Dynamic Obstacle Avoidance and Path Planning for an Omnidirectional Robot Using 2-Dimensional Voronoi Diagram and Delaunay Triangulation

This paper is to fulfill IF2211 Algorithm Strategies course assignment, Informatics program, Institut Teknologi Bandung

Dionesius Agung Andika P.
Informatics Department
School of Electrical Engineering and Informatics, Institut Teknologi Bandung
Bandung, Indonesia
13516043@std.stei.itb.ac.id

***Abstract*—this paper proposes and elaborates a new method for a path planning and obstacle avoidance in a dynamic environment for a RoboCup Middle Size League soccer playing robot. It involves the construction of Voronoi diagram and Delaunay triangulation, as well as using divide and conquer algorithm to calculate the best path towards a goal point and simultaneously avoiding incoming obstacles.**

Keywords—Delaunay triangulation; divide and conquer; soccer robot; omnidirectional robot; Voronoi diagram

I. Introduction

MSL or Middle Size League is a robot soccer competition initiated by RoboCup. It is played on an indoor soccer field with goals of reduced size by teams of five fully autonomous soccer playing robots that compete against one another. No human intervention is allowed during a match with the exception of taking robots on and from the field or in case of emergency and unexpected robots behavior.

In order to perceive objects in its world, an MSL robot has camera sensor (either omnivision or frontal). A stream of images from the camera are processed real-time to detect objects such as field, ball, and other robots. Auxilliary sensors can also be used for localization means such as distance or infrared sensor, compass, and encoder from the motors that will turn its wheels.

Fig. 1 MSL Robots from TU Eindhoven competing in the RoboCup.

An important ability in this robot, which are usually omnidirectional, should have is obstacle avoidance. The main obstacle in this robot's world are opponent robots as well as sometimes its teammates.

A known obstacle avoidance algorithm is by calculating velocity and position vectors of both the robot and the obstacle and then returning a velocity vector $\vec{v}$ in which direction the robot should go. An example of this algorithm is explained by Robert L. Williams in [1].

Another approach proposed by the author is by utilizing Voronoi diagram and its corresponding Delauny triangulation. A Voronoi diagram and its corresponding Delaunay triangulation graph can be utilized to plan a path from one point to another point in the field with dynamic obstacles. One advantage of using this method is that the Voronoi diagram is only constructed once during the initialization, thus reducing the computing cost during the gameplay.

By using Voronoi diagram and Delaunay triangulation, position control is used as opposed to velocity control to control the behavior of the robot. This, however, has a little drawback, as the motion generated by the algoritm is a bit less smooth than the obstacle avoidance algorithm in [1].

II. Voronoi Diagram

A. Definition

The partitioning of a plane with n points into convex polygons such that each polygon contains exactly one generating point and every point in a given polygon is closer to its generating point than to any other. A Voronoi diagram is sometimes also known as a Dirichlet tessellation. The cells are called Dirichlet regions, Thiessen polytopes, or Voronoi polygons.

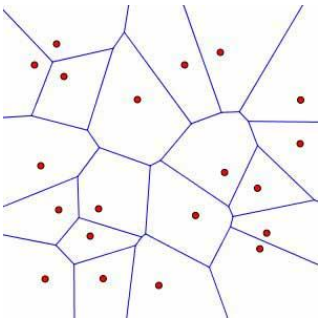

Fig. 2 Voronoi diagram with 20 points.

B. History

The first recorded use of this diagram can be traced back to French mathematician René Descartes in 1644. In 1850, Peter Gustav Lejeune Dirichlet used 2-dimensional and 3-dimensional Voronoi diagrams in his study of quadratic forms [2].

Voronoi diagram is named after the mathematician Georgy Fedosievych Voronoi of the Russian Empire who first defined and studied the general n-dimensional case of this problem. Some equivalent names for this concept are Voronoi polyhedron, Voronoi cell, domain of influence, Voronoi tesselation, and Dirichlet tesselation.

