Analyzing Locomotion Synthesis with Feature-Based Motion Graphs

Mentar Mahmudi, *Student Member, IEEE* and Marcelo Kallmann, *Member, IEEE*

Abstract—We propose feature-based motion graphs for realistic locomotion synthesis among obstacles. Among several advantages, feature-based motion graphs achieve improved results in search queries, eliminate the need of post-processing for foot skating removal, and reduce the computational requirements in comparison to traditional motion graphs. Our contributions are threefold. First, we show that choosing transitions based on relevant features significantly reduces graph construction time and leads to improved search performances. Second, we employ a fast channel search method that confines the motion graph search to a free channel with guaranteed clearance among obstacles, achieving faster and improved results that avoid expensive collision checking. Lastly, we present a motion deformation model based on Inverse Kinematics applied over the transitions of a solution branch. Each transition is assigned a continuous deformation range that does not exceed the original transition cost threshold specified by the user for the graph construction. The obtained deformation improves the reachability of the feature-based motion graph and in turn also reduces the time spent during search. The results obtained by the proposed methods are evaluated and quantified, and they demonstrate significant improvements in comparison to traditional motion graph techniques.

Index Terms—Computer Animation, Locomotion, Motion Capture, Human-like Motion Planning

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.

Given the achieved simplicity and quality of produced motions, motion graphs remain a popular method for motion synthesis. The fact that motion graph applications often require the use of search algorithms makes real-time performance to be difficult to achieve, in particular when searching for long motions. The focus of this work is to improve the performance of motion graphs for synthesis of locomotion among obstacles.

We propose a feature-based approach for constructing motion graphs. Feature-based motion graphs can be constructed significantly faster than traditional motion 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 character's skeleton. Once these suitable frames are detected, transitions are evaluated with the usual threshold-based comparison metric, thus maintaining the same quality of results when blending the two motion segments defined by one transition.

Depending on the application, different feature detectors can be employed. A wide range of applicable feature detectors have been proposed by Müller et al [1]. 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 [2], 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 local clearance triangulation of the environment [3]. For any given start and goal positions, the triangulation returns a collision-free channel that is used to prune the branches that lie outside the channel during the unrolling of a motion graph search. This results in improved search times, especially in environments with many obstacles.

One inherent problem of discrete search methods is the difficulty of reaching precise targets. In order to address this problem we introduce a simple and efficient Inverse Kinematics (IK) deformation technique. Each branch of the search is treated as a kinematic chain

• *The authors are with the School of Engineering of the University of California, Merced, CA, 95343.*
E-mail: {mmahmudi,mkallmann}@ucmerced.edu

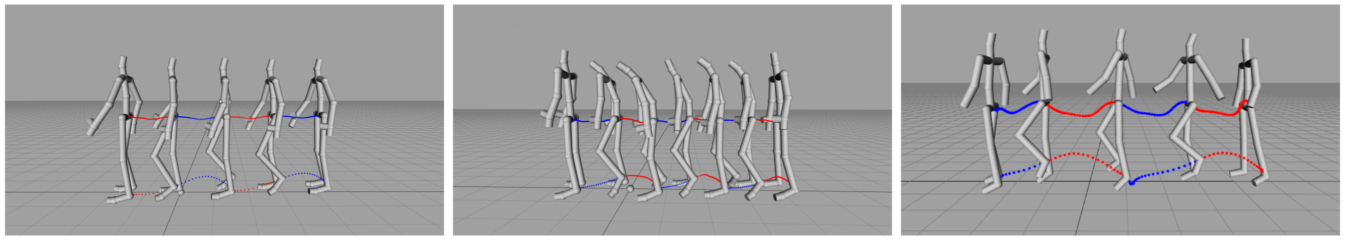

Fig. 1. Feature segmentation robustly segments walking motions into walk cycles. The images here show the correct segmentation obtained for a normal walking motion, sad walking motion and happy walking motion. Alternating colors indicate segmentation points.

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 search termination in several cases. These benefits do not sacrifice the quality of the motions as the quality of the deformed transitions will not exceed the same predefined threshold error used to construct the graph.

Lastly, we present extensive experiments with our methods solving locomotion synthesis among obstacles, and our results demonstrate significant improvements in time of computation, in finding solutions, and in the quality of the results.

This paper is an extended version of our previous work on the same topic [4]. This extended version describes the proposed methods in greater detail and includes several new experiments demonstrating and quantifying the advantages of the proposed techniques. For instance, we include a revised presentation of the Inverse Branch Kinematics method with a precise description of the transformations needed to model motion transitions as joints in our Inverse Kinematics solver. Most importantly, extensive new experiments are reported for evaluate the advantages of the proposed feature-based motion graphs in comparison to standard motion graphs: in respect to the total number of nodes (Figure 5), in respect to the branching factor (Figure 6), in respect to the length of paths (Figure 13), and in respect to the improvements obtained with channel pruning (Figure 14). Experiments quantifying the combined improvements using both channel pruning and IK deformation are also reported (Table 4), and an extended discussion of the overall method is included in Section 8.

2 Related Work

A large body of work has been devoted to animating characters using human motion capture data [5], [6], [7], [8], [9], [10], [11], [12], [13]. Motion graphs [5], [6], [8], [14] represent a popular approach that is based on the simple idea of 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. [5] 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 [6] build a hierarchy of graphs and use a randomized search to satisfy user constraints. Arikan et al. [7] use dynamic programming to search for motions satisfying user annotations. Lee et al. [8] 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 pairs of frames in the database. Our method improves on this operation by considering connections only between selected pairs of frames.

Many improvements to motion graphs have already been proposed. Ikemoto et al [15] use multi-way blends to create quick and enhanced transitions. Ren et al [16] combine motion graphs with constrained optimization. Shin and Oh [17] combine groups of parametrized edges into a fat edge for interactive control. Wang and Bodenheimer [18] evaluate cost functions for selecting transitions. Beaudoin et al [19] use a string-based model to organize large quantities of motion capture data in a compact manner. Our contributions focus on optimizing the motion graph construction, representation, and search, 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 [20], [21], [22], [23], [24], [25], [26]. In particular, Lau and Kuffner [27] manually build a behaviour-based finite state machine of motions, which in later work is precomputed [23] to speed up the search for solutions. Choi et al [28] combine motion segments with probabilistic roadmaps. These methods however require the user to manually organize motion examples in suitable ways.

The approach of *interpolated motion graphs* [29], [30], [31] is based on an interpolation of two time-scaled paths through the motion graph. Although the method increases the solution space, this comes with the expense of increasing the time spent to build and search the graph structure.

