Fast path planning under polygonal obstacle constraints

Salles Viana Gomes Magalhães
Rensselaer Polytechnic Inst.
Troy, NY, USA
vianas2@rpi.edu

Marcus Vinicius Alvim Andrade
Univ. Federal de Viçosa
Viçosa, MG, Brazil
marcus@ufv.br

W. Randolph Franklin
Rensselaer Polytechnic Inst.
Troy, NY, USA
mail@wrfranklin.org

Wenli Li
Rensselaer Polytechnic Inst.
Troy, NY, USA
liw9@rpi.edu

ABSTRACT

This paper presents UPLAN, an efficient algorithm for path planning on road networks with polygonal constraints. UPLAN is based on the A* algorithm and it uses a uniform grid for efficiently indexing the obstacles. As shown in the experiments, UPLAN can quickly compute shortest paths on maps and, thanks to the uniform grid, it is can efficiently process very many polygonal obstacles. UPLAN was one of the top 3 finishers in the ACM GISCUP 2015 competition.

Categories and Subject Descriptors

F.2.2 [**Nonnumerical Algorithms and Problems**]: Geometrical problems and computations

General Terms

Algorithms, Experimentation, Performance

Keywords

Path planning, obstacle-avoidance, uniform grid

1. INTRODUCTION

Path planning, an important application in GIS, mobile systems, and robotics, has been studied for a long time. However the increasing volume of data available poses a new challenge that requires new strategies. For example, the NASA's *Earth Observatory System* satellites collect about 1 TB of spatial data every day. There is also a huge volume of data collected by GPS-enabled mobile devices.

This paper presents an efficient method for path planning that, given a road network and a set of polygons representing obstacles, determines the shortest path connecting two nodes (the source and the destination) on the road network, while avoiding all obstacles.

2. RELATED WORK

This is a well-established research area with applications in several domains in two or three dimensions. Many works [5, 6, 8, 12] have focused on finding the shortest path on regions represented by grid points (for example, terrains). In general, the algorithms process a 2D matrix of cells where some cells are marked as obstacles that cannot be visited. This problem is a common application in the robotics, GIS, and virtual environment communities; the goal is to find the path that avoids obstacles and minimizes the total number of cells visited or the safest path, that is, the path that keeps as far away from obstacles as possible. In [5, 6, 12] a sophisticated cost function is implemented for a path over terrrain that first minimizes distance traveled inside forbidden regions, second minimizes distance traveled uphill, and third minimizes horizontal distance traveled. One application assumes that the forbidden regions are the viewsheds of observers, and that a traveler wants to remain hidden as much as possible.

Many other algorithms [1, 11, 14] perform path planning on road maps represented as graphs. Again, the obstacles are represented as polygons and the goal is to determine "the best" path, that is, the shortest or the fastest, avoiding the obstacles. Most of these algorithms are based on a variation of the A* search [10] or Dijkstra's algorithm [4].

The A* algorithm is a general-purpose guided search algorithm, which can be adapted to efficiently solve path-planning problems. This serves as the basis for UPLAN.

3. UNIFORM GRID

The uniform grid is a simple indexing data structure that superimposes a grid over a polygon, a map or any set of segments, and then identifies the grid cells intersected by each segment. Then, given a query segment, a superset of other segments that are close to (or intersect) the query can be efficiently determined. If the grid size is chosen to make the expected number of segments per cell to be constant, then the expected query time is also constant. Even if the number of segments per cell varies, perhaps under a Poisson distribution, because the segments are chosen from an independent and identical probability distribution, the algorithm is still efficient. The uniform grid has been used in many applications including map overlay [3, 9], map generalization [7], and segment intersection [2].

For the GISCUP 2015 [13] problem solution, we used a uniform grid for obstacle indexing. During the path compu-

23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL) 2015 Seattle, WA, USA

tation, each road explored by the algorithm is checked for intersection with any obstacle. More precisely, a grid with a given resolution is created, and the line segments representing the obstacles are inserted into the intersecting grid cells. Then, when the shortest path computation algorithm processes an edge (road) on the graph, it selects the grid cells intersected by the polyline l representing this road and verifies if any line segment in these cells intersects l. If an intersection is detected, the edge is ignored.

Thus, the uniform grid is used to quickly identify those map roads that can't be used because they intersect an obstacle.

4. PATH PLANNING

First, we create a directed graph G from the shapefile representing the road network. The vertices in G represent the road intersections. The edges are created such that a one-way road connecting the intersection u to v is represented by a directed edge (u,v) and a two-way road is represented by two edges (u,v) and (v,u). Since graphs representing roads are usually sparse, G is represented by an adjacency list.

