Robot Motion Planning Using Generalised Voronoi Diagrams

MILOŠ ŠEDA, VÁCLAV PICH
Institute of Automation and Computer Science
Brno University of Technology
Technická 2, 616 69 Brno
CZECH REPUBLIC

Abstract: - In robot motion planning in a space with obstacles, the goal is to find a collision-free path of robot from the starting to the target position. There are many fundamentally different approaches, and their modifications, to the solution of this problem depending on types of obstacles, dimensionality of the space and restrictions for robot movements. Among the most frequently used are roadmap methods (visibility graphs, Voronoi diagrams, rapidly exploring random trees) and methods based on cell decomposition. A common feature of all these methods is the generating of trajectories composed from line segments. In this paper, we will show that generalised Voronoi diagrams can be used for fast generation of smooth paths sufficiently distant from obstacles.

Key-Words: - motion planning, cell decomposition, sampling methods, roadmap method, generalised Voronoi diagram

1 Introduction

The task of planning trajectories of a mobile robot in a scene with obstacles, has received considerable attention in the research literature [2, 10, 12, 14].

There are three basic types of robot motion planning algorithms [9] based on potential fields, cell decompositions and roadmaps. The potential-fields methods are used rarely because they often converge to a local minimum. Cell decompositions avoid this drawback and, in a special case with the scene divided into squares, then they can be simply used for 8-directional (horizontal, vertical and diagonal) robot motion. Their main drawbacks are combinatorial explosion, limited granularity and generation of infeasible solutions. The combinatorial explosion and time of computation may be partially reduced using a case-based reasoning procedure [5].

There are several different methods for developing the roadmap such as visibility graphs and Voronoi diagrams [9]. These methods do not have the drawbacks of the previously-mentioned ones and thus are more promising for the task under investigation. However, all these methods generate non-smooth trajectories and need not be suitable for real robots with kinematic and dynamic restrictions. Therefore, we will search for ways of avoiding this problem.

First of all, we will outline some of the traditional approaches. A robot is usually represented by a single point or a circle. Algorithms may identify a robot with a mere point enlarging the obstacles in the workspace accordingly to avoid having to consider the robot's size.

2 Cell Decomposition

First, let us consider robot motion planning reduced to navigating a point in a free space F. Then the cell decomposition can be stated as follows [10]:

1. Divide F into connected regions called *cells*.
2. Determine which cells are adjacent and construct an adjacency graph. The vertices of this graph are cells, and edges join cells that have a common boundary.
3. Determine which cells the start and goal lie in, and search for a path in the adjacency graph between these cells.
4. From the sequence of cells found in the last step, compute a path connecting certain points of cells such as their midpoints (centroids) via the midpoints of the boundaries.

Fig. 1 presents a trapezoidal decomposition. A set of vertical lines that do not cross obstacles is constructed and their centres are determined. Connections of the centres of neighbouring vertical lines create a set of graph edges. To this set, an edge joining the starting position with the closest centre and an edge joining the target position with the closest centre are added. In this graph, a shortest

path between the starting and target positions can be found easily. It is evident that, due to polygonal obstacles, regions determined by vertical lines, obstacle edges and scene boundaries, the decomposition regions have a trapezoidal shape.

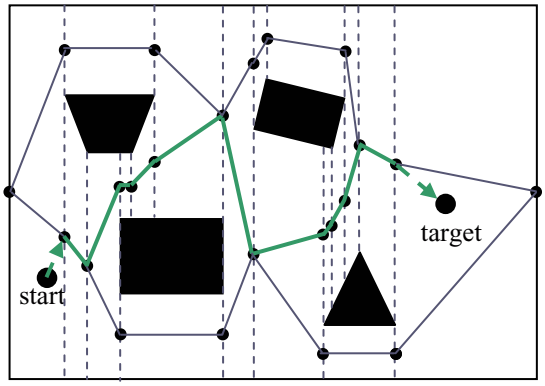

Fig. 1: Trapezoidal decomposition

3 Sampling Methods

Assume that R is a robot of a convex shape, $R(x,y)$ denotes its reference point at (x,y) and the obstacles are also convex. (Non-convex obstacles, as in Fig. 5, can be easily divided into several convex parts.) The *obstacle region* (or *configuration-space obstacle* or *C-obstacle*) of an obstacle P and the robot R is defined as the set of points in the configuration space such that the corresponding placement of R intersects P. Denote C_P the obstacle region of an obstacle P. Then

$$C_P = \{(x,y) \mid R(x,y) \cap P \neq \varnothing\} \qquad (1)$$

In order to avoid the explicit construction of the obstacle region, *sampling-based motion planning* algorithms have been proposed such as *probabilistic roadmap methods* or *rapidly exploring random trees* [6].