In respect to our proposed IK motion deformation technique, a related approach has been explored before by Kanoun et al. [32] for the problem of footstep planning for humanoid robots. The work describes an IK formulation that deforms a kinematic chain connecting footstep locations under specific constraints ensuring

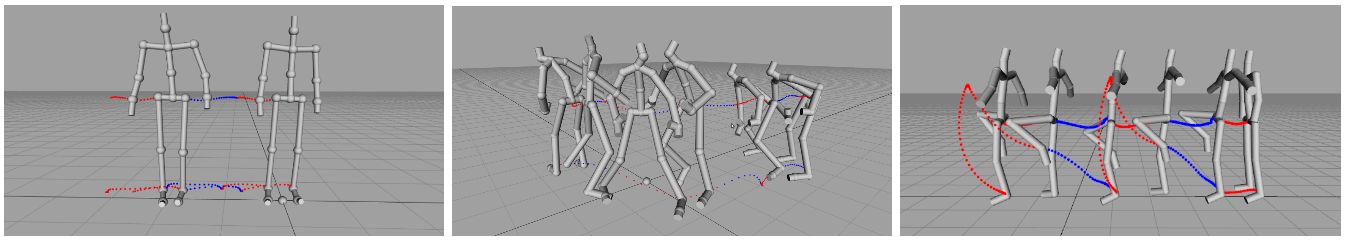

Fig. 2. Feature segmentation is robust for different types of motions. The left image shows the segmentation of a lateral stepping motion, the center image shows the segmentation of a basketball motion and the right image shows the segmentation for a ballet motion. For clarity, the character posture is shown only at every second segmentation for the lateral and basketball motions.

motion generation. Our method, however, is designed to solve a quite different problem, that of deforming motion transitions under a global motion transition threshold, and without generating foot skating artifacts.

In general, the main drawback of traditional 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 [33]. This paper presents new techniques for addressing these problems and together they significantly improve the performance of motion synthesis from motion graphs.

3 Feature-Based Motion Graph

As usual, we build our motion capture database by concatenating multiple motions together. Each motion M 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 the orientation of the i-th joint. A motion is defined in respect to a skeleton S, which defines the joint hierarchy and contains the joint offsets. When a frame F_i of a motion M is applied to its skeleton S, the global joint positions of the skeleton can then be calculated.

We start by segmenting the motions into semantically similar clips using feature detectors. A feature detector $\phi(F_i) \rightarrow \{0, 1\}$ is a binary function that evaluates whether a frame satisfies a feature property as defined by the feature detector. A motion is segmented at a frame F_i if $\phi(F_i) \neq \phi(F_{i+1})$.

In this manner, 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 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 Figures 1 and 2. Our motion capture database also contains lateral stepping motions. For these type of motions, we devised a feature detector that segments the motion when the velocity of both the left and right ankle joints is close to zero. See Figure 2 (left) 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. These 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 segmentation 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 transition 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 pair of segmented clips. A transition is created whenever the frame comparison metric returns a value under the transition threshold pre-specified by the user. We use the same distance metric and alignment transformation as in the original motion graph work [5].

We also compute during construction time the allowed rotational range at each transition, as required by our IK-based deformation method (described in Section 7). 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 pre-defined threshold. The achieved range is stored at each transition and defines the allowed rotational range during IK motion deformation.

As a final step, the largest strongly connected subgraph of the graph is selected. The obtained subgraph represents the final feature-based motion graph.

It is important to observe the significance of devising good feature detectors capable of picking appropriate transition points for the types of motions to be used in the motion graph. If the presented feature detectors are not adequate for a given type of motion, the resulting graph may have poor connectivity and achieve poor performances in search queries. The presented methods were devised to work well with locomotion sequences and an extensive analysis of the obtained results are presented in the next sections.

4 Analyzing Feature-Based Graphs

Figure 3 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 in 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 3 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.

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 IK deformation method since more connections (and at suitable frames in the walk cycle) are considered for deformation.

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 4 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.

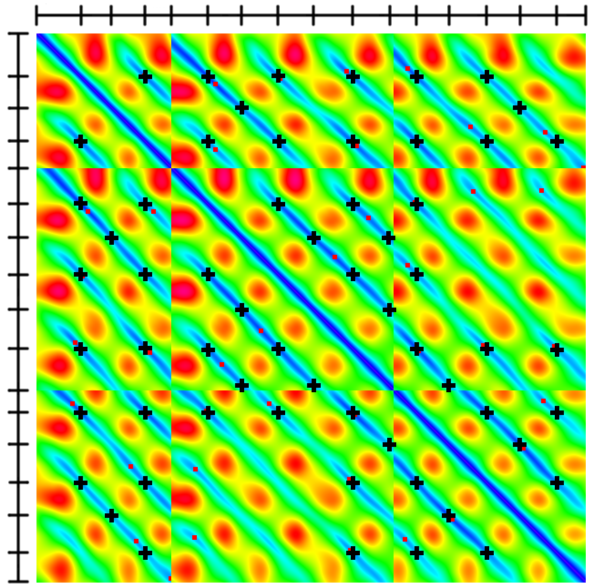

Fig. 3. 2D error image between the frames of 3 walking motion cycles containing 693 frames sampled at 60 Hz. The red regions represent highest error and the blue regions represent lowest error. The red points marked are the local minima and the black crosses are transitions detected by the feature segmentation. 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.

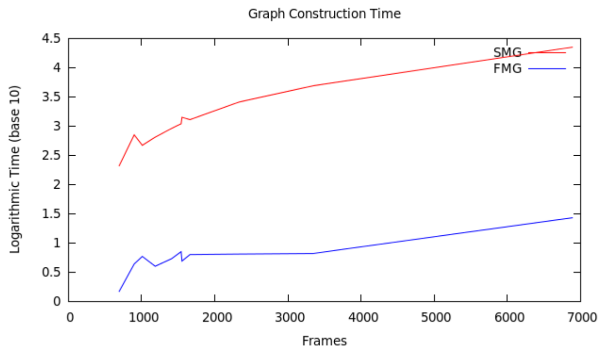

Fig. 4. Construction time spent for SMG (top/red line) and FMG (blue/bottom line) as a function of the number of frames. The vertical axis represent time on a logarithmic scale (base 10). See also Table 1.

For instance, it took 27s to create a feature-based graph from 6887 motion frames while the standard motion-graph took about 6h.

We have also analyzed the obtained size and connectivity in respect to the selected transition threshold. Figure 5 shows the number of nodes and average branching factor as a function of the transition threshold. The average branching factor is computed as the average number of edges per node. It can be observed that FMGs

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	Construction time (s)	Nodes	Edges	BF	Construction time (s)	Nodes	Edges	BF
693	208.1	42	16	1.27	1.5	51	80	1.61
1009	467.2	69	28	1.28	5.9	22	11	1.33
1539	1107.4	137	67	1.32	7.0	33	137	1.81
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

are more compact and better connected than SMGs. The higher branching factor for the FMGs is a direct consequence of checking transitions at selected suitable frames, allowing more than one transition to occur in a same lowest error connected component of the error image.

