Robot Motion Planning in Eight Directions

Miloš Šeda and Tomáš Březina

***Abstract*—**In this paper, we investigate the problem of 8-directional robot motion planning where the goal is to find a collision-free path from the starting to the target position in the 2D space containing point and rectangular obstacles. In contrast to the traditional approaches frequently based on decomposition methods combined with heuristic methods, we propose a method for solving this problem using rectilinear Voronoi diagrams whose bisectors are restricted only to horizontal, vertical and diagonal directions.

***Index Terms*—**decomposition methods, case-based reasoning, motion planning, Voronoi diagram**.**

I. Introduction

The task of planning trajectories of a mobile robot, has received considerable attention in the research literature [1]-[3]. This task can be formulated in many ways depending on various conditions, e.g. on the fact whether the terrain contains obstacles, which shape they have, or whether the obstacles are movable. Further constraints may represent knowledge of the scene (complete or partial), the metric under consideration and so on. In this paper, we concentrate on a special case of motion planning in the 2D completely known scene with static point and polygonal obstacles that can be composed from rectangular parts and where possible movements of a robot are reduced only to horizontal, vertical and diagonal directions. This problem is usually solved by heuristics applied to a grid representation of the plane e.g. [2] and can include a case-based reasoning procedure [4], [5]. We will briefly sketch this approach for the case of 8-directional motion using genetic algorithm and discuss its limitations.

Unfortunately the cardinality of the search space of possible paths in the grid has exponential dependence on the granularity of the plane.

Therefore we propose an entirely different approach based on an application of a rectilinear Voronoi diagram using only steps of polynomial time complexity and avoiding all the other drawbacks of the previous approach. In contrast to [6], we will start with the classical Voronoi diagram in the Euclidean plane and later adapt it to the rectilinear case and define a way of replacing its diagonal segments to apply it also to constructing a horizontal/vertical trajectory between the starting and target position.

Manuscript received June 26, 2009. This work has been supported by the Czech Science Foundation GA ČR in the frame of GA ČR 102/09/1668 project "Control Algorithm Design by Means of Evolutionary Approach" and by the Ministry of Education, Youth and Sports of the Czech Republic under research plan MSM 0021630518 "Simulation Modelling of Mechatronic Systems".

M. Šeda and T. Březina work in the Institute of Automation and Computer Science, Faculty of Mechanical Engineering, Brno University of Technology, Technická 2896/2, CZ 616 69 Brno, Czech Republic (e-mail: seda@fme.vutbr.cz, brezina@fme.vutbr.cz).

II. Decomposition Methods

In this section, we will assume 8-directional robot motion in the plane with static rectangular obstacles. In such case, the scene can be easily modelled by a grid and then we only concentrate on navigating the robot from the starting to the target position choosing allowed directions without collisions with obstacles. That means that the path is defined as a sequence of adjacent cells between start and target to given constraints and its total length is expressed by the sum of distances between adjacent cells.

If there are more feasible solutions (i.e. paths between start and target satisfying defined constraints), then we try to determine the paths of a minimal value of a cost function considering both the length and the difficulty of a path. For calculations, it is necessary to assign values to possible directions, e.g. by Fig. 1.

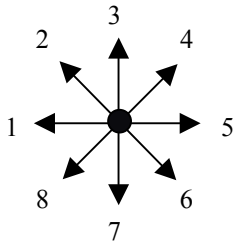

Fig. 1: Valid directions of robot motion

A grid representation of the plane with obstacles is shown in Fig. 2. The robot is represented by a little disk and its starting and target positions are situated in cells in the left upper and right lower corners.

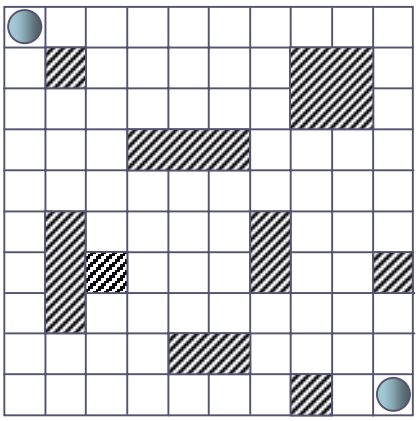

Fig. 2: Grid representation of 2D space with starting and target positions of the robot and static obstacles.

In Fig. 3 a path from a starting to a target position in the configuration from Fig. 2 and its coding is shown.