The rapidly exploring tree grows from the starting position that initialises the tree. The principle of its extension is shown in Fig. 2. At each step, a point is randomly generated and, by a shortest possible way, connected to the current tree. Fig. 2 shows a tree containing four points to which two points, q_1 and q_2, should be added. First q_1 is connected by a new edge to the nearest point of the current tree. In the second case, the nearest point lies on an edge and thus the edge is split into two parts and a new vertex is inserted into the tree.

Of course, the points generated into the regions occupied by obstacles and the points whose line connections with the nearest points would intersect an obstacle are not accepted. Since the starting and target positions determine the direction in which the robot should move, it is possible to consider this fact, e.g., in such a way that, for generated points, a maximal deviation from this direction is determined.

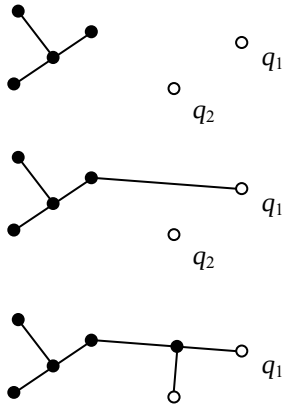

Fig. 2: Construction of rapidly exploring tree

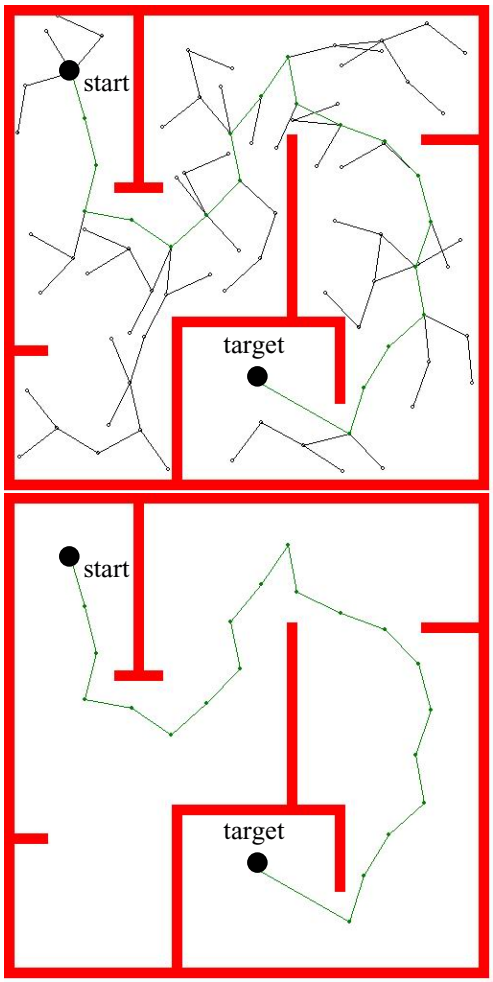

Fig. 3: Planning trajectory generated from the rapidly exploring tree

As a rule the distance of a newly generated point from the current tree is restricted, see [4, 6]. If a

point of the exploring tree reaches a position in a sufficiently small distance from the target, then it is directly connected to the target and further points are not generated.

A certain complication is that we must implement a function for finding the nearest point of the tree. There are exact and approximate methods for such computation and both of them have some drawbacks [6].

If there is an obstacle, the edge travels up to the obstacle boundary, as far as allowed by the collision detection algorithm [6].

The drawback of the rapidly exploring trees is a high number of generated edges resulting in a broken trajectory, see Fig. 3 [4]. It can be smoothed, to a certain extent, using splines.

