Path Planning for Crowds: From Shared Goals to Individual Behaviors

J. Pettre and D. Thalmann

VRlab, EPFL, Switzerland

Abstract

This paper presents a novel approach for path planning in the context of crowds animation. The solution produces several paths joining each user-defined pair of locations in the environment. Pairs are possible initial/goal locations for the virtual characters. The obtained paths diversity enables individual behaviorial diversity, while ensuring the achievement of potentially complex goals. The solution is general, derives from the Probabilistic Roadmap motion planning technique developed in Robotics, and proceeds in two stages. First a dense roadmap is build from the 3D definition of the environment and individual's bounding box, then, given a specific problem, a set of feasible paths is extracted from the roadmap using a Diskstra's algorithm implementation and an edge deletion technique. Resulting paths are plausible, and covers widely the environment given that short paths are found as well as less optimal ones. The method is illustrated and demonstrated all along the paper with a generic example.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Until recently, real-time Virtual Worlds were almost uninhabited and an observer was proposed to see or interact with few virtual characters. Then, appeared solutions for controlling, animating and rendering huge crowds of characters [UT02, LD04, DHOO05]. In this paper, we focus on large-size Virtual Worlds (e.g. virtual cities, buildings, museums,...) populated with numerous characters having individual and complex goals that they may share (e.g. several characters having a same busy destination such as a metro station, a shop, etc...). One can observe that in the real life, people go from locations to others both following different directions (e.g. different sequence of streets in cities) and different paths (e.g. covering the whole sidewalk in a given street). We address the problem of finding such a variety of paths given two locations to join (joined locations pairs are then possible initial/destination locations for any character). The inputs to our problem are the 3D definition of a virtual world and locations coordinates to join (many pairs should be defined for a virtual world). Our goal is to compute automatically a set of diverse solution paths, from the shortest ones (followed by the most hurried characters) to the less optimal ones (for those wanting to stroll).

Many crowd simulation methods rely on particle systems [BG96, HFV00, BMdOB03] or flocking systems [Rey87] to control crowds. Such solutions fit simulation of groups sharing simultaneously common goals. As the behavioral model of each agent is simple (for performance related reasons), flocks or particle systems are supervised to enable achievement of high-level goals. An example of a supervised potential field technique is given in [LMM03]. The solution relies on a 2D grid for pedestrians inter-collision avoidance, and pre-computed paths for achieving higher-level goals. In such approaches, behavioral diversity is obtained thanks to agents interactions in crowded zones. However, an observer may feel (or even see) the paths guiding pedestrians in less crowded zones, and large parts of the environment may stay empty.

Consequently, our contribution is to propose a method for planning motion for crowds of characters (having individual but potentially shared goals), relying on a single and efficient structure allowing achievement of complex locomotion tasks (obstacle avoidance in potentially complex environments), ensuring behaviorial diversity, combinable with level-based solutions for animating characters, and providing exploitable relationships between time, environment geography and characters' situation.

Definitions of the data structure and of the basic tools used by our solution is provided in Section 2.1. Then, the two-stage process for computing paths is described: first, the method for constructing roadmaps (Section 2.2), and second, the path extraction technique (Section 2.3). An optional stage consists in filtering the obtained paths (Section 2.4). Examples of obtained paths are displayed and commented in Section 3. The approach specificities are exposed and discussed in Section 4, before Conclusion.

2. Roadmap Construction and Paths Synthesis

Our path generation method relies on the Probabilistic Roadmap (PRM) [KSLO96], a general principle for motion planning, reusing the same tools and data structure for quite opposite goals: whereas a classical PRM tends to explore efficiently a high-dimensional configuration space to answer a submitted problem with a single solution path, we explore redundantly a low-dimensional workspace (an horizontal plane) to obtain multiple solution paths. Like for PRM, both roadmap and paths depend only on the environment and character-related parameters (bounding-box) and as a result, all the method presented below consists in precomputations and their results should be stored/loaded for future reutilizations.

