Multi Robot Motion Planning using Compact Voronoi Diagrams

Suyash Yeotikar
M. Engg. Robotics
University of Maryland
College Park, USA

Mohammad Salman
M. Engg. Robotics
University of Maryland
College Park, USA

Sarvesh Thakur
M. Engg. Robotics
University of Maryland
College Park, USA

Abstract—In this paper, we implement Path Planning strategy for Multi Robots using Compact Voronoi Diagrams[1]. Size of Generalized Voronoi diagrams depends upon total vertices of polygons in the workspace. Current paper focuses on implementing Voronoi-like diagram for which size is a function of the number of polygons only with time complexity of O(log n)[1] per each modification. Certain important features of Voronoi diagrams are maintained instead of complete structure for path planning for each robot. Current implementation can be extended to path planning of a swarm in a dynamic environment.

Keywords** - **Path Planning, Voronoi, Deformation Retract, Multi Robots, CVD, Delaunay Triangulation

I. INTRODUCTION

Path Planning is a challenging problem for multi robot applications[9]. For multi robots, each robot needs to maintain a copy of the environment and other robots for safe traversal. Voronoi diagrams are an efficient tool to find the best regions of operations for each robot. It also allows finding a feasible, non-collidable path for each robot[4].

Georgy Voronoi introduced Voronoi Tessellations for the first time in 1868 and since then it has been in various applications throughout. Voronoi tessellations efficiently divide the workspace into smaller regions according to points of interests. Each point of interest is at the center of a sub-region such that all locations inside that region can be traced fastest from this point. The points on this diagram split the map into Voronoi regions. All points in this region are closer to the site present in that region, than to any other site on the map[2]. The points which form the GVD, however, are equidistant to two or more sites, and these lines, therefore, create such Voronoi diagrams. Basically, these lines are perpendicular bisectors between those two sites. These lines are referred to as Voronoi edges.

Voronoi diagrams have wide applications in Mathematics, Computer Science, Biology, and Geography. Since, it is a diagram of n convex polygons, size of Voronoi diagram depends on the total number of vertices as well as the number of polygons in the diagram[1]. There are famous sweep line algorithms like Fortune's algorithm[6] to generate Voronoi diagrams. Points at which Voronoi edges intersect are Voronoi vertices. A type of Voronoi vertices are the junction vertices. These are the points of intersection of three or more Voronoi edges.

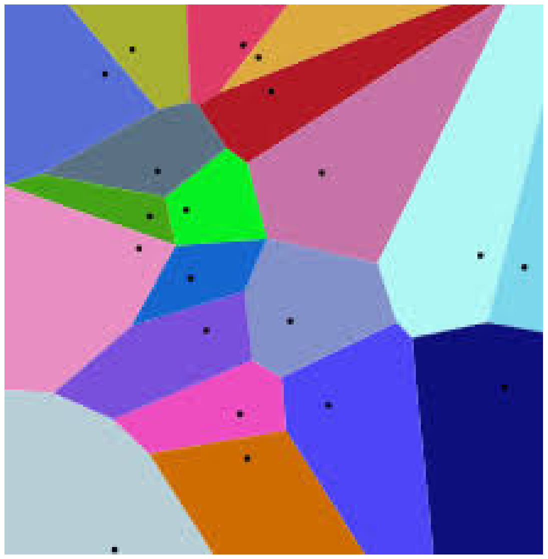
Fig. 1 Voronoi diagram for n point cluster

These points are equidistant from three or more sites (since they essentially lie on three or more Voronoi edges or perpendicular bisectors). Another type of Voronoi vertices is interior vertices which are separate the closest feature pairs of obstacles.[10]

In CVD, Voronoi edges exist between junction vertices, without any interior vertices, whereas in GVD, the edges are made using both types of vertices. For multi-robot path planning, each robot requires a map of the environment to localize itself. GVD is good for this purpose but they take up more size. This is where Compact Voronoi diagrams are useful as they maintain the same quality information but in a lesser size. Concept of Compact Voronoi diagrams can be utilized for Retraction.[1]

II. METHOD

A. *Path Planning using Generalised Voronoi Diagrams*

Our workspace consists of convex obstacles and homogeneous convex robots (red squares). The obstacle space is accounted for the rigid shape of robots by adding Minkowski dilation.

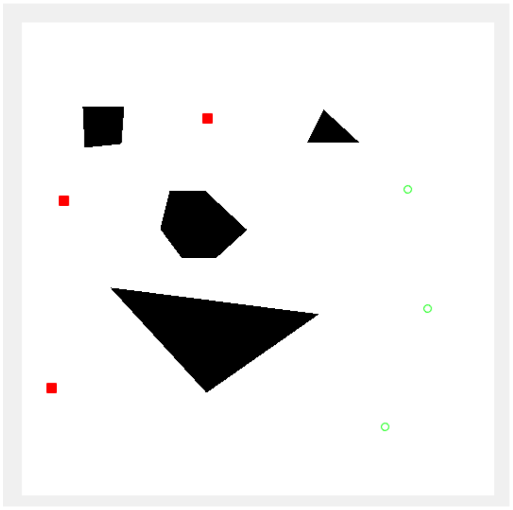

Fig 2. Workspace

In the first run, general Voronoi diagrams are created for each robot. For each robot, other robots are treated as obstacles. To visualize the creation of Voronoi diagrams, imagine each obstacle expanding uniformly. As points on these expanding boundaries start meeting, lines are formed through those points. To implement this, Fortune's algorithm is the most sought-after. But this can be implemented using distance transform[11] of the image of workspace, and then finding the edges using image derivatives. This creates a GVD for each robot in the first instance. This GVD is the safest route for a robot to go at its respective destination without colliding with other robots and obstacles.

