In *Proceedings of the Workshop on Reasoning, Representation, and Learning in Computer Games*, International Joint Conference on Artificial Intelligence (IJCAI), Edinburgh, Scotland, July 31, 2005, 49-54.

Path Planning in Triangulations

Marcelo Kallmann
USC Institute for Creative Technologies
13274 Fiji Way
Marina del Rey CA 90292
kallmann@ict.usc.edu

Abstract

This paper presents in detail how the techniques described in my previous work [Kallmann *et al.*, 2003] can be used for efficiently computing collision-free paths in a triangulated planar environment.

The method is based on a dynamic Constrained Delaunay Triangulation (CDT) where constraints are the obstacles in the planar environment. The main advantage of relying on a triangulated domain is that the size of the adjacency graph used for searching paths is usually much smaller than in grid-based search methods. As a result much more efficient planning can be achieved.

1 Introduction and Related Work

The Delaunay Triangulation and its constrained version [Preparata *et al.*, 1985] [Anglada, 1997] [Floriani *et al.*, 1992] are important tools for representing planar domains and have been used in several applications.

This work focuses on using the Constrained Delaunay Triangulation (CDT) as the main data structure to represent planar environments composed of polygonal obstacles, in order to efficiently determine collision-free paths.

Obstacles are considered to be constraints, and specific constraint insertion and removal routines are available for updating the CDT whenever obstacles appear, disappear or change position [Kallmann *et al.*, 2003].

Let n be the total number of vertices in a given set of obstacles. The initial construction of the CDT can be done in optimal O(n log n) time using a divide-and-conquer algorithm [Chew, 1987]. After the initial construction, the CDT can be updated in order to reflect eventual changes in the obstacle set. For this matter, several insertion and removal routines are available in the Computational Geometry literature.

Having the CDT up-to-date, a path joining any two given points (or a failure report) is obtained in two phases. First, a graph search performed over the CDT triangles adjacency graph determines a sequence of free triangles (a *channel*) joining both points. Finally, a linear pass algorithm computes the shortest path inside channels. The obtained path is the shortest in the homotopy class [Hershberger *et al.*, 1994] determined by its channel, but it might not be the globally shortest one.

The main advantage of the method is that, due to the underlying CDT, the size of the graph used for searching for paths is O(n), reducing the time required for determining shortest paths to the time required by the graph search itself, which can be performed in O(n log n) time with any Dijkstra-type search method [Cormen *et al.*, 1993]. In addition to that, as already mentioned, the CDT can be efficiently updated when obstacles change position.

The main application that motivated the development of this method is the navigation of characters in planar environments for virtual reality and game-like applications. Based on high-level decision mechanisms, characters decide where to go and determine a goal position to walk to. The path planner module is responsible to find a collision-free path towards the desired position. In such applications, handling dynamic environments and fast determination of paths are important; guaranteed shortest paths are not required.

The classical problem of finding shortest paths in the plane [Mitchell *et al.*, 1998] has been studied since a long time. Efficient sub-quadratic approaches are available [Mitchell *et al.*, 1996] and an optimal algorithm has been proposed taking O(n log n) time and space, where n is the total number of vertices in the obstacle polygons [Hershberger *et al.*, 1999].

However practical implementations are still based on grid-based search [Koenig, 2004] or on visibility graphs [Kreveld *et al.*, 2000]. Unfortunately, grid-based methods lead to large graphs when fine grids are used and visibility graphs can have $\Omega(n^2)$ edges in the worst case.

Even if not popular in real applications, good alternatives are available, as the *pathnet* graph [Mata *et al.*, 1997], constructed from a planar subdivision. The sparsity of pathnets can be controlled in order to get as close as desired to the global shortest path. This control is done by choosing the number k of rays emanating from the source node, resulting in a graph of size O(kn).

The method presented herein is also based on a planar subdivision, but speed is preferred over control of the global optimality of paths. A graph of fixed size O(n) is used, which is implicitly defined by the subdivision. This choice allows faster determination of paths and allows to dynamically update the subdivision in order to cope with dynamic environments.

2 Method Overview

The method can be divided in three main steps.