Next, we create a uniform grid to index the obstacle edges.

Third, an implementation of A* [10] is used to find the best path between the source node s and the target node t; see Algorithm 1. This algorithm computes the cost of the shortest path between s and t; the actual path can be computed by keeping track of the parent of each node during the search procedure.

The basic idea is to explore the graph using a priority queue such that, in each step, the non-explored vertex that is "closest" to the source node s is processed. For performance purposes, the A* algorithm uses a heuristic function *heuristic(v)* that returns a lower bound for the cost between the node v and the target node t. When a node v is inserted into the priority queue, the priority of this node is incremented with *heuristic(v)* and, therefore, this heuristic is used to guide the search such that nodes that have more potential to be in the shortest path are processed first.

A* gives the best performance when the heuristic returns high estimates for the lower bound (that is, when the estimates for the cost are similar to the real cost of the paths) [10]. If the heuristic function returns small estimates (for example, numbers close to zero), even though the algorithm will always compute the shortest path, it may need to explore a larger portion of the graph (behaving similarly to Dijkstra's algorithm). If the heuristic returns an estimate larger than the actual cost, the resulting path is not guaranteed to be the shortest one.

The heuristic for computing the path based on *distance* was defined to be the length of a straight path between the current node and the target node. Since the time needed to traverse the paths depends on the speed and length of the roads, the heuristic for computing the shortest path based on *time* is defined to be the ratio of the length of the straight path between the current node and the target node and the highest speed of the roads on the maps. These two functions will always return lower bounds for the cost of the paths, and, thus, the A* will always find the best paths.

In order to avoid obstacles, Algorithm 1 never explores an edge whose corresponding road (polyline) intersects an edge of the obstacle polygons. We could have preprocessed G to remove edges intersecting obstacles. However, we decided not to do this because the A* algorithm is a guided search that does not explore the whole graph and, therefore, not all edges in G need to be tested for intersection. This strategy improves the performance of UPLAN mainly in situations where the shortest path is small when compared to the size of the graph and, therefore, only a few edges are explored.

Finally, since neither the index nor the road graph changes during the path planning, paths between different pairs of nodes or using different criteria for the cost may be easily computed in parallel. Since in the GISCUP competition, the algorithm was required to compute both the path based on traveling time and the one based on distance, we computed these two paths in parallel (using OpenMP).

Algorithm 1 Path planning algorithm.

```
1: G: graph representing the input road network
2: s: start node
3: t: end node.
4: heuristic(v): heuristic function
5: cost[v]: cost of the shortest path from s to v.
6: PQ ← priority queue of vertices sorted in ascending order
7: cost[v] ← 0 if v = s or infinity otherwise
8: while PQ is not empty do
9:     v ← front of PQ
10:    if v = t then
11:        Return the priority of v
12:    end if
13:    Label v as explored
14:    cost[v] ← v's priority
15:    for each edge e = (v, u) in G do
16:        if e crosses an obstacle or u was explored then
17:            Continue the for loop
18:        end if
19:        u's priority ← cost[v] + |e| + heuristic(u)
20:        if e isn't in PQ then
21:            Insert u into PQ
22:        else
23:            Update u's priority in PQ
24:        end if
25:    end for
26: end while
```

5. EXPERIMENTS

UPLAN was tested on a computer with a dual E5-2687 Intel Xeon CPU and 128 GB of RAM. Since we wanted to understand the performance of each step of UPLAN, the experiments used a single-threaded version of UPLAN. However as mentioned above, UPLAN can easily compute many paths in parallel.

The experiments were performed on the sample dataset road map provided by the GISCUP 2015 challenge [13]. This map contains 42407 nodes and 96849 polylines, containing 302785 edges. To evaluate UPLAN with varying numbers of obstacles, we used four different polygonal obstacle datasets. Dataset 1 is the polygonal map provided by the GISCUP; it contains 4 edges representing a quadrilateral. Datasets

2 and 3 were created manually and contain, respectively, 15783 edges (91 polygons) and 1414861 edges (2658 polygons). Dataset 4 contains the same number of edges as Dataset 3, but the obstacles were positioned further from the main roads used in the paths. This was done to evaluate the performance of the algorithm on maps with a large number of constraints that are far from the roads (since ideally such distant constraints should not affect the performance of the algorithm).