2.1. Definition

A roadmap is a *graph* where *nodes* are collision-free locations for the characters' bounding box and *edges* are collision-free position evolutions of the bounding box between two nodes (i.e. two locations). A *solution path* is a set of contiguous edges connecting two given nodes. A cost value is associated to each edge in order to enable *shortest paths* searches. In our case, the cost function is the Euclidean distance between two directly connected nodes. Then, roadmap construction invokes a set of basic tools:

- a *collision checker* determining whether a position (node) or a position evolution (edge) is collision free or not,
- a *node generator* sampling the environment randomly in order to insert new nodes in the roadmap graph,
- a *steering method* computing the position evolution of the bounding box between two locations (nodes),
- an *edge generator* elaborating a strategy for selecting and attempting to connect pairs of nodes,
- a *path finder* to solve path search problems.

As mentioned above, our goal is to compute a **set of different solution paths** and consequently, built roadmaps should be *dense* and contain *redundant* connections between nodes. To fulfill these requirements, we developed specific implementations of the tools cited above.

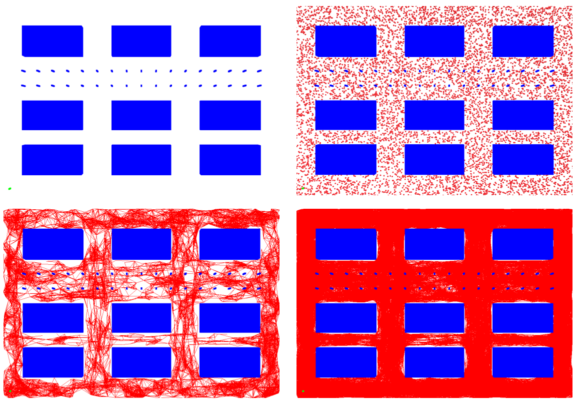

top-left: considered environment. *top-right*: nodes generator result. *bottom-left*: edge generator result (intermediate). *bottom-right*: edge generator result.

Figure 1: *Roadmap Construction Stages*

2.2. Roadmap Building Method

The method for building roadmaps is schematized in Figure 1. First Image (top-left) displays the considered 3D-environment seen from above, where blue boxes are obstacles (and white zones are walkable): this environment can be compared to a virtual city where the rectangular boxes are buildings ($50 \times 30 \times 10$ units), the smaller square ones are trunks ($1 \times 1 \times 10$ units) while characters' bounding-box is $1 \times 1 \times 2$ units. A user-defined number (or density) of nodes are created invoking the node generator (top-right Image). Edges are then inserted into the graph invoking the edge generator (bottom-left and right Images). Our strategy for creating edges (providing the best compromise found between computation-time and graph quality) consists in attempting to connect nodes within a user-defined distance. To improve the required collision checks at this stage, we use an approximate test where 4 rays draw a corridor between two nodes. Such a test fits many classes of environments where size proportions between obstacles and characters respect given criteria and furthermore, introducing few colliding edges is not harmful as they can be removed at a following stage (where required collision checks are less numerous: see Section 2.4). Finally, using the same technique, we introduce user-defined nodes into the roadmap corresponding to the individuals' possible goals or initial locations. Next Section describes how paths are generated between the initial/goal locations.

2.3. Paths Search

Paths search is necessary for each user-defined pair of initial/goal locations previously inserted in the graph. The objective is to compute a set of paths captured by the roadmap joining these locations, ranging from the shortest to the less optimal ones. We designed an iterative method to accomplish this task. At each iteration, a Dijkstra's algorithm computes the shortest path contained into the graph and next, a random

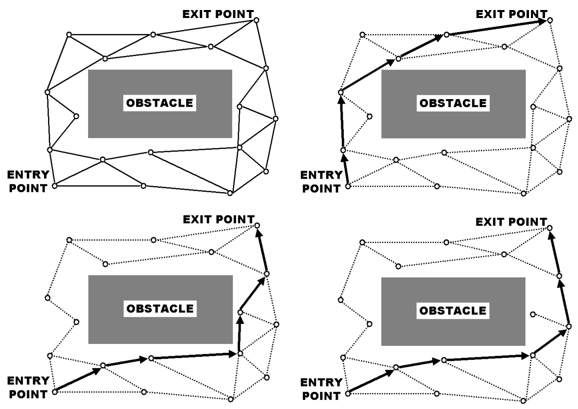


Figure 2: *Path Search Iterative Method*

group of contiguous edges composing the solution path is removed from the graph (a random number of edges, at a random place in the solution path). Dijkstra's algorithm and edge removal stage is reiterated until the considered pair of locations is disconnected (or a criteria is fulfilled, such as a number of paths found or a relative path length overpassed).