4 Roadmap Methods

The most important approaches included in roadmap methods are based on visibility graphs and Voronoi diagrams.

A *visibility graph* is a graph whose vertices include the start, target and the vertices of polygonal obstacles [2, 7]. Its edges are the edges of the obstacles and edges joining all pairs of vertices that can see each other. Unfortunately, the shortest paths computed by using visibility graphs touch obstacles at the vertices or even edges of obstacles and thus are not very good in terms of safety. This drawback can be removed using Voronoi diagrams.

A *Voronoi diagram* of a set of sites in the plane is a collection of regions that divide up the plane. Each region corresponds to one of the sites and all the points in one region are closer to the site representing the region than to any other site [1, 3, 10]. An example of the Voronoi diagram is shown in Fig. 4.

Let $d(p_i, p_j)$ denote the distance between two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ in the plane. Then more formally, we can define Voronoi diagrams in mathematical terms.

Definition: Let $P=\{p_1, p_2, \dots, p_n\} \subset \Re^2$ be a set of points with the Cartesian coordinates (x_1, y_1), … , (x_n, y_n) where $2 < n < \infty$ and $p_i \neq p_j$ for $i \neq j$. We call the region

$$V(p_i) = \left\{ x \in \Re^2 \mid d(x, p_i) \leq d(x, p_j) \text{ for } j \neq i \right\} \quad (2)$$

the *planar Voronoi polygon* associated with p_i (or the Voronoi polygon of p_i) and the set given by

$$V = \{V(p_i), \dots, V(p_n)\} \quad (3)$$

the *planar Voronoi diagram* generated by P (or the Voronoi diagram of P). We call p_i of $V(p_i)$ the *site* or *generator point* or *generator* of the i-th Voronoi polygon and the set $P=\{p_1, p_2, \dots, p_n\}$ the *generator set* of the Voronoi diagram V. Hence we get

$$V(P) = \bigcup_{p_i \in P} V(p_i) =$$

$$= \bigcup_{p_i \in P} \left\{ x \in \Re^2 \mid d(x, p_i) \leq d(x, q) : \forall q \in (P - \{p_i\}) \right\} = \quad (4)$$

$$= \bigcup_{p_i \in P} \left[\bigcap_{q \in P - \{p_i\}} \left\{ x \in \Re^2 \mid d(x, p_i) \leq d(x, q) \right\} \right]$$

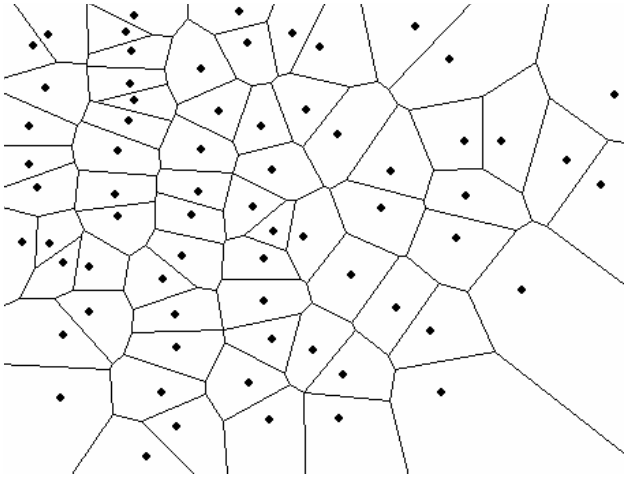

Fig. 4: Voronoi diagram

For reasons of time complexity, it is necessary to know the properties of the Voronoi diagrams and the algorithms of their constructions. We only mention the most substantial properties:
(i) The number of vertices in a Voronoi diagram of a set of n point sites in the plane is at most $2n-5$ and (ii) the number of edges is at most $3n-6$.

The algorithms used to construct Voronoi diagrams (*divide and conquer*, *incremental* and *plane sweep*) need $O(n \log n)$ time.

If a generator set of a Voronoi diagram represents point obstacles and other obstacles are not present in the plane, then the robot can walk along the edges of the Voronoi diagram of P that define the possible channels that maximise the distance to the obstacles, except for the initial and final segments of the tour. This allows us to reduce the robot motion problem to a graph search problem: we define a subgraph of the Voronoi diagram consisting of the edges that are passable for the robot. However, some of the edges of the Voronoi diagram may be impassable. Then these edges must be omitted from the diagram.