It is useful to observe why FMGs are more compact than SMGs with an example. In SMGs transitions can happen at any point $p = (i, j)$, where i and j are frame numbers. In FMGs transitions are only allowed at points generated by the feature segmentation. Suppose a motion M with 20 frames (enumerated from 0 to 19) is given as input and the transitions from the local minima segmentation are: $T_m = \{(5,7),(8,14),(15,10)\}$. In this case M will be segmented in clips $C_m = \{0-5, 5-7, 7-8, 8-10, 10-14, 14-15, 15-19\}$, resulting in a graph with 7 nodes. FMGs will first generate a set S of feature-based segmentation points, for example $S = \{5, 7, 10, 14\}$. Then, all transitions $\{(5,5),(5,7),(5,10),...,(14,10),(14,14)\}$ will be candidates but only those below the similarity threshold will be kept. Given that M is the same in both cases, it is highly probably that the transition set will be $T_f = \{(5,7),(7,14),(14,10)\}$. Note that $(8,14) \in T_m$ is close to $(7,14) \in T_f$ and $(15,10) \in T_m$ is close to $(14,10) \in T_f$. Given transitions T_f, motion M will be segmented in clips $C_f = \{0-5, 5-7, 7-10, 10-14, 14-19\}$; resulting in a graph of 5 nodes instead of 7. This example illustrates how the structured segmentation of FMGs often lead to graphs with fewer nodes, but also well representing the same motion capture database.

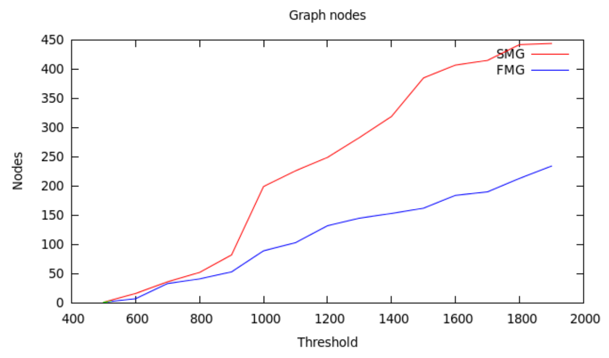

Fig. 5. Number of nodes in the standard motion graph (SMG, in red/top) and in feature-based motion graph (FMG, in blue/bottom) as a function of the transition threshold.

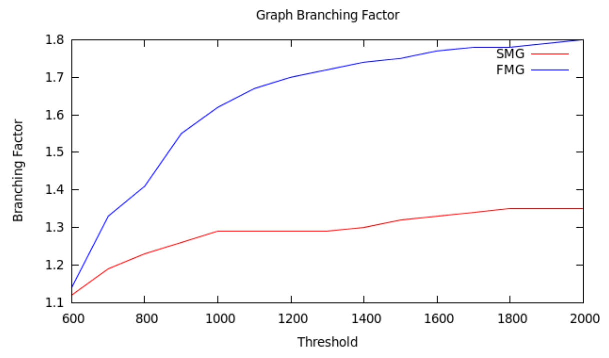

Fig. 6. Average branching factor in SMG (red/bottom) and in FMG (blue/top) as a function of the transition threshold.

5 Locomotion Synthesis

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.

Figure 13 illustrates the solution space of both graphs using color-coded error comparisons similar to the comparison methods by Reitsma et al. [33]. Each error image spans an environment with dimensions of 10 m by 10 m, and with cells of dimensions of 10 cm by 10 cm. For these comparisons we have not employed the triangulation-based pruning technique (Section 6) and the IK-based deformation (Section 7) as we are only interested in measuring the difference between FMGs and SMGs without any optimizations.

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 318 nodes with an average branching factor of 1.31, whereas FMG had 79 nodes with an average branching factor of 1.59. 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. A maximum expansion count of 100,000 was chosen for all the queries. If the maximum expansion count was reached and a solution was not found, a failure was reported. For sake of fairness, we only compared queries when both SMGs and FMGs were successful. The comparisons for SMGs and FMGs are presented in percentages and they range between -100% and 100%.

For the length comparisons we have defined the length error as follows: let P be the length of the 2D path that needs to be followed and let L be the length of the projection of the root joint trajectory of the solution motion that follows P. The length error is defined as $E = L/P$. Since the same set of trials were used for the comparison experiments, the value of P is the same, and therefore we only use L as a metric for comparing the length quality of the generated trials.

As Table 2 shows, in average, FMGs expand less nodes, spend less time searching and produce smaller errors in all of the cases. In average, FMGs show an improvement between 1-3% in length error and between 40-60% in search time in comparison to SMGs thanks to the increased solution space achieved with the higher connectivity. Our results show that choosing transitions at local minima does not always yield better results. Instead, transitions at key points detected by the feature segmentation generate better results and lead to faster searches.

6 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 [29], [34], 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 [3]. 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 to reach the goal is available (see Figure 7).

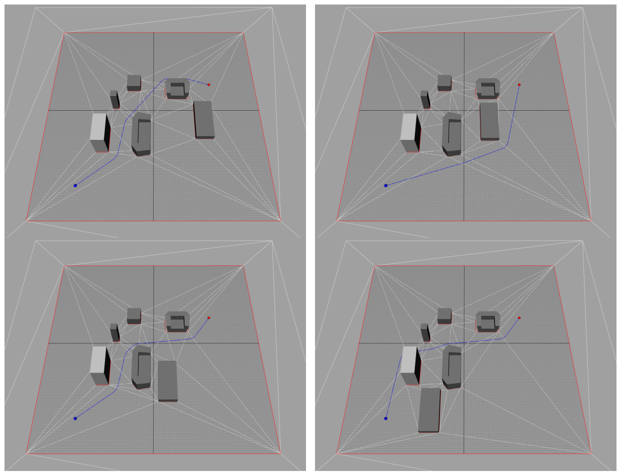
Fig. 7. Example of paths with clearance. The sequence of images illustrate that the floor triangulation can be updated very efficiently if obstacles move (under a millisecond in an average computer when up to about 100 vertices are involved). Therefore, channel pruning does not impose any requirements on the environment to be static and no other computation prior to a search query is needed.

