Planning Plausible Human Animation with Environment-aware Motion Sampling

Je-Ren Chen and Anthony Steed

Department of Computer Science,
University College London
{J.Chen,A.Steed}@cs.ucl.ac.uk

Abstract. The creation of plausible human animation remains a perennial problem in computer graphics. To construct long animations it is common to stitch together a sequence of motions from a motion database. Typically, this is done in two stages: planning a route and then sampling motions from the database to follow that route. We introduce an environment-aware motion sampling technique that combines the planning and sampling stages. Our observation is that in the traditional approach the route generated in the first stage over-constrains the motion sampling so that it is relatively implausible that a human would follow this animation. We combine the motion sampling and planning and show that we can find shorter and more plausible animations.

Keywords: Computer animation, motion sampling, motion planning

1 Introduction

The creation of animations for virtual humans is an essential part of many computer graphics-based productions. One important goal is to create animations that looks plausible. By plausible we mean that the animation would be feasible for a similarly-proportioned real human and it would be likely that a real human would choose a similar motion in an analogous real-world situation. Motion capture can be used to create animations (motion clips), but it is difficult to capture all the necessary animations for a production such as a video game. Thus, the common practise is to build a database of motion clips and then construct longer animations from these shorter clips.

To construct an animation from a motion database, a query to the database must be formulated. The query is formulated as a set of constraints such as starting position, end position, specific animations to play at certain points, waypoints to travel through, etc. It is common that the query is under-constrained in that there may be many sequences of clips that achieve the goal. (e.g., "walk from the door to the chair avoiding the table"). Given that we can find a set of candidate solutions that meet the constraints, we select between these by typically choosing the one with the shortest total path length, assuming this is the most plausible.

One way to search the motion database is to exhaustively search all possible sequences of clips, but this works only for short sequences. A typical situation involving longer sequences is when the character needs to navigate through the environment. A common strategy in this situation is to decouple the query into a path-finding stage and a path-following stage. The first stage searches for a path, i.e. a 2-D or 3-D curve, that travels through the free space between obstacles and reaches the goal. The second stage queries the motion database to find a sequence of clips that follows this curve. We can then ask two questions: can we generate an animation that follows the curve? Will the resulting animation be plausible? Our first observation is that in order to search for a path, one must make a conservative assumption about how far to stay away from the obstacle. We might thus inadvertently exclude some animations from consideration because an individual motion clip might be able to pass the obstacle more closely. Our second observation is that once a path is generated, we need to approximate it by motion clips. In doing this approximation we might need to select clips that would not be plausible from a high-level view of what sequence of motions a real human would take. Our belief is that both of these problems result from the decoupling of the path-finding stage from the path-following stage.

We propose an environment-aware motion sampler which allows us to generate plausible virtual human animations that navigate through an environment with obstacles. We propose to couple the path-finding and path-following stages in a search of a motion database. In Section 2 we discuss related work. In Section 3 we present an overview of our approach, and the new environment-aware motion sampling (EMS) algorithm. In Section 4 we explain how path-following can be achieved with explicit constraints. The main contribution is in Section 5 where we show how obstacle avoidance can be introduced as an implicit constraint. Section 6 then shows how we can find animations where the previous standard technique cannot in typical navigation problems.

2 Related Work

The purpose of motion planning is to generate motions which moves a subject from one location to another without any collisions with least effort. Commonly this is simplified as finding a shortest collision-free path to the goal. One way of finding such a path is to approximate the space by decomposing the free space into connected proximities [4] or using a grid to store discretized free space information [2]. In 2-D or 3-D it is efficient enough to search for paths connecting free cells in approximate representations, but these are inefficient when the subject has many DOFs, such as a human-like character. Thus, they are suitable for generating ground paths at a coarser level which we then can use as constraints to query a motion database to follow it.

Another approach is to take random samples of the free space and summarise it as roadmaps [12]. For example, Choi et al. [3] approximate footprints on the a roadmap and re-target biped motions on a given route, but they only evaluate the collision of the footprints rather than entire body motions. It is also possible to represent motions as sampled points with tree structures for later determination. For example, precomputed search trees [7] have been applied with motion data by plotting them into point clouds so they are intersected with obstacles to eliminate invalid branches. Our research shares a similar goal with theirs, i.e., planning animation with collision detection using motion data. However, we estimate collision errors as a soft constraint instead of using the binary decision of accepting or rejecting a solution.

