Roadmap Methods vs. Cell Decomposition in Robot Motion Planning

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

Abstract: - The task of planning trajectories plays an important role in transportation, robotics, information systems (sending messages), etc. In robot motion planning, the robot should pass around obstacles from a given starting position to a given target position, touching none of them, i.e. the goal is to find a collision-free path from the starting to the target position. Research on path planning has yielded many fundamentally different approaches to the solution of this problem that can be classified as roadmap methods (visibility graph method, Voronoi diagram) and methods based on cell decomposition. Assuming movements only in a restricted number of directions (eight directional or horizontal/vertical) the task, with respect to its combinatorial nature, can be solved by decomposition methods using heuristic techniques. We present drawbacks of this approach (combinatorial explosion, limited granularity and generating infeasible solutions). Then, using the Voronoi diagrams, we need only polynomial time for finding a solution and, choosing a Euclidean or rectilinear metric, it can be adapted to tasks with general or directional-constrained movements.

Key-Words: - motion planning, cell decomposition, roadmap method, visibility graph, 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 [2,6,11,13]. A robot is usually represented by a single point or a circle. There are three basic types of robot motion planning algorithms [9].

The first type is the *potential field method*. The goal has an attractive potential and the obstacles have a repulsive potential. The robot moves in the direction of the gradient of a potential field produced by the goal configuration and the obstacles. Unfortunately, this algorithm often converges to a local minimum in the potential field and therefore we will not deal with it.

The second type is the *cell decomposition method*. Here, the scene is decomposed into cells and the outcome of the search is a sequence of adjacent cells between start and target from which a continuous path can be computed. The square cell decomposition can be used for 8-directional (horizontal, vertical and diagonal) robot motion in the plane with static rectangular obstacles. Unfortunately, this approach has many drawbacks such as combinatorial explosion, limited granularity and generating infeasible solutions as we briefly show in the next paragraph. This approach can be slightly improved using a case-based reasoning procedure [5].

The third type of motion planning algorithm is referred to as a *roadmap method*. The roadmap is built by a set of paths where each path consists of collision free area connections. There are several different methods for developing the roadmap such as visibility graphs and Voronoi diagrams [7]. As these methods do not have the drawbacks of the previously-mentioned ones, we will study them in more detail trying to combine them.

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 [9]:

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 shows a decomposition using rectangular strips. The cells joining the start and target are shaded. The resolution of the decomposition is chosen to get a collision-free path dependent on the sensitivity of controlling our robot's motion.

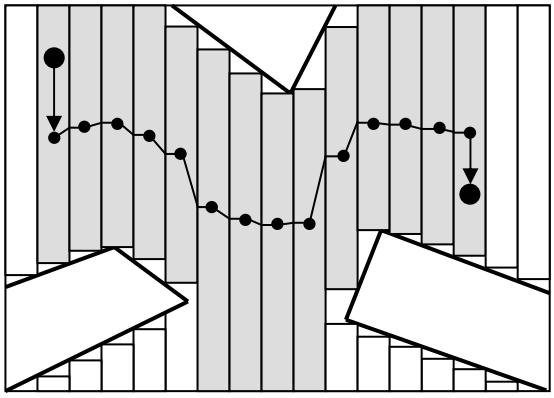

Fig. 1: Vertical strip cell decomposition of the scene with 3 polygonal obstacles.

This approach is not usable for 8-directional motion planning. For movements in 8 directions, the scene decomposition shown in Fig. 2 may be used. Here, all cells have the same square shape.

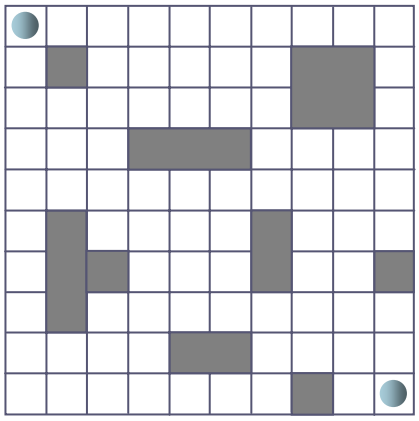

Fig. 2: Grid representation of 2D space with starting and target positions of the robot and static obstacles

Fig. 3 shows one of the possible paths from the starting to the target position.