Of course, when we first construct the configuration space for obstacles, it is not necessary to test whether a robot can walk along the edges of the Voronoi diagram.

For scenes with point, straight-line and polygonal obstacles, the simplest way of finding optimal trajectories is to compute ordinary Voronoi diagrams for vertices of obstacles and then remove those of its edges that intersect obstacles. We get more precise solutions by approximating the polygonal edges by line segments and then applying the previous approach [11,13].

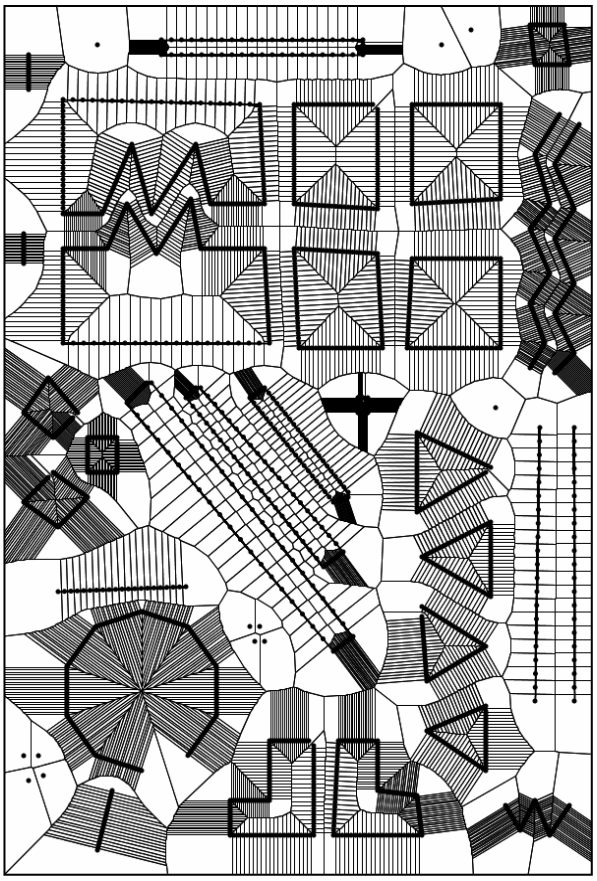

Fig. 5: Voronoi diagram with redundant edges

An implementation of this approach is described in [13]. Using this program, we can determine the number of line segments that approximate the edges of polygonal obstacles and compute the final Voronoi diagram with more precise edges.

The last two figures demonstrate the results. Fig. 5 and 6 show the Voronoi diagram for point, line and polygonal obstacles with 20 edge segments before and after removing redundant edges.

5 Generalised Voronoi Diagrams

In the previous section, polygonal obstacles were substituted by sets of their vertices and approaches for ordinary Voronoi diagrams were applied. However, the resulting trajectories were not smooth. If we deal with obstacles as sets of their boundaries approximated by lines, then we can precisely compute the bisectors between point and line or between two lines as follows.