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 expensive collision checks, which represent 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 an A* heuristic 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 expansion count exceeds a certain limit, in which case failure is reported. In our experiments d is set to 10cm and the expansion count limit is set to 1 million. The expansion count limit is necessary because otherwise the unrolling process could

TABLE 2
Statistics when searching for locomotion sequences in different environments. The same transition threshold was used in both graphs. “Total” is the total number of attempted queries. “Mutual” is the number of mutually successful queries. “Perc” is the percentage ratio between “Mutual” and “Total”. “‘Time” is the average time spent searching for each solution in seconds, “Length” is the average arc-length of the solution motions measured along the character’s root trajectory projected on the floor, and “Exp” is the average number of node expansions during each search. The last six columns show the mean and standard deviation of improvements (in percentage ratios) of FMGs over SMGs.

	Searches			SMG			FMG			Mean (%)			Std. Dev. (%)		
Env.	Total	Mutual	Perc.	Time	Length	Exp.	Time	Length	Exp.	Time	Length	Exp.	Time	Length	Exp.
1	7920	7386	93.26	0.41	602.51	3623	0.32	582.88	1805	45.43	3.13	66.99	40.08	6.69	35.32
2	6435	5930	92.15	2.18	777.84	17463	1.85	752.11	9498	39.48	3.85	59.76	39.72	5.41	36.45
3	5092	4423	86.86	2.81	890.79	22259	0.94	879.28	4429	60.84	1.31	76.07	16.65	2.13	13.48
4	4286	3451	80.52	2.91	862.18	23128	0.96	850.87	4565	59.31	1.33	75.20	18.70	2.40	15.12

continue indefinitely.

Another advantage of finding a path in the environment before starting the unrolling process is to avoid local minima. In general, the A* search will expand nodes blindly towards the goal without any consideration on the placement of the obstacles in the environment. As such, the search spends a lot of time expanding nodes towards one local minimum which might not be a part of the final solution. Such cases occur, for example, when the goal is right behind an obstacle. With channel pruning, a collision free path is already known in the triangulated free space of the environment. Therefore, the path returned by the triangulation serves as a guide to the A* search by confining the search within the free channel.

Figure 8 clearly depicts the advantage of confining the search within the computed channel. For the four environments showed in Figure 14, the improvements in search times were up to 40%. For a more complicated environment, such as the bottom example in Figure 8, the pruning technique was able to find a solution 10 times faster.

Failure in the described search procedure can happen if the motion graph does not have suitable connectivity. In other words, failure will occur if the search frontier runs out of leaves due to the obstacles or channel pruning. Feature-based motion graphs present themselves as a better choice for preventing the search to stop before reaching the goal location of the path. For example, Figure 9 illustrates a typical situation where FMGs are able to produce a locomotion sequence successfully following the entire path, while the SMG structure makes the search to run out of connections before reaching the goal. Figure 14 shows several comparison results obtained with the channel-based search procedure. It is possible to notice that, due the higher connectivity obtained, FMGs practically always produce better results.

7 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.

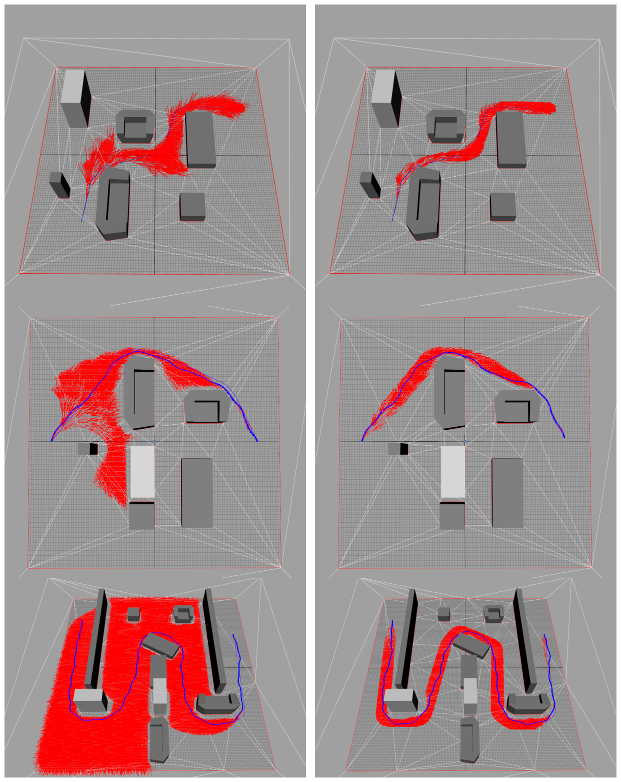
Fig. 8. Unrolled branches in two different environments by a motion graph search with channel pruning disabled (left column) and enabled (right column).

As previously mentioned in Section 3, we compute a lower and upper limit for the rotational component of each created transition during the graph construction step. A transition $Tr(i,j)$ is generated between the ith and jth frame of the motion capture database by using a transformation T_m that minimizes the distance between the interpolated motions as defined by the employed frame similarity metric. The limits are computed by measuring the error associated to rotational increments. For each new angle increment, the new transformation is calculated in the following manner:

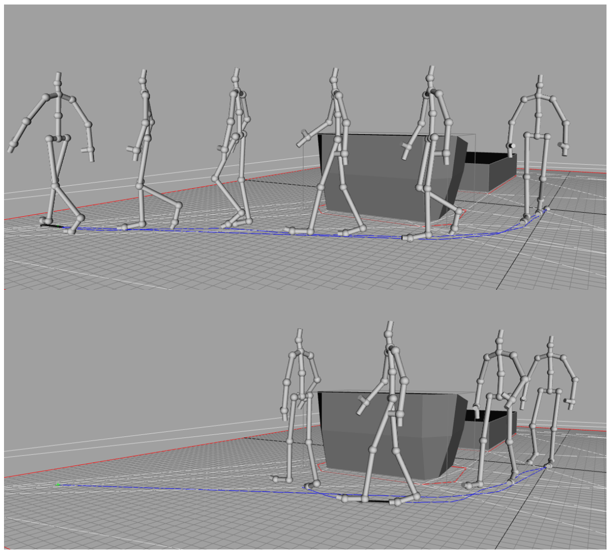

Fig. 9. Top image: the channel search procedure on a feature-based graph successfully computed a motion following the entire path. Bottom image: the same search failed in the standard motion graph. The same motion capture database and transition threshold were used in both cases.

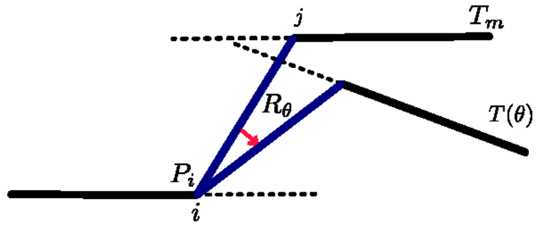

Fig. 10. Computing rotational joint limits for the IBK search procedure. The blue segment represents a transition from ith to the jth frame of the motion capture database.

$$T(\theta) = P_i^{-1}\ R_\theta\ P_i\ T_m, \tag{1}$$

where T_m is the transformation as returned by the original metric introduced by Kovar et al [5], R_θ is the rotational transformation of angle θ about the Y axis and P_i is the global position of the root joint of the ith frame of the motion database (see Figure 10). This procedure is repeated for both the upper and lower rotational limits while the transition cost does not exceed the global transition error threshold.