The cell decomposition-based path planning in 8 directions has the following drawbacks:

- Robot size must be smaller than cell size. In the opposite case, we are not able to determine uniquely the robot position. This decreases the possible range of grid.
- If we use stochastic heuristic techniques (genetic algorithms, simulated annealing, tabu-search, …) for finding trajectories, then their crossover, mutation and neighbourhood operators generate many infeasible solutions (movements out of grid, collisions with obstacles). In Fig. 4 we can see that, although the neighbouring cells are free, the robot cannot move between them without colliding with obstacles.
- Increasing the range of the grid, satisfying the first condition results in combinatorial explosion. Assume m=n (square grid). Then the cardinality of the search space is equal to $8^{2n} = (2^3)^{2n} = 2^{6n}$, which, even for not very high values of m and n, leads to a rather intractable amount of possible paths, for m=n=20, for example, we get $2^{6n} = 2^{120} = (2^{10})^{12} = (1024)^{12} > 10^{36}$ paths, which gives no chance to achieve the optimal solution in a reasonable amount of time.

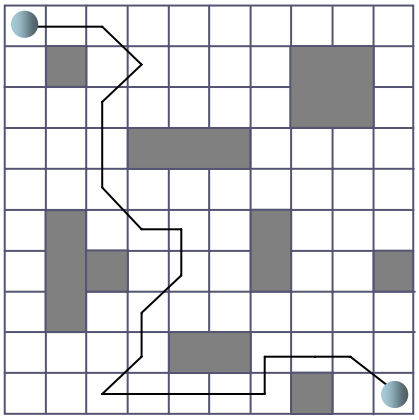

Fig. 3: A path with movements in 8 directions

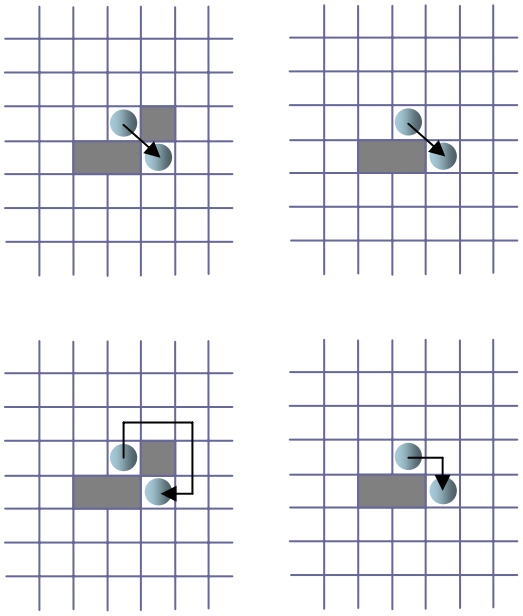

Fig. 4: Collisions with obstacles and collision-free paths

The computation may include a case-based reasoning procedure [5].

Case-based reasoning (CBR) is based on the retrieval and adaptation of old solutions to new problems.

A general CBR cycle may be given by the following steps:

- Retrieve the most similar case or cases;
- Reuse the information and knowledge in that case to solve the problem;
- Revise the proposed solution;
- Retain the parts of this experience likely to be useful for future problem solving.

If, for a given start cell c_s^0 and a given goal cell c_g^0, the case-base does not contain a path leading from c_s^0 to c_g^0, a similar path is retrieved according to the formula

$$P(c_s',\ c_g') = \arg\min \left\{ \begin{array}{l} F\left(P(c_s,\ c_g)\right), \\ d(c_s^0,\ c_s) \le \delta,\ d(c_g^0,\ c_g) \le \delta \end{array} \right\} \quad (1)$$

The problem is that the new solution gained as an adaptation of the most similar case in old solutions can be worse than a new computation that is not based on the stored cases as Fig. 5 shows.

