# ROBOT MOTION USING DELAUNAY TRIANGULATION

**by**
**Ioana-Maria Ileană**

**Abstract**. Robot motion is nowadays studied from a lot of different perspectives. This paper is based on the assumption of a fully known environment, allowing the supervisor to plan the robot trajectory before the actual motion. The auxiliary structure we mainly use is the Delaunay triangulation. We analyze the implementation details, as well as the limits of the geometrical model we use.
**Keywords:** robot motion, geometrical model, Delaunay triangulation.

## 1. Initial environment data; geometrical model

One of the most basic tasks in robotics is navigation through an obstacle-constrained space. Leaving aside the real-time adjusting models, we place our investigation under the hypothesis of a fully known map, which can be processed by the robot user to build useful guiding structures – namely, the Delaunay triangulation and its dual, the Voronoi diagram.

In order to quickly obtain an operational model, we choose the simplest geometrical frame: two-dimensional environment, obstacles as two-dimensional points, circular shape (radius 1, knowing that any other radius value can be reduced to the unit, using a global scaling factor) for the mobile robot.

## 2. Algorithms; sketched proofs

We remind briefly the definition of the main two structures used in this paper. Given a set S of two-dimensional points, we define the Voronoi diagrams as a set of partially or totally closed polygonal cells, each cell corresponding to one and only one point in S, and constructed as follows: for every point q, its cell is the reunion of all points that are closer to q than to any other point in S.

The Delaunay triangulation is Voronoi diagrams' dual graph: every two points having adjacent Voronoi cells will be connected by a Delaunay edge. It is shown that this way we obtain a correct triangulation of the points in S; although we define Delaunay triangulation starting from Voronoi cells, the actual building process works 'in reverse': it is much easier to implement and update the Delaunay structures.

We assume the reader is familiar with the basic theoretical results as well as the generic algorithms concerning Delaunay triangulation. Knowing that obtaining this structure was not the main purpose of our study, we have chosen to code the triangulation using the incremental algorithm, which provides a satisfactory complexity (n^2 whereas n log n is proved to be the best complexity level) and seems most appealing due to its simplicity.

We reformulate our problem as follows: we shall further study the motion of a mobile point (the robot center) inside an environment blocked by circular obstacles. Moreover, we model a polygonal rectangular box by placing close obstacles on the borders (the alternatives will be considered in the last section).

. We build the Delaunay triangulation and the Voronoi diagrams of the set of obstacles (viewed as points). A graphical overview is shown in the figure below:

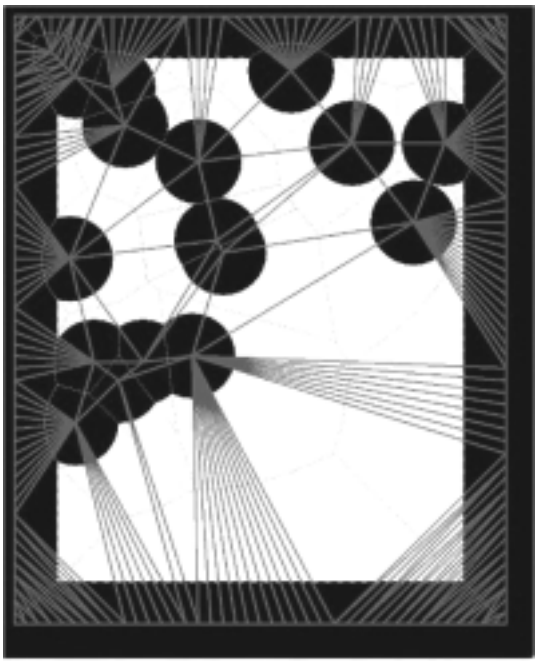

Fig. 1. Initial environment: circular obstacles, Delaunay triangulation, Voronoi diagrams

We shall further prove that using data providing a valid start and end point, any valid trajectory (we refer to a trajectory as valid if it does not cut any circular obstacles) of the robot center can be 'transformed' into a trajectory based only on Voronoi edges.