Step 1 Given a set of polygonal obstacles, a CDT having as constraints the edges of the obstacles is constructed. In order to consider discs of arbitrary radius r, obstacles can be grown by r [Laumond, 1987] before insertion in the CDT, reducing the problem to planning paths for a point [Latombe, 1991] (see Figure 1).

During run-time obstacles are allowed to be inserted, removed or displaced in the CDT as required. The CDT is able to dynamically take into account these changes and detailed algorithms are available in previous work [Kallmann *et al.*, 2003].

The CDT implicitly defines the polygonal domain used by the path search. It fills with triangles both the interior and the exterior of obstacles and ensures that the edges of obstacles are also edges of the triangulation (see Figure 2). If an edge of the triangulation is an obstacle edge, the edge is said to be *constrained*. Therefore triangulation edges can be of two types: constrained or non-constrained.

Step 2 Given two points p_1 and p_2, a graph search is performed over the adjacency graph of the triangulation, defining the shortest channel (according to the graph) connecting p_1 and p_2. This process first locates the triangle containing p_1, and then applies an A* search in the adjacency graph until the triangle containing p_2 is found. If the entire graph is searched and p_2 is not reached, a failure report is generated. Section 3 describes this process.

Step 3 Obtained channels are equivalent to triangulated simple polygons, and thus the *funnel algorithm* [Chazelle, 1982] [Lee *et al.*, 1984] can be applied in order to determine the shortest path joining p_1 and p_2 inside the channel. This takes linear time with respect to the number of vertices in the channel. For completeness purposes, the funnel algorithm is briefly described in section 4.

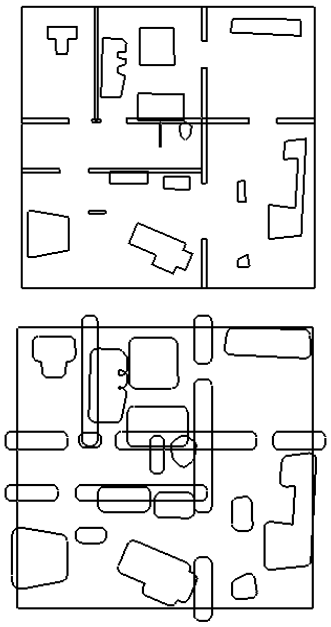
Figure 1. Obstacles inside a rectangular domain (top) and their grown versions (bottom). Growing obstacles ensures that paths maintain a given distance from the original objects.

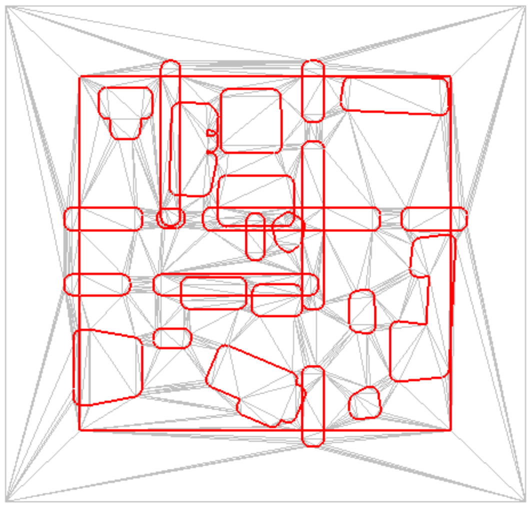
Figure 2. Grown obstacles inserted in the CDT. Edges of obstacles become constrained edges.

3 Channel Search

The polygonal domain considered by the path planner is implicitly defined as all triangles sharing non-constrained edges, starting from one given triangle.

Point Location Given two points p_1 and p_2, the first step is to determine the triangle t_1 that contains p_1. A point location routine is required for finding t_1. For robustness purposes, the same routine may also determine if p_1 lies outside the CDT domain.