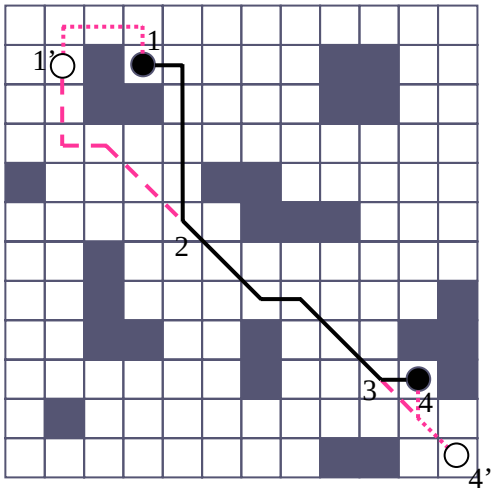

Fig. 5: 1-2-3-4 = old solution;
1'-1-old solution-4-4' = new solution;
1'-2-3-4' = optimal solution

3 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 [1,3,9].

More formally, we can define Voronoi diagrams in mathematical terms. The distance $d(p_i, p_j)$ between two points $p_i = (x_i, y_i)$ and $p_j = (x_j, y_j)$ in the plane can be defined by the Euclidean metric

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad (2)$$

or rectilinear (or Manhattan) metric

$$d(p_i, p_j) = |x_i - x_j| + |y_i - y_j| \quad (3)$$

Definition: Let $P=\{p_1, p_2, \ldots, p_n\} \subset \Re^2$ 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

$$V(p_i) = \left\{ q \in \Re^2 \mid d(q, p_i) \le d(q, p_j) \text{ for } j \neq i \right\} \quad (4)$$

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

$$V = \{V(p_i), \ldots, V(p_n)\} \quad (5)$$

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 $P=\{p_1, p_2, \ldots, p_n\}$ the *generator set* of the Voronoi diagram V.

Using the selected metric, we divide Voronoi diagrams into two classes: the Euclidean and rectilinear Voronoi diagrams. If we use the rectilinear metric for a Voronoi diagram, then, due to the rectilinearity, each straight-line segment of a bisector in the now rectilinear Voronoi diagram will be either horizontal, vertical, or inclined at 45° or 135° to the positive direction of the x-axis [4,10]. This finding suggests using the rectilinear Voronoi diagram for the 8-directional motion planning.

For time complexity considerations it is necessary to know the properties of the Voronoi diagrams and algorithms of their constructions. Therefore, we will briefly summarise them in the next paragraphs.

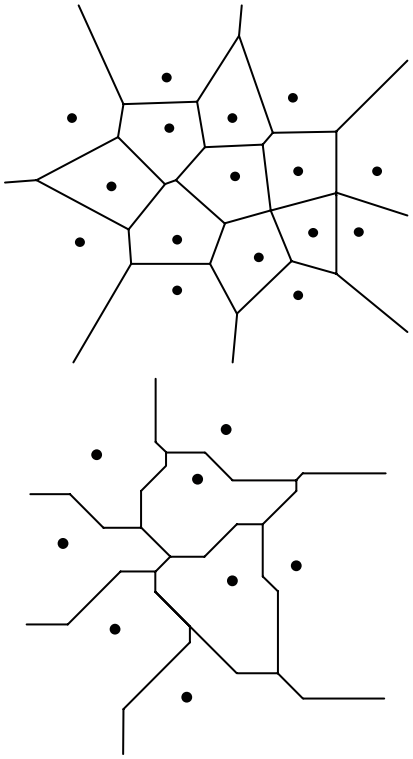

Fig. 6: Examples of the Euclidean and rectilinear Voronoi diagrams

Assume that Voronoi diagrams are non degenerate (no four or more of its Voronoi edges have a common endpoint). Then the following is satisfied [1,2]:

- Every vertex of a Voronoi diagram $V(P)$ is a common intersection of exactly three edges of the diagram.
- A point q is a vertex of $V(P)$ if and only if its *largest empty circle* $C_P(q)$ contains three points on its boundary.
- The bisector between points p_i and p_j defines an edge of $V(P)$ if and only if there is a point q such that $C_P(q)$ contains both p_i and p_j on its boundary but no other point.
- For any q in P, $V(q)$ is convex.
- Voronoi diagram $V(P)$ of P is planar.
- Polygon $V(p_i)$ is unbounded if and only if p_i is a point on the boundary of the convex hull of the set P.
- The number of vertices in the Voronoi diagram of a set of n point sites in the plane is at most $2n–5$ and the number of edges is at most $3n–6$.

