1-1-1990

Planar Graphs and Sparse Graphs from Efficient Motion Planning in the Plane

L Paul Chew
Dartmouth College

PLANAR GRAPHS AND SPARSE GRAPHS FROM EFFICIENT MOTION PLANNING IN THE PLANE

L. Paul Chew

Technical Report PCS-TR90-146

Planar Graphs and Sparse Graphs for Efficient Motion Planning in the Plane

(extended abstract)

L. Paul Chew
Department of Math and Computer Science
Dartmouth College
Hanover, NH 03755

Abstract

Given a source, a destination, and a number of obstacles in the plane, the Motion Planning Problem is to determine the best path to move an object (a robot) from the source to the destination without colliding with any of the obstacles. For us, motion is restricted to the plane, the robot is represented by a point, and the obstacles are represented by a set of polygons with a total of n vertices among all the polygonal obstacles. Currently, the best method for finding the optimal path (the path with the shortest length) uses the visibility graph which can be constructed in time $O(n^2)$. We show that one can determine a reasonably good path, with length at most twice that of the optimal path, in time O(n log n). This technique uses a planar graph based on a special kind of Delaunay triangulation. An extension of this technique uses k different triangulations to construct, in time O(kn log n), a sparse graph which can be used to find a path with length at most $1+\pi/k$ times that of the optimal path.

1. Introduction

Given a source (A), a destination (B), and a set (S) of obstacles, the Motion Planning Problem is to determine the best path to move an object (a robot) from A to B without colliding with any of the obstacles of S. In this paper, we study the special case in which all motion is confined to the plane, the object to be moved is a point, and the obstacles are polygons. We use n to represent the total number of vertices, vertices of all the polygonal obstacles as well as the points A and B.

Currently, the best method for finding the optimal A-to-B path (i.e., the A-to-B path with minimum length) uses a particular graph called the *visibility graph*. The vertices of this graph consist of the points A and B plus the vertices of the obstacles. There is an edge between a pair of vertices iff the vertices can "see" each other; in other words, there is an edge between two vertices iff a straight line between the vertices does not cross any of the obstacles. It can be shown that the optimal shortest path must lie along edges of the visibility graph; thus, the shortest A-to-B path within the visibility graph is the optimal (shortest possible) path from A to B. Dijkstra's algorithm can be used to determine the shortest path within the n vertex visibility graph in $O(n^2)$ time and the visibility graph itself can be constructed in $O(n^2)$ time [We85, AAGHI85]. This time bound is optimal since the visibility graph can as many as $n(n-1)/2$ edges.

My own approach to the Motion Planning Problem is to drop the goal of an optimal path and to look instead for a reasonable path that can be found very efficiently. In this paper, we show how to construct a graph with the properties that (1) the graph is planar (in fact, the graph is a triangulation), (2) the graph can be constructed in $O(n \log n)$ time using $O(n)$ space, and (3) the length of the shortest path within the graph is at most twice that of the optimal path. It follows from property (1) that the

shortest path within the graph can be found in O(n log n) time. The graph with the above properties is based on a type of triangulation, called here a triangle-distance (TD) triangulation. These results improve on results presented in [Ch86] in which we developed a graph with properties 1 and 2 and a weaker version of property 3 (a bounding factor of $\sqrt{10}$ instead of 2).

In addition, we extend this technique, combining k triangulations to produce a graph with the properties that (1) the graph is sparse (it has O(kn) edges), (2) the graph can be constructed in O(kn log n) time using O(kn) space, and (3) the length of the shortest path within the graph is at most $1+\pi/k$ times that of the optimal path. It follows from property (1) that the shortest path within the graph can be found in O(kn log n) time using straightforward Dijkstra's algorithm or in time O(kn + n log n) using Dijkstra's algorithm with a Fibonacci heap [FT84].

The properties of some of the various methods for finding a path among obstacles in the plane are summarized in the following table. Running time is the time needed to find the shortest path within the graph once the graph has been created. The bounding factor shows the bound on the length of the shortest path within the graph; in other words, the length of the shortest path within the graph is bounded by "bounding factor" times "the length of the optimal path".