Good results were obtained with the visibility walk approach [Devillers, 2001]. Starting with a seed triangulation vertex *v*, one triangle *t* adjacent to *v* is selected. Then, *t* is switched to the adjacent triangle *t'* such that the common edge of *t* and *t'* divides p_1 and *t* in different semi planes. If more than one edge can be selected, a random choice is taken in order to avoid possible loops. Intuitively, *t'* is now closer to p_1 than *t*. This process of switching triangles is continuously repeated. At a certain point it is no more possible to switch of triangles, and the last triangle *t* visited contains p_1. Rare cases may produce an exponential search, and for avoiding that, whenever the visibility walk is detected to traverse the total number of triangles a linear search over the remaining triangles is performed. The point location routine needs to be carefully crafted, specifically in relation to the used geometric primitives.

Graph Search Once triangle t_1 is found, a graph search over the triangulation adjacency graph is performed, starting from t_1 until the triangle containing p_2 is found, without traversing constrained edges. Note that it is essential to have the triangulation described by an efficient data structure [Guibas *et al.*, 1985], permitting to retrieve all adjacency relations in constant time. This is the case not only for the graph search step, but also for several other computations presented in this paper.

The considered connectivity graph is depicted in figure 3. A starting node has the same position as p_1. This node is then connected to the midpoint of each non-constrained edge of triangle t_1. This process is continuously repeated, expanding each node of the graph to the two opposite edges of the same triangle, if these edges were not yet reached and are not constrained. At each step, the edge with less cost accumulated is selected to be expanded. The cost is the Euclidian distance measured along the graph edges. The search finishes when the triangle containing p_2 is reached, and the shortest channel is determined by the history of traversed triangles in the branch from p_1 to p_2.

An additional A* heuristic cost function was included based on the Euclidian distance from the current leaf node to p_2. Additional cost information can be associated to triangles in order to indicate, for instance, different properties of the terrain being traversed. Note that constrained edges are not expanded guaranteeing that a path will never traverse them.

The graph shown in figure 3 captures the cost of a canonical path passing through the center of the non-constrained triangulation edges. This solution has shown to be more accurate than using the center of each triangle.

At the end of the search process, a channel joining p_1 and p_2 is determined. Figure 4 illustrates such a channel. We define the first and last triangles of the channel by connecting additional edges to p_1 and p_2 respectively.

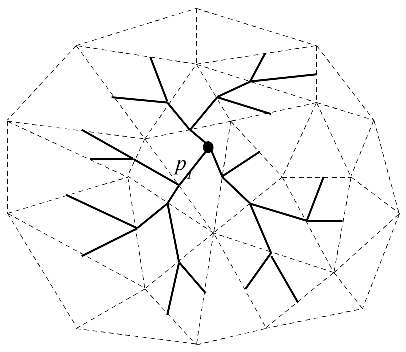

Figure 3. The connectivity graph (solid lines) is implicitly defined by the triangulation (dashed lines).

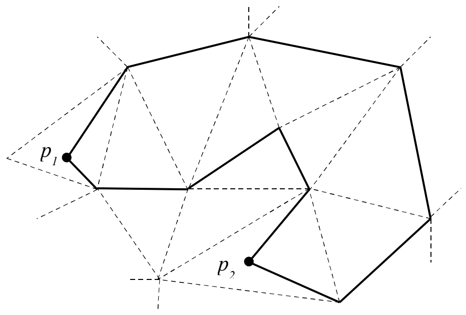

Figure 4. Channel (solid lines) joining point p_1 to p_2.

4 Paths Inside Channels

With the channel determined, the problem is now reduced to find the closest path inside a triangulated simple polygon.

For this, the *funnel algorithm* [Chazelle, 1982] [Lee *et al.*, 1984] can be applied for linearly determining the shortest path inside the channel. This algorithm is briefly reviewed here for completeness purposes, following the description of Hershberger and Snoeyink [Hershberger *et al.*, 1994].

Let *p* be a point and *uv* be a segment (figure 6). The shortest paths from *p* to *v* and from *p* to *u* may travel together for a while. At some point *a* they diverge and are concave until they reach *u* and *v*. The funnel is the region delimited by segment *uv* and the concave chains to *a*, and *a* is its *apex*. The vertices of the funnel are stored in a double-ended queue, a *deque*.