Indeed, we can decompose the trajectory into segments so that every segment belongs to one and only one Voronoi diagram. Given a segment, its validity means it is placed at a distance d >=1 (obstacle radius) from the point p to which the cell corresponds. We proceed to simply replace the segment with the sequence of the cell bord starting with the 'entering' point of the segment and ending with its 'exit' point (where the orientation is induced by the trajectory). We're sure to stay valid: the 'border' points are (due to the Voronoi diagrams definition) placed on a distance to the cell center greater or equal that any 'interior' point (the segment points as well).

This simple remark gives us the amount of operational directives needed to build a trajectory: step 1, we find a path from the initial point to a Voronoi vertex p1, step 2, we proceed similarly to obtain a path from the end point to a Voronoi vertex p2, step 3, we find a path between p1 and p2 using Voronoi edges. The initial observation guaranties us that whenever a valid path exists, a path constructed as mentioned exists.
The implementation of the first two steps goes as follows: we find the closest obstacle p of the analyzed point (start or end point); we trigonometrically turn around this obstacle using Voronoi vertexes, until we find vertexes v(j) and v(j+1) so that the analyzed point is placed inside the triangle p v(j) v(j+1). We examine the validity of the edge v(j) v(j+1) and we choose between v(j) and v(j+1).

The validity of a Voronoi edge shall be also used to filter the original Voronoi graph and allow or not the edge to be placed on our trajectory. We establish this validity by analyzing if the edge is or not cut by a circular obstacle. In fact, the Voronoi properties make it possible to examine only the two extremities of the edge to mark it valid or non-valid.

The final path between p1 and p2 is calculated as an usual shortest path, in the valid edges' graph.

## 3. Graphical results

We give a graphical perspective on the results obtained in the figures below. We show the initial obstacle structure, the Delaunay triangulation and Voronoi diagrams. We also show the three steps in building a valid trajectory.

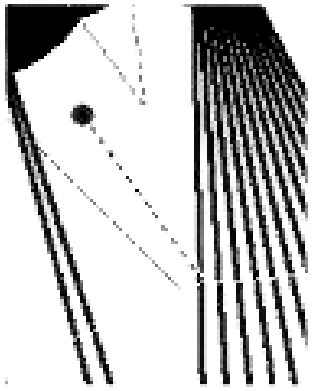

Fig 2. Step 1 in building a trajectory. The starting point is marked with a small full disk.

