applications including entertainment applications (such as games and movies), educational applications, and training simulators. Motion graphs emerged as a very promising technique for automatic synthesis of human motion [AF02, KGP02, LCR*02, PB02]. A motion graph is constructed by taking as input a set of motion capture clips and adding transitions between similar frames in these clips. Once constructed, a path through this graph represents a multi-behavior motion.

A number of papers [RP07, SH07, GSKJ03] that use mo-

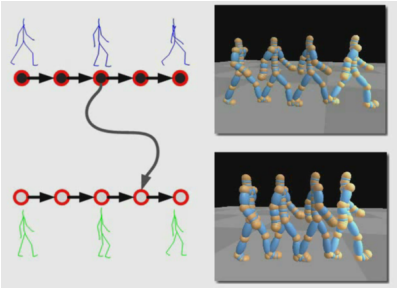


Figure 3: *Poses from a walk with a long step length (above) and from a walk with a short step length (below)*

tion graphs for new motion synthesis note that the results depend largely on the quality of the graph. To synthesize good motions, motion graphs require *smooth* transitions and good *connectivity*. A transition is *smooth* if it does not introduce visual discontinuity to the motion. *Connectivity* is a measure of how quickly one can transition from a pose in one behavior to a pose in another behavior. Good connectivity is important for both interactive control applications where the quick transitions provide real-time user control, and for off-line motion synthesis where the generated motion can follow a user sketch without suffering from long intermediate motion segments. Unfortunately, it is often difficult to achieve both smooth transitions and good connectivity at the same time. Reitsma and Pollard [RP07] did an extensive evaluation of motion graphs for character animation and demonstrated that motion graphs have poor responsiveness to user control. For example, changing from a running motion to an evasive action (ducking) in their motion graph took an average of 3.6s. In industry, hand-constructed blend trees are often used [RenderWare 2001]. They have better connectivity, but require a long time to set up.

In this paper, we present an algorithm that automatically constructs an unstructured motion graph with very smooth transitions and much better connectivity than offered by standard motion graphs (Figures 1 and 2). We call this graph wcMG (well-connected motion graph). To illustrate the main idea, consider two motions M_1 and M_2, with different step lengths (Figure 3). They consist of poses which are similar, but not similar enough to create smooth transitions. If we construct a motion graph from M_1 and M_2, there will be no transitions between them. On the other hand, a set of motions computed by interpolating M_1 and M_2 with different weights contain many more similar poses. These poses allow for smooth transitions between M_1 and M_2 (Figure 2).

In general, our algorithm first computes a set of *interpolated* poses that are interpolations of the original poses from the motion data. It then creates graph wcMG from all original poses and a subset of the interpolated poses. The choice of this subset affects the quality and the size of the motion graph. Our algorithm constructs this subset in such a way as to: (a) guarantee the physical correctness of the transitions in the motion graph; (b) grow the overall graph size as little as possible while maintaining the best connectivity among the original poses. Moreover, the user can pick an upper bound μ on the amount by which the connectivity is sacrificed to trade for further decrease in graph size. The resulting graph wcMG uses exactly the same representation as a standard motion graph, a well studied data-structure with a variety of applicable motion synthesis algorithms.

Our experimental results show that graph wcMG achieves much better connectivity and generates much smoother motions than standard motion graphs. These results are based on visual perceptions as well as computing several metrics, some of which were previously proposed by other researchers. In addition, the motions generated from graph wcMG require no post-processing, a tedious and often imperfect task of removing visual artifacts such as foot sliding. Finally, the evaluation of graph wcMG in an interactive character control scenario demonstrates better responsiveness to user control and better visual quality of generated motions than standard motion graphs.

2. Background

Inspired by the technique of Schödl and his colleagues [SSSE00] that allowed a long video to be synthesized from a short clip, motion graphs were developed simultaneously by several research groups in 2002 and were extended in subsequent years. Motion graphs emerged as a very promising technique for automatic synthesis of human motion for both interactive control applications [LCR*02, KG03, PSS02, PSKS04, KS05] and for off-line sketch-based motion synthesis [AF02, LCR*02, KGP02, AFO03, IAF07, WK88, LP02, FP03, SHP04, SHP04, SH07]. The motion graph construction technique we present in this paper is therefore beneficial for both interactive control and off-line sketch-based motion synthesis applications.

The interpolated motion graph (IMG) introduced by [SH07] is created by taking a "product" of two identical standard motion graphs. The quality of graph IMG, therefore, depends highly on the quality of the standard motion graph. If the standard motion graph does not contain quick and smooth transitions between different behaviors (from walking to picking, for example), graph IMG will not contain them either. Our method adds quick and smooth transitions to a standard motion graph (resulting in graph wcMG). If desired, graph IMG can then be computed by taking a "product" of two graphs wcMG. Our method is, therefore, complimentary to the method of [SH07].