Figure 5 illustrates the insertion process of a new vertex *w*. Points from the *v* end of the deque are popped until *b* is reached, because the extension of edge *ab* is not below *w* as occurred with previous popped points. If the apex of the previous funnel is popped during the process, then *b* becomes the new funnel apex. Note that edge *bw* is on the shortest path from *p* to *w*. A similar symmetrical process is performed if the new vertex is between the extended edges of the upper concave chain of the funnel. Figures 7 and 8 show some examples of paths and channels obtained from CDTs.

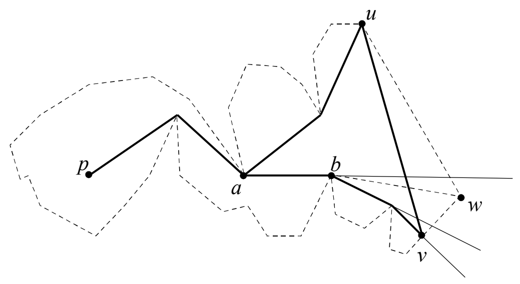

Figure 5. The funnel algorithm.

5 Results and Extensions

Examples of obtained channels and paths are presented in Figures 6, 7, 8 and 9. For instance it can be noticed in Figure 6 that the number of cells (i.e. triangles) in the triangulation is much smaller than the number of cells that would be required in a fine grid. Furthermore, the contour of obstacles is precisely described and not subject to a cell resolution choice.

Direct Visibility Test Given points p_1 and p_2, the obtained path joining the two points is not necessarily the globally shortest one. For instance, it is possible to obtain a case where p_1 and p_2 are visible through a straight line, but the path obtained from the planner is a different path. This kind of situation mostly happens when several possible solutions with similar lengths are available, as in Figure 7. A specific *direct visibility* test was implemented in order to detect such cases. When this test is activated, before performing the graph search to find a path, a straight walk in the triangulation [Devillers, 2001] is performed. The walk consists in traversing all the triangles that are intersected by the line segment p_1p_2, starting from p_1, towards p_2. If during the traversal a constrained edge is crossed, the test fails. Otherwise, the triangle containing p_2 is reached and the test succeeds, meaning that a straight line segment is the global shortest path between p_1 and p_2.

This test has shown to be beneficial in particular for the application of controlling characters in virtual environments when users usually remark if characters don't choose a straight line path whenever possible.

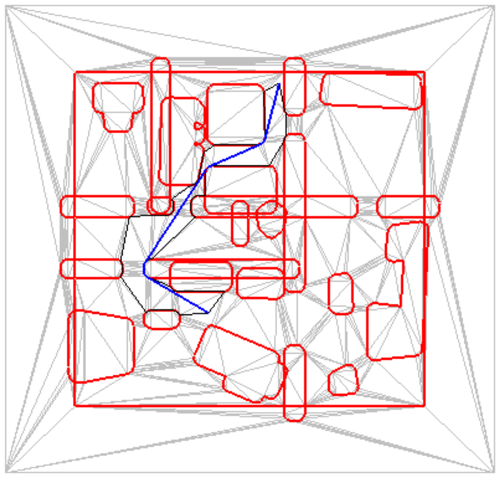

Figure 6. Example of a path and its channel.

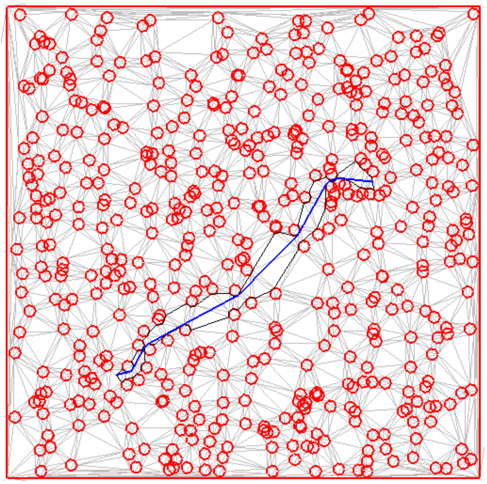

Figure 7. A path, its channel, and the CDT of 500 heptagons.