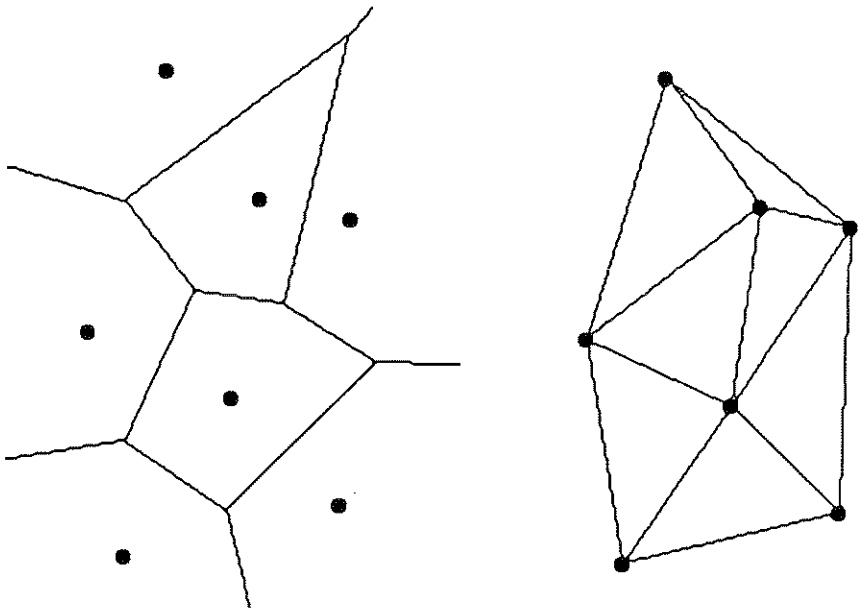
	preprocessing	space	running time	bounding factor
visibility graph	$O(n^2)$	$O(n^2)$	$O(n^2)$	1
TD triangulation	O(n log n)	O(n)	O(n log n)	2
k triangulations	O(kn log n)	O(kn)	O(kn + n log n)	$1+\pi/k$

Frederickson's [Fr83] results on shortest paths in planar graphs can be used to further reduce the running time for the TD triangulation, bringing the time down to O(n log*n) by using some additional preprocessing. This

additional preprocessing requires time O(n log n). Unfortunately, the constant hidden by the big-O notation is large enough to make this method impractical.

2. Background

The graphs we develop for efficient motion planning are based on a form of *Delaunay triangulation*. The Delaunay triangulation is the straight line dual of the *Voronoi diagram*. See [PS86] for definitions and a large number of Voronoi diagram applications.



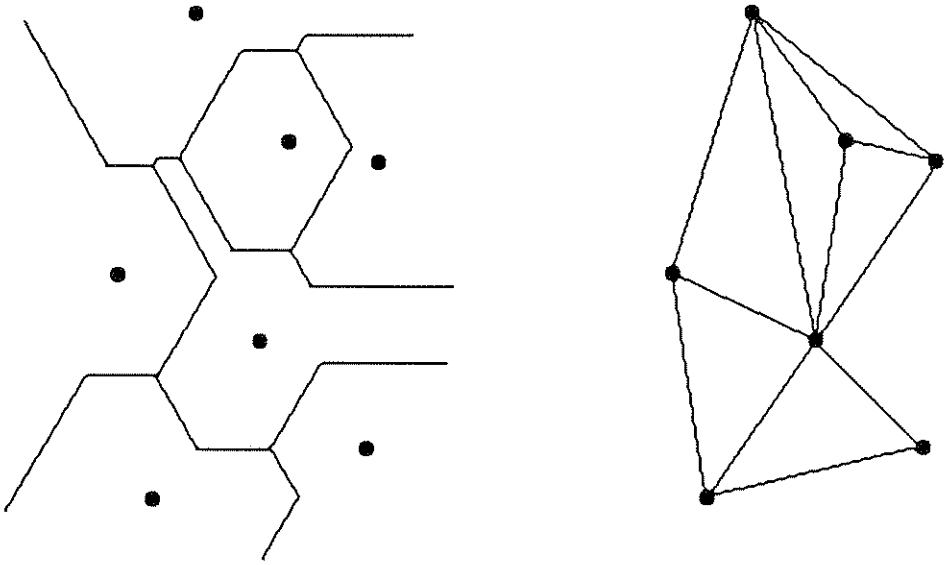
A Voronoi diagram and the corresponding Delaunay triangulation.