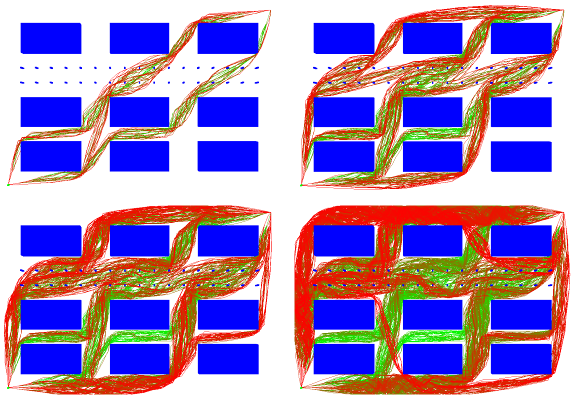
Figure 2 schematizes the paths search method. First, a complete roadmap is displayed (top-left Image), in which the shortest path between an entry and an exit point is found (top-right Image) composed of 5 edges. Randomly, 3 edges are removed (second, third and fourth one), and a shortest path computed again (bottom-left Image). Edges are removed from this new path, and the Dijkstra's algorithm result on this reduced roadmap is displayed on the last image of the figure.

2.4. Path Set Refinements

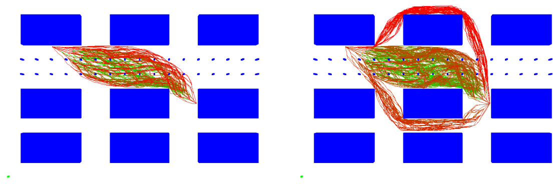
Among all the generated paths, some may present undesired characteristics. Indeed, we used approximate collision checks, and residual collisions may appear. Also, paths may contain u-turns or high-frequency orientation changes. Path are numerous enough to filter and simply remove the undesired ones. However, corrections are feasible especially when u-turns occur, where shortcuts can be easily searched (trying to skip solution path edges) and replace parts of a solution path.

3. Results

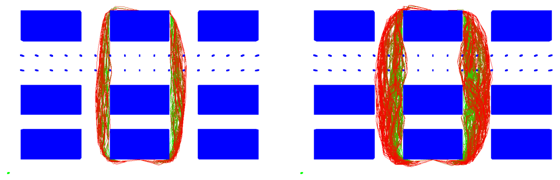
Figure 3 illustrates our first results on 3 different examples (3 different init/goal pairs). For each example, a fixed number of paths is displayed in the chronological order of their computation, easing the figure readability and illustrating that paths are found in length order. We pushed the search at the limit on the first example, showing that found paths cover entirely the environment. The same roadmap was used for all examples - 10 000 nodes and 238 000 edges, computed in a minute on a 1.5GHz pentium, 1GB ram. With such a huge roadmap, each path is computed in half a second. Memory usage is low: 2.8MB for the roadmap, 1.2MB for the 3200 paths of Example 1. Technically, we used the *coldet library (http://photoneffect.com/coldet/)* for collision checks and the *boost library (http://www.boost.org/)* implementation of the Disjkstra's algorithm. Complexity for each stage is $O(N)$ for the node generator, $O(N^2)$ for the edge generator, and $O(PElogN)$ for path synthesis assuming that all nodes are accessible one from each other (common case), where N is the number of nodes, E the number of edges and P the number of paths.



Example 1: successive display of the 200, 800, 1600 and 3200 first paths found (from green for the shortest to red for the longest



Example 2: 500 and 1000 paths



Example 3: 500 and 1000 paths

Figure 3: *Resulting Paths*

4. Discussion and Future Work

First results illustrating our directions for controlling individual motions in the crowd simulation context are promising. Although this is still work in progress, we identified how a roadmap-based structure could benefit a crowd simulation architecture. Next paragraphs describe both these advantages through a discussion and our future work directions.

The roadmap structure creates true relationships between the environment topology/geography and individuals. Any location can be linked to nearby nodes or nearby edges and next, passing-by paths and passing-by characters are efficiently deductable. These relationships are crucial for enhancing characters' capabilities and enabling **high-level directives** ranging from the **individual-level** to the **crowd-level** such as for example: "cross zone A", "avoid zone A", "go from A to B via zone C", "go fast between A and B", "wander between A and B" etc. To increase crowds autonomy, general strategies can be elaborated for assigning a path to each character: based on a probability distribution (e.g. according to the relative path length), based on individuals' characteristics (hurried businessmen, strolling tourists, etc.), based on characters' motivations, or on reactions to events (e.g. zones are forbidden if becoming dangerous or attractive if a cultural event occur nearby).

