In Proceedings of the 4th International Conference on Motion In Games, Edinburgh, UK, 2011

Feature-Based Locomotion with Inverse Branch Kinematics

Mentar Mahmudi and Marcelo Kallmann

University of California, Merced
Merced, CA, 95343, USA
{mmahmudi,mkallmann}@ucmerced.edu
http://graphics.ucmerced.edu

Abstract. We propose a novel Inverse Kinematics based deformation method that introduces flexibility and parameterization to motion graphs without degrading the quality of the synthesized motions. Our method deforms the transitions of a motion graph-like structure by first assigning to each transition a continuous rotational range that guarantees not to exceed the predefined global transition cost threshold. The deformation procedure improves the reachability of motion graphs to precise locations and consequently reduces the time spent during search. Furthermore, our method includes a new motion graph construction method based on geometrical segmentation features, and employs a fast triangulation based search pruning technique that confines the search to a free channel and avoids expensive collision checking. The results obtained by the proposed methods were evaluated and quantified, and they demonstrate significant improvements in comparison with traditional motion graph approaches.[1]

Keywords: Character Animation, Locomotion, Motion Capture

1 Introduction

The realistic animation of virtual characters remains a challenging problem in computer animation. Successful approaches are mostly data-driven, using motion capture data, and often involving motion blending and search. While blending operations over motion segments adequately grouped are suitable to real-time performances, search techniques based on motion graphs provide minimum distortion of the captured sequences, and naturally allow the exploration of solutions in complex environments. Even if real-time performance is most often lost, techniques based on motion graphs can still be employed in off-line production phases for several purposes. For instance, the proposed deformation of locomotion sequences can be used to build blendable clips for real-time steering control. Little work has been done on developing deformation models that maintain minimum distortion to the captured motions. In this paper we present a simple and efficient Inverse Kinematics (IK) branch deformation technique to address this

[1] Accompanying video at http://graphics.ucmerced.edu/videos/11-mig-fmg.m4v

problem. Each branch of the search is treated as a kinematic chain of motion segments with joint limits representing the allowed rotational deformation at transitions. This technique produces motions precisely reaching given targets, and at the same time leads to an earlier successful termination of each search. These benefits do not sacrifice the quality of the motions as the error introduced in the deformed transitions does not exceed the same predefined threshold error used to construct the graph.

Our deformation method is also free of feet sliding artifacts thanks to a proposed new feature-based approach for constructing improved motion graphs. Feature-based motion graphs can be constructed significantly faster than traditional motions graphs by avoiding the pairwise comparison between all frames in the database. Instead, only suitable pairs of frames are chosen for transition evaluation. The chosen pairs are the output of feature detectors encoding relationships of interest among key joint positions of the given skeleton. Once these suitable frames are detected, transitions are evaluated with the usual threshold-based comparison metric, thus achieving the same quality of results.

Depending on the application, different feature detectors can be employed. A wide range of feature detectors have been proposed by Müller et al [14]. These feature detectors can be quickly evaluated and are based on spatial relationships between the joints of the character at any given frame. We have noticed that, for the purpose of locomotion synthesis, a very small set of feature detectors is sufficient to successfully segment various walking motions. For example, the forward walk detector checks for a crossing event at the ankle joints, leading to motion segments with one foot always planted on the floor. This is a desirable property as it eliminates the need for post-processing due foot-skating artifacts. Similar strategies were demonstrated by previous authors [20], but employing manually crafted clips.

We also present a search pruning technique based on planar channels with guaranteed clearance from obstacles. This is achieved by projecting all obstacles on the floor plane and maintaining a triangulation of the environment [7]. For any given start and goal positions, the triangulation returns a collision-free channel that is used to prune the branches unrolled by the motion graph search that lie outside of the channel. This results in improved search times, especially in environments with many obstacles.

Finally, extensive evaluations are presented for synthesizing locomotion among obstacles, and our results demonstrate significant improvements in time of computation, in finding solutions, and in the quality of results.

2 Related Work

A large body of work has been devoted to animating characters using human motion capture data. Motion graphs [9, 1, 12, 2, 6] represent a popular approach that is based on connecting similar frames in a database of motion capture examples. Once a motion graph is available, graph search is performed in order to extract motions with desired properties.