Later, for each motion search query, 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, the 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. See Figure 11 for an example. In our experiments, we have set d_i to have the same value as parameter d used in Section 6 (10cm), and we have set d_o to 50cm. Parameters d and d_i are kept the same in order to achieve meaningful results in our performance evaluations.

We have determined the values of d, d_i and d_o empirically after experimenting with the method in a few motion queries. These values have worked well in all our examples and we have not observed the need to modify them according to each query. One important factor that influences the choice for these values is the average length of the motions being computed. If improvements are needed, one possible extension is to fine-tune the predefined value of d_o as a function of the length of each obtained branch. In this way it is possible to have longer deformable branches being able to reach larger areas.

Depending on the nature of the transitions, the chain might have different joint limits. In Figure 11, the transition between the third and fourth 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 [35]. We have in particular experimented with a Jacobian-based pseudo-inverse solver, however, in our highly constrained 2D kinematic 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 11 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 the previous iteration, which is given by how much closer the end-effector is to the target.

The CCD iterations stop when 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

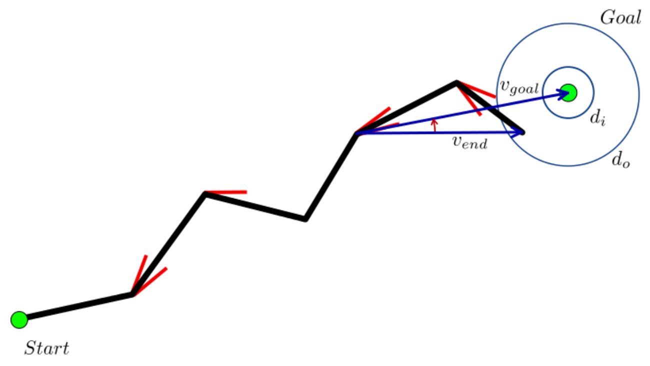

Fig. 11. A graph branch represented as a kinematic chain. Motion transitions are represented as rotational joints and the red segments represent the joint limits which are identical to the corresponding rotation limits stored in the transitions.

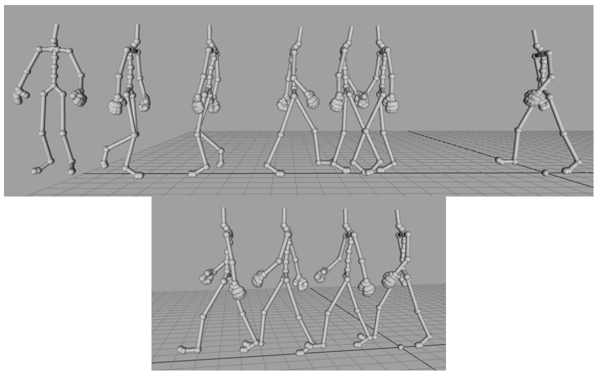

Fig. 12. The top image shows a typical problematic motion graph solution where an overly long motion is returned as solution to a given target. The bottom image shows the correct solution obtained by coupling the search with the IBK solver, which is able to deform nearby motions to the exact target without degrading the quality of the solution.

is evaluated for collisions. If no collisions occur, success is reported otherwise the optimization search continues until another candidate branch is obtained. Figure 12 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 time is 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.

The IBK optimization does not impose noticeable performance degradation since only a few iterations are performed in each search query and iterations only require extremely simple 2D operations.

The IBK deformation procedure not only improves search time and convergence to the goal point, but it is also formulated in a way to not introduce any undesirable artifacts such as foot skating. IK rotations are only allowed at transition points, which are always blended with the re-parenting strategy to the support foot in order to not generate foot skating. As a result, although IBK may introduce additional rotations to transitions, the resulting motion will automatically remain without foot skating artifacts.

8 Discussion

We now present several evaluations demonstrating the advantages of the proposed feature segmentation, channel pruning and IBK deformation. A video illustrating the presented results is available from the website of the authors [1].

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 60% improvement on the time spent searching in all four environments.

In addition to the significant improvement in construction time, feature-based segmentation also introduces semantics. For example, when specific feature detectors for forward and lateral steps are employed, each created motion segment in the graph can carry the label of its generating detector. This ability allows the user, for instance, to control the number of motion segments per feature type and to achieve compact graphs from large motion capture databases, or to specify search queries with only given types of motion segments in the solution. The employed feature detectors have also shown to robustly segment walking motions in several different styles.

Feature-based segmentation can furthermore help the user in determining the frame similarity threshold. For example, in the forward walk segmentation, all motion clips start and end with only one foot planted on the floor, with alternated support foot at the start and end frames. Naturally, there should not be a transition from

1. http://graphics.ucmerced.edu/publications.html

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 Pruning			Without Channel			Channel Pruning			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

TABLE 4
Improvements gained against SMGs for four different environments as the proposed techniques are deployed. All values are represented as percentages as explained in Table 3.

	None			Small			Medium			Large		
Technique	Length	Exp.	Time	Length	Exp.	Time	Length	Exp.	Time	Length	Exp.	Time
FMG	8.7	60.1	31.0	10.3	61.9	33.4	13.4	60.8	27.2	14.7	68.5	42.9
FMG + Channel	12.9	79.0	49.2	13.8	78.8	37.1	15.1	74.3	15.3	16.7	78.2	27.1
FMG + Channel + IBK	10.4	91.8	67.3	11.2	87.0	42.6	11.4	14.2	19.9	14.6	85.3	34.7

the end to the beginning of a same motion clip, i.e. a reflexive transition. Lowering the threshold until there are no reflexive transitions in the graph is a good way to determine an initial suitable transition threshold. The segmentation also naturally prevents foot sliding effects during transition blending because all transition points are suitable for re-parenting the skeleton to the support foot.

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 channel pruning technique was able to significantly speed up the search without affecting the optimality of the solutions. The procedure eliminates the need for collision checking and guides the A* search by confining the search to the collision-free channel and, thus, avoiding getting stuck in local minima.

Figure 8 and Figure 14 show that FMGs present themselves as a better option for motion synthesis in comparison to SMGs since they are less likely to run out of leaves and terminate the search prematurely. This is due to the fact that FMGs create suitable transitions. While SMGs minimize the transition cost, FMGs consider transitions from pair of frames detected by meaningful feature detectors (without exceeding the same error threshold as used in SMGs). Selecting transitions in this manner also enables the A* search to further benefit from channel pruning.