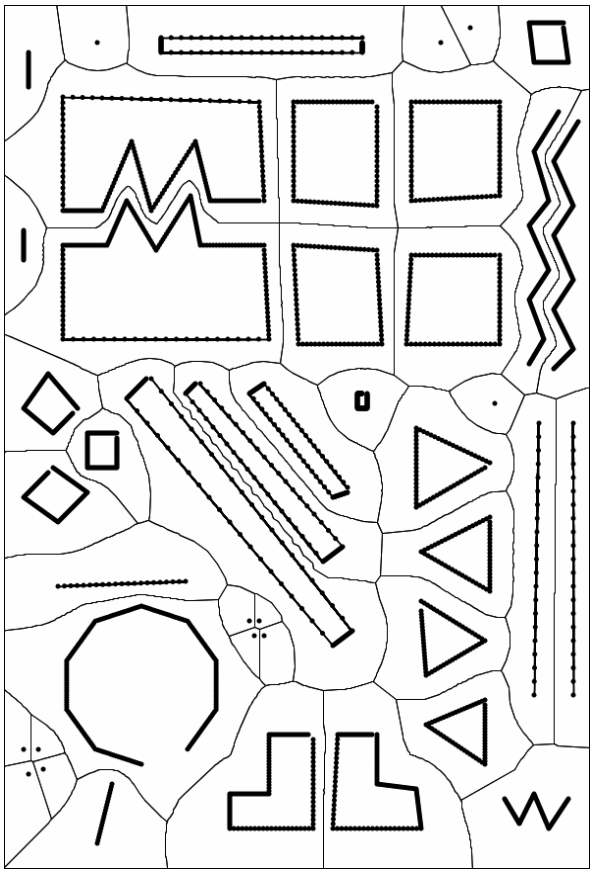

Fig. 6: Voronoi diagram for point, line and polygonal obstacles after avoiding redundant edges

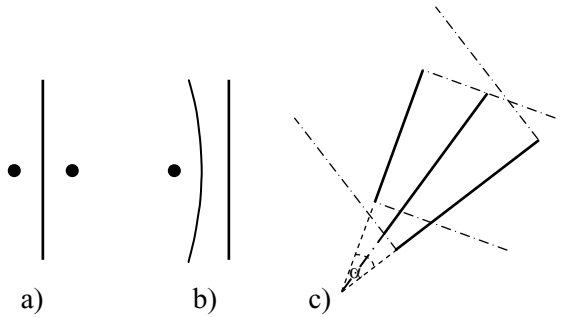

Fig. 7: Edges in a generalised Voronoi diagram

Fig. 7 shows that a) bisector for two points is given by an axis perpendicular to the centre of their connection, b) bisector for a point and line is given by a parabolic arc and the point is its locus, and c) bisector of two lines is given by the line dividing the

angle between the given lines. Therefore, the edges of the generalised Voronoi diagrams are composed by straight lines and parabolic arcs.