Other useful routines have been efficiently implemented based on the underlying CDT: ray-obstacle intersections, point-in-obstacle queries, and Boolean operation of obstacles.

Obstacles may also be defined as open polygons, i. e. as polygonal lines. Polygonal lines can be grown and inserted in the CDT, similarly to closed polygons. Figure 8 exemplifies one case based on line segments.

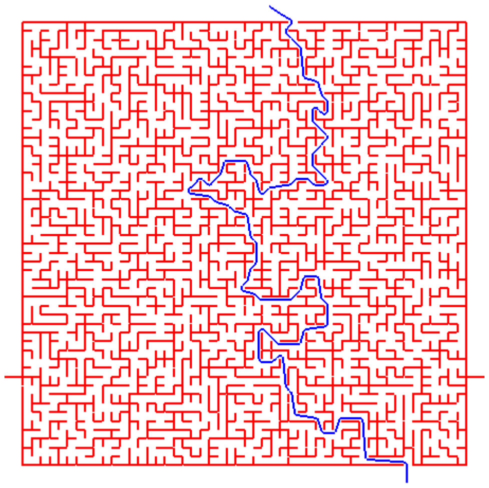

Figure 8. A maze composed of 2600 segments and one example path obtained. Each segment is considered to be one open obstacle (the CDT is not shown for clarity).

6 Conclusions

This paper presents methods for fast path planning in triangulated planar environments. The presented techniques were fully implemented.

The algorithms presented here are also useful for several other related purposes. For instance, the algorithm does not only computes shortest paths, but also the channels containing the paths. Such information can be very useful for spatial reasoning algorithms, and for bounding steering maneuvers when following planned paths.

The implemented software is being integrated with several other grid-based search methods for the purpose of evaluation, and will be soon available for research purposes.

Acknowledgments

The project or effort described here has been sponsored by the U.S. Army Research, Development, and Engineering Command (RDECOM). Statements and opinions expressed do not necessarily reflect the position or the policy of the United States Government, and no official endorsement should be inferred.

References

[Anglada, 1997] M. V. Anglada. "An Improved Incremental Algorithm for Constructing Restricted Delaunay Triangulations", Computer & Graphics, 21(2):215-223, 1997.

[Chazelle, 1982] B. Chazelle. A Theorem on Polygon Cutting with Applications. In Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science, 339-349, 1982.

[Chew, 1987] L. P. Chew, "Constrained Delaunay Triangulations", Proceedings of the Annual Symposium on Computational Geometry ACM, 215-222, 1987.

[Cormen *et al.*, 1993] T. Cormen, C. Leiserson, and R. Rivest. "Introduction to Algorithms", MIT Press, Cambridge, MA, 1993.

[Devillers, 2001] O. Devillers, S. Pion, and M. Teillaud, "Walking in a Triangulation", ACM Symposium on Computational Geometry, 2001.

[Floriani *et al.*, 1992] L. de Floriani and A. Puppo. "An On-Line Algorithm for Constrained Delaunay Triangulation", Computer Vision, Graphics and Image Processing, 54:290-300, 1992.

[Guibas *et al.*, 1985] L. Guibas and J. Stolfi. "Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams", ACM Transaction on Graphics, 4:75-123, 1985.

[Hershberger *et al.*, 1994] J. Hershberger and J. Snoeyink. "Computing Minimum Length Paths of a given Homotopy Class", Computational Geometry Theory and Application, 4:63-98, 1994.

[Hershberger *et al.*, 1999] J. Hershberger and S. Suri. An optimal algorithm for Euclidean shortest paths in the plane. *SIAM J. Comput.*, 28(6):2215-2256, 1999.

[Kreveld *et al.*, 2000] M. V. Kreveld, M. Overmars, O. Schwarzkopf, and M. de Berg. "Computational Geometry: Algorithms and Applications", ISBN 3-540-65620-0 Springer-Verlag, 2000.

[Latombe, 1991] J.-C. Latombe. Robot Motion Planning. Kluwer Academic Publishers, ISBN 0-7923-9206-X, Boston, 1991.