A number of other approaches have been developed over the past few years that combine motion graphs and interpolation techniques [PSS02, PSKS04, KS05, SO06, HG07, TLP07]. These techniques pre-process motions into similar behaviors (for example, walk cycles that all start from the

same leg or martial arts motion segments with similar rest poses [SO06]). Similar segments are then grouped together to create interpolation spaces and smooth transitions are created insides these spaces. In such structured, behavior-based graphs, behaviors can only connect at the end of the motion clips. Moreover, transition blending requires a large amount of motion capture data with good variations for accurate user control [TLP07]. In our work, we also try to find transitions in the interpolated space, however, we do it for standard and unstructured motion graphs. Therefore, our graph can transition from different places inside behaviors. In addition, we construct our motion graphs from a small set of representative motions with a mixture of behaviors.

Ikemoto and her colleagues [IAF07] presented another relevant work for creating quick (1 second long) transitions between all frames in the motion database. They split the motions into short segments (1 second long) and separate them via clustering by similarity. To synthesize a transition from frame i to frame j, they find motion segments that are similar to the motion segments right after frame i and right before frame j. They then search over all possible interpolations of these segments to find the most natural transition from frame i to frame j. To find natural motion segments, they define discriminative classifiers to distinguish natural and unnatural motions. In our work, we also construct a graph that allows for quick transitions between frames, but our construction process is simple, fully automatic, and very similar to the standard motion graph construction process. We do not require a classifier to distinguish natural motions, just a similarity measure commonly used in standard motion graphs. In addition, the transition lengths are not fixed. They evolve naturally from the dynamics of the motion data.

An additional difference between graph wcMG and other variants of motion graphs that were introduced in recent years([PSKS04, KS05, SO06, HG07, IAF07]), is that graph wcMG has exactly the same representation as a standard motion graph. A standard motion graph is a well studied data-structure with a variety of existing motion synthesis algorithms applicable to it.

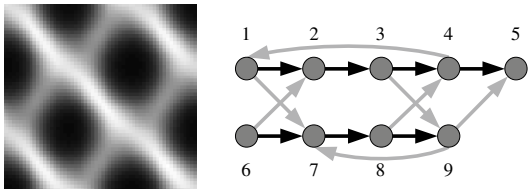
A number of on-line methods that use variants of motion graphs together with reinforcement learning [BEL57, Nil71] have been recently proposed. Lee and Lee [LL04] precompute policies that indicate how the avatar should move for each possible control input and avatar state. Their approach allows interactive control with minimal run-time cost for a restricted set of control input. McCann and Pollard [MP07] achieve better immediate control of the character by integrating a model of user behavior into the reinforcement learning. Treuille and his colleagues [TLP07] use reinforcement learning to learn from the continuous control signals. They use basis functions to represent value functions over continuous control signals and show that very few basis functions are required to create high-fidelity character controllers. This permits complex user navigation and obstacle-avoidance tasks. Using these reinforcement learning techniques, we evaluate the performance of our graph wcMG for on-line interactive control.

3. Standard Motion Graphs (MG)

Standard motion graphs (MG) are constructed by taking as input a set of motion capture clips. Each frame in these motion clips is treated as a graph node and two nodes are connected by a directed edge if there exists a smooth transition from one node to the other.

First, one needs to define a metric for computing smooth transitions between the nodes in the graph. Current popular approaches compare the similarity of two nodes i and j, and if the similarity is lower than a user specified value, node i can be connected to node $j+1$ and node j can be connected to node $i+1$ (where node $i+1$ and node $j+1$ are the successors of node i and node j respectively in their own motion clips). There are a number of approaches to measure this similarity. We adopt Kovar et al.'s [KGP02] point cloud metric with weights on different joints from Wang and Bodenheimer's work [WB04].

After thresholding the similarity value by a user specified value for naturalness, a dense transition matrix is obtained (Figure 4), where rows and columns are the frames from the motion clips. The transition value at row i and column j is $t = exp(-s/\sigma)$, where s is the similarity value between frame i and frame $j-1$. A standard process is then to leave only transitions that represent local maxima in the transition matrix and compute the largest strongly connected component (SCC) of the graph to remove dead ends.



(a) (b)

Figure 4: *Motion Graph construction. (a): Transition matrix. The entry at (i,j) contains the transition value from node i to node j. Bright values indicate good transitions and dark values indicate poor transitions. (b): A simple motion graph. Black edges are from the motion clips and gray edges are created through the similarity measurement. The largest strongly connected component is {1,2,3,4,7,8,9}.*

Nodes in the motion graph are naturally connected to their successor nodes according to the clip sequence. Motion graphs become more interesting and useful if nodes can be connected in an order other than just the original motion sequence, for example, connections that create loops inside a clip and transitions between different clips. Usually the