Mizuguchi et al., [11], describe how motions are authored and constructed for character motion control in game production. Motions representing in-game actions and the transitions between them are hand-edited by animators such that motions are blended together to control the character at run-time. To reduce the burden of manual input, many researchers have investigated different ways of detecting and querying transitions. The term "motion graphs" ([6][1][8]) are used to refer various techniques that detect motion transitions by making use of frame distance matrices. To generate an animation that satisfies user-specified goals, a motion graph defines constraints, such as desired locations and orientations, and a cost function to search for an optimal edge sequence that minimises the cost. Hence, navigation animation is typically done by identifying all necessary constraints along a feasible route obtained from a preceeding planning stage.

One way of searching for a solution that navigates through obstacles is to enumerate the states in the control-parameter space and connect these states with corresponding paths of the motion graph. Reitsma and Pollard [13] embed a motion graph into a 4-D grid space that consists of the ground position, orientation, and the motion type such that they are able to evaluate the motion graph coverage in the scene and the motion path quality of the search. We also measure the cost of an animation by calculating the ratio of the actual travel distance to an expected path length. Safonova and Hodgins [14] also apply a an embedded motion graph but they allow the interpolation between two source motions rather than a single motion type in each state so that they are able to improve the optimality of the animation. However, the main drawback of these methods is that they need to re-embed the motion graph whenever the scene is changed.

Another way of finding a motion graph solution is using randomised search. Arikan and Forsyth [1] applied a Monte Carlo simulation on motion graph search by generating an initial random graph walk and repeatedly re-enumerating sub-optimal edges with alternatives towards satisfying the constraints. We adapt their method to our motion query in this research. Whilst their method works with multiple constraints that follows a curve, we incorporate collision errors as implicit costs in the optimisation.

Reinforcement learnining techniques are also applied to generate navigation animation ([15] [9] [10]). They do so by incorporating the obstacle configurations in the parameter space so that they can pre-compute the costs for each condition and select the best one to avoid the obstacle at run-time. However, due to limited dimensionality of the parameter space, they can only allow very few numbers of obstacle configurations to be considered. Our implicit cost function estimates

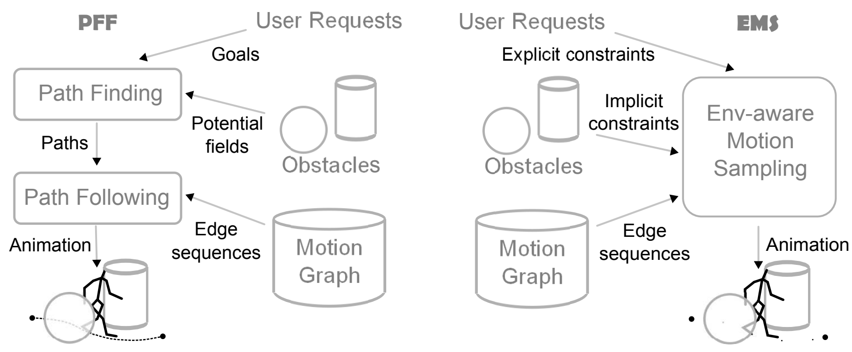

Fig. 1. The approach of decoupled search (left) and joint optimisation (right).

the collision error directly in the work space and can be scaled with any number of obstacles.

3 Overview

We wish to generate plausible full-body human motions for navigating a virtual environment. One typical way is to query a motion database, such as a motion graph (see Section 2), by specifying as many waypoints as needed to avoid the obstacles until we reach the goal. The waypoints are obtained by some path finding technique in a preceding stage. We propose to formulate navigation animation generation as a planning problem by jointly optimising both explicit (user-specified goals) and implicit (environmental obstacles) constraints. The aim of the optimisation is to sample a sequence of clips that minimises the explicit and implicit costs.

In this paper, we compare our joint optimisation method, called environment-aware motion sampling (EMS), with a typical decoupled search approach, called path finding and following (PFF). Figure 1 compares the two approaches. Both approaches have the same inputs: the user-requested constraints such as locations, orientations, and desired actions; the virtual environment; and a motion graph. The obstacles in the scene are primitive geometries, and the motion graph is built by connecting similar poses between each motion sequence in the locomotion library. The EMS and PFF methods both rely on a motion graph to query a motion database. They also both employ potential-field path finding [2] to select a route through the environment. While PFF applies potential fields to generate a collision-free path and searches for animation that follows this path, EMS synthesises a full-body collision-free animation directly using the collision errors within the cost optimisation.