C. Significance in Research and Study Fields

Voronoi diagrams used in geophysics and meteorology to analyze spatially distributed data (e.g. rainfall measurements) are called Thiessen polygons after American meteorologist Alfred H. Thiessen. In condensed matter physics, such tessellations are also known as Wigner-Seitz unit cells. Voronoi tessellations of the reciprocal lattice of momenta are called Brillouin zones. For general lattices in Lie groups, the cells are simply called fundamental domains. In the case of general metric spaces, the cells are often called metric fundamental polygons.

D. Mathematical Properties

The mathematical properties of a Voronoi diagram, assuming that it is set in an Euclidean plane and space, according to [3] and [4] are as follows.

- The dual graph for a Voronoi diagram corresponds to the Delaunay triangulation for the same set of points.
- The closest pair of points corresponds to two adjacent cells in the Voronoi diagram.
- Two points are adjacent on the convex hull if and only if their Voronoi cells share an infinitely long side.
- If the space is a normed space and the distance to each site is attained, each Voronoi cell can be represented as a union of line segments emanating from the sites.
- Under relatively general conditions, Voronoi cells enjoy a certain stability property: a small change in the shapes of the sites, e.g., a change caused by some translation or distortion, yields a small change in the shape of the Voronoi cells. This is the geometric stability of Voronoi diagrams.

III. Delaunay Triangulation

A. Definition

In mathemtics and computational geometry, a Delaunay triangulation is a triangulation $DT(P)$ for a given set P of discrete points in a plane such that no point in P is inside the circumcircle of any triangle in $DT(P)$.

Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation, i.e. they tend to avoid skinny triangles.

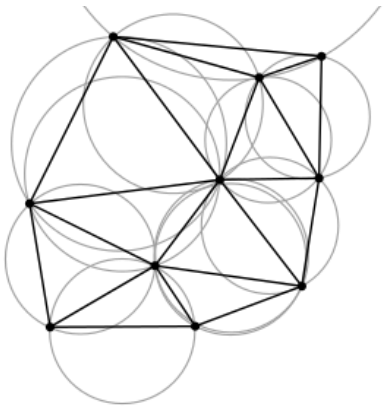

Fig. 3 Delaunay triangulation for a set of 10 points.

B. Mathematical Properties

The mathematical properties of a Delaunay triangulation for n number of points in d-dimension according to [5], [6], [7], and [8] as follows.

- The union of all simplices in the triangulation is the convex hull of the points.
- The Delaunay triangulation contains $O(n^{\lceil d/2 \rceil})$ simplices.
- In the plane, if there are b vertices on the convex hull, then any triangulation of the points has at most $2n - 2 - b$ triangles, plus one exterior face.
- If points are distributed according to a Poisson process in the plane with constant intensity, then each vertex has on average six surrounding triangles.
- In the plane, the Delaunay triangulation maximizes the minimum angle. Compared to any other triangulation of the points, the smallest angle in the Delaunay triangulation is at least as large as the smallest angle in any other. However, the Delaunay triangulation does not necessarily minimize the maximum angle. The Delaunay triangulation also does not necessarily minimize the length of the edges.
- A circle circumscribing any Delaunay triangle does not contain any other input points in its interior.

- If a circle passing through two of the input points doesn't contain any other of them in its interior, then the segment connecting the two points is an edge of a Delaunay triangulation of the given points.
- Each triangle of the Delaunay triangulation of a set of points in d-dimensional spaces corresponds to a facet of convex hull of the projection of the points onto a $(d+1)$-dimensional paraboloid, and vice versa.
- The closest neighbor b to any point p is on an edge bp in the Delaunay triangulation since the nearest neighbor graph is a subgraph of the Delaunay triangulation.
- The Delaunay triangulation is a geometric spanner: the shortest path between two vertices, along Delaunay edges, is known to be no longer than $\frac{4\pi}{3\sqrt{3}}$ times the euclidean distance between them.