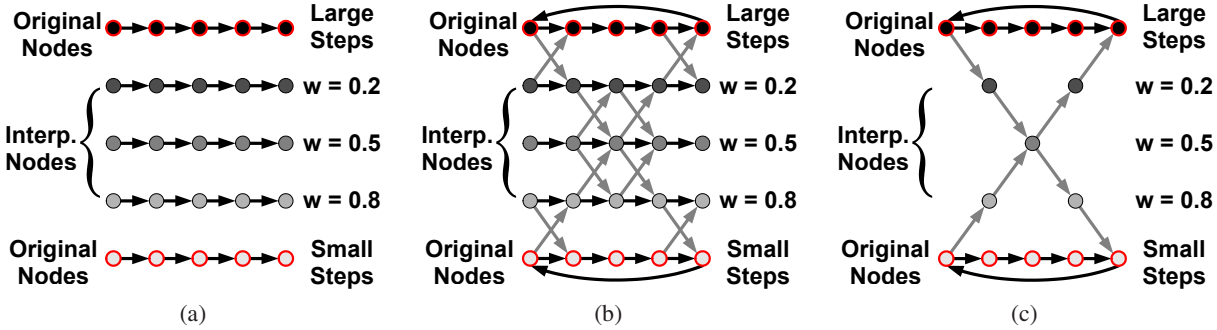


Figure 5: *A simple example illustrating the construction steps. (a) Interpolation Step. (b) Transition Creation Step. (c) Node Reduction Step. All original nodes are kept. Interpolated nodes and edges outside the* best *paths set are removed.*

denser the connectivity, the better the performance of motion graphs, because quick transitions become possible between different behaviors. However, to achieve denser connectivity, we need to introduce transitions between less similar poses. Therefore, there is a trade-off between good connectivity in a motion graph and smoothness of the added transitions.

One major challenge in creating a good motion graph is picking a threshold value that would result in both smooth transitions and good connectivity between different behaviors. Usually a low threshold results in very few transitions, most of which occur within the same behavior, but not across behaviors. A high threshold, on the other hand, often results in poor quality transitions. As we show in the experimental section, to get good connectivity between different behaviors in standard motion graphs, it is necessary to pick a threshold that results in transitions with noticeable visual discontinuity. Our approach on the other hand, achieves both good connectivity and smooth transitions for a variety of motion databases, including databases that are especially hard for standard motion graphs. These databases contain motions of different styles, from different people, and of many different behaviors.

4. Well-connected Motion Graphs (wcMG)

In this section, we first describe the construction process for our well-connected motion graph. We then extend our algorithm to allow for trade-offs between degree of connectivity and graph size.

4.1. Construction Process

The construction of graph wcMG is divided into three steps: the interpolation step, the transition creation step, and the node reduction step. We explain these three steps as follows.

Interpolation Step: Given a motion data set, we first separate the motions into segments based on contact with the environment. We use the technique from Lee and his colleagues [LCR*02] to identify the contacts. Contact information could also be computed using one of the other published techniques [IAF06, CB06].

The interpolation technique described by Safonova and Hodgins [SH05] is then used to interpolate all pairs of segments with the same contact information. This interpolation technique produces close to physically correct motions. We denote the frames in the motion segments from the original motion capture clips as *original nodes* and those from the interpolated segments as *interpolated nodes*. The original nodes , the interpolated nodes, and the natural transitions between them are all added to graph wcMG(Figure 5(a)).

Transition Creation Step: At this stage, graph wcMG contains only natural (original or interpolated) edges between the original and the interpolated nodes. Additional edges are added to the graph using the same process used in constructing standard motion graphs (Figure 5(b)). The only difference is that graph wcMG contains both the original and the interpolated nodes and the natural transitions between them.

From our experiments, a large number of smooth transitions can be added to graph wcMG with the help of the interpolated nodes. The interpolated nodes provide connections between poses that are similar, but not similar enough to create smooth transitions, such as poses from walks with different step lengths. The interpolated nodes also provide transitions between different behaviors even if such transitions are not explicitly captured. For example, a transition from jumping to ducking is possible, because the landing steps from jumping and the steps that lead to ducking can be interpolated, even when no such motion capture data exists. Therefore, it is possible to choose a very low threshold for a variety of behaviors to create a graph with very good connectivity. Section 5 gives a detailed analysis of the connectivity and threshold choices for graph wcMG.

Node Reduction Step: Theoretically, the number of nodes in graph wcMG equals to $N = n \times n \times w$, where n is the number of frames in the motion database and w is the number of interpolation weights used. The number of nodes and edges grow quadratically with the size of the original data

set. However, assuming that the choice of the initial data set reflects the user's application need, we can remove all nodes and edges from the graph that are not necessary for connecting the original nodes.

Therefore, in this step we compute a subset of nodes and edges, S and E respectively, that best connect the original nodes. To find this subset, we compute the *best* paths between the original nodes. The measure of *best* can be application based. For example, if quick responsiveness is the goal, we look for the shortest transitions; if motion quality is the goal, we look for the smoothest transitions. In our implementation, we used a weighted average of length and the smoothness of the transitions.

First, S is initialized to include all the original nodes. Next, for each of the original nodes, we find the best transition paths to all other original nodes in graph wcMG. This is done by setting the transition cost according to the metric described above and running a Dijkstra's shortest path algorithm for each of the original nodes iteratively (we use the Dijkstra algorithm instead of the Floy-Warshall algorithm because we do not want to waste time computing best paths between the interpolated nodes). All nodes and edges that belong to the best transition paths between the original nodes are added to S and E respectively (Figure 5(c)). By only keeping the nodes and edges in S and E, we do not change the connectivity between the original nodes. Moreover, this node reduction step makes the final motion graph size increase linearly with the size of the motion data set as shown in Section 5.5.