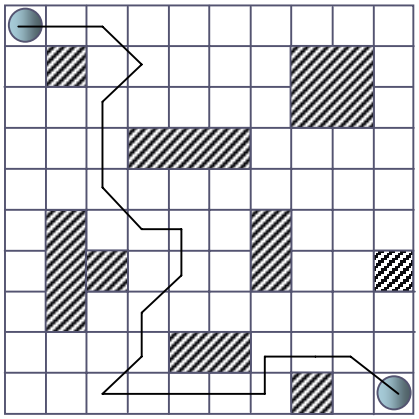

Fig. 3: A path with coding
(5,5,6,8,7,7,6,5,7,8,7,8,5,5,5,5,3,5,5,6)

It is obvious that the problem is of a combinatorial nature and its time complexity depends on the granularity grid and distribution of obstacles. Even if we restrict our considerations to the case where paths have fixed lengths, the complexity remains exponential.

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.

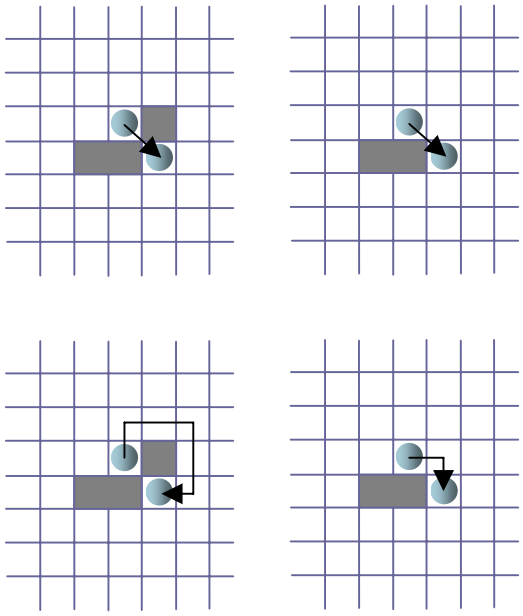

Fig. 4: Collisions with obstacles and collision-free paths

However, besides the exponential explosion, the cell decomposition-based path planning in 8 directions has many other drawbacks as follows:

- 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.

The computation may include a case-based reasoning procedure [4], [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.

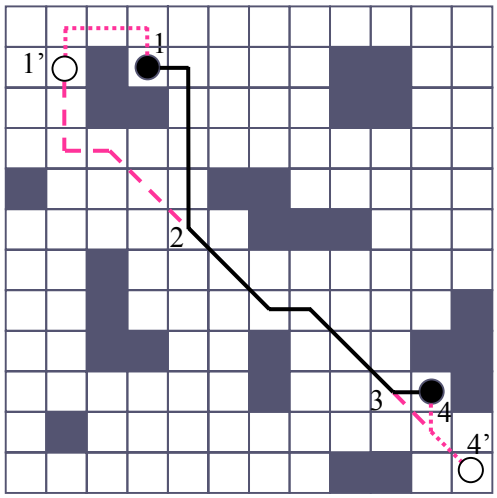

Fig. 5: 1-2-3-4 = old solution;
1'-1-old solution-4-4' = new solution;
1'-2-3-4' = optimal solution

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

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

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.

The approach demonstrated by Fig. 5 has also another drawback that occurs mainly in databases that contain only a few solutions where a new problem cannot be very similar to the solutions stored in a database. It can be partially improved if, in addition to taking into consideration the start and target positions of the old solutions, we first try to investigate the neighbouring cells of the new start and target. If these neighbours intersect some of the paths stored in a database, then these intersections are used for joining to the old solutions. This is shown in Fig. 6.

Robot motion planning algorithms can be modified by considering a robot as a point and enlarging the obstacles in the workspace accordingly. We "add" the size of robot to the obstacles using Minkowski sums [1] and thus the robot is reduced to a point.

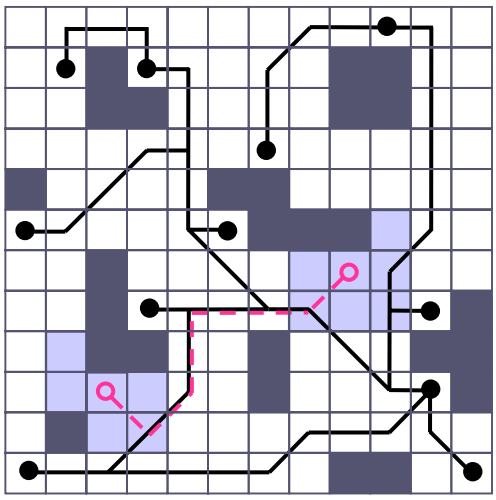

Fig. 6: "Neighbour-intersection" strategy

III. Roadmap Methods

Now we will describe another approach that does not guarantee finding the shortest path but takes into consideration motion safety in the sense of minimising possible collisions with obstacles. It is based on geometric data structures and therefore we define necessary notions.

A Voronoi diagram of a set of points (called *sites*) in the Euclidean 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.