C. *Relationship with Voronoi Diagram*

Actually, the Voronoi diagram is just the straight-line dual graph of the Delaunay triangulation, i.e. we can go from the Voronoi diagram to the Delaunay triangulation by drawing in the edges which are perpendicular to the region boundaries and vice versa.

The Delaunay triangulation of a discrete point set P in general position corresponds to the dual graph of the Voronoi diagram for P. Special cases include the existence of three points on a line and four points on circle.

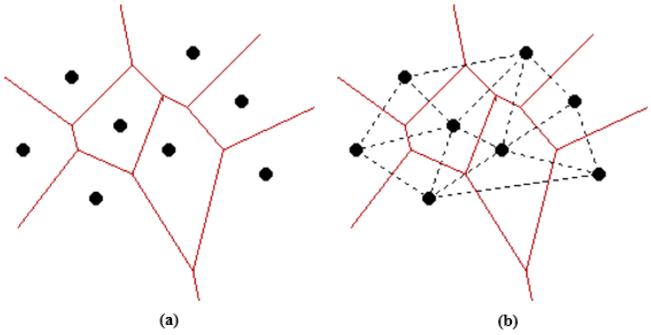

Fig. 4 Voronoi diagram in red lines (a) and its corresponding Delaunay triangulation in dashed lines (b).

IV. Divide and Conquer Strategy

In computer science, divide and conquer is a problem solving method that works by breaking down a problem into smaller sub-problems of the same type, solving the sub-problems independently, then combining the sub-problems' solutions into a whole solution for the original problem [9].

Precisely, divide and conquer algorithm is comprised of 3 main processes:

1. divide: breaking down the problem into smaller sub-problems of the same type,
2. conquer: solving each and every sub-problem recursively, and
3. combine: combining solutions of each sub-problem into one whole solution for the original problem.

A. *Advantages*

The divide and conquer method gives 2 main advantages [9]. Firstly, this method provides a simpler approach for more conceptually complex problems such as the classic Tower of Hanoi problem by recursively reducing the size of the problem until it becomes sub-problems that can be solved trivially.

Secondly, even if the exact solution is known (e.g. sorting, polynomial problems in brute force), divide and conquer method can substantially decrease a problem's complexity. For instance, a problem with $O(n^2)$ time complexity and size n, and combining two sub-problems requires $O(n)$ time, divide and conquer decreases the time complexity into $O(n \log n)$.

B. *Design and Schematics of Divide and Conquer*

Details of divide and conquer is characterized by (1) threshold of sub-problem size n_0, the size of which cannot be broken down anymore, (2) sub-problem size, (3) total number of sub-problems, and (4) the algorithm to combine all sub-problem solutions.

Value of n_0 is usually referred to as base value (where the recursive ends). Sub-problem size is the ratio of the original problem to the sub-problem size, which is usually 2 or, for more complex problems, 4.

Below is a c-style pseudocode for general schematics of divide and conquer.

```
void d_and_c(int n, Type sub_problem) {
  int r;
  int k;

  if (n < n0) {
    // sub-problems are small enough to
be solved
    solve(sub_problem);
  }
  else {
    sub_problem = divide(sub_problem);
    s1 = d_and_c(n/2, sub_problem[0]);
    s2 = d_and_c(n/2, sub_problem[1]);
    combine(s1, s2);
  }
}
```

V. Constructing Voronoi Diagram using Divide and Conquer

A. *Algorithm Description*

In divide and conquer approach for this problem, first, the set of points P is broken down into subsets P_L and P_R of roughly the same size by a dividing line. Then, the Voronoi diagram $Vor(P_L)$ of subset P_L and $Vor(P_R)$ of subset P_R are computed recursively.

The vital part of algorithm consists of finding the split line, and merging $Vor(P_L)$ and $Vor(P_R)$, to obtain $Vor(P)$ of

original set P. If computing the split line and merging of two Voronoi diagrams could carried out in time $O(n)$ then the overall running time is $O(n \log n)$.

Calculation of vertical or horizontal split lines is straightforward during recursion if the points in P are sorted by their x and y coordinates beforehand. Any optimal sorting algorithm such as heap sort does this task in $O(n \log n)$ time.