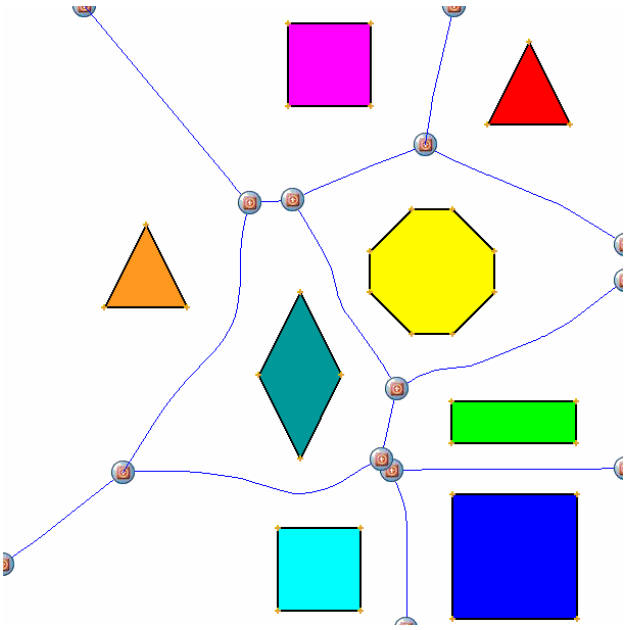

Fig. 8: Generalised Voronoi diagram for a scene with 8 polygonal obstacles

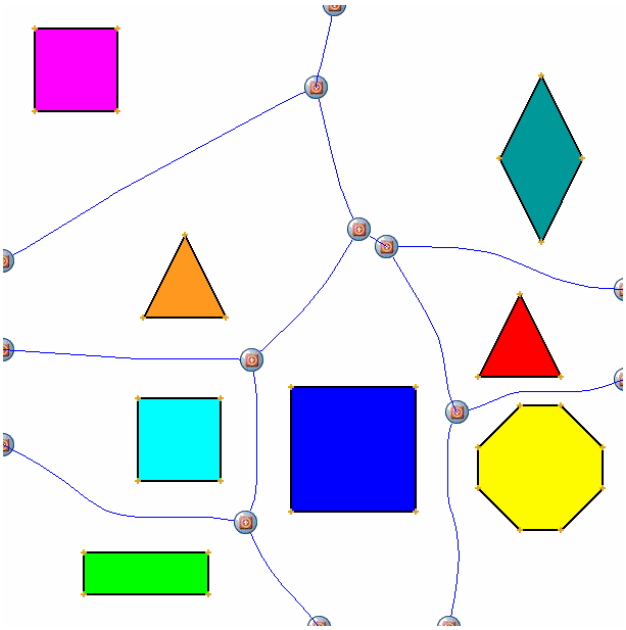

Fig. 9: Generalised Voronoi diagram for a modified configuration

Fig. 8 and Fig. 9 show generalised Voronoi diagrams for two configurations of the same scene with 8 polygonal obstacles from the implementation of V. Pich [8]. Since the redrawing of the diagram for moving obstacles runs in real time, we can assume that it could be used for robot motion planning in a dynamic scene.

6 Conclusions

In this paper, we briefly summarised the main approaches to robot motion planning and showed that they generate broken trajectories that require frequent changes of direction of the moving robot. As algorithms for constructing the Voronoi diagrams run in polynomial time, a promising way of avoiding this drawback may be to adapt the algorithms using ordinary Voronoi diagrams to a general case with not only straight line boundaries.

If a scene contains movable obstacles and these obstacles move along continuous curves, then the corresponding generalised Voronoi diagram also changes continuously and, therefore, the position of the robot will be changed continuously.

In the future, we will do more experiments with movable obstacles and try to determine their maximal number for real-time control.

Acknowledgments

The results presented have been achieved using a subsidy of the Ministry of Education, Youth and Sports of the Czech Republic, research plan MSM 0021630518 "Simulation modelling of mechatronic systems".

References:

[1] F. Aurenhammer, Voronoi Diagrams – A Survey of a Fundamental Geometric Data Structure, *ACM Computing Surveys*, Vol.23, No.3, 1991, pp. 345-405.

[2] M. de Berg, M. van Kreveld, M. Overmars and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, Berlin, 2000.

[3] S. Fortune, Voronoi Diagrams and Delaunay Triangulations, in: D.A. Du and F.K. Hwang (eds.), *Euclidean Geometry and Computers*, World Scientific Publishing, Singapore, 1992, pp. 193-233.

[4] J. Krejsa and S. Věchet, Rapidly Exploring Random Trees Used for Mobile Robot Path Planning, *Engineering Mechanics*, Vol.12, No.4, 2005, pp. 231-237.

[5] M. Kruusmaa and J. Willemson, Covering the Path Space: A Casebase Analysis for Mobile Robot Path Planning, *Knowledge-Based Systems*, Vol.16, 2003, pp. 235-242.

[6] S.M. LaValle, *Planning Algorithms*, University Press, Cambridge, 2006.

[7] A. Okabe, B. Boots, K. Sugihara and S.N. Chiu, *Spatial Tessellations and Applications of Voronoi Diagrams*, John Wiley & Sons, New York., 2000.

[8] V. Pich, *Application of Voronoi Diagrams in Robot Motion Planning* (in Czech), Master Thesis, Brno University of Technology, 2008, 64 pp.

[9] M. Ruehl and H. Roth, Robot Motion Planning by Approximation of Obstacles in Configuration Space, *16th IFAC World Congress*, Prague, 2005, 6 pp., submitted.

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, Prentice Hall, New Jersey, 1995.

[11] M. Šeda, A Comparison of Roadmap and Cell Decomposition Methods in Robot Motion Planning, *WSEAS Transactions on Systems and Control*, Vol. 2, Issue 2, 2007, pp. 101-108.

[12] K. Sugihara and J. Smith, Genetic Algorithms for Adaptive Planning of Path and Trajectory of a Mobile Robot in 2D Terrains, *IEICE Transactions on Information and Systems*, Vol.E82-D, No.1, 1999, pp. 309-317.

[13] P. Švec, *Using Methods of Computational Geometry in Robotics*, PhD. Thesis, Brno University of Technology, 2007, 135 pp.

[14] A. Zilouchian and M. Jamshidi, *Intelligent Control Systems Using Soft Computing Methodologies*, CRC Press, Boca Raton, 2001.