The fundamental algorithms and their modifications include the *incremental algorithm*, *random incremental algorithm*, *divide and conquer algorithm* and *plane sweep algorithm* (or *Fortune's algorithm*). More details can be found e.g. in [1,2,3,6]. The time complexity of the incremental algorithm is $O(n^2)$ in the worst case, and $O(n \log n)$ for the other three algorithms.

4 Voronoi Diagrams with Obstacles

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.

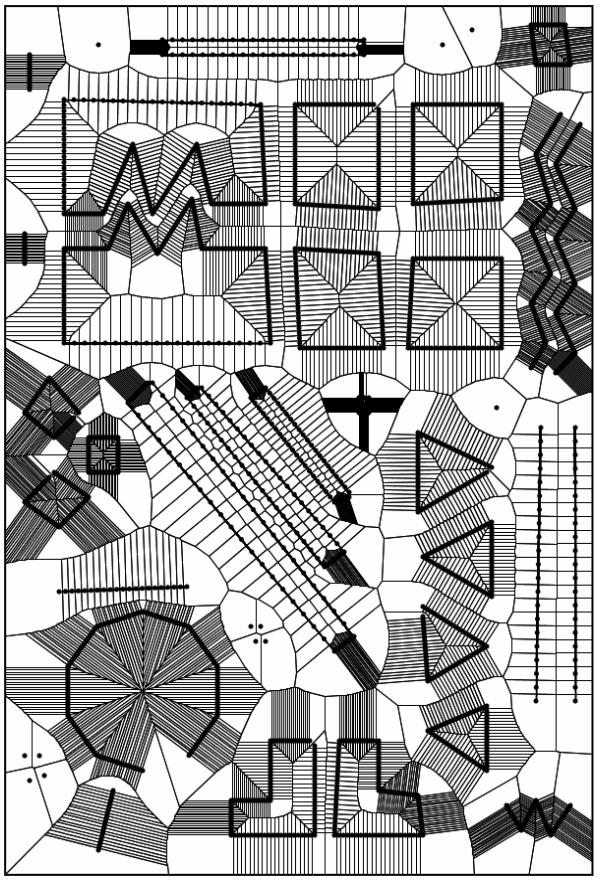

Fig. 7: Voronoi diagram with redundant edges

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.