Kovar et al. [9] cast the search as an optimization problem and use a branch and bound algorithm to find motions that follow a user specified path. Arikan and Forsyth [1] build a hierarchy of graphs and use a randomized search to satisfy user constraints. Arikan et al. [2] use dynamic programming to search for motions satisfying user annotations. Lee et al. [12] construct a cluster forest of similar frames in order to improve the motion search efficiency. All these methods require quadratic construction time for comparing the similarity between all the frames in the database. Our method improves on this operation by connecting only selected frames.

Many improvements to motion graphs have already been proposed. Ren et al [16] combine motion graphs with constrained optimization. Shin and Oh [18] combine groups of parametrized edges into a fat edge for interactive control. Beaudoin et al [3] use a string-based model to organize large quantities of motion capture data in a compact manner. Our contributions focus on deforming motions without accruing additional error, improved construction and representation and can be used to improve any previous method relying on a motion graph structure.

Motion capture data has also been extensively used for locomotion planning among obstacles [5, 15, 11, 21, 19]. In particular, Lau and Kuffner [10] manually build a behaviour-based finite state machine of motions, which in later work is precomputed [11] to speed up the search for solutions. Choi et al [4] combine motion segments with probabilistic roadmaps. These methods however require the user to manually organize motion examples in suitable ways.

Other approaches [17, 23, 13] are based on the interpolation of two time-scaled paths or involve methods that solve a linear system of constraints [8]. Although these methods increase the solution space, they come with the expense of further distorting synthesized motions and they often increase the involved computation time and complexity.

In general, the main drawback of motion graph structures is the prohibitively large amount of data that may be needed in order to address practical applications involving obstacles and precise placements. The solutions presented in this paper address these problems.

3 Feature-Based Motion Graph

As usual, we build our motion capture database by concatenating multiple motions together. Each motion is composed of a sequence of frames. A frame $f = (p_r, q_0, q_1, ..., q_k)$ defines the pose of the character where p_r is the root position and q_i is the orientation of the i-th joint.

We start by segmenting the motions into semantically similar clips using feature detectors. For each motion type of interest, a feature detector is assigned to it. For example, for forward walking motions (straight and turning), the segmentation extracts from the motion capture database relatively small clips, each containing one walk cycle. Each frame is tested against a foot crossing binary test which looks whether the right ankle of the character is behind or in front

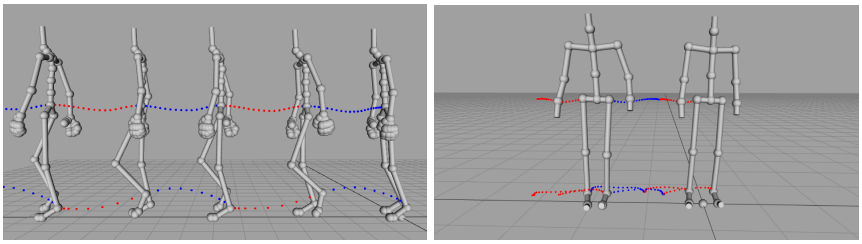

Fig. 1. Left: The alternating colors in the trajectories represent the segmentation of a walking motion into walk cycles. **Right:** Segmentation of steps similar to lateral steps. For clarity, only every second segmentation is displayed.

of the plane created by the left hip, right hip and the left ankle joint positions. This rule leads to a robust segmentation procedure as shown in Figure 1. Our motion capture database also contains lateral stepping motions. For these type of motions, we devised a feature that segments the motion when the velocity of both the left and right ankle joints is close to zero. See Figure 1 for an example.

The segmented clips start and end with frames that are very similar to the equivalent ones in the other segmented motion clips. This segmentation procedure is therefore suitable for a motion graph construction. Additional feature detectors can be designed in order to segment motions of different nature. Adding new feature detectors is straightforward and simple rules can achieve robust segmentation. For the purpose of locomotion synthesis, the two described segmentation criteria suffice to ensure that useful clips are obtained. The obtained clips are semantically similar in form and length and could potentially be categorized, parametrized or dropped if sufficient amount of motion segments have already been segmented. Another advantage of feature-based motion graphs is the automatic avoidance of foot-skating artifacts. Since blending operations are performed only at the extremities of each segmented clip, where there are only frames with one foot in contact with the floor, the skeleton can be re-parented at the contact foot before the blending operations, ensuring that the contact foot is not altered from its original position. This is done during transition generation avoiding any IK-based post-processing step for foot-skating correction. Motions extracted from a feature-based motion graph contain no foot-skating.