In conclusion, graph wcMG offers a fully automatic way to construct a motion graph from a set of representative motion capture data with good connectivity and responsiveness. The low similarity threshold only introduces smooth transitions (FIgure 2) and the size of the graph grows linearly with the size of the data set.

4.2. Trade-off between Size and Graph Quality

In this section, we extend our algorithm to allow a user to further decrease the size of graph wcMG obtained from the previous section at the expense of the best transition path quality between the original nodes.

We denote the motion graph generated in the previous section as the *optimal graph*, because it includes nodes and edges that belong to the best paths among the original nodes. We then introduce a suboptimality scalar μ to the previous best path search process. μ allows us to compute suboptimal graphs with a smaller number of nodes and edges but at the expense of the optimal paths between the original nodes.

First, S is initialized to include only the original nodes as in the previous section. However, during each iteration of the Dijkstra's shortest path search, the cost c of each edge e is computed differently now:

$$c = \begin{cases} t, & \text{if } i, j \in S, \quad (1a) \\ \mu \cdot t, & \text{otherwise.} \quad (1b) \end{cases}$$

where i and j are the nodes that edge e connects and t is the base cost of the edge as described before. After each iteration, the nodes along the shortest path are added to the set S and this process repeats for every original node.

This suboptimality scalar μ compromises between graph quality and graph size. As μ increases, it forces the search to reuse the nodes which are already in S, even though paths that pass through these nodes have worse quality than those paths outside S. As a result, the size of S decreases as μ increases. This process does not affect the reachability between the original nodes, but will reduce the quality of transition (smoothness and responsiveness) since paths outside of S are penalized by μ times, even if they are of lower transition cost or shorter length. Fortunately, graph wcMG's low similarity threshold gives a nice upper bound on the transition discontinuity. Section 5.6 gives a detailed analysis on the suboptimality trade-off and shows that the size of the graph decreases drastically as μ increases.

5. Experimental Analysis

In this section, we analyze the performance of graph wcMG. Table 1 shows the motion data sets we used in our analysis. Dataset1 contains walking motions from one person with different step lengths, ranging from small steps to exaggerated large strides. Dataset2 contains walking motions from four different people with approximately the same step length. We chose these two data sets, because generating well connected motion graphs between different styles and for different people is known to be difficult. Dataset3 contains a large set of motions commonly used to construct motion graphs for interactive control applications and for off-line motion synthesis applications. Each motion in the data set contains a series of different behaviors and natural transitions between the behaviors. For example, in a single capture, the actor walks, jumps, walks again, ducks and so on. Even though each motion contains natural transitions between behaviors, the standard motion graph computed from this database has poor connectivity. Our graph wcMG on the

Table 1: *Data Set Description*

Name	Frames	Content
Dataset1	194	walks with different step lengths
Dataset2	264	walks from four different people
Dataset3	2721	a mixture of behaviors: walking, idling, jumping, running, turning, ducking and picking up objects and captured transitions between them.

Table 2: *Connectivity vs. Similarity Threshold*

	Data1		Data2		Data3	
Thre	wcMG	MG	wcMG	MG	wcMG	MG
0.2	42%	1%	84%	21%	96%	7%
0.4	70%	19%	91%	21%	98%	10%
0.6	77%	24%	94%	21%	99%	64%
0.8	81%	40%	95%	21%	99%	86%
1.0	84%	59%	96%	21%	99%	92%
2.0	91%	64%	96%	81%		
5.0	92%	80%	97%	92%		
6.0	96%	92%	98%	92%		

other hand, achieves much better connectivity. All data is resampled to 30 frames per second.

The computation time for each construction step of graph wcMG in Section 4.1 is as follows: 7 seconds for Interpolation Step, 4.5 hours for Transition Creation Step and between 0.5 to 1 hour for Node Reduction Step (depending on the suboptimal parameter value). The computation times are for Database3, our largest data set, and are spent only once as a pre-processing step for graph wcMG.

In the experiments shown in Section 5.1, we synthesize a number of motions consisting of a mixture of behaviors using graph wcMG. We show that even with no post-processing the motions computed using graph wcMG are very smooth and natural-looking. In Sections 5.2, 5.3, 5.4 we analyze the connectivity of graph wcMG and compare it with the standard motion graph (MG). As most motion graph construction algorithms go through the process of computing the largest strongly connected component (SCC), in Sections 5.2, we compare the size of the largest SCC for both graphs, MG and wcMG, at varying threshold values. In Sections 5.3 and 5.4, we analyze and compare the responsiveness of graph MG and graph wcMG to interactive user input by computing the time it takes to transition between different frames in the motions and between different behaviors. In Section 5.5, we analyze the size of graph wcMG and in Section 5.6 we show that one can decrease the size of graph wcMG at the cost of its responsiveness.

5.1. Motion Smoothness

We generated a variety of motions using graph wcMG to visually evaluate its quality. We generated these motions by computing the *best* paths between user selected poses in a specified order. The *best* metric is defined as in Section 4.1. The resulting animations are shown in the accompanying movie. Figure 1 shows a smooth motion consisting of 4 different behaviors (walking, running, jumping and ducking). Figure 2 shows smooth transitions between a walk with a short step length and a walk with a long step length.

