Integrating Local Collision Avoidance with Shortest Path Maps

Ritesh Sharma, Renato Farias and Marcelo Kallmann

Computer Graphics Lab, University of California Merced, USA

Abstract

The effective integration of local collision avoidance with global path planning becomes a necessity when multi-agent systems need to be simulated in complex cluttered environments. This work presents our first results exploring the new approach of integrating Shortest Path Maps (SPMs) with local collision avoidance in order to provide optimal paths for agents to navigate around obstacles toward their goal locations. Our GPU-based SPM implementation is available.

CCS Concepts

• ***Computing methodologies*** → ***Collision detection; Multi-agent planning;***

1. Introduction and Related Work

Local collision avoidance is an integral part of making multiple agents able to interact with each other in a fluid and realistic way. However, being restricted to only local knowledge of the environment means that agents may get trapped in obstacles. In such cases global path planning techniques need to be integrated. Popular approaches for global path planning are mostly based on graph search, where the graph is derived from a representation of the environment ranging from navigation meshes to grid representations or roadmap graphs. In order to favor simplicity and speed of computation, these approaches are mostly heuristic in nature and sacrifice global optimality in the Euclidean sense.

In this work we show how globally-optimal paths can be efficiently obtained from Shortest Path Maps (SPMs) and easily integrated with local collision avoidance behavior. A SPM [Mit91, Mit93] encodes all globally-optimal shortest paths from defined goal locations to every point in an environment. While SPMs are well-known in the computational geometry area, the complexity in computing them has limited their adoption in applications. Based on our previous work [CK14, FK19] relying on GPU rasterization procedures we are able to efficiently produce a SPM representation that integrates well with multi-agent simulations. Our SPMs can in addition handle polygonal lines as goal locations. A single SPM is therefore able to optimally route multiple agents around obstacles and towards their closest points in their closest goal polygonal line.

2. Shortest Path Maps

SPMs are structures computed with respect to "sources", which in our case are arbitrary polygonal lines representing goal locations in our scenarios. A SPM computed for a particular planar environment encodes a globally-optimal Euclidean Shortest Path for *all* points in that environment. See Figure 1 for an example.

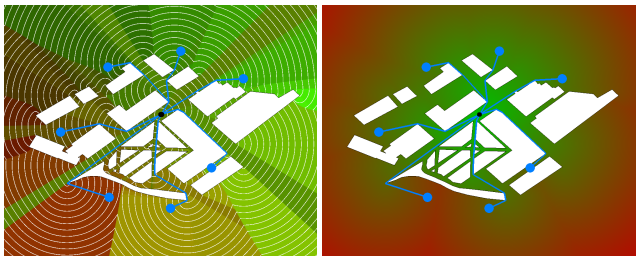

Figure 1: *Example SPM. Blue lines represent shortest paths from each blue agent to a source point in the center of the environment. SPM regions (left) share a same parent vertex towards the source. A distance field is also naturally represented (right).*

Our approach computes the SPM entirely in a GPU buffer. It is based on propagating costs with rasterized clipped cones using OpenGL's rendering pipeline and custom shaders [FK19]. The environment is partitioned into the regions sharing the same sequence of vertices on the shortest path to the closest source region. Since every pixel in the buffer stores its parent point, agents can retrieve their next heading direction in constant time. This constant time buffer mapping is an advantage over CPU implementations, which require a point localization procedure to determine which region contains the query point. Furthermore, the entire shortest path can be constructed in linear time with respect to the number of vertices in the shortest path. Our SPM implementation has also been successfully applied to solve continuous max flows [FK18].

3. Integration with Collision Avoidance

We have integrated our SPM implementation with the well-known Reciprocal Velocity Obstacle (RVO) [vdBLM08] approach for local collision avoidance. At the beginning of every time step, each