[Laumond, 1987] J.-P. Laumond, "Obstacle Growing in a Nonpolygonal World", Information Processing Letters 25, 41-50, 1987.

[Lee *et al.*, 1984] D. T. Lee and F. P. Preparata. Euclidean Shortest Paths in the Presence of rectilinear barriers. Networks. 14(3):393-410, 1984.

[Mitchell *et al.*, 1996] J. S. B. Mitchell. "Shortest paths among obstacles in the plane", International Journal on Computation Geometry Applications 6, 309-332, 1996.

[Mitchell *et al.*, 1998] J. S. B. Mitchell. "Geometric shortest paths and network optimization", in J.-R. Sack and J. Urrutia, editors, Handbook of Computational Geometry, Elsevier Science, Amsterdam, 1998.

[Mata *et al.*, 1997] C. S. Mata, and J. S. B. Mitchell. "A New Algorithm for Computing Shortest Paths in Weighted Planar Subdivisions". Proceedings ACM Symposium on Computational Geometry, 264-273, Nice, France, 1997.

[Preparata *et al.*, 1985] F. P. Preparata and M. I. Shamos. Computational Geometry: An Introduction. Springer-Verlag, ISBN 3540961313, 1985.

[Kallmann *et al.*, 2003] M. Kallmann, H. Bieri, and D. Thalmann, "Fully Dynamic Constrained Delaunay Triangulations", In Geometric Modelling for Scientific Visualization, G. Brunnett, B. Hamann, H. Mueller, L. Linsen (Eds.), ISBN 3-540-40116-4, Springer-Verlag, Heidelberg, Germany, pp. 241-257, 2003.

[Koenig, 2004] S. Koenig, "A Comparison of Fast Search Methods for Real-Time Situated Agents", AAMAS'04, July 19-23, New York, 2004.

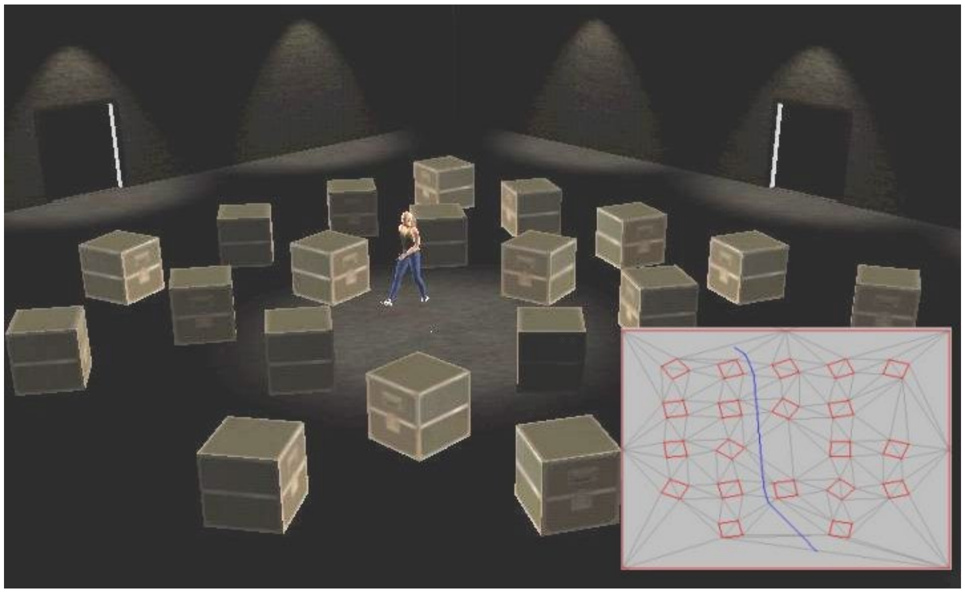

Figure 9. The image shows an interactive application where the virtual human is able to walk to a selected location without colliding with the boxes inside the room. Note that in this application the polygons representing the boxes are not grown before insertion in the CDT. Therefore found paths are further optimized to maintain a desired *clearance distance* from the obstacles. This illustrates a possible tradeoff between the number of triangles considered during the channel search and additional computation required to derive paths for the given channels.