More formally [1], [7], [9]:

Definition 1 Let P be a set of n points in the plane. For two distinct sites $p_i, p_j \in P$, the *dominance* of p_i over p_j is defined as the subset of the plane that is at least as close to p_i as to p_j. Formally,

$$\mathrm{dom}(p_i, p_j)=\{x\in\Re^2 \mid d(x, p_i)\le d(x, p_j)\}, \qquad (2)$$

where d denotes the Euclidean distance. □

Clearly, $\mathrm{dom}(p_i, p_j)$ is a closed half-plane bounded by the perpendicular bisector of p_i and p_j.

Definition 2 *Voronoi region* (or *Voronoi polytope*, *Voronoi cell*, *Voronoi face*, *Dirichlet polygon*, *Thiessen polygon*) of a site $p_i\in P$ is a close or open area $V(p_i)$ of points in the plane such that $p_i\in V(p_i)$ for each p_i, and any point $x\in V(p_i)$ is at least as close to p_i as to any other sites in P (i.e. $V(p_i)$ is the area lying in all of the dominances of p_i over the remaining sites in P).

Formally,

$$V(P_i)=\left\{x\in\Re^2 \;\middle|\; d(x,p_i)\le d(x,q):\forall q\in(P-\{p_i\})\right\}= \\ =\bigcap_{q\in P-\{p_i\}}\mathrm{dom}(p_i,q) \qquad (3)$$

□

Since the Voronoi regions are formed by intersecting $n-1$ half planes, they are convex polygons. Thus the boundary of a region consists of at most $n-1$ *edges* (maximal open straight-line segments) and *vertices* (their endpoints). Points on the boundary of $V(p_i)$ and $V(p_j)$ are equidistant to p_i and p_j.

Definition 3 A *Voronoi diagram* (or *Voronoi tessellation*) for a given set $P=\{p_1, p_2, \dots, p_n\}$ of points (or sites) is a polygonal partition of the plane into Voronoi regions $V(p_1),V(p_2), \dots, V(p_n)$. The vertices of polygons $V(p_i)$ are called the *vertices of the Voronoi diagram*, and their edges are called the *edges of the Voronoi diagram*. A Voronoi diagram is called *degenerate* if four or more of its Voronoi edges have a common endpoint. □

Clearly, each edge of the Voronoi diagram belongs to just two Voronoi regions and

$$V(P)=\bigcup_{p_i\in P}V(P_i) \qquad (4)$$

Consider a disc-shaped robot. If a set of obstacles is only by points, these point obstacles can be considered as sites of a Voronoi diagram and the robot can use for its tour the shortest path along the Voronoi diagram edges that represent passable channels among obstacles, see Fig. 7.

This allows us to reduce the robot motion problem to a graph search problem again. However, in practice, obstacles very often have more general shapes than point ones and thus we must generalise the algorithm for constructing Voronoi diagrams. We take the vertices of polygonal obstacles to be point obstacles, then compute the corresponding Voronoi diagram and, finally, remove from the diagram all the edges intersecting the obstacles.

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.

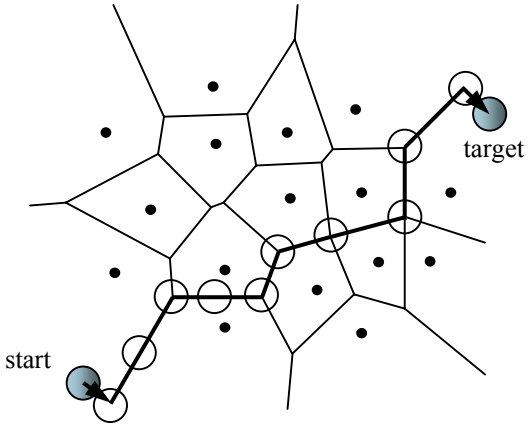

Fig. 7: Motion planning in a scene with point obstacles

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], [7]-[9]:

- 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], [7]-[9]: 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.