Motion Graphs. As discussed in Section 2, a motion graph is a database technique that can generate a sequence of motions from a library. To query the database, we specify certain constraints and search for a solution that minimise a cost function. We implement the Monte Carlo simulation-based motion graph search method (MCMG) [1]. This method begins a search with a random edge sequence. In each iteration, MCMG proposes the best few candidates by scoring their costs and optimises these best-so-far solutions by modifying their sub-optimal edges with their alternatives. To search the graph more efficiently, the edges are arranged as tree structures such that edge modifications, or *mutations*, only enumerate the root-level edges of the trees (hop mutations), while finer tuning of the transitions can still be made by replacing an edge with its children (demote mutations). Once an edge sequence is obtained, the animation is derived by stitching together the corresponding motion clips. Please refer to the original work for technical details [1]. This paper will focus on how we design the explicit and implicit constraints and the cost functions in order to facilitate the search algorithm for character navigation. The algorithm we describe would be applicable to types of query other than navigation.

Potential-field Path Finding. For path finding, we use potential fields [2] since it is easy to implement and is considered to be effective in low-dimensional path finding problems. The potential field method finds a path by discretising the workspace into a grid of cells and propagating a gradient of ascending distances in the free space from the goal point. The cells in this field, the potentials, are then used as a heuristic to search for a connected chain of cells from a given initial location by moving to a lower potential neighbour cell. The chain is converted into a curve (a poly-line) by connecting the centroids of consecutive cells. The curve is further straightened if any segments does not collide with an obstacle. Such property maintains the shortest path length for this route. Although we can obtain a short and collision-free path by using potential fields, we cannot query an animation to follow this path as it does not consider the volume of the character. A common workaround is to sample the animation along the curve with a safety distance to an obstacle. To achieve this in PFF, we estimate a maximum bounding cylinder from the library, and use its radius to thicken obstacles when building potential fields. We will describe how potential fields are applied to obtain the path to be followed for PFF in Section 4 and how it is used to estimate the minimal travel distance for EMS in Section 5. Note that the potential field path finding method used in this research can be replaced by other 2-D path finding techniques.

Decoupled Search versus Joint Optimisation. An animation query begins with a pair of explicit constraints: initial location, orientation and action (i.e., a specific motion clip), and goal location, orientation and action. In decoupled search, PFF first finds a poly-line that connects initial and goal locations in the path finding stage. During path following, intermediate constraints are densely planted along this path as waypoints. PFF then uses an objective function to search for a solution that strictly travels through these exact waypoints within a certain radius by minimising $Cost_{PFF} = Error_{smooth} + Error_{penalisedExplicit}$, where $Error_{penalisedExplicit}$ is the total deviation between the solution and waypoints (described in Section 4). The term $Error_{smooth}$ maintains the smoothness of the transitions and is the sum of the cost of a transition estimated by the dissimilarity between poses obtained during motion graph construction ([1]). For joint optimisation, EMS also performs path finding but does not enforce path following. Instead, EMS only considers the vertices of the poly-line as approximate waypoints to sample towards a preferable short route and to estimate the expected travel distance. Thus, EMS searches for an edge sequence that travels close to these sparse waypoints and avoid obstacles by optimising $Cost_{EMS} = Error_{smooth} + Error_{explicit} + Error_{implicit}$. We will describe how to estimate $Error_{implicit}$ in Section 5.

4 Animation Control using Explicit Constraints

We first consider sampling motions with explicit user specifications but without the implicit constraints of collision avoidance. We begin by considering only two explicit constraints. We then describe how to sample motions with multiple constraints, and how to partially enforce constraints to allow sampling an animation along a series of constraints such as those implied by the curve generated by the PFF method.

Explicit Constraints. To animate a character moving from one spot to another, we define an explicit constraint c to be a set of triple such that $c = (act, loc, ori)$ to specify required *action*, *location*, and *orientation* constraints. Actions are labelled poses (frames) to specify a motion to be happened at a particular point. Actions are hard constraints and are obeyed by searching among the edge sequences that contain these required poses. A location constraint specifies the required position of the character's centre of mass projected on the ground (x-z plane). The orientation constraint specifies the required angle between the character's facing direction (obtained by projecting the local z-axis of the body on the ground) and the world z-axis. The location and orientation constraints are used to estimate the deviation of sampled edge sequences in the explicit objective function.