Our motion graph is then formed by performing a pairwise test between the initial and final frames of each segmented clip. A transition is created whenever the frame comparison metric returns a value under a threshold pre-specified by the user. We use the same distance metric and alignment transformation as in the original motion graph work [9]. In order to compute the rotation range for the transition, as will be required by our IK-based deformation model, we start with the initial transformation as returned by the metric and change the rotational component about the vertical axis in incremental steps in both clockwise and counterclockwise directions until before the transitional cost exceeds the predefined threshold. The achieved range is stored on each transition and defines

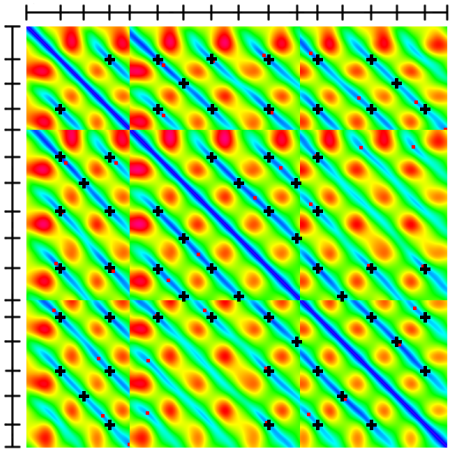

Fig. 2. 2D error image between the frames of 3 walking motion cycles containing 693 frames sampled at 60 Hz. The red regions (circular) represent highest error and the blue regions (thin and long) represent lowest error. The red points marked are the local minima and the black crosses are transitions detected by the feature segmentation. There are 57 black transitions and 42 red transitions. The bars at the top and left of the image indicate the frames that were selected during the feature segmentation phase. Black transitions are always located at intersections of segmented frames.

the allowed rotational range during motion deformation. As a final step, the largest strongly connected subgraph of the graph is selected. The remaining subgraph represents the final feature-based motion graph.

4 Analyzing Feature-Based Graphs

Figure 2 compares the transition locations selected with the standard motion graph procedure against the proposed feature segmentation. These transitions are superimposed on the 2D error image of the entire motion database. The same frame comparison threshold is used for both methods. Note that the computation of the 2D error image is required by the standard motion graph but not by the feature-based motion graph. The transitions selected by the standard motion graph are selected by detecting local minima in the 2D error image; whereas for feature-based graphs, candidate transitions are determined by the feature segmentation directly and are independent of the 2D error image.

Figure 2 shows that the feature-based transitions often occur in similar locations as the local minima ones. Since the possible transition frames are segmented by the same feature detectors, they satisfy the same geometrical constraints and thus have high chances of forming transitions. The motion capture database is also segmented at less points; moreover, our experiments show that the feature segmentation criterion will result on transition points that have higher connectivity with other transition points.

Table 1. Numerical comparison between standard motion graphs (SMG) and feature-based motion graphs (FMG). Column 'BF' illustrates the connectivity of each graph with its average branching factor, which is computed as the number of edges divided by the number of nodes.

	SMG				FMG			
Frames	Constr. time (s)	Nodes	Edges	BF	Constr. time (s)	Nodes	Edges	BF
693	208.1	42	16	1.27	1.5	51	80	1.61
1185	650.0	198	102	1.34	4.0	20	51	1.72
1660	1297.7	135	59	1.30	6.3	35	125	1.78
2329	2577.5	188	81	1.30	6.5	22	46	1.68
3347	4853.5	310	211	1.40	6.6	19	47	1.71
6887	22272.5	1245	933	1.42	27.0	100	403	1.80

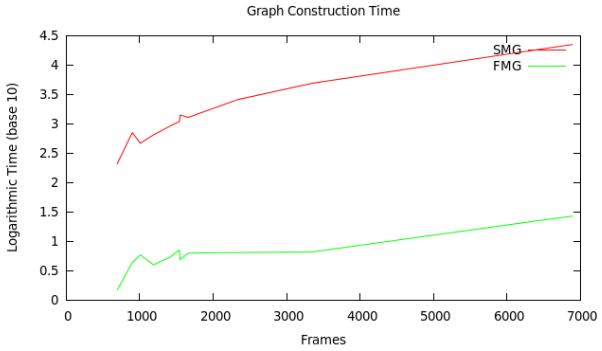

Fig. 3. Construction time spent for SMG (top line) and FMG (bottom line) as a function of the number of frames. The vertical axis represent time on a logarithmic scale (base 10). See also Table 1.