The IK-based procedure represents a novel, simple, and effective way to optimize motions composed of concatenations of motion capture segments. If more deformation capability is needed, the IK deformation can be extended to consider every frame of a motion as a joint and to also include translational deformations. However, feet slide cleaning operations would be required after deformation in order to maintain the original quality of the motions. By only considering rotational deformation at transitions, a simple, clean and reasonably effective deformation method is achieved.

Overall, the presented evaluations demonstrate that our methods significantly improve the usability of motion graphs. The simplicity and motion quality obtained by motion graphs are excellent and difficult to surpass with other methods. Motion graphs do not require manual preparation of motions and can be built automatically from a single parameter defining the allowed error in transitions. Motion deformation is only needed to form transitions (with blending), and thus the overall distortion is minimum. This is also the case when using the proposed IK-based deformation since it was designed to always respect the overall allowed error threshold.

Limitations and Avenues for Future Work

The main drawback of the proposed method is that relying on feature segmentation requires good feature detectors. If a feature detector is not adequate for a given set of motions, sufficient transitions may not be found. For example, the described walking feature detector will probably not be applicable to cartwheel motions, and if applied, it will probably lead to a motion graph with poor connectivity and, thus, not able to efficiently synthesize new motions. However, it is our experience that effective feature detectors can be developed with little effort, by determining where the most logical transition point should be. For instance, we have also used the presented walking feature detector to successfully segment different styles of walking and even ballet motions (see Figures 1 and 2).

An extensive collection of relevant feature detectors has been developed for the problem of motion comparison and retrieval [1], and they could be well applicable to segment a variety of motions. We have not analyzed how additional feature detectors would impact the construction of a motion graph from generic motions. However, it is also possible to devise a hybrid approach where the presented robust locomotion features are responsible for segmenting locomotion clips, and the standard local minima metric is used to find generic segmentation points in areas not covered by the feature segmentation. The investigation of such a hybrid strategy is, however, left as future work.

Another point to observe is that FMGs lead to less nodes in the motion graph and this could lead to an increased response time for interactive character control, since less transition points are available for switching to a new motion when required. We have taken this possible implication into consideration when choosing our locomotion features. For motions involving locomotion, the character will most often face a choice only when one foot is planted on the floor, therefore, the feature segmentation should not affect interactivity. Similar observations were made by previous methods using manually crafted motion clips [2].

More generally, FMGs are more compact because transitions are packed into "hub frames" that avoid over segmentation of nodes at many locations. The real aspect influencing interactivity would be the distance (in time) between nodes and not the number of nodes. Having poor interactivity performance would also imply poor search performance since less transition nodes would be available for finding solutions. FMGs have lower number of nodes but higher branching factors, the involved trade-offs were thoroughly examined in this paper and the results demonstrate only benefits. This indicates that our method, at least for locomotion, should not loose interactivity.

In terms of performance, the computation time taken during search can still be significantly improved, possibly getting close to real-time performances, with the pre-computation of limited-horizon search trees, a method that has been already explored for manually-built move graphs [23]. We suspect that the more structured segmentation of FMGs will be well suited for achieving pre-computed trees that can connect to each other even when built from an unstructured database. We intend to investigate this topic as future work.

9 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 significantly improved results in all cases.

The major result demonstrated in this paper is that choosing transitions at local minima will not lead to optimal reachability or optimal search performance in the resulting graph. Instead, well-designed feature detectors will lead to improved motion graphs in several aspects, in particular with superior performances in search queries.

Acknowledgments

This work was partially supported by NSF Award IIS-0915665.

References

[1] M. Müller, T. Röder, and M. Clausen, "Efficient content-based retrieval of motion capture data," in *Proceedings of SIGGRAPH*. New York, NY, USA: ACM Press, 2005, pp. 677–685.
[2] A. Treuille, Y. Lee, and Z. Popović, "Near-optimal character animation with continuous control," in *Proceedings of ACM SIGGRAPH*. ACM Press, 2007.
[3] M. Kallmann, "Shortest paths with arbitrary clearance from navigation meshes," in *Proceedings of the Eurographics / SIGGRAPH Symposium on Computer Animation (SCA)*, 2010.
[4] M. Mahmudi and M. Kallmann, "Feature-based locomotion with inverse branch kinematics," in *Proceedings of the 4th International Conference on Motion In Games (MIG)*, 2011.
[5] L. Kovar, M. Gleicher, and F. H. Pighin, "Motion graphs," *Proceedings of SIGGRAPH*, vol. 21, no. 3, pp. 473–482, 2002.
[6] O. Arikan and D. A. Forsyth, "Synthesizing constrained motions from examples," *Proceedings of SIGGRAPH*, vol. 21, no. 3, pp. 483–490, 2002.
[7] O. Arikan, D. A. Forsyth, and J. F. O'Brien, "Motion synthesis from annotations," *Proceedings of SIGGRAPH*, vol. 22, no. 3, pp. 402–408, 2003.
[8] J. Lee, J. Chai, P. Reitsma, J. K. Hodgins, and N. Pollard, "Interactive control of avatars animated with human motion data," *Proceedings of SIGGRAPH*, vol. 21, no. 3, pp. 491–500, July 2002.
[9] K. Pullen and C. Bregler, "Motion capture assisted animation: Texturing and synthesis," *Proceedings of SIGGRAPH*, pp. 501–508, 2002.
[10] Y. Li, T.-S. Wang, and H.-Y. Shum, "Motion texture: a two-level statistical model for character motion synthesis," *Proceedings of SIGGRAPH*, vol. 21, no. 3, pp. 465–472, 2002.
[11] S. I. Park, H. J. Shin, and S. Y. Shin, "On-line locomotion generation based on motion blending," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*. New York, NY, USA: ACM Press, 2002, pp. 105–111.
[12] T. Kwon and S. Y. Shin, "Motion modeling for on-line locomotion synthesis," in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*. New York, NY, USA: ACM Press, 2005, pp. 29–38.
[13] C. Rose, B. Bodenheimer, and M. F. Cohen, "Verbs and adverbs: Multidimensional motion interpolation," *IEEE Computer Graphics and Applications*, vol. 18, pp. 32–40, 1998.
[14] M. Gleicher, H. J. Shin, L. Kovar, and A. Jepsen, "Snap-together motion: assembling run-time animations," in *Proceedings of the symposium on Interactive 3D graphics and Games (I3D)*, NY, USA, 2003, pp. 181–188.

[15] L. Ikemoto, O. Arikan, and D. Forsyth, “Quick transitions with cached multi-way blends,” in *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, ser. I3D ’07. New York, NY, USA: ACM, 2007, pp. 145–151.

[16] C. Ren, L. Zhao, and A. Safonova, “Human motion synthesis with optimization-based graphs,” vol. 29, no. 2, 2010.