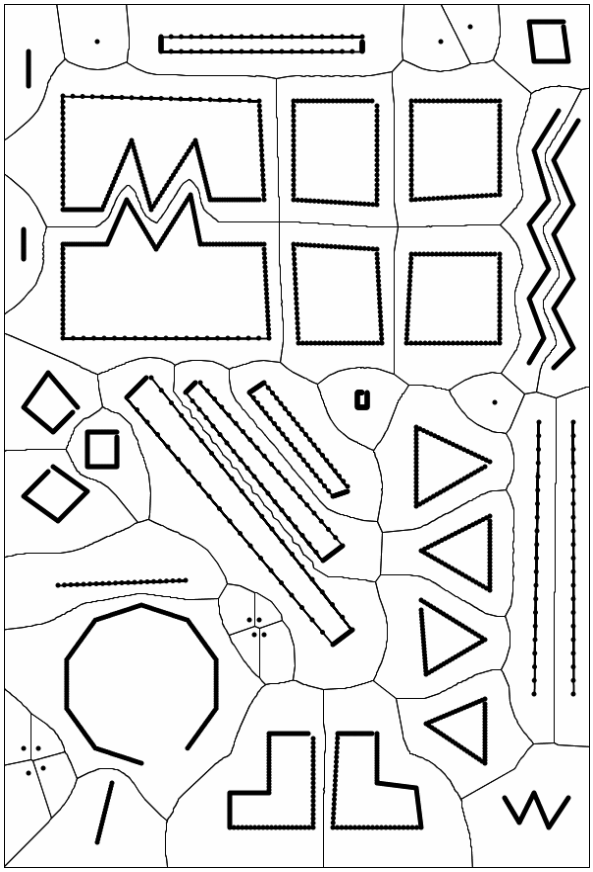

Fig. 8: Voronoi diagram for point, line and polygonal obstacles

An implementation of this approach is described in [12]. 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. 7 and 8 show the Voronoi diagram for point, line and polygonal obstacles with 20 edge segments before and after removing redundant edges.

Fig. 9 shows the same situation as in Fig. 8, but for a 40-line-segment approximation, and it also includes the shortest path between two positions along the Voronoi diagram edges.

This principle can also be used for rectilinear Voronoi diagrams that build maps for 8-directional robot motion planning.

5 Conclusions

In this paper, we compared cell decomposition and roadmap methods with respect to their time complexity and proposed applications of the Voronoi diagrams to general and 8-directional motion planning.

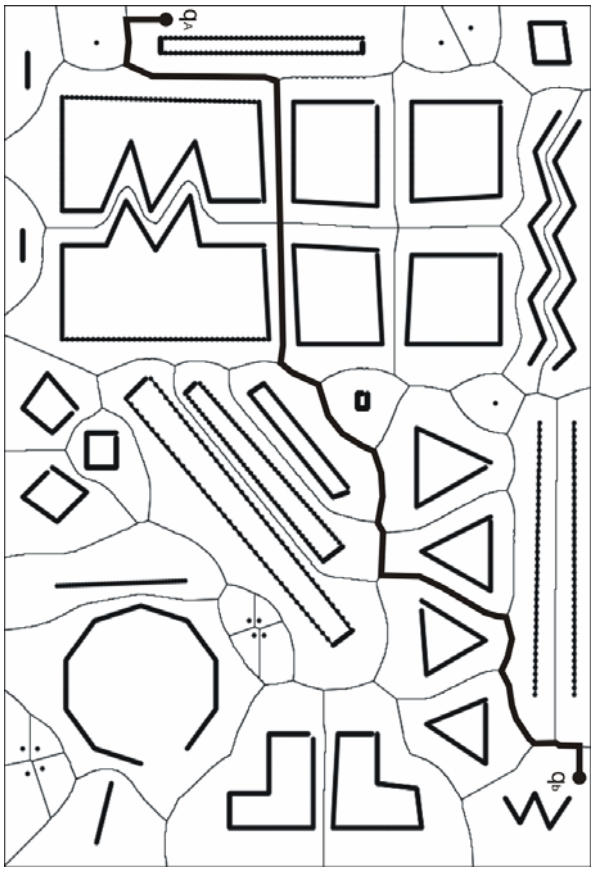

Fig. 9: Voronoi diagram-based path between two positions

As algorithms for constructing the Voronoi diagrams run in polynomial time, the number of their edges is linearly dependent on the number of obstacles and algorithms for searching the shortest paths in graphs are polynomial, too. Since this holds for all additional operations for finding a collision-free path of a robot (replacements, extensions of the rectangular obstacles), the overall time complexity of all the algorithms proposed is polynomial. This approach eliminates all the drawbacks of classical methods (combinatorial explosion, low boundaries for grid representation and generating many infeasible solutions).

In the future, we will try to generalize this approach for cases of more complex shapes of obstacles and movable obstacles.

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] S. Guha and I. Suzuki, Proximity Problems for Points on a Rectilinear Plane with Rectangular Obstacles, *Algorithmica*, Vol.17, 1997, pp. 281-307.

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

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

[10] M. Šeda, Rectilinear Voronoi Diagram-Based Motion Planning in the Plane with Obstacles, *Elektronika* (Poland), No. 8-9, 2004, pp. 24-26.

[11] 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.

[12] P. Švec, Robot Motion Planning Using Computational Geometry (in Czech), *Ph.D. Thesis*. Brno University of Technology, 2005.

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