This experiment was done solely to show the visual quality of the motion. We do not claim this to be a useful application. In Section 6 we demonstrate the applicability of graph wcMG to interactive control applications by generating motions in real-time according to user input. The resulting animations shown in the accompanying movie are responsive and smooth.

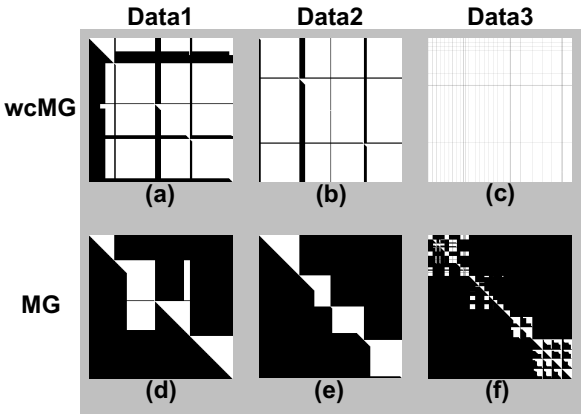


Figure 6: *Transitive Closure Matrix. Ideally, we would like to have a plain white image which indicates that every frame can transition to every other frame. The coverage of white area gives an estimate of the* connectivity.

5.2. Size of the Largest Strongly Connected Component

In this section, we analyze the size of the largest strongly connected component (SCC) as a percentage of the original data set size for both graphs MG and wcMG at varying threshold values. This shows how well the nodes in the data set are connected to each other and how well the motion graphs are utilizing the motion data set.

Table 2 shows the percentage of largest SCC size for both graphs wcMG and MG at different similarity thresholds. The first column shows the similarity threshold. Experiments show that, the best paths computed according to Section 5.1 almost always produce smooth motions with no visual discontinuity, when the threshold is below 0.6 (colored in green), and often contain visually noticeable discontinuity when the threshold is above 0.6 (colored in red). The results are shown for all three data sets. Graph MG requires 3 to 12 times higher threshold values than graph wcMG does in order to achieve over 90% data set utilization. These entries are highlighted in blue in the table.

Figure 6 shows the transitive closure matrix for both graphs wcMG and MG for all three data sets at a 0.4 threshold. The transitive closure matrix is a square matrix in which element (i, j) is 1 (shown in white) if there is a path between pose i and pose j and 0 otherwise (shown in black). As shown in the figure, graph wcMG has paths between the majority of nodes for all databases at a low similarity threshold that always results in smooth transitions. Graph MG, on the other hand, has very poor connectivity at the same threshold

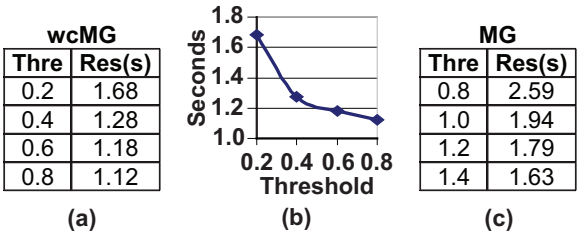


Figure 7: *Average Transition Time vs. Threshold. (a)Average Transition Time of graph wcMG at different similarity thresholds. (b) Plot of Average Transition Time vs. threshold for graph wcMG shows an exponential decrease. (c)Average Transition Time of graph MG at different similarity thresholds.*

for all three databases. The results for Dataset1 and Dataset2 show that it is hard to connect motions of different styles in graph MG. For Dataset2, graph MG only contains transitions between the frames from the same person, but not across frames from different people. As a result, Figure 6e shows no white area outside the diagonal blocks. Graph wcMG on the other hand, can transition between styles from different people, as Figure 6b is almost fully covered by white dots. Dataset3 contains a mixture of behaviors. Graph wcMG is able to create transitions between them (almost all white in Figure 6c). At the same threshold value, graph MG is only able to explore transitions within the motions from similar behaviors such as walks and turns. These similar behaviors are indicated by the clustered off diagonal white areas in Figure 6f. In conclusion, graph wcMG offers better data set utilization with low similarity thresholds.

5.3. Time Between Frames

In this section, we compute the average transition time between all pairs of the original nodes. This measure provides insight into how quickly it is possible to transition between different poses in the data set. Quick transitions are important for both interactive user control and off-line motion synthesis. We present results at different similarity thresholds for Dataset3, but similar results are achieved for all three data sets. For graph wcMG, we vary the threshold from 0.2 to 0.8. For graph MG, we vary the threshold from 0.8 to 1.4, because below the 0.8 threshold, not every behavior present in the data set is connected to every other behaviors. Figure 7 shows the comparison. Transition time decreases monotonically as the similarity threshold increases, because lower quality transitions become possible (Figure 7b). Graph wcMG obtains as good a transition time at the 0.2 threshold as graph MG at the 1.4 threshold (A 7 times increase in threshold value).