Explicit Objective Function. Let $\Omega = (e_1, ..., e_n)$ be a proposed edge sequence sampled from the motion graph with a pair of initial and goal condition as $C_{s,t} = (c_s, c_t)$ and let f be a frame of the animation derived from Ω. We define functions $locate(f)$ and $orient(f)$ to estimate the accumulated location and orientation of any f. Since we require the animation to be derived from the initial constraint c_s, the error between Ω and $C_{s,t}$ is estimated by calculating the positional and rotational deviations between the end frame f_t and the goal constraint c_t as

$$deviate(\Omega, c_t) = w_l \cdot (|locate(f_t) - loc_t|)^2 + w_o \cdot ((orient(f_t) - ori_t))^2, \quad (1)$$

where weights w_l and w_o are chosen such that the error yielded by $30cm$ in location is equivelent to the error of $10°$ in orientation, as suggested in [1].

Multiple and Partial Explicit Constraints. We may also wish to generate an animation that satisfies more than two constraints. We insert multiple constraints between c_s and c_t and accumulate the deviations raised at each constraint with Equation 1. However, to calculate the deviation, we need to determine the corresponding frame for the action of each constraint since there might exist more than one frame that satisfy the action constraint. Let $C = (c_1, ..., c_n)$ be a series of explicit constraints. We find f_i for each c_i by selecting frames in Ω that contain act_i and choose the one with the least deviation by

$$f_i = \arg\min_F deviate(\Omega, c_i), \tag{2}$$

where F is the set of frames that are of action type act_i found in the animation that corresponds Ω. We then chose the deviation of f_i to be the the error increased by c_i.

Occasionally, we only require the character to travel through certain locations regardless of their actions or orientations. We can partially enforce an explicit constraint by discarding its action constraint and orientation deviation in Equation 1. Thus, similar to Equation , for each partial explicit constraint, we find an f_i among all frames that minimises $deviate(\Omega, c_i)$ regardless the orientation. Finally, the total explicit error is

$$explicit(\Omega, C) = \sum_1^n deviate(\Omega, c_i). \tag{3}$$

Path Following using Explicit Constraints. Once we can sample motions with multiple and partial explicit constraints, we can generate an animation that follows a curve with the motion graph. We plant intermediate partial constraints as waypoints along the curve obtained from the path-finding stage with step size d_{step}. However, as mentioned in Section 3, we cannot simply find and follow a path regardless of the volume of the body. Thus, we estimate a minimum bounding cylinder from the library and apply the radius r_{body} of the cylinder to expand the obstacles in the potential field. To ensure we sample the animation moving only inside this corridor, we penalise those frames that are far away from the intermediate partial constraints by adding

$$penalise(f_k, c_i) = (\frac{|locate(f_k) - loc_i|}{r_{body}})^{p_{follow}}, \tag{4}$$

where $p_{follow} >= 1$ increases the penalty. Finally, we are able to generate an animation to follow the path by using $explicit(\Omega, C)$ for $Error_{penalisedExplicit}$ for $Cost_{PFF}$ in Section 3.

5 Collision Avoidance with Implicit Constraints

We now present our new approach where we introduce two implicit constraints to the search: the environmental obstacles and minimal travel distance constraints.

Implicit Constraints. We define two types of implicit constraints: one that enforces obstacle avoidance and one that selects minimal travel distance. We design the obstacle constraints using three parametric primitives (spheres, cylinders and cubes) so that collision detection can use efficient distance measurements. Scenes can be approximated by sets of transformed primitives. For each frame, we collide a point cloud of the joint positions with the obstacles to estimate the collision error. The minimal travel distance constraint maintains a minimal cost from taking longer alternative motions to avoid obstacles. In reality, humans can move along straight lines between two locations. However, this is often not the case when sampling an animation as we may not be able to find exact motions in the library. We suggest linear distances between goals as an expected travel distance and we wish to sample motions that move along these straight lines as close as possible. Hence, the minimal travel distance constraint is an error computed as the ratio of the required travel distance to the expected travel distance.