In order to evaluate our approach, we have computed both a standard motion graph (SMG) and a feature-based motion graph (FMG) from 6 different motion capture databases containing an increasing number of frames. Table 1 shows numerical comparisons between the structures, using the same transition threshold. It is possible to notice that FMGs have less nodes and more edges in most of the cases. The 'BF' column in the table shows the average branching factor obtained in each case. This column clearly indicates that FMGs exhibit higher connectivity compared to SMGs in all cases. This property makes FMGs particularly suitable for our deformation method.

FMGs are also computed much faster. The time spent for constructing a feature-based motion graph is often improved from several minutes to just a few seconds. Figure 3 depicts this comparison in logarithmic scale (with a base of 10) and shows that FMGs can be constructed 2 to 3 orders of magnitude faster than SMGs. For instance, it took 27s to create a feature-based graph from 6887 motion frames while the standard motion-graph took about 6h.

Table 2. Statistics when searching for locomotion sequences in different environments. The same threshold was used for both of the graphs. 'Size' is the number of nodes and 'BF' is the branching factor in each graph. 'Length' represents the average arc-length of the solution motions, measured with the character's root position projected to the floor. 'Exp' is the average number of node expansions during each search and 'Time' is the average time spent searching for each solution in seconds. The last three columns show the percentage improvement of FMGs over SMGs.

	SMG					FMG					Comparison		
Env.	Size	BF	Length	Exp.	Time	Size	BF	Length	Exp.	Time	Length	Exp.	Time
1	344	1.30	239.6	2391	145.3	130	1.66	219.8	1015	115.4	8.28	57.55	20.57
2	344	1.30	250.3	2903	148.1	130	1.66	229.2	1072	97.8	8.44	63.07	33.91
3	344	1.30	267.6	3745	145.6	130	1.66	243.0	1263	85.0	9.20	66.28	41.59
4	344	1.30	259.0	3671	121.9	130	1.66	234.3	1292	74.5	9.53	64.79	38.88

As showed in Table 1, FMGs contain less nodes and higher connectivity between nodes. The higher connectivity is key for improving the solutions generated from search queries. In order to quantify and evaluate the solution space of the graphs, we now present several experiments measuring and comparing the quality of our solutions in different environments with obstacles.

Table 2 summarizes the obtained statistics from 4 different environments with increasing number of obstacles. Both SMG and FMG were constructed using the same transition threshold. SMG had 344 nodes with an average branching factor of 1.30, whereas FMG had 130 nodes with an average branching factor of 1.66. Three metrics were used to compare the performance of the graphs: average optimal solution length, average expansion count and average search time (in seconds) spent during the graph search. As it can be seen on the table, FMG in average expands less nodes, spends less time searching and produces smaller errors in all of the cases. In average, FMGs show an improvement between 8-9% in length error and between 20-40% in search time in comparison to SMGs thanks to the increased solution space achieved with the higher connectivity.

5 Improving Search Among Obstacles with Channels

The search procedure used in the previous section represents the standard search solution for extracting locomotion sequences from a given motion graph. We now describe our improved search for faster motion extraction, which is based on constraining the search to pre-computed channels. While the idea of pruning the search has been explored before [17], our approach employs a new technique based on fast geometrically-computed channels.

We first compute a free 2D path on the floor plane between the current position of the character and the target position using an available triangulation-based path planning technique [7]. The computed paths are obtained well under a millisecond in the presented environments. The path is computed with guaranteed clearance, therefore guaranteeing that sufficient clearance for the character

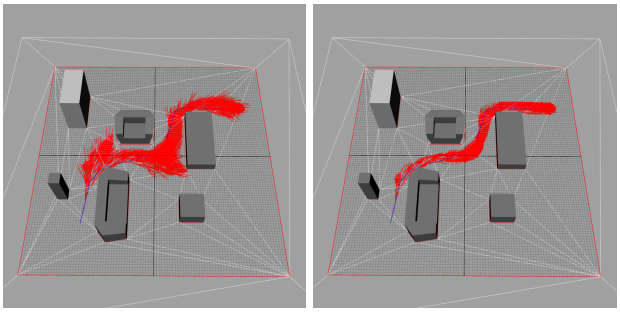

Fig. 4. Unrolled branches by a motion graph search with channel pruning disabled (left) and enabled (right).