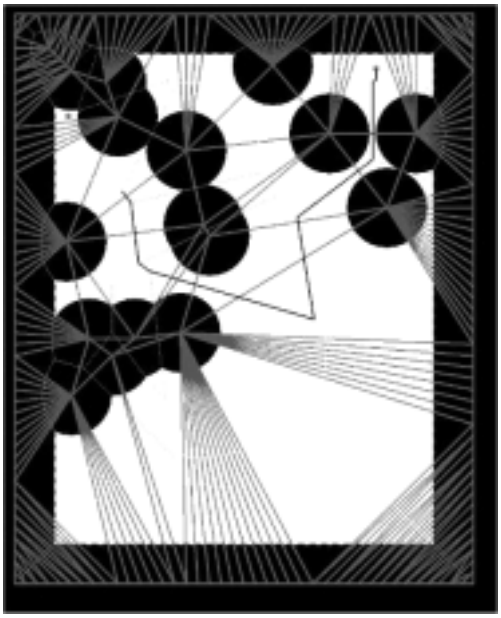

Fig. 3. First example: partial trajectory

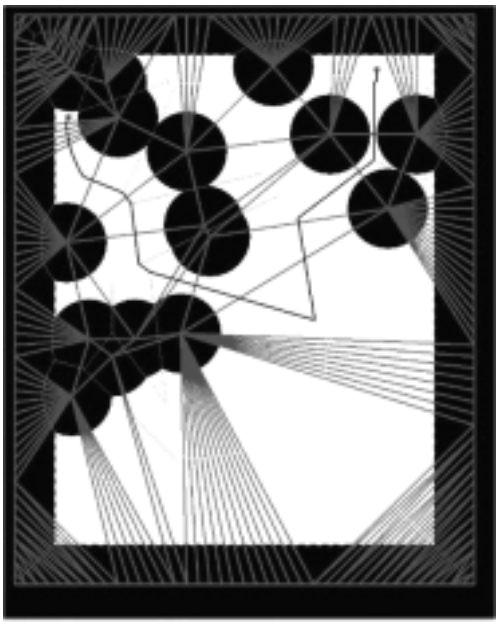

Fig. 4. First example: complete trajectory

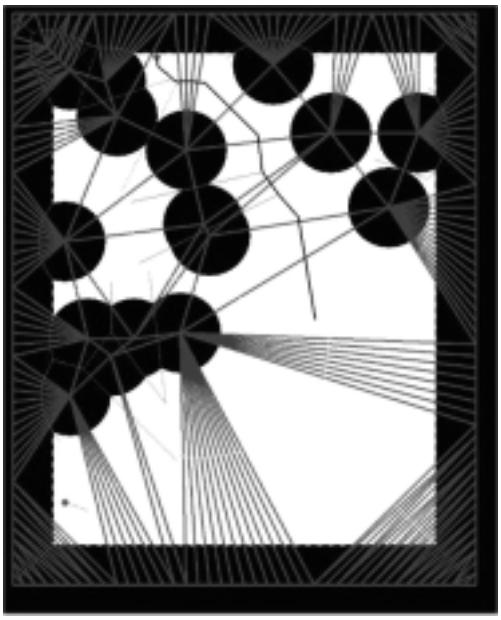

Fig. 5. Second example, partial trajectory

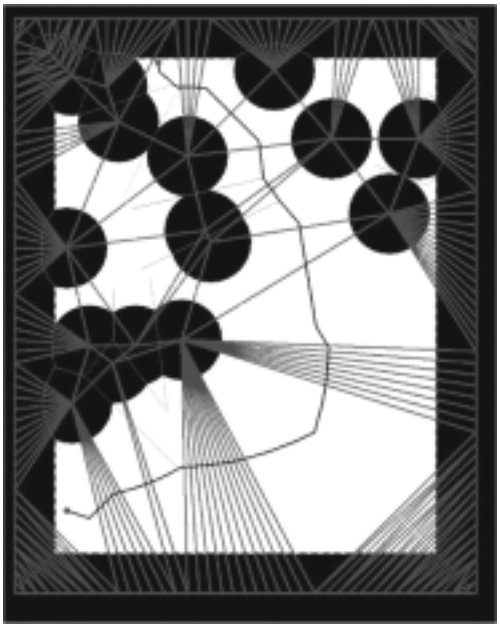

Fig. 6. Second example, complete trajectory

## 4. Problems, questions, limits

When trying to build the triangulation, we had to deal with degenerated situations. The incremental algorithm is based on a first triangulation of the convex closure, problem for which there is a well known algorithm (Tsai's algorithm) that does not make coherent decisions when facing a degenerated situation. We therefore improved Tsai's algorithm, and also coded a probabilistic approach which is known to be very quick and convenient in practice.

A major choice in the final problem modeling was the border modeling: we have reminded briefly this point, and we re-insist upon in this section.

The first choice had in sight a convex polygonal box, to which we would have had to intersect the Voronoi edges of the open cells. This procedure came up to be very complex in practice, therefore we restrained our model to 'conveniently placed' obstacles. The placement of these obstacles is in itself an interesting problem, in a certain way asking for 'reverse' calculus (finding points starting with a partial edge set). As we have already mentioned, we have conducted our experiment placing 'very close' obstacles on borders (at a distance smaller then the obstacle diameter), thus arriving to a perhaps unnecessary complexity.

There are algorithms for building the Delaunay triangulation of a set of more complex-shaped objects. We think our study has very well functioned as a first approach, but also consider further developments that will allow us to overcome the limits of this very simple and basic model.

**References**

[1].Olivier Devillers- Geometrie et synthese des images, Ecole Polytechnique, 2003

**Author:**

Ileană Ioana-Maria, Ecole Polytechnique, Paris, France, ileana@poly.polytechnique.fr