All experiments were performed by computing the shortest path between pairs of nodes in three sets containing 20 pairs of nodes each. The *Close* and *Far* sets were generated by randomly creating 3000 pairs of edges and choosing the 20 pairs whose shortest path had, respectively, the smallest and largest lengths. Similarly, the *Median* set was created by choosing from the 3000 pairs of edges the 20 pairs whose shortest paths are closest to the median shortest distance.

UPLAN was configured to choose the uniform grid resolution such that the average number of edges per grid cell is 3. (In previous applications, the precise grid size has not been important.) Preliminary experiments show that this configuration leads to a suitable performance.

Table 1 presents the average running time, and several other statistics, of each step of UPLAN on the 4 different datasets considering the three sets of pairs of nodes.

As expected, the time to create the uniform grid depends on the number of edges in the obstacle polygons. For Datasets 3 and 4, most of the processing time was spent creating the grid. However, in these two datasets the obstacles were artificially created with many edges (5 times more edges than the number of line segments in the input road map) in order to stress-test the index. Also, the index is usually created once and then reused for many queries.

The most important factor determining the time to compute the shortest paths is the distance between the start and end nodes, and so the number of nodes that need to be explored. As expected, queries with start and end nodes that are far from each other (see node set *Far*) take much more time (and need to visit more vertices) than queries with close nodes.

For the same sets of start and end nodes, the time for computing the shortest paths varies by no more than a factor of 2, suggesting that, thanks to the uniform grid indexing, even large sets of obstacles have little influence on the performance.

It is interesting to observe the performance of the path planning in Dataset 4: this dataset was created to test the performance of UPLAN when there are many obstacles that are distant from the shortest paths. Here, the performance of UPLAN was similar to its performance on Dataset 1 where the number of obstacles was very small. This suggests that, if the obstacles are not close to the edges explored by the algorithm, even many obstacles do not significantly slow things down.

We also tested the effect of disabling the heuristic. When enabled, the time for computing the shortest path based on distance is usually smaller than or equal to the time for computing the path based on shortest traveling time. This happens because the heuristic function for the path based on distance presents a better lower bound than the heuristic for the path based on time (this latter heuristic depends on the range of speeds on the map). This difference is larger for the *Median* set of pairs of nodes.

If the heuristic function is disabled, the time for computing the paths based on shortest distance increases, becoming more similar to the time for computing the path based on shortest time. Considering the path based on time, even though the average number of vertices explored increases when the heuristic function is disabled, this increase is not enough to make difference in the processing time.

These results open possibilities for future work: evaluate the importance of the heuristic function on larger road networks, and explore different heuristic functions. If processing time is a critical factor, one possible way to improve the performance of computing the shortest time path is to use a heuristic based on the average speed on the roads (instead of the maximum speed). However, even though this may improve the performance (since the estimates obtained by the heuristic will be larger), the resulting path will be only an approximate shortest path since the heuristic will not be guaranteed to always return a lower bound.

6. CONCLUSIONS

We have presented UPLAN, an efficient algorithm for path planning on maps represented by road networks with polygonal obstacle constraints. UPLAN uses a uniform grid for indexing the obstacles and, as showed in the experiments, it can quickly compute shortest paths even when the number of obstacles is large. In fact, UPLAN's performance was affected by a maximum factor of 2 even if a set of obstacles with more than one million edges is used (5 times the number of line segments representing the roads).

In the future, we intend to explore other strategies for defining the heuristic function used for planning paths based on shortest traveling time. Allowing the path to cross an obstacle, but at an increased cost, also looks interesting.

This research was partially supported by NSF grant IIS-1117277, by CAPES (Ciencia sem Fronteiras) and FAPEMIG.

7. REFERENCES

[1] I. Abraham, D. Delling, A. V. Goldberg, and R. F. Werneck. A hub-based labeling algorithm for shortest paths in road networks. In *Proceedings of the 10th International Conference on Experimental Algorithms*, SEA'11, pages 230–241, Berlin, Heidelberg, 2011. Springer-Verlag.

[2] V. Akman, W. R. Franklin, M. Kankanhalli, and C. Narayanaswami. Geometric computing and the uniform grid data technique. *Comput. Aided Design*, 21(7):410–420, 1989.

[3] S. Audet, C. Albertsson, M. Murase, and A. Asahara. Robust and efficient polygon overlay on parallel stream processors. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 304–313, New York, NY, USA, 2013. ACM.

Table 1: Running-time (in milliseconds) for each step of UPLAN considering the computation of the shortest path based on travel time and of the shortest path based on distance. Column *Time grid* represents the time for initializing the uniform grid. Columns *Nodes visit.* and *#Tests* represent, respectively, the number of nodes that were visited by the algorithm during the shortest path computation and the number of intersection tests performed to filter roads traversing obstacles.