to reach the goal is available. Once a 2D path is available, we perform a graph search by unrolling the motion graph in the environment and expanding only the nodes that remain close enough to the path. Since the path is guaranteed to be free of obstacles within its channel (i.e. within a distance r to the path), only nodes generating motion clips completely inside the free channel are expanded, and collision tests with the environment are not needed. As a result, faster searches are achieved by avoiding collision checking, which represents a major computational bottleneck when employed.

We test if a motion clip remains inside the free channel by projecting the position of key extreme joints of the character (like the hand and feet joints) to the floor, and measuring if their distances to the path are smaller than r. The projected positions are taken from the final frame and few intermediate frames of each motion clip. Collision tests are therefore reduced to point-path distance computations.

The overall search procedure starts from the node in the motion graph containing the initial character pose. This node is expanded, and every valid expansion remaining inside the free channel is inserted in a priority queue Q storing the expansion front of the search. The priority queue is sorted according to a function $f(node) = g(node) + h(node)$, where $g(node)$ is the cost-to-come value and $h(node)$ is the distance to the goal. The search stops when a node is within a distance d to the goal or when the priority queue is empty, in which case failure is reported. In our experiments d is set to 10cm. Figure 4 clearly depicts the advantage of confining the search within the computed channel.

6 IK-based Motion Deformation

Combined with our FMG and triangulation based pruning technique, a new *Inverse Branch Kinematics* (IBK) procedure is proposed for improving the obtained solutions. As previously mentioned, section 3, we compute a lower and upper limit for the rotational component of each created transition during the graph construction step.

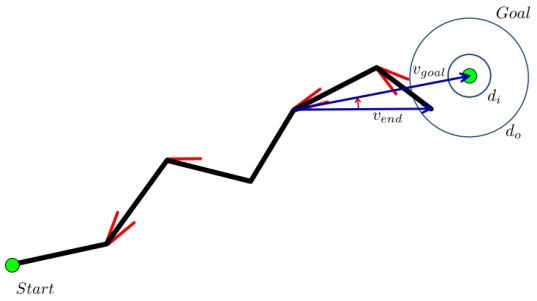

Fig. 5. A graph branch represented as a kinematic chain. Motion transitions are represented as rotational joints and the red lines represent the joint limits which are identical to the corresponding rotation limits stored in the motion graph transitions.

The search procedure is performed as previously described, stopping when a branch becomes close enough to the target in respect to a user-specified distance d_o. Then, our IBK solver is employed to iteratively optimize the solution towards the exact target location, up to a given tolerance d_i. Therefore, when $h(node) < d_o$ and $g(node) > dist(start, goal)$, $d_i < d_o$, the IBK procedure is invoked. In other words, when the distance between the node being expanded and the goal is under d_o and the length of the current path is longer than the Euclidean distance between the start and goal positions (meaning that there is room for a branch deformation), the search is then paused and the branch is deformed as a 2D kinematic chain with joint limits taken from the transition limits stored in the transitions. In our experiments we have set d_o to 50cm, d_i to 10cm. See Figure 5 for an example.

Depending on the nature of the transitions, the chain might have different joint limits. In Figure 5, the transition between the third and four node of the branch is not flexible. Thus this joint of the chain remains fixed. Also, the lower and upper joint limits do not have to be symmetric. For example, the second link has only room to move in respect to the upper limit. Once a candidate solution chain with its joints limits is obtained, the IBK solver can then evaluate rotational values at the joints in order to reach the target with the end-node of the search path. Several experiments were performed and our solver achieved best results with a Cyclic Coordinate Descent (CCD) solver [22]. We have in particular experimented with a Jacobian-based pseudo-inverse solver, however, in our highly constrained 2D kinematics chains the much simpler CCD solver was faster.