Table 3: *Local Maneuverability*

MG			wcMG	
Thre	**LM**	**Size**	**Thre**	**LM**
0.8	1.33	2334	0.2	0.79
1.0	0.98	2516	0.4	0.54
1.2	0.88	2617	0.6	0.49
1.4	0.79	2645	0.8	0.45

5.4. Local Maneuverability Measurement(LM)

An alternative way to measure the graph connectivity is the local maneuverability measurement (LM) proposed by Reitsma and Pollard [RP07]. Instead of computing the transition time between any two frames, they compute the average minimum time needed to transition to a particular behavior:

$$LM_k = \frac{1}{\|C\|} \sum_{c \in C} (0.5 * D_c + MDMPC_{c,k}) \qquad (2)$$

where C is the set of all the motion segments in the motion graph, $MDMPC_{c,k}$ is the minimum time to transition from the end of motion segment c to any instance of behavior k, and D_c is the motion segment length.

In addition, we compute an average of local maneuverability measurement over all behaviors:

$$LM = \frac{1}{\|K\|} \sum_{k \in K} (LM_k) \qquad (3)$$

where K is all the behaviors in the motion graph.

Table 3 shows the LM measure change with respect to the similarity threshold change for both graphs MG and wcMG on Dataset3. For graph MG, as the similarity threshold increases, the size of the largest SCC increases (Table 3(left column)). For graph wcMG, the size of the largest SCC practically does not change, as it contains almost all the necessary data even at the lowest threshold.

For both graphs MG and wcMG, as the graph size increases, the LM measure decreases (Reitsma and Pollard [RP07] obtained similar results). Even at a very low similarity threshold (0.2), graph wcMG is able to produce an LM value as low as 0.79 seconds. We could not compute the LM value for graph MG at the same threshold, because behaviors are not fully connected to each other. At the 0.8 similarity threshold, graph wcMG has an LM value of 0.45 seconds which is three times smaller than the LM value for graph MG at the same threshold. This shows that graph wcMG provides very responsive transitions between different behaviors while preserving good motion quality. Therefore, it is much better suited for interactive applications. We present results for Dataset3 in this section, but similar results are achieved for all three data sets.

Table 4 shows the behavior to behavior LM measure for both graphs wcMG and MG at the similarity threshold = 0.8.

Table 4: *Behavior to Behavior LM measure*

wcMG

	idle	walk	jump	run	duck	pick
idle	0.00	0.03	0.27	0.53	0.03	0.13
walk	0.03	0.00	0.03	0.03	0.03	0.03
jump	0.73	0.03	0.00	0.50	0.17	0.20
run	1.60	0.90	0.03	0.00	1.17	1.23
duck	0.07	0.03	0.17	0.20	0.00	0.07
pick	0.17	0.03	0.20	0.27	0.07	0.00

MG

	idle	walk	jump	run	duck	pick
idle	0.00	0.03	1.73	1.40	0.03	1.37
walk	0.03	0.00	0.03	0.03	0.03	0.03
jump	0.90	0.03	0.00	1.73	1.07	0.63
run	2.70	0.90	0.03	0.00	2.83	2.70
duck	0.37	0.03	0.87	0.60	0.00	0.40
pick	0.97	0.03	0.90	0.67	0.07	0.00

Each entry is the average minimum time in seconds to transition from the behavior in row to the behavior in column. In general, highly dynamic motions connect poorly to low dynamic motions, for example, running to idle. The table is not symmetric, because the motion capture data does not contain transitions in symmetric directions. Graph wcMG introduces interpolated poses and transitions to help find extra connections, and therefore, outperforms graph MG in all cases.

The LM measure we compute in this section is a theoretical measurement. Some of the minimum-duration transitions may become impossible when the motion graph is unrolled into the environment. In addition to theoretical LM measure, Reitsma and Pollard compute practical LM measure which takes the environment into account. As Reitsma and Pollard show in their work, practical LM measure is usually higher than the theoretical measure. However, they also note that poor theoretical LM values typically indicate poor practical LM values.

5.5. Graph Size

In this section, we analyze the number of nodes and edges in graph wcMG with respect to the size of the data set used to compute the graph. For this experiment, we use Dataset3 and vary the amount of motion data used. Figure 8 shows that before the node reduction step, the number of nodes (blue empty squares) and the number of edges (orange empty diamonds) grow quadratically as the motion data size increases. However, after the node reduction step, which fully preserves the *connectivity* among the original nodes, the number of nodes (red filled diamonds) and the number of edges (green filled squares) grow linearly as the motion data size increases. The linear coefficients for the node and edge increase are 15.4 and 19.3 respectively. In conclusion, the node reduction step provides a nice linear bound to the size of graph wcMG.

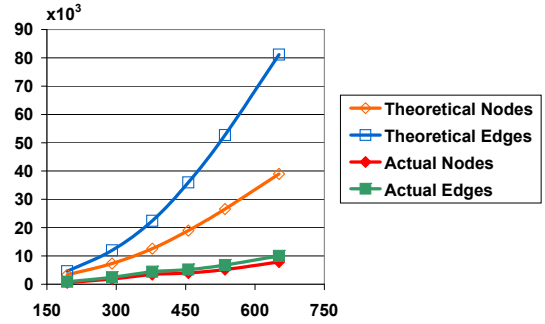


Figure 8: *Graph Size vs. Motion Data Size. The horizontal axis is the total number of frames in the database and the vertical axis is the number of nodes and edges.*