The merge step involves computing the set of perpendicular bisector s of sets P_L and P_R, i.e. $B(P_L, P_R)$, of all Voronoi edges of $Vor(P)$ that separates the sites in P_L from regions of sites in P_R. The idea of merging based on the fact that the edges of $B(P_L, P_R)$ form a single y-monotone polygonal chain. This can be proved by taking any edge b of $B(P_L, P_R)$ and two points l belongs to L and r belongs to R of two regions adjacent to b. The smaller x-coordinate of l with respect to x-coordinate of r implies that b cannot be horizontal. Thus, stitch together the part of $Vor(P_L)$ to the left of $B(P_L, P_R)$ with the part of $Vor(P_R)$ to the right of $B(P_L, P_R)$ to get $Vor(S)$. Find a starting edge at infinity, and by tracing $B(P_L, P_R)$ through $Vor(P_L)$ and $Vor(P_R)$ in order to construct the polygonal chain $B(P_L, P_R)$.

B. *Support Line Computation*

This algorithm requires convex polygons and calculation of common support line. The algorithm runs in $O(n)$ time. In fact, there has already been an $O(\log n)$ algorithm for finding the common support by Overmars and Leevwen [10]. But since this sub-problem is not a bottleneck of the total time complexity, following $O(n)$ algorithm is quite enough.

VI. Path Planning with Voronoi Diagram and Delaunay Triangulation

A. *Design Motivation*

It is possible to design a path planning with dynamic obstacle using Voronoi diagram and Delaunay triangulation. The field on which the robots are playing can be considered as a 2-dimensional plane with x and y coordinates and a Voronoi diagram can be layered on top of it.

The Voronoi diagram will describe regions on which the objects exist on the field and every point is closest to a particular point p in a predefined subset of points P which lies on the field. The Delaunay triangulation will describe the actual path the robot will take in navigating the field of play during gametime.

B. *Algorithm Design*

Assuming the coordinates of robot and obstacle are known relative to the world, firstly plot the coordinates of starting point, goal point, and positions of robot and obstacle on the world then mark the regions which the four points are on. These regions can either be overlapping or disjoint.

Secondly, calculate the ideal path of the robot if there is no obstacle to avoid. This can be done by generating a graph with Delaunay triangulation sides as the edges and set of points P as the vertices then calculating the shortest path between the starting region and goal region. Many path finding algorithms can do this task easily such as A* algorithm.

Thirdly, iterate every period of time. In every iteration, check whether the current region of the robot and region of obstacle is within a certain threshold. The threshold is defined based on the velocity of both objects.

If the objects are within a certain region (or even overlapping), divert the path of the robot to another region farther from the region of the obstacle. The direction of diversion can be adopted from [1]. Again, it depends on the direction of both the robot and the obstacle as well as their respective velocity vector.

If the objects are beyond a certain region, this means it is safe for the robot to go on its initial course. Repeat this step until the robot reaches the goal region or it switches its state.

C. *Voronoi Diagram Design*

As stated in the rulebook of RoboCup Middle Size League [11], the maximum width and height of the robot is 52×52 centimeters. Draw points on the field 52 centimeters apart from each other. See figure below for details.

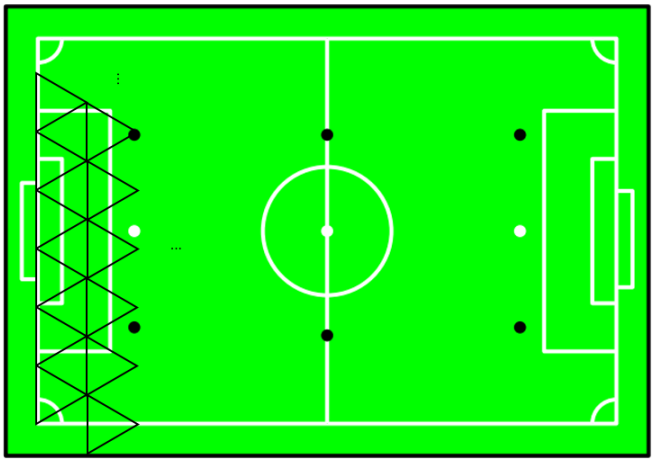

Fig. 5 Each point of the triangles represent 1 point in set P.