Implicit Objective Function. Let $O = (o_1, ..., o_m)$ be a set of obstacles, we define a boolean function $isInside(o_i, p)$ to determine whether a point p is inside an obstacle o_i. The collision error between the obstacle set O and each frame is estimated by

$$err_{collide}(O, f) = \sum_{i=1}^{m} \sum_{j=i}^{n} \frac{isInside(o_i, jointPos(f, j))}{n}, \tag{5}$$

where m and n are obstacle and joint numbers and $jointPos(f, j)$ is a function to obtain the world position of a joint at frame f. In practise, we perform a coarse bounding-box collision between obstacles and the point cloud to select a smaller set of O such that we do not collide with every single obstacle. The minimal travel distance error is the ratio of the required travel distance to the expected travel distance:

$$err_{travel}(\Omega, C, O) = \frac{arclength(\Omega) + |locate(f_m) - loc_n|}{arclength((c_1, ..., c_n))}. \tag{6}$$

where $arclength(\Omega)$ is the travelled distance derived from the animation and $arclength((c_1, ..., c_n))$ is the total length of the poly-line formed by the explicit constraints from initial to final goal. The distance between $locate(f_m)$ and loc_n is a penalty applied to solutions that are short but are not close to the final goal. Finally, the total cost of the implicit errors is calculated as:

$$implicit(\Omega, C, O) = (\sum_{k=1}^{n} err_{collide}(O, f_k))^2 + (w_d \cdot (err_{travel}(\Omega, C, O) - 1))^2, \tag{7}$$

where n is the number of frames of Ω and w_d is chosen by experiment given that w_l and w_o in the explicit errors (see Equation 1) are fixed.

Scene Navigation via Joint Optimisation. By introducing $implicit(\Omega, C, O)$ to $Cost_{EMS}$ described in Section 3, we are able to estimate the cost of a sampled solution. For joint optimisation, the EMS algorithm also applies potential path finding to obtain a path from the explicit constraints in the initialisation stage. Unlike path-finding in PFF, we do not expand obstacles since we require the path to be the shortest possible: the path arclength will be used to estimate the expected travel distance in Equation 6. Also, instead of strictly following the path, we only insert the vertices of this path as intermediate partial constraints to converge to a solution with a shorter route.

6 Implementation and Results

We have implemented both methods using the C++ programming language and integrated them as a plug-in for the Maya animation software. We use the potential-field path finding technique to generate a path from the initial and goal constraints specified by the user. We found $d_{cell} = 5cm$ to be a reasonable size for cells in the potential field to generate good quality paths in our experiments. For both PFF and EMS, we generate an initial random edge sequence that connects two poses of the required actions in the motion graph. During optimisation, we choose the best three candidates in each iteration. We only search in the root-level edges, i.e., the hop mutations, as we favour search speed over smoothness of the animation, although we did not find any noticeable artefacts in our results. We also maintain a hash table of explored solutions to avoid simulating the same edge sequence mutated from different candidates. The search terminates when no better mutation can be found and the optimal solution is the one that has the minimal cost among local minima. For path-following in PFF, we found $d_{step} = 30cm$ and $pow_{follow} = 2$ to be a reasonable step size and penalty to obtain an animation that follows the poly-line closely.

We have designed *two-corner turning* and *object-cluttered space* scenes to evaluate the results of both methods. We performed our experiments on an Intel P9700 2.8 GHz processor on a 64-bit Windows laptop machine with 6 GB memory. For our motion library, we chose ten locomotions from the CMU Motion Capture Database [5] (details available on first author's website). We re-sampled motions to 30 frame-per-second to reduce the size of the library. Our library contains 1091 frames and our motion graph has 9066 root-level edges. The quality of the animation is measured by the transition numbers and costs to determine its smoothness, the location and orientation errors to the goal constraint, and the ground trajectory for travelled distance and required frames. We also compare the required motion-graph search time of both methods.

Two-corner Turning. In this experiment, we design a corridor scene that has two consecutive turns, as seen in Figure 2. The initial and goal constraints are

	Average			Optimal		
	PFF	**EMS**	**Diff**	**PFF**	**EMS**	**Diff**
# of transitions	12.50	11.30	*1.20*	12	9	*3*
Transition costs	3.74	3.88	*-0.14*	3.97	3.23	*0.74*
# of frames	338.40	327.60	*10.80*	335	297	*38*
Location(cm)	10.10	11.16	*-1.06*	9.74	28.41	*-18.67*
Orientation(deg)	2.03	7.2	*-5.17*	0.05	0.16	*-0.11*
Travel dist.(cm)	1058.36	966.41	*91.95*	1075.70	907.89	*167.81*
Search time(sec)	106.20	39.03	*67.17*	70.43	21.34	*49.09*

Table 1. Performance comparison between PFF and EMS in "Two-corner turning". Column *Average* is the mean of 10 trials while *Optimal* is the solution that has the minimal cost among 10 trials. Column *Diff* is calculated by subtracting the result of EMS from PFF in order to show the improvements.