With increasing time steps, as other robots move and robot under consideration itself moves, its GVD is recalculated and paths are updated. Since we are doing the same operations at every time stamp, it takes up more time. In next part, when we use CVDs, we will eliminate the need to update the whole CVD by modifying only a part of it.

B. *Path Planning using Deformation Retracts*

In CVDs we use only junction vertices instead of all vertices to maintain the compact version of the GVD.

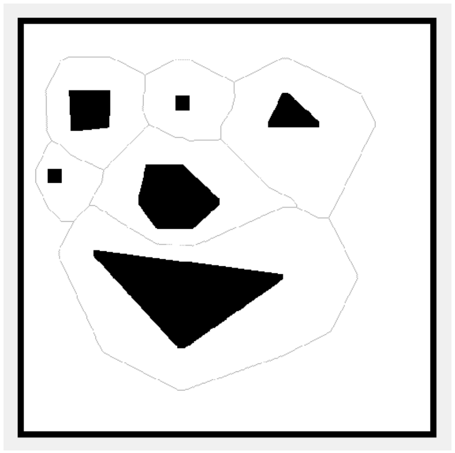

Fig 3. GVD for Robot 1(bottom left robot in fig 1)

The following figure illustrates this idea.

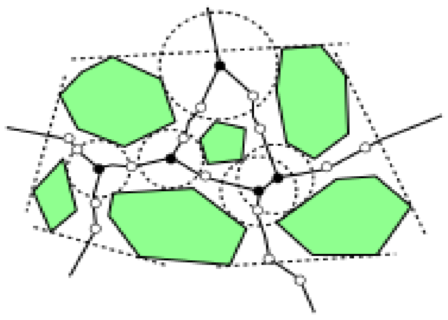

Fig 4. GVD of workspace[1]

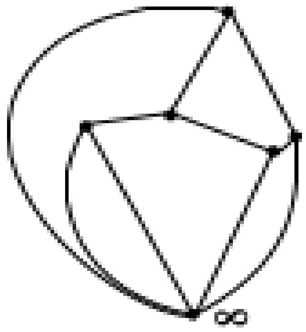

Fig 5. CVD for the above diagram[1]

To keep a check that current CVD is a good approximation of initial GVD, we can maintain local certificates (explained in Previous Studies section) such as junction triangles being locally Delaunay.

When robot moves we keep track of the junction vertices which are the centers of circles touching 3 polygonal obstacles. As these junctions change position, we observe with which polygons the triangle touches next. This way only a small portion of the map is updated instead of modifying the whole map.

Thus, maintaining CVD takes less space in memory than GVD and suits the multi-robot path planning problem.

We use CVD to make sure there are no collisions. If adjacent corridors overlap, it means there is a collision between objects. We also utilize for Retraction Motion Planning by determining narrowest passage between two convex chains in the corridor.

III. PREVIOUS STUDIES AND IMPLEMENTATION

Reference paper[1] proposes Compact Voronoi Diagrams over Generalised Voronoi Diagrams. To understand the results few key terminologies are important to understand:

1. *Local Certificate*

They are certain local conditions which need to be met at all times to guarantee that CVD is a good approximation of GVD. At regular intervals, they are

2. *Junction Points*

Point of intersection of three or more Voronoi edges. Junction points location is critical for maintenance of CVD.

3. *Witness Circles*

A circle drawn from junction point which touches three polygons and does not include any other polygon.

4. *Junction Triangles*

It is a triangle inscribed in Witness Circle. Junction triangles are critical for maintaining the local certificates for compact Voronoi diagrams.

5. *Cocircularity Events*

Circumcircles of two or more junction triangles are coincident. So these two junction triangles share a common edge.

Taking reference from the above definitions, following text explains implementation details in MATLAB:

For finding the junction vertices (fig. 6), we use *branchpoints* morphological operation on the GVD of Minkowski dilated obstacle space. At every junction, we draw a witness circle (fig. 7) tangent to three obstacle polygons. This is followed by creating a Delaunay triangulation (fig. 8) of junction triangles associated with three features (points lying on polygons).

Points within corridor (free space excluding junction triangles and obstacles) space that lie on the line bisecting closest pairs of features (edges or vertices) of two obstacles are stored to find routes that are maximally distant from obstacles.

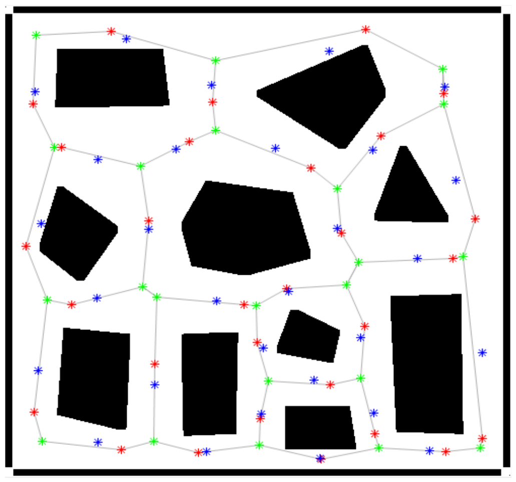
Fig. 6 Junction Vertices(green)