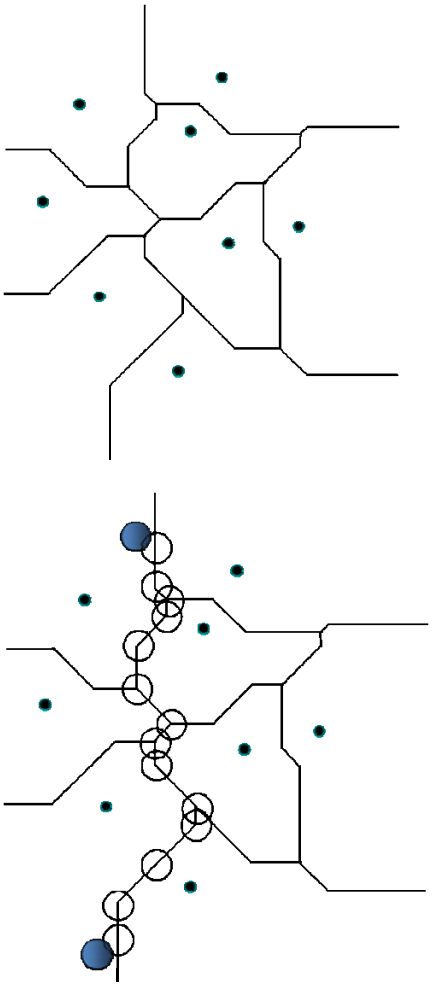

Fig. 8: Motion planning in 8 directions in a scene with point obstacles

Consider *rectilinear* (or *Manhattan*) *metric*

$$d(p_i, p_j) = |x_i - x_j| + |y_i - y_j| \qquad (5)$$

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 [6]. This finding suggests using the rectilinear Voronoi diagram for the 8-directional motion planning. This approach avoids all the drawbacks of classical plane decomposition methods (combinatorial explosion, low boundaries for grid representation and generating many infeasible solutions).

The rectilinear Voronoi diagram can be constructed by a simple modification of the random incremental algorithm for the Euclidean metric.

In Fig. 8 an example of disc-shaped robot motion planning in eight directions for a set of eight point obstacles is shown.

Now consider a more general case of a scene where, besides point obstacles, rectangular obstacles occur. Fig. 9 shows such a scene with the rectilinear Voronoi diagram constructed for point obstacles only. After the construction of the rectilinear Voronoi diagram for point obstacles, we increase the width and height of the rectangular obstacles by the diameter of the disc-shaped robot and a certain small number representing a reserve for finding a collision-free path. For the optimal path between the starting and target position, we search in the graph whose edges create the edges of the "extended" rectangular obstacles and edges of the rectilinear Voronoi diagram without their parts inside the extended obstacles.

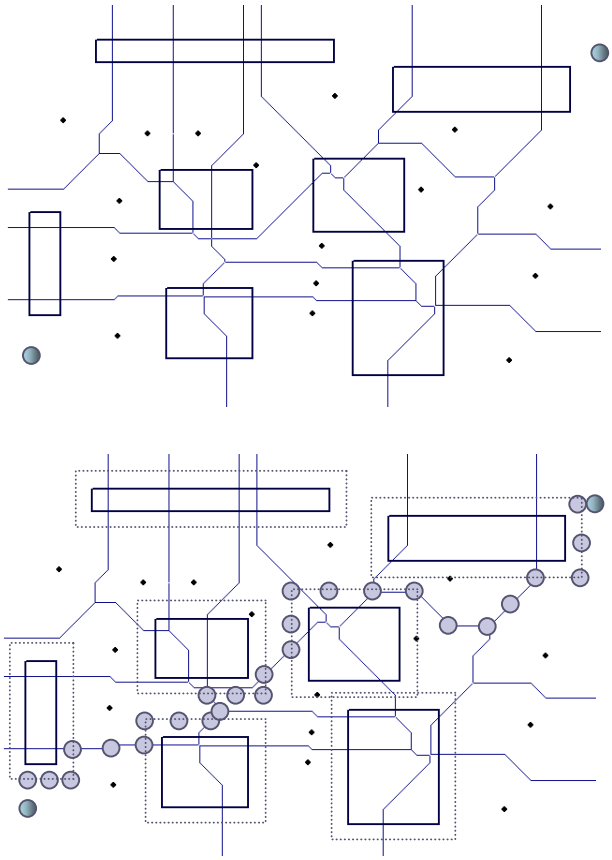

Fig. 9: Motion planning in 8 directions in a scene with point and rectangular obstacles

IV. Conclusion

In this paper, we proposed applications of the Voronoi diagrams to 8-directional motion planning. As algorithms for constructing the Voronoi diagrams run in polynomial time, the number of their edges is linearly dependent on the number of obstacles, algorithms for searching the shortest paths in graphs are also polynomial, and 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 proposed algorithms is

polynomial. This approach avoids all the drawbacks of classical methods (combinatorial explosion, low boundaries for grid representation and generating many infeasible solutions).

In future, we will try to generalize this approach for cases of more complex shapes of obstacles and movable obstacles. Further investigating will also include the case when the environment is totally or partially unknown, varying over time or a combination of both.

REFERENCES

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

[2] 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, vo.1, 1999, pp. 309-317.

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

[4] J. Dvořák and P. Krček, Using Case-Based Reasoning and Graph Searching Algorithms for Mobile Robot Path Planning, in *Proceedings of the 12th International Conference on Soft Computing MENDEL 2006*, Brno, Czech Republic, 2006, pp. 151-156.

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

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

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

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