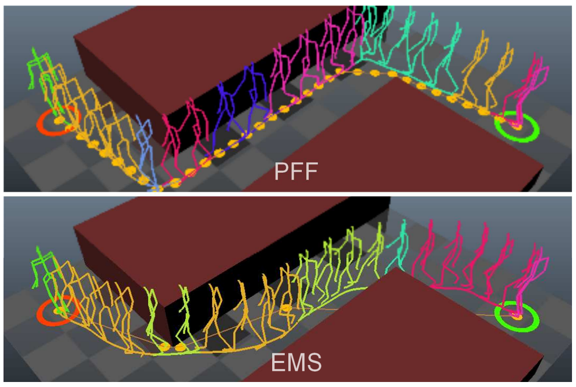

Fig. 2. Two-corner turning using PFF (top) and EMS (bottom).

placed at both ends of the corridor. Since the initial random edge sequence has an impact on the graph search time, we perform ten trials of searches. For each trial, we randomly choose an initial edge sequence for both methods and average their results, as shown in Table 1. For both methods, we select an optimal solution that has the lowest cost among ten trials and compare them in the *Optimal* column. In general, we found EMS can generate animation that has fewer transitions and shorter travel distance with less search time in both *Average* and *Optimal* columns. Although PFF seems to have smaller location and orientation errors, the difference between two methods is actually very small (within $20cm$ and $6°$). Visual improvements can also be seen from the animation in Figure 2, where the result from EMS turns at the corners smoothly while PFF always makes two sharp turns in order to strictly follow the path.

Object-cluttered Space. In the second experiment, we evaluate the performance of both methods when navigating through multiple random obstacles. First, we perform a single trial of motion search for both methods in a grid of

	Average			**Optimal**			**Grid**		
	PFF	**EMS**	**Diff**	**PFF**	**EMS**	**Diff**	**PFF**	**EMS**	**Diff**
# of Transitions	22.7	11.4	*11.3*	18	5	*13*	27	6	*21*
Transition costs	6.5	3.41	*3.08*	5.63	1.60	*4.03*	5.5	1.28	*4.22*
# of frames	490.1	275	*215.1*	394	230	*164*	287	230	*57*
Location(cm)	32.88	30.13	*2.75*	20.59	8.86	*11.72*	24.45	11.67	*12.78*
Orientation(deg)	12.99	5.82	*7.17*	0.48	0.05	*0.44*	0.93	2.86	*-1.93*
Travel dist.(cm)	1481.69	831.87	*649.82*	1211.80	810.11	*401.69*	854.44	807.23	*47.21*
Search time(sec)	740.82	38.13	*702.68*	686.94	3.85	*683.09*	332.3	3.86	*328.44*

Table 2. Performance comparison between PFF and EMS in "Object-cluttered space". Column *Grid* is the result before the boxes are randomly transformed.

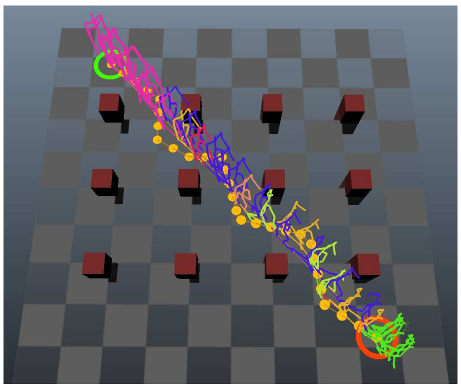

Fig. 3. Search in a grid of boxes. Both methods generate animations that travel via a similar route.