Each CCD iteration increments rotations at each joint, starting from the base joint towards the end-effector joint. At each joint two vectors v_{end} and v_{goal} are calculated. Vector v_{end} is from the current joint to the end-effector and v_{goal} is the vector from the current joint to the goal. These two vectors are shown in Figure 5 for the fifth link of the chain. The angle between the two vectors is incremented to the current joint and the result clipped against the joint limits. The last step of the CCD iteration consists of calculating the improvement from

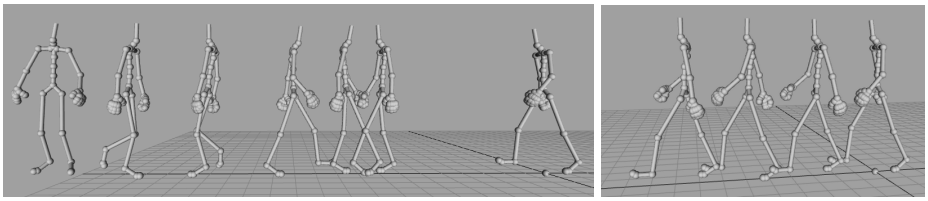

Fig. 6. The left image shows a typical problematic motion graph solution where an overly long motion is returned as solution to a given target. The right image shows the correct solution obtained by coupling the search with our IBK solver, which is able to deform nearby motions to the exact target without degrading the quality of the solution.

Table 3. Improvements gained when deploying IBK during search. Comparisons for both SMGs and FMGs with and without channel pruning for four different environments are shown. All values are represented as percentages. Each value is calculated as follows: if v is the value measured without deploying IBK and v_{ibk} is the value with IBK deployed then the reported percentage p is calculated a $p = -(v_{ibk} - v)/v$.

	SMG						FMG					
	Channel Prunning			Without Channel			Channel Prunning			Without Channel		
Env.	Length	Exp.	Time	Length	Exp.	Time	Length	Exp.	Time	Length	Exp.	Time
1	17.9	73.3	36.8	1.5	55.0	29.2	18.0	80.0	57.2	5.0	49.1	30.8
2	17.6	67.4	29.9	2.5	53.3	21.8	16.6	75.6	43.9	4.5	49.3	13.5
3	17.6	62.7	23.4	5.6	58.1	23.9	13.7	67.1	19.9	4.2	56.5	19.0
4	17.6	67.3	29.9	5.6	48.1	15.1	12.9	55.2	6.2	4.4	47.4	14.6

the previous iteration, which is given by how much closer the end-effector is to the target.

The CCD iterations stop if no improvements are detected after a number of iterations. At this point, if the distance between the end-effector and the goal is less then d_i then the solution with its new rotation values are evaluated for collisions. If no collisions occur, success is reported otherwise the search continues until another candidate branch is obtained. Figure 6 illustrates that in several cases the IK deformation is able to achieve a solution that otherwise the alternative solution without deformation would not be acceptable.

Table 3 shows the effect of employing the IBK solver for SMGs and FMGs with both the channel pruning enabled and disabled. As it can be seen from the table, IBK improves the generated solutions and reduces the search time in all the cases. The average improvement in the length of the motions when channel pruning is enabled is about 17% and 5% when pruning is disabled. The improvement on average on the search time is 31% when channel pruning is enabled and 21% when channel search is disabled. The reduced search times are a direct consequence of being able to terminate the search process early. This is possible because branches that are close to the goal can be deformed to meet the goal precisely.

7 Discussion and Conclusions

The presented evaluations demonstrate many advantages of the proposed feature segmentation, channel pruning and IBK deformation.

The first obvious advantage of the proposed FMG is that the construction time is dramatically improved in comparison to the standard motion graph procedure as our method does not need to compute a full 2D error image of the motion capture database (see Table 1). The fact that we do not search for transitions in the quadratic space of possibilities does not impose any drawbacks. On the contrary, we have shown that feature-based graphs have more connectivity and most often lead to improved results when applying search methods for locomotion synthesis around obstacles, which is always a challenging problem for discrete search structures to address. For instance, Table 2 shows up to 41% improvement on the time spent searching for all four environments.

We have also showed comparisons demonstrating the several improvements obtained by the channel pruning and the IK-based deformation technique (See Table 3). In all scenarios, these methods were able to improve both the quality of the synthesized solutions and the time taken during the search process. The IK-based procedure in particular represents a novel, simple, and effective way to optimize motions composed of motion capture segments.