	Node	Osbatcles		Grid	Grid init.	Path based on distance			Path based on travel time		
	Set	Dataset	# Edges	size	Time	Time	#Vert.visit.	#Tests	Time	#Vert.visit.	#Tests
With heuristic	Close	1	4	1	0	0.1	48	0	0.1	127	0
		2	15784	72	1	0.1	48	1	0.1	127	1
		3	1414861	686	59	0.1	45	1	0.1	123	2
		4	1414861	686	58	0.1	48	0	0.1	127	0
	Median	1	4	1	0	0.9	3799	0	2.0	10149	11
		2	15784	72	1	1.8	4157	436	3.6	10303	966
		3	1414861	686	59	2.0	4265	411	3.9	10723	768
		4	1414861	686	58	1.1	3732	7	2.3	9899	21
	Far	1	4	1	0	4.2	21642	36	5.3	28752	38
		2	15784	72	1	7.7	22566	2201	8.8	28467	2697
		3	1414861	686	59	8.5	21581	1920	8.4	26833	2136
		4	1414861	686	57	4.1	21640	22	4.9	28748	34
Without heuristic	Close	1	4	1	0	0.1	229	0	0.1	227	0
		2	15784	72	1	0.1	229	2	0.1	227	2
		3	1414861	686	59	0.2	224	4	0.1	222	3
		4	1414861	686	57	0.2	229	0	0.1	227	0
	Median	1	4	1	0	2.4	14142	20	2.3	14569	19
		2	15784	72	1	3.5	14145	1251	3.6	14623	1282
		3	1414861	686	58	3.9	13696	1003	4.0	14468	1057
		4	1414861	686	57	2.4	13731	30	2.3	14200	31
	Far	1	4	1	0	4.7	30833	41	4.9	30671	41
		2	15784	72	1	7.6	30735	2803	6.9	30509	2754
		3	1414861	686	71	9.4	28148	2190	9.4	27999	2173
		4	1414861	686	57	4.8	30772	101	4.8	30598	104

[4] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[5] W. R. Franklin, M. Inanc, Z. Xie, D. M. Tracy, B. Cutler, and M. V. A. Andrade. Smugglers and border guards: the geostar project at RPI. In *15th ACM International Symposium on Geographic Information Systems, ACM-GIS 2007, November 7-9, 2007, Seattle, Washington, USA, Proceedings*, page 30, 2007.

[6] W. R. Franklin, D. M. Tracy, M. Andrade, J. Muckell, M. Inanc, Z. Xie, and B. Cutler. Slope accuracy and path planning on compressed terrain. In *Symposium on Spatial Data Handling*, Montpellier FR, June 2008.

[7] M. G. Gruppi, S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and W. Li. An efficient and topologically correct map generalization heuristic. In *ICEIS 2015 - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 1, Barcelona, Spain, 27-30 April, 2015*, pages 516–525, 2015.

[8] N. Lebeck, T. Mølhave, and P. K. Agarwal. Computing highly occluded paths using a sparse network. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Dallas/Fort Worth, TX, USA, November 4-7, 2014*, pages 3–12, 2014.

[9] S. V. G. Magalhães, M. V. A. Andrade, W. R. Franklin, and W. Li. Fast exact parallel map overlay using a two-level uniform grid. In *Proceedings of the 4rd ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, BigSpatial '15, New York, NY, USA, 2015. ACM.

[10] N. J. N. P. E. Hart and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[11] O. Salzman, D. Shaharabani, P. K. Agarwal, and D. Halperin. Sparsification of motion-planning roadmaps by edge contraction. *I. J. Robotic Res.*, 33(14):1711–1725, 2014.

[12] D. M. Tracy, W. R. Franklin, B. Cutler, F. T. Luk, and M. Andrade. Path planning on a compressed terrain. In *Proceedings of the 16th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '08, pages 56:1–56:4, New York, NY, USA, 2008. ACM.

[13] R. R. Vatsavai, X. Chen, and S. Ravada. GISCUP - ACM SIGSPATIAL CUP 2015. http://research.csc.ncsu.edu/stac/GISCUP2015/index.php (accessed on Oct–2015).

[14] L. Wu, X. Xiao, D. Deng, G. Cong, A. D. Zhu, and S. Zhou. Shortest path and distance queries on road networks: An experimental evaluation. *Proc. VLDB Endow.*, 5(5):406–417, Jan. 2012.