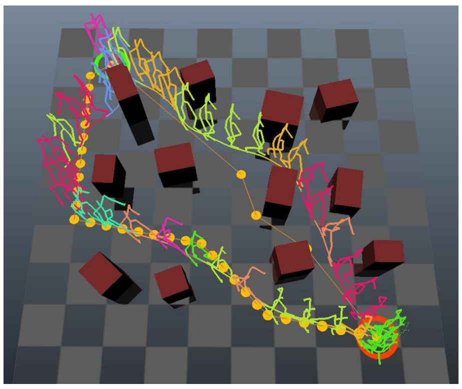

Fig. 4. While PFF can only find an animation travelling through larger gaps, EMS is able to find motions moving around obstacles closely in the object-cluttered scene.

4×3 boxes ($40 \times 40 \times 40cm$) with a distance of $4 \times r_{body}$ between each box (see Figure 3). While PFF requires more transitions to stay inside the corridor, the results of both methods are very similar. We then randomly change the size, orientation, and location of each box for ten trials, and for each random configuration, we compare the the average and optimal results generated from both methods, as shown in Table 2. Note that although this scene is more complicated than the previous, the search time of EMS does not increase substantially. On the contrary, PFF requires much more search time to sample transitions as there are many turns in the paths. While PFF can only navigate through obstacles with an in-between distance larger than $2 \times r_{body}$, EMS is able to generate animations that travel through narrower gaps. Finally, unlike PFF, EMS is not over-constrained by the intermediate constraints even if it fails to find a solution that reaches all waypoints. Figure 4 shows an example where EMS manages to find alternative motions that reach the goal and the result is still a short route. We also include more results of our two experiments in the supplementary video.

7 Conclusion

In this paper, we proposed an environment-aware motion sampling method to generate plausible human animation in a non-trivial virtual environment. The coupling of collision avoidance and motion-graph queries allows us to sample motions that reach the goals and avoid the obstacles in a more human-like way. One key reason for this is that we don't have to conservatively plan to avoid obstacles. The results show that although it is possible to apply a path finding technique to obtain explicit constraints to query a motion graph, the combination of planning and querying allows us to find more feasible solutions with fewer transitions and shorter animations to achieve the same goal.

Although we demonstrate our approach for navigation animations, the technique could be extended to other situations that involve planning motion in cluttered environments. For example, complex manipulations of objects that involve reaching into cluttered space, or planning motions that involve collision with only part of the body such as ducking or raising a hand to avoid collision with street furniture. The performance of our methode could also be improved by scoring the solutions in multiple processes or on the GPU since each simulation is independent.

References

1. O. Arikan and D. A. Forsyth. Interactive motion generation from examples. *ACM Trans. Graph.*, 21:483–490, July 2002.
2. J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. pages 1012 –1017 vol.2, jun. 1991.
3. M. G. Choi, J. Lee, and S. Y. Shin. Planning biped locomotion using motion capture data and probabilistic roadmaps. *ACM Trans. Graph.*, 22(2):182–203, 2003.
4. H. Choset and J. Burdick. Sensor-based exploration: The hierarchical generalized voronoi graph, 2000.
5. CMU Graphics Lab. Carnegie mellon university motion capture database, 2009. [Online; announced April 15 2009].
6. L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM.
7. M. Lau and J. J. Kuffner. Precomputed search trees: planning for interactive goal-driven animation. In *SCA '06: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 299–308, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.
8. J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. *ACM Trans. Graph.*, 21(3):491–500, 2002.
9. Y. Lee, S. J. Lee, and Z. Popović. Compact character controllers. *ACM Trans. Graph.*, 28:169:1–169:8, December 2009.
10. W.-Y. Lo and M. Zwicker. Real-time planning for parameterized human motion. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer*

Animation, SCA '08, pages 29–38, Aire-la-Ville, Switzerland, Switzerland, 2008. Eurographics Association.
11. M. Mizuguchi, J. Buchanan, and T. Calvert. Data driven motion transitions for interactive games, 2001.
12. D. Nieuwenhuisen, A. Kamphuis, and M. H. Overmars. High quality navigation in computer games. *Sci. Comput. Program.*, 67:91–104, June 2007.
13. P. S. A. Reitsma and N. S. Pollard. Evaluating motion graphs for character animation. *ACM Trans. Graph.*, 26(4):18, 2007.
14. A. Safonova and J. K. Hodgins. Construction and optimal search of interpolated motion graphs. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 106, New York, NY, USA, 2007. ACM.
15. A. Treuille, Y. Lee, and Z. Popović. Near-optimal character animation with continuous control. *ACM Trans. Graph.*, 26, July 2007.