[17] H. J. Shin and H. S. Oh, “Fat graphs: constructing an interactive character with continuous controls,” in *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer animation (SCA)*, 2006, pp. 291–298.

[18] J. Wang and B. Bodenheimer, “An evaluation of a cost metric for selecting transitions between motion segments,” in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, ser. SCA ’03, 2003, pp. 232–238.

[19] P. Beaudoin, S. Coros, M. van de Panne, and P. Poulin, “Motion-motif graphs,” in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA)*, 2008, pp. 117–126.

[20] C. Esteves, G. Arechavaleta, J. Pettré, and J.-P. Laumond, “Animation planning for virtual characters cooperation,” *ACM Transaction on Graphics*, vol. 25, no. 2, pp. 319–339, 2006.

[21] J. Pan, L. Zhang, M. Lin, and D. Manocha, “A hybrid approach for synthesizing human motion in constrained environments,” in *Conference on Computer Animation and Social Agents (CASA)*, 2010.

[22] J. J. Kuffner and J.-C. Latombe, “Interactive manipulation planning for animated characters,” in *Proceedings of Pacific Graphics*, Hong Kong, October 2000, poster paper.

[23] M. Lau and J. J. Kuffner, “Precomputed search trees: planning for interactive goal-driven animation,” in *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)*. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2006, pp. 299–308.

[24] M. Sung, “Continuous motion graph for crowd simulation,” in *Technologies for E-Learning and Digital Entertainment*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007, vol. 4469, pp. 202–213.

[25] B. J. van Basten, A. Egges, and R. Geraerts, “Combining path planners and motion graphs,” *Computer Animation and Virtual Worlds*, vol. 21, pp. 1–22, 2011.

[26] M. Sung, L. Kovar, and M. Gleicher, “Fast and accurate goal-directed motion synthesis for crowds,” in *Proceedings of the Symposium on Computer Animation (SCA)*, jul 2005.

[27] M. Lau and J. J. Kuffner, “Behavior planning for character animation,” in *2005 ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, Aug. 2005, pp. 271–280.

[28] M. G. Choi, J. Lee, and S. Y. Shin, “Planning biped locomotion using motion capture data and probabilistic roadmaps,” *Proceedings of SIGGRAPH*, vol. 22, no. 2, pp. 182–203, 2002.

[29] A. Safonova and J. K. Hodgins, “Construction and optimal search of interpolated motion graphs,” *ACM Transactions on Graphics (Proceedings. of SIGGRAPH)*, vol. 26, no. 3, 2007.

[30] L. Zhao and A. Safonova, “Achieving good connectivity in motion graphs,” in *Proceedings of the 2008 ACM/Eurographics Symposium on Computer Animation (SCA)*, July 2008, pp. 127–136.

[31] S. K. Liming Zhao, Aline Normoyle and A. Safonova, “Automatic construction of a minimum size motion graph,” in *Proceedings of ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2009.

[32] O. Kanoun, J.-P. Laumond, and E. Yoshida, “Planning foot placements for a humanoid robot: A problem of inverse kinematics,” *I. J. Robotic Res.*, vol. 30, no. 4, pp. 476–485, 2011.

[33] P. S. A. Reitsma and N. S. Pollard, “Evaluating motion graphs for character animation,” *ACM Trans. Graph.*, vol. 26, October 2007.

[34] J. Chestnutt and J. Kuffner, “A tiered planning strategy for biped navigation,” in *Proceedings of the IEEE - RAS / RSJ Conference on Humanoid Robots*, November 2004.

[35] L.-C. Wang and C. Chen, “A combined optimization method for solving the inverse kinematics problem of mechanical manipulators.” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 4, pp. 489–499, 1991.

Mentar Mahmudi is currently a PhD student at University of California, Merced. He received his BSc degree (2006) in Electrical Engineering and Computer Science from Jacobs University in Bremen, Germany. His research interest are in human-like motion planning, computer animation and artificial intelligence.

Marcelo Kallmann is associate professor and founding faculty at the University of California Merced, and adjunct faculty at the University of Southern California. He received his PhD in computer science from the Swiss Federal Institute of Technology in Lausanne, Switzerland. His areas of interest include motion and path planning, computer animation, virtual reality and humanoid robotics.