Generate the Voronoi diagram of this set of points P. The cells in the said Voronoi diagram will mark the regions that will be used.

Construct a Delaunay triangulation for the Voronoi diagram and it will be used as the discrete path of the robot.

VII. Conclusion

It is possible to construct a robust path planning and dynamic obstacle avoidance algorithm with the help of Voronoi diagram and Delaunay triangulation. Using this technique is also simpler than calculating velocity vector for every iteration.

That being said, however, it is still not so effective to use it as the path will be rough and a high amount of diversion and changes of direction is to be expected.

VIII. Further Works

The author hopes to optimize this algorithm in order to improve the robot movement on the field as well as minimize the amount of changes of direction as it might harm the robot's actuators.

Acknowledgment

The author would like to thank God Almighty for helping him and giving him the strength to pull an all-nighter to finish this paper after an exhausting 3 days of robotics competition in Jakarta, and thus not being able to finish or even do this paper beforehand. The author would also like to thank the entire Dagozilla ITB MSL Robotics team for the support and ideas given to the author during the production process of this paper. Lastly, the author would like to thank his trusty friend Michael and his printer for helping the author in printing this paper so that it can be submitted.

References

[1] R. L. Williams II and J. Wu, "Dynamic Obstacle Avoidance for an Omnidirectional Mobile Robot," *Journal of Robotics*, vol. 2010, p. 14, 2010.

[2] G. L. Dirichlet, "Über die Reduktion der positiven quadratischen Formen mit drei unbestimmten ganzen Zahlen (On the Reduction of Positive Quadratic Forms with Three Indefinite Integers)," *Journal für die reine und angewandte Mathematik (Journal for Pure and Applied Mathematics)*, vol. 40, pp. 209-227, 1850.

[3] D. Reem, "An Algorithm for Computing Voronoi Diagrams of General Generators in General Normed Spaces," in *Proceedings of the sixth International Symposium on Voronoi Diagrams in science and engineering (ISVD 2009)*, Copenhagen, 2009.

[4] D. Reem, "The geometric stability of Voronoi diagrams with respect to small changes of the sites," in *Proceedings of the 27th Annual ACM Symposium on Computational Geometry (SoCG)*, Paris, 2011.

[5] R. Seidel, "The upper bound theorem for polytopes: an easy proof of its asymptotic version," *Computational Geometry*, vol. 5, no. 2, pp. 115-116, 1995.

[6] R. A. Dwyer, "Higher-dimensional Voronoi diagrams in linear expected time," *Discrete and Computational Geometry*, vol. 6, no. 3, pp. 343-367, 1991.

[7] H. Edelsbrunner, T. S. Tan and R. Waupotitsch, "An O(n2 log n) time algorithm for the minmax angle triangulation," *SIAM Journal on Scientific and Statistical Computing*, vol. 13, no. 4, pp. 994-1008, 1992.

[8] J. M. Keil and C. A. Gutwin, "Classes of graphs which approximate the complete euclidean graph," *Discrete and Computational Geometry*, vol. 7, no. 1, pp. 13-28, 1992.

[9] R. Munir, Diktat Kuliah IF2211 Strategi Algoritma, Bandung: Program Studi Teknik Informatika ITB, 2007.

[10] M. H. Overmars and J. v. Leeuwen, "Maintenance of configurations in the plane," *Journal of Computer and System Sciences*, vol. 23, no. 2, pp. 166-204, 1981.

[11] MSL Technical Commitee 1997-2018, Middle Size Robot League Rules and Regulations for 2018, RoboCup MSL, 2017.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Mei 2018

Dionesius Agung Andika Perkasa
13516043