In conclusion, we have presented new segmentation, search and deformation techniques for improving motion graphs. Our techniques significantly reduce the time spent for constructing the graph and at the same time lead to better solutions from search queries. We have demonstrated these benefits with several experiments in environments with obstacles, using both standard search procedures and the proposed channel pruning and IK-based motion deformation techniques. The proposed methods have showed to produce superior results in all cases.

Acknowledgments: this work was partially supported by NSF Award IIS-0915665.

References

1. Okan Arikan and David A. Forsyth. Synthesizing constrained motions from examples. *Proceedings of SIGGRAPH*, 21(3):483–490, 2002.
2. Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *Proceedings of SIGGRAPH*, 22(3):402–408, 2003.
3. Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, pages 117–126, 2008.
4. Min Gyu Choi, Jehee Lee, and Sung Yong Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *Proceedings of SIGGRAPH*, 22(2):182–203, 2002.
5. Claudia Esteves, Gustavo Arechavaleta, Julien Pettré, and Jean-Paul Laumond. Animation planning for virtual characters cooperation. *ACM Transaction on Graphics*, 25(2):319–339, 2006.

6. Michael Gleicher, Hyun Joon Shin, Lucas Kovar, and Andrew Jepsen. Snap-together motion: assembling run-time animations. In *Proceedings of the symposium on Interactive 3D graphics and Games (I3D)*, pages 181–188, NY, USA, 2003.
7. Marcelo Kallmann. Shortest paths with arbitrary clearance from navigation meshes. In *Proceedings of the Eurographics / SIGGRAPH Symposium on Computer Animation (SCA)*, 2010.
8. Manmyung Kim, Kyunglyul Hyun, Jongmin Kim, and Jehee Lee. Synchronized multi-character motion editing. *ACM Trans. Graph.*, 28(3):1–9, 2009.
9. Lucas Kovar, Michael Gleicher, and Frederic H. Pighin. Motion graphs. *Proceedings of SIGGRAPH*, 21(3):473–482, 2002.
10. Manfred Lau and James J. Kuffner. Behavior planning for character animation. In *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, pages 271–280, August 2005.
11. Manfred Lau and James J. Kuffner. Precomputed search trees: planning for interactive goal-driven animation. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*, pages 299–308, 2006.
12. Jehee Lee, Jinxiang Chai, Paul Reitsma, Jessica K Hodgins, and Nancy Pollard. Interactive control of avatars animated with human motion data. *Proceedings of SIGGRAPH*, 21(3):491–500, July 2002.
13. Sanjeev Khanna Liming Zhao, Aline Normoyle and Alla Safonova. Automatic construction of a minimum size motion graph. In *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.
14. Meinard Müller, Tido Röder, and Michael Clausen. Efficient content-based retrieval of motion capture data. In *Proceedings of SIGGRAPH*, pages 677–685, New York, NY, USA, 2005. ACM Press.
15. Jia Pan, Liangjun Zhang, Ming Lin, and Dinesh Manocha. A hybrid approach for synthesizing human motion in constrained environments. In *Conference on Computer Animation and Social Agents (CASA)*, 2010.
16. Cheng Ren, Liming Zhao, and Alla Safonova. Human motion synthesis with optimization-based graphs. *Computer Graphics Forum (In Proc. of Eurographics 2010, Sweden)*, 29(2), 2010.
17. Alla Safonova and Jessica K. Hodgins. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics (Proceedings. of SIGGRAPH)*, 26(3), 2007.
18. Hyun Joon Shin and Hyun Seok Oh. Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation (SCA)*, pages 291–298, 2006.
19. Makyu Sung, Lucas Kovar, and Michael Gleicher. Fast and accurate goal-directed motion synthesis for crowds. In *Proceedings of the Symposium on Computer Animation (SCA)*, jul 2005.
20. A. Treuille, Y. Lee, and Z. Popović. Near-optimal character animation with continuous control. In *Proceedings of ACM SIGGRAPH*. ACM Press, 2007.
21. Ben J.H. van Basten, Arjan Egges, and Roland Geraerts. Combinining path planners and motion graphs. *Computer Animation and Virtual Worlds*, 21:1–22, 2011.
22. L.-C.T. Wang and C.C. Chen. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 7(4):489–499, 1991.
23. Liming Zhao and Alla Safonova. Achieving good connectivity in motion graphs. In *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation (SCA)*, pages 127–136, July 2008.