The path structure (sequence of nodes) fits the level-based design of other architecture components. Indeed, culling characters given their distance or visibility from the user's point of view can be eased as their motion is time-predictable. Also, the way the characters move from a node to another - i.e. the steering method - can be level-designed. For example, characters should follow smooth trajectories ([PLS03, Bou05]) when close to the camera, they should follow linear trajectories (expressible analytically) between two nodes when far, their location should be estimated at a given node for a given time when invisible.

Yet, we do not consider the local agent/agent collision avoidance problem. Based on a roadmap, we can envisage three sorts of method for addressing this problem. First, paths can be assigned to agents according to time scale in such a way that collision-free motions are ensured (this would need new pre-computations). Such solutions faces complexity-related problems but fits applications were the user's point of view is frequently and widely changed. Second, priority-rules for path execution can be used to solve local conflicts (guaranteing dead-locks avoidance is then a crucial issue). Third, a potential fields based technique from literature can be applied. These two last approaches fit the collision avoidance in the vicinity of the user's point of view.

Another required improvement concerns initial/goal locations, actually modelled as nodes. These locations should be modelled as areas. A first solution consists in masking initial/goal nodes into buildings. At their entry/exit, paths should provide sufficient diversity. Another solution consists in removing firsts and lasts edges of the solutions paths. Characters then follow an interactively computed path to go from initial/goal areas to the new paths extremities.

5. Conclusion

We presented a novel approach to address path planning problems for crowds. The method is easily implementable and roadmap-based motion planning solutions already demonstrated their efficiency on many class of problems. The methods fits the crowd context on many points, commented extensively in the paper and to be demonstrated in future developments and experiments.

Acknowledgements

This work is supported by the EU FP6-2004-Mobility-5 Programme (Marie Curie Intra-European Fellowship).

References

[BG96] BOUVIER E., GUILLOTEAU P.: Crowd simulation in immersive space management. In *Proc. of the Eurographics workshop on Virtual environments and scientific visualization '96* (1996), pp. 104–110.

[BMdOB03] BRAUN A., MUSSE S., DE OLIVEIRA L., BODMANN B.: Modeling individual behaviors in crowd simulation. In *Proc. of Computer Animation and Social Agents (CASA)* (2003).

[Bou05] BOULIC R.: Proactive steering toward oriented targets. In *Eurographics Short Presentations* (2005).

[DHOO05] DOBBYN S., HAMILL J., O'CONOR K., O'SULLIVAN C.: Geopostors: A real-time geometry/impostor crowd rendering system. In *Proc. of the ACM SIGGRAPH 2005 Symposium on Interactive 3D Graphics and Games* (2005).

[HFV00] HELBING D., FARKAS I., VICSEK T.: Simulating dynamical features of escape panic. *Nature 407* (2000), 487–490.

[KSLO96] KAVRAKI L., SVESTKA P., LATOMBE J., OVERMARS M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Tr. on Robotics and Automation 12*, 4 (1996), 566–580.

[LD04] LAMARCHE F., DONIKIAN S.: Crowds of virtual humans : a new approach for real time navigation in complex and structured environments. *Computer Graphics Forum* (2004).

[LMM03] LOSCOS C., MARCHAL D., MEYER A.: Intuitive crowd behaviour in dense urban environments using local laws. In *Theory and Practice of Computer Graphics* (2003).

[PLS03] PETTRÉ J., LAUMOND J., SIMÉON T.: A 2-stages locomotion planner for digital actors. In *Proc. of the ACM SIGGRAPH Symposium on Computer Animation (SCA'03)* (2003).

[Rey87] REYNOLDS C.: Flocks, herds, and schools: A distributed behavioral model. *ACM Computer Graphics 21*, 4 (jul 1987), 25–34.

[UT02] ULICNY B., THALMANN D.: Towards interactive real-time crowd behavior simulation. *Computer Graphics Forum 21*, 4 (dec 2002), 767–775.