The results discussed here use a Delaunay triangulation based a different distance function than standard Euclidean distance. We use a *convex distance function* based on an equilateral triangle. Convex distance functions, often called Minkowski distance functions, were first used by Minkowski in 1911. The distance for such a distance function can be defined in terms of a unit "circle". "Circle", here, is printed with quotes

because this circle can be any convex shape. To find the distance from point p to point q, we center the unit "circle" at point p and expand (or contract) the circle until its boundary intersects q. By definition, the distance from p to q is the factor by which the circle changed. If the circle is a true circle, with its "center" in the usual place, then we get the usual Euclidean distance. If the "circle" is an arbitrary convex shape with a "center" anywhere in its interior then we get a convex distance function. Note that the distance defined in this manner is not necessarily a metric, since the symmetry property (the distance from p to q is the same as the distance from q to p) holds only if the unit circle is symmetric about its center. The convex distance function that we have found to be particularly useful for motion planning uses an equilateral triangle as the distance-defining convex shape. (Since we are concerned only with relative distances, any size equilateral triangle will do.)

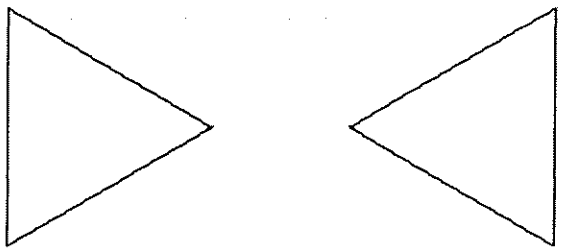
The TD (equilateral triangle convex distance function) Voronoi diagram is defined just like the standard (Euclidean) Voronoi diagram except the equilateral triangle convex distance function is used to calculate distances. A useful intuition is to think of circular waves (i.e., waves in the shape of the distance defining "circle") expanding from the set of data points; where the waves collide, we have a Voronoi boundary [CD85]. Like the standard Voronoi diagram, the boundaries of the TD Voronoi regions form a planar graph. My work with R. L. Drysdale [CD85] has shown that, like the Euclidean Voronoi diagram, the TD Voronoi diagram (and Voronoi diagrams based on many other distance functions) can be constructed in O(n log n) time where n is the number of points in the set S. The TD Delaunay triangulation can be derived from the corresponding Voronoi diagram in O(n) time, or, alternately, it can be built directly using a method similar to the Euclidean-case method presented in

[LS80]. It is also possible to build such diagrams using the sweep-line technique of [Fo86].



A TD Voronoi diagram and the corresponding TD Delaunay triangulation.

An important property of Delaunay triangulations is that for any empty triangle of the triangulation, the circumscribed circle contains no points of S in its interior. Of course, for the TD Delaunay triangulation, the circle in question is based on the equilateral triangle described above. An examination of the previous diagram should convince the reader that a Delaunay-triangulation empty circle is a reflection of the distance-defining circle.



Orientations of the distance-defining circle and a Delaunay-triangulation empty circle.

See [CD85] for more information on convex distance functions and their relationship to Voronoi diagrams and Delaunay triangulations.

3. Properties of TD Delaunay triangulations

The results of this paper are based on the following theorem. The proof of this theorem uses techniques similar to those developed in [Ch86] for the proof of a similar theorem about the L_1 Delaunay triangulation. (The L_1 metric is a convex distance function in which the unit "circle" is a square tipped at 45°. The result developed in [Ch86] has a bounding factor of $\sqrt{10}$ instead of the better bounding factor of 2, presented here.) The complete proof of this theorem will have to wait for a longer version of this paper.

Theorem 1. Let S be a set of points in the plane and let T be the TD Delaunay triangulation of S. For any points A and B of S, the shortest A-to-B path along edges of T has length $\leq 2\,|AB|$, where |AB| is the Euclidean straight-line distance from A to B.

The Theorem as presented above is not immediately applicable to motion planning since it does not allow for obstacles. One way to take obstacles into account is to create a *constrained Delaunay triangulation* (called an obstacle triangulation in [Ch86] or a generalized Delaunay triangulation in [Le78]).