DOI: 10.2312/egp.20201037

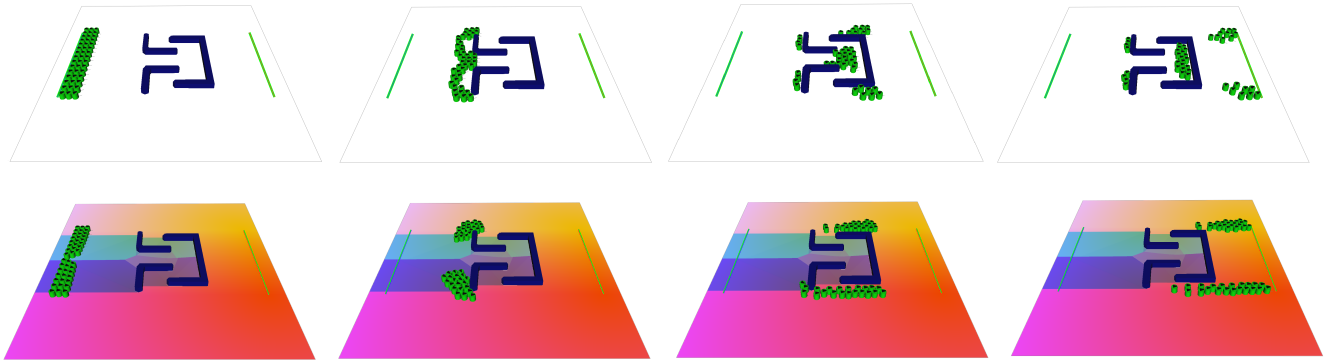

Figure 2: *Agents are trapped when relying only on collision avoidance (top) but not when extracting directions from the SPM (bottom).*

agent makes a query from its current position to the SPM in order to retrieve in constant time the preferred velocity vector, which is the vector leading to the next vertex along its shortest path to the goal. Once this vector is retrieved, a collision-free velocity vector is determined from the RVO method. First, the set of velocity regions using the preferred velocity vector for each agent in the neighborhood of the current agent is computed. Then, to account for their relative motion, the velocity obstacles for every pair of agents is computed. Finally the boundary of the union of all velocity obstacles is determined and each agent is routed to its nearest point on the boundary. This nearest point is the safest velocity vector that each agent should take.

Given that the initial preferred velocity for each agent is extracted from the SPM at every time step, the result naturally balances following the shortest path while avoiding agents nearby which are as well following similar trajectories when going around obstacles.

4. Results

To evaluate our approach we have implemented two different scenarios: one with a single group of agents and the other with multiple groups of agents. In both scenarios the groups of agents have to navigate from a source region to a goal region. Figure 2 reveals that despite using an effective collision avoidance technique, the agents cannot navigate around the obstacles and are easily stuck in traps. This does not happen when the agents are guided by the velocity vector extracted from the SPM of the environment. When multiple groups of agents are defined, as shown in Figure 3, each group relies on a SPM specific to its goal region. In each case, when agents rely on their SPM they do not get stuck in the "pockets" of the environment.

In summary our results show that SPMs are efficient and effective for multi-agent and crowd simulation in cluttered environments for situations where groups of agents share the same goals.

Code Availability

Our SPM implementation is available as the libsigspm module of the SIG toolkit hosted at https://bitbucket.org/mkallmann/sig/.

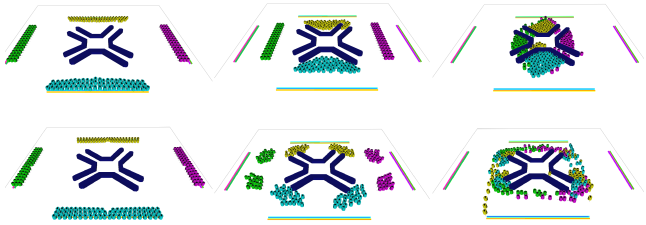

Figure 3: *Top: Four groups of agents navigating toward the center area using RVO-based local behavior without any global path planning. Bottom: SPMs are applied to provide preferred velocity vectors aligned with shortest paths in order to guide the agents.*

Acknowledgements

This research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-17-1-0463. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government.

References

[CK14] CAMPORESI C., KALLMANN M.: Computing shortest path maps with GPU shaders. In *Proceedings of Motion in Games (MIG)* (2014), ACM, pp. 97–102. 1

[FK18] FARIAS R., KALLMANN M.: GPU-based max flow maps in the plane. In *Proceedings of Robotics: Science and Systems (RSS)* (2018). 1

[FK19] FARIAS R., KALLMANN M.: Optimal path maps on the GPU. *IEEE TVCG* (2019). 1

[Mit91] MITCHELL J. S. B.: A New Algorithm for Shortest Paths Among Obstacles in the Plane. *Annals of Mathematics and Artificial Intelligence 3*, 1 (1991), 83–105. 1

[Mit93] MITCHELL J. S. B.: Shortest Paths Among Obstacles in the Plane. In *Proceedings of SOCG* (New York, NY, USA, 1993), SCG '93, ACM, pp. 308–317. 1

[vdBLM08] VAN DEN BERG J., LIN M. C., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE ICRA* (2008), IEEE, pp. 1928–1935. 1