5.6. Parameter μ

The suboptimality scalar μ allows one to decrease the size of the graph further at the expense of the transition quality and the connectivity of the graph. In this experiment, we construct a series of graphs wcMG by varying the parameter μ from 1.0 to 100.0. We use Dataset3 and a similarity threshold of 0.2 in all the cases. Table 5 shows the analysis in terms of number of nodes (NodeSize), graph size in comparison to graph MG (SizeFactor), LM measure in seconds (LM), and memory requirement in mega-bytes (Mem). The first row shows the values for the optimal graph. As μ increases, the graph size and the memory requirement decrease exponentially, producing graphs with size from 35.7× to 1.4× the size of graph MG. At the same time, the graph quality (smoothness and connectivity) gets worse. On the positive side, suboptimal graphs still contain smooth transitions because the similarity threshold is below 0.2 for all graphs. The connectivity of the graph does become worse but at a log scale. For comparison, graph MG constructed from Dataset3 has 2721 nodes and requires 4 MB memory.

Table 5: *The effects of the suboptimality parameter μ*

μ	NodeSize	SizeFactor	LM(s)	Mem(MB)
Opt.	97201	35.7	0.79	190
1.1	71013	26.1	0.79	135
1.3	52967	19.5	0.80	98
1.5	44879	16.5	0.80	81
1.7	40419	14.9	0.81	73
2.0	36085	13.3	0.82	65
3.0	28646	10.5	0.85	51
5.0	16098	5.9	0.97	28
10.0	6950	2.6	1.24	11
20.0	4989	1.8	1.56	8
50.0	3940	1.4	1.88	6
100.0	3789	1.4	2.07	6

In conclusion, the user can choose an appropriate value for μ to balance between memory requirements and transition quality. $\mu = 5.0$ was used in our interactive control experiments in Section 6, as it provides the best compromise between size and responsiveness.

6. Application to Interactive Control

In this section, we show the applicability of graph wcMG to interactive control applications. We use value iteration methods to learn good control policies following the reinforcement learning techniques proposed by [MP07, TLP07, LL04]. The main advantage of the value iteration approach is that the online operation reduces to a simple fast table lookup for an action. The resulting animations shown in the accompanying movie are responsive and smooth.

We compare the performance of graphs wcMG and MG for interactive control applications. We compute a graph wcMG consisting of walks with different turns, jumps, and a stopping motion. The user can control the character by changing its yaw orientation around the vertical axis and by changing its behavior among walking, jumping and stopping. The orientation control happens only during the walking behavior.

To evaluate the performance of graphs wcMG and MG, we use an approach similar to the one described in [MP07]. We train a control policy for both graphs wcMG and MG. When computing graph wcMG, we use a very low threshold (0.2), because it already provides good connectivity between the nodes. When computing graph MG, we set the threshold very high to allow low quality transitions following the observation made by McCann and Pollard [MP07] , that low quality transitions are required for better interactive user control. McCann and Pollard [MP07] show that if all low quality transitions are removed from the graph, the overall performance of the interactive control decreases. We then train control policies using the value iteration technique for both graph wcMG and graph MG.

Similar to McCann and Pollard [MP07], we first collect desired user traces (i.e., a sequence of keyboard commands) and then evaluate the performance of the policy trained for both graph wcMG and graph MG on these traces. Table 6 shows the result. When evaluating a given trace from the user, we compute both the transitions smoothness in the resulting motion (Transition column in Table 6) and how well the motion follows the user input (Control column in Table 6) - similarly to [MP07]. The column labeled Total is the multiplication of the two measures. As can be seen from the table, even with a much lower threshold (15 to 50 times lower than that for graph MG), our graph wcMG outperforms graph MG in terms of both transition quality and responsiveness to user control.

Table 6: *Interactive Control Evaluation*

wcMG

Thre	Control	Transition	Total	SCC Size
0.2	0.74	0.95	0.70	561

MG

Thre	Control	Transition	Total	SCC Size
0.2	0.22	0.99	0.22	36
1.0	0.24	0.92	0.22	244
5.0	0.50	0.84	0.42	524
10.0	0.51	0.87	0.44	552

7. Discussion

In this paper, we presented an algorithm for constructing an unstructured motion graph with much better connectivity than standard motion graphs. The method first builds a set of interpolated motion clips, then constructs a motion graph, and reduces its size by minimizing the number of interpolated poses present in the graph. The outcome of the algorithm is a standard motion graph, which can benefit from all the standard research techniques for motion graphs. Our experimental results show that well-connected motion graphs outperform standard motion graphs across a number of measures, result in very good motion quality, allow for high responsiveness in interactive controls, and even require no post-processing to the synthesized motions.

The current method for trading off size for graph quality (Section 4.2) is greedy, because the order in which the original nodes are processed to compute the best paths to other nodes affects the size of the final graph. In the future, we would like to develop a more optimal algorithm. Even though our current Interpolation Step (Section 4.1) targets the motions that contain changes in contact, we believe that with better motion segmentation and appropriate interpolation techniques, our method can produce well-connected motion graphs for other data as well.