Intuitively, the *constrained Delaunay triangulation* is the triangulation (on the vertices of the obstacles) that is as close as possible to the Delaunay triangulation with the restriction that the obstacle edges

must be included in the triangulation. Lee [Le78] presented an $O(n \log^2 n)$ time algorithm for constructing such a triangulation. In [Ch87], we show that a constrained Delaunay triangulation can be built in $O(n \log n)$ time using a method similar to that used by Yap [Ya84] for building the Voronoi diagram of a set of simple curved segments.

Definition. Let G be a straight-line planar graph. A *constrained Delaunay triangulation* of G is a triangulation T such that each edge of G is an edge of T and for each remaining edge e of T there exists a "circle" C with the following properties

1) the endpoints of edge e are on the boundary of C, and
2) if any vertex of G is in the interior of C then it cannot be "seen" from either endpoint of e (i.e., a line segment drawn from such an interior vertex to an endpoint of e must cross an edge of G).

It follows from the definition that if G has no edges then the constrained Delaunay triangulation is the same as the (unconstrained) Delaunay triangulation. A constrained Delaunay triangulation can be defined for any convex distance function - only the meaning of the word circle changes. We use the constrained TD (equilateral triangle convex distance function) Delaunay triangulation.

We show how a constrained TD Delaunay triangulation can be used for motion planning. Let T be the constrained TD Delaunay triangulation for our set of obstacles. If we choose a source and a destination among the obstacles then the optimal path is made up of straight line segments between obstacle vertices. (Recall that the the term "optimal path" refers to the shortest source-to-destination path that does not go through an obstacle; this path is not necessarily restricted to edges of the

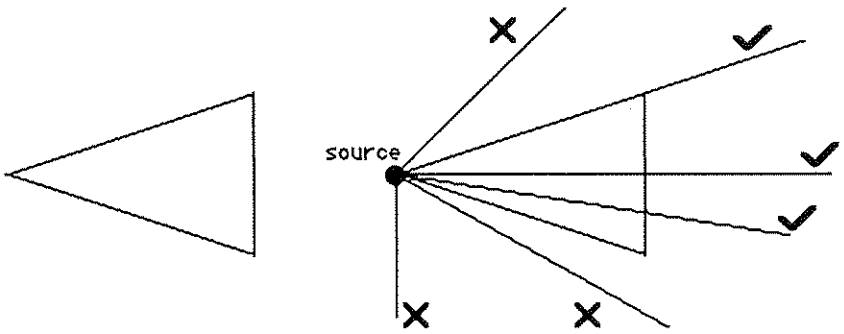
triangulation.) Let A and B be two obstacle vertices that are adjacent along the optimal path. Then the portion of T between A and B has properties similar enough to the (unconstrained) TD Delaunay triangulation for a version of Theorem 1 to apply; thus, there is an A-to-B path along edges of T with length $\leq 2\,|AB|$. Since this can be done for each segment of the optimal path, we conclude that there is a path along edges of T with length ≤ 2 times the length of the optimal path.

4. Using multiple triangulations

Instead of an equilateral triangle, an isosceles triangle can be used to define a convex distance function and to define, in turn, a Delaunay triangulation. Such an ITD (isosceles triangle convex distance function) Delaunay triangulation has properties similar to those stated in Theorem 1 for the TD Delaunay triangulation. However, because of the asymmetries of the isosceles triangle, the corresponding theorem has a preferred direction. (The proof is similar to the proof of Theorem 1.)

Theorem 2. Let Δ be an isosceles triangle (oriented as in the following figure) with an angle $\beta<60°$ between the two equal sides. Let S be a set of points in the plane and let T be the ITD (using Δ as the distance-defining "circle") Delaunay triangulation of S. For any points A and B of S for which segment AB is at an angle (from horizontal) between $-\beta/2$ and $+\beta/2$, the shortest A-to-B path along edges of T has length $\leq (1 + 2\sin(\beta/2))\,|AB|$.

Intuitively, the isosceles triangle can be thought of as a flashlight beam projected from our source vertex. Vertices that are illuminated by the flashlight are destinations for which we can find a reasonably good path.



Orientation of an empty "circle" and some directions in which good shortest paths can be found.