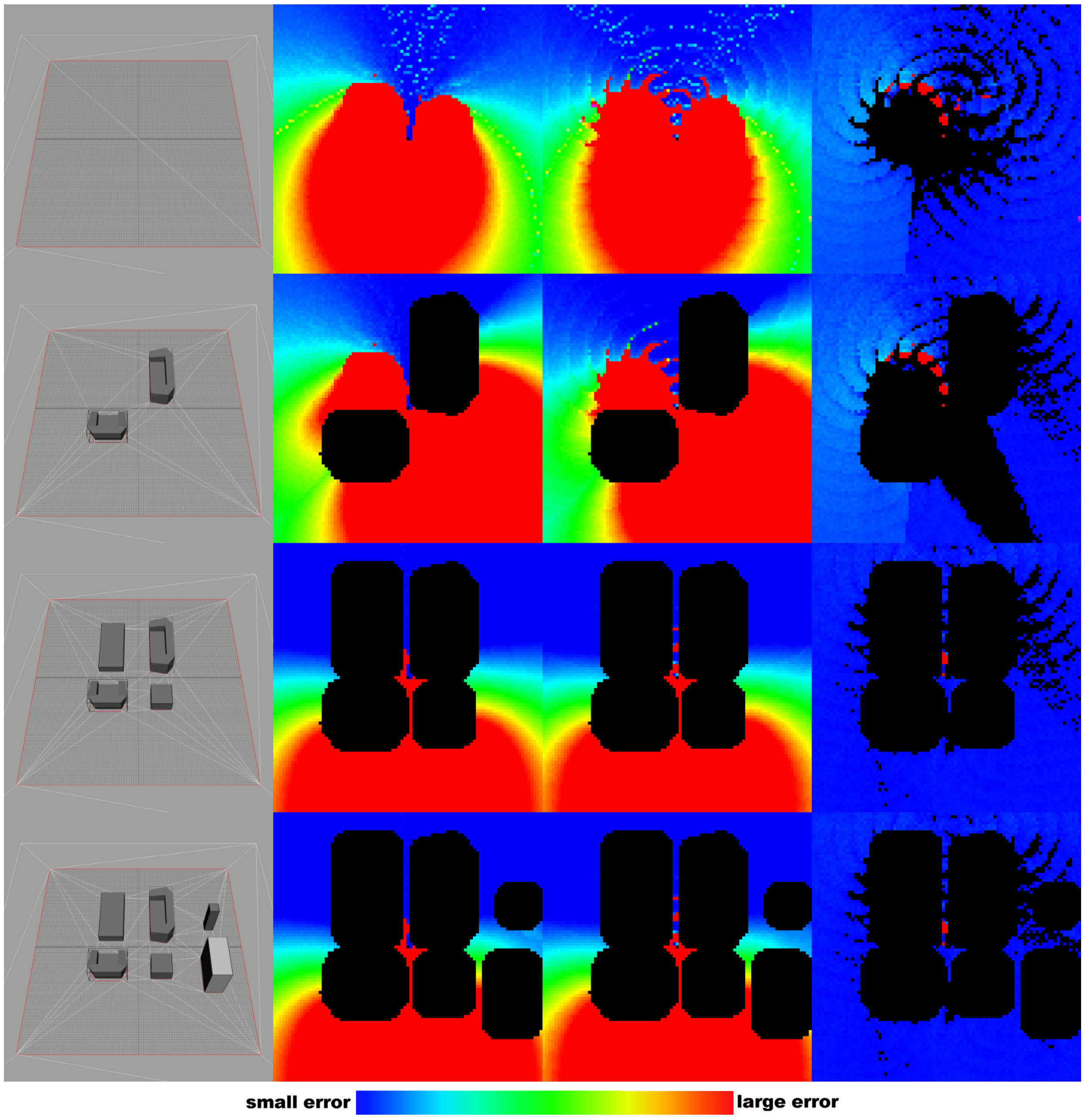

Fig. 13. Color-coded error comparisons when searching for locomotion sequences. The first column shows the used environments. The second and third columns show the errors obtained with the SMG and FMG respectively. The error $e = l/p$ is the ratio between the length l of the obtained motion from each method and the length p of the Euclidean shortest path on the floor plane (including a path clearance). A blue color means a solution length very similar to the shortest path, a red color means maximum error, which here is set to 2 times the length $(2p)$ of the shortest path. The character starts by facing the up direction and needs to first rotate down in order to reach the lower targets, which explains the large red areas in the images. The right-most column shows the error ratio $E = e_{smg}/e_{fmg}$ between both methods. A cell with black color means either an obstacle or that SMG is better, i.e. $E < 1$. For $1 < E < 2$, when FMG is better, the color scale is used ranging from blue, meaning slightly better, to red, meaning 2 times better. It is possible to notice that the errors obtained with the FMG are almost always lower.

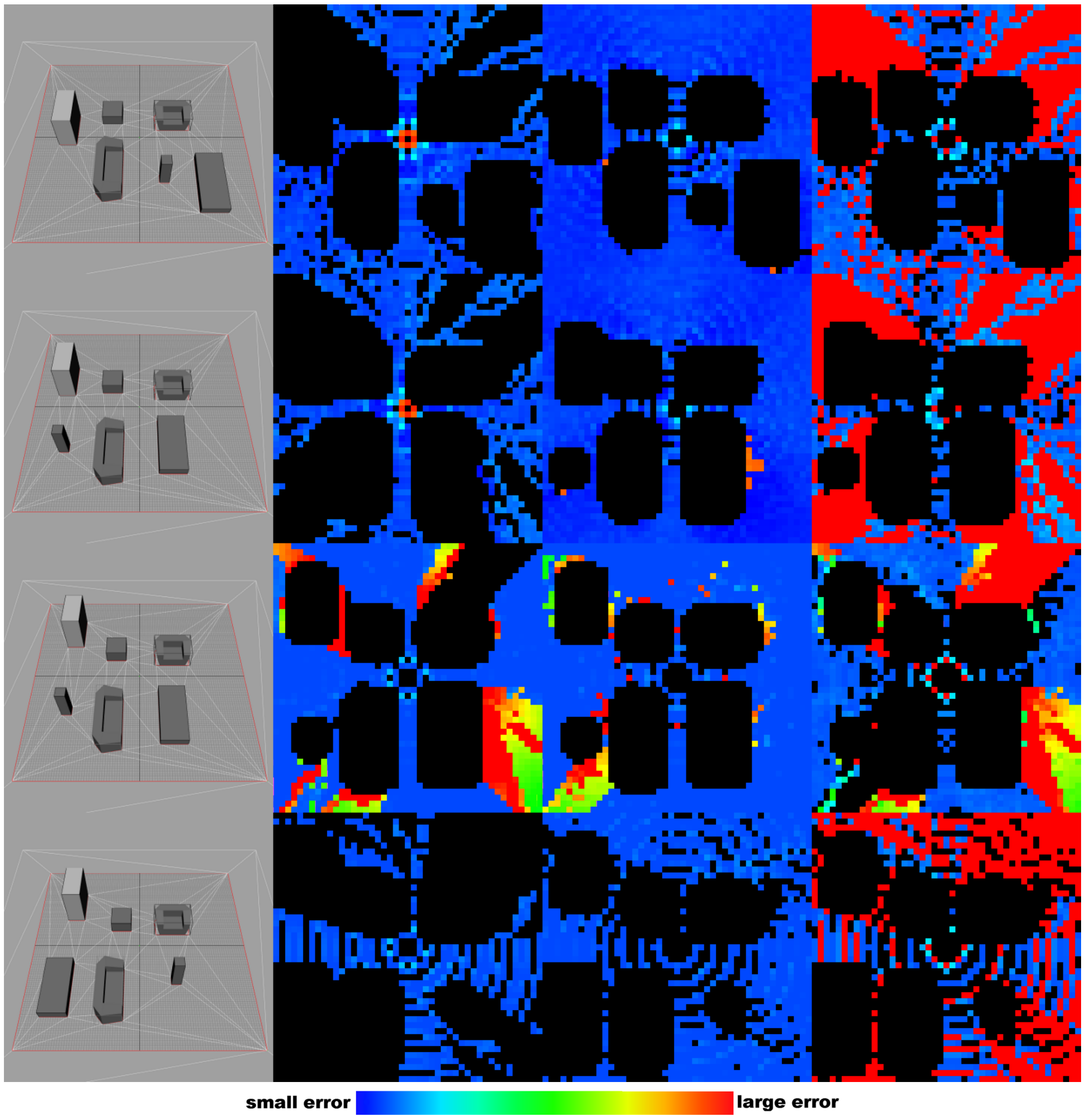

Fig. 14. Color-coded error comparisons when searching for locomotion sequences with the precomputed channel pruning enabled. The color coding for the comparisons here is the same as in Figure 13, with the exception that in the right-most column a cell is set to red (maximum error) when FMG successfully finds a solution but SMG fails. Another difference in respect to Figure 13 is that, instead of always starting oriented upwards, here the character's starting orientation is set to face the channel prior to the search, what makes the character to always start with a forward walking motion. The presented comparisons show that FMGs have lower errors and higher success rates than SMGs in most of the cells.