In this paper we analyzed the performance of our graph for interactive control applications. We believe that our graph will also perform well for off-line search-based methods and plan to evaluate it for sketch-based synthesis applications. In addition, it is part of the future work to analyze the performance of our graph when it is unrolled into the environment (as was done by Reitsma and Pollard [RP07]).

8. Acknowledgement

The authors would like to thank Matt Kuruc, Joe Kider, Jan Allbeck and Amy Calhoun for their help with our video and paper, and the anonymous reviewers for their thoughtful comments.

References

[AF02] ARIKAN O., FORSYTH D. A.: Interactive motion generation from examples. *ACM Trans. on Graphics 21*, 3 (2002), 483–490.

[AFO03] ARIKAN O., FORSYTH D. A., O'BRIEN J. F.: Motion synthesis from annotations. *ACM Trans. on Graphics 22*, 3 (2003).

[BEL57] BELLMAN R.: *Dynamic Programming*. Princeton University Press, 1957.

[CB06] CALLENNEC B. L., BOULIC R.: Robust kinematic constraint detection for motion data. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (2006), pp. 281–290.

[FP03] FANG A. C., POLLARD N. S.: Efficient synthesis of physically valid human motion. *ACM Trans. on Graphics 22*, 3 (2003), 417–426.

[GSKJ03] GLEICHER M., SHIN H., KOVAR L., JEPSEN A.: Snap-together motion: Assembling run-time animation. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (2003), pp. 181–188.

[HG07] HECK R., GLEICHER M.: Parametric motion graphs. In *ACM Symposium on Interactive 3D Graphics* (2007), pp. 129–136.

[IAF06] IKEMOTO L., ARIKAN O., FORSYTH D.: Knowing when to put your foot down. In *ACM Symposium on Interactive 3D Graphics* (2006), pp. 49–53.

[IAF07] IKEMOTO L., ARIKAN O., FORSYTH D.: Quick transitions with cached multi-way blends. In *ACM Symposium on Interactive 3D Graphics* (2007), pp. 145–151.

[KG03] KOVAR L., GLEICHER M.: Flexible automatic motion blending with registration curves. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (Aug. 2003), pp. 214–224.

[KGP02] KOVAR L., GLEICHER M., PIGHIN F.: Motion graphs. *ACM Trans. on Graphics 21*, 3 (2002), 473–482.

[KS05] KWON T., SHIN S. Y.: Motion modeling for on-line locomotion synthesis. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (July 2005), pp. 29–38.

[LCR*02] LEE J., CHAI J., REITSMA P. S. A., HODGINS J. K., POLLARD N. S.: Interactive control of avatars animated with human motion data. *ACM Trans. on Graphics 21*, 3 (2002), 491–500.

[LL04] LEE J., LEE K. H.: Precomputing avatar behavior from human motion data. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (2004), pp. 79–87.

[LP02] LIU C. K., POPOVIĆ Z.: Synthesis of complex dynamic character motion from simple animations. *ACM Trans. on Graphics 21*, 3 (2002), 408–416.

[MP07] MCCANN J., POLLARD N. S.: Responsive characters from motion fragments. *ACM Trans. on Graphics (SIGGRAPH 2007)* (2007).

[Nil71] NILSSON N.: *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, 1971.

[PB02] PULLEN K., BREGLER C.: Motion capture assisted animation: texturing and synthesis. *ACM Trans. on Graphics 22*, 2 (2002), 501–508.

[PSKS04] PARK S. I., SHIN H. J., KIM T. H., SHIN S. Y.: On-line motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 125–138.

[PSS02] PARK S. I., SHIN H. J., SHIN S. Y.: On-line locomotion generation based on motion blending. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (July 2002), pp. 105–112.

[RP07] REITSMA P. S. A., POLLARD N. S.: Evaluating motion graphs for character animation. *ACM Trans. Graph.* (2007), 18.

[SH05] SAFONOVA A., HODGINS J. K.: Analyzing the physical correctness of interpolated human motion. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (2005), pp. 171–180.

[SH07] SAFONOVA A., HODGINS J. K.: Construction and optimal search of interpolated motion graphs. In *ACM Trans. Graph.* (2007), p. 106.

[SHP04] SAFONOVA A., HODGINS J. K., POLLARD N. S.: Synthesizing physically realistic human motion in low-dimensional, behavior-specific spaces. *ACM Trans. on Graphics 23*, 3 (2004), 514–521.

[SO06] SHIN H. J., OH H. S.: Fat graphs: Constructing an interactive character with continuous controls. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (Sept. 2006), pp. 291–298.

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D. H., ESSA I.: Video textures. In *Proceedings of ACM SIGGRAPH 2000* (July 2000), Computer Graphics Proceedings, Annual Conference Series, pp. 489–498.

[TLP07] TREUILLE A., LEE Y., POPOVIĆ Z.: Near-optimal character animation with continuous control. *ACM Trans. Graph.* (2007), 7.

[WB04] WANG J., BODENHEIMER B.: Computing the duration of motion transitions: an empirical approach. In *ACM SIGGRAPH/Eurographics Symp. on Comp. Animation* (2004), pp. 335–344.

[WK88] WITKIN A., KASS M.: Spacetime constraints. *Computer Graphics (Proceedings of SIGGRAPH 88) 22*, 4 (1988), 159–168.