Of course, given a source and a destination, the problem is symmetric and we have the option of switching our terminology, changing source to destination and destination to source. For example, in the previous diagram, directions opposite the checkmarked directions are also directions for which a good path can be found.

By using several ITD Delaunay triangulations, each of which has a preferred direction for good paths, we can handle all the directions. We divide a semicircle by marking off equal portions of its circumference to define k isosceles triangles, each with an angle of π/k at the center. Each triangle can be used to build an ITD Delaunay triangulation that produces good paths in the directions covered by that triangle (using the flashlight analogy, the directions in which it shines and the opposite directions). By using a set of triangles that fill a semicircle, we guarantee that for each direction there is an isosceles triangle that handles that direction.

Given a set of obstacles in the plane, a semicircle full of isosceles triangles (as described above) is used in the following way. For each of the k triangles, we form the constrained ITD Delaunay triangulation. This produces k different triangulations which can be combined to make a single sparse (O(kn) edges) graph H. Using Theorem 2 and arguments similar to those used for the equilateral triangle case, it follows that for

any source and destination there is a path along edges of H with length $\leq$ $(1 + 2 \sin(\pi/2k))$ times the length of the optimal path. Some simple properties of trigonometric functions can be used to simplify this, giving an H-path with length $\leq (1+\pi/k)$ times that of the optimal path. The time bounds for this whole process are listed in the Introduction.

5. Conclusions and further research

The motion planning methods presented here for finding a good suboptimal path have significant advantages over available techniques for finding the optimal path. The savings in preprocessing (the time to build the graph), storage space, and running time (the time to compute the shortest path within the graph) are particularly important in cases where a large number of motion planning problems must be done for a single set of obstacles. The savings in time and space outweigh the fact that the techniques do not necessarily produce the optimal path, especially since the paths produced are guaranteed to be reasonably good.

There may be graphs that are better for motion planning than the TD Delaunay triangulations used here. For instance, the standard Euclidean Delaunay triangulation may have a bounding factor of $\pi/2$. An attempt to prove this conjecture led to the results presented in this paper.

A particularly interesting possibility is that of extending some of these results to problems in 3 dimensions. It may be possible to develop an algorithm for motion planning in 3 dimensions that runs in time $O(n^2)$ and that produces a path with length bounded by a small constant time the length of the optimal path.

References

[AAGHI85] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, Visibility-polygon search and Euclidean shortest paths, Proc. 26th IEEE Symposium on Foundations of Computer Science (1985), 155-164.

[CD85] L. P. Chew and R. L. Drysdale, Voronoi diagrams based on convex distance functions, Proceedings of the 1st Symposium on Computational Geometry, Baltimore (1985), 235-244. (Revised version submitted to Discrete and Computational Geometry.)

[Ch86] L. P. Chew, There is a planar graph almost as good as the complete graph, Proceedings of the Second Annual Symposium on Computational Geometry, Yorktown Heights (1986), 169-177.

[Ch87] L. P. Chew, Constrained Delaunay triangulations, in preparation.

[Fo86] S. Fortune, A sweepline algorithm for Voronoi diagrams, Proceedings of the Second Annual Symposium on Computational Geometry, Yorktown Heights (1986), 313-322.

[Fr83] G. N. Frederickson, Shortest path problems in planar graphs, Proceeding of the 24th Annual Symposium on Foundations of Computer Science, IEEE Computer Society (1983), 242-247.

[FT84] M. L. Fredman and R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, Proceedings of the 25th Annual Symposium on Foundations of Computer Science, IEEE Computer Society (1984), 338-346.

[Le78] D. T. Lee, Proximity and reachibility in the plane, Technical Report R-831, Coordinated Science Laboratory, University of Illinois (1978).

[LS80] D. T. Lee and B. Schachter, Two algorithms for constructing Delaunay triangulations, International Journal of Computer and Information Sciences, 9:3 (1980), 219-242.

[PS85] F. P. Preparata and M. I. Shamos, Computational Geometry, Springer-Verlag (1985).

[Ya84] C. K. Yap, An O(n log n) algorithm for the Voronoi diagram of a set of simple curve segments, Technical Report, Courant Institute, New York University (Oct. 1984).

[We85] E. Welzl, Constructing the visibility graph for n line segments in $O(n^2)$ time, Information Processing Letters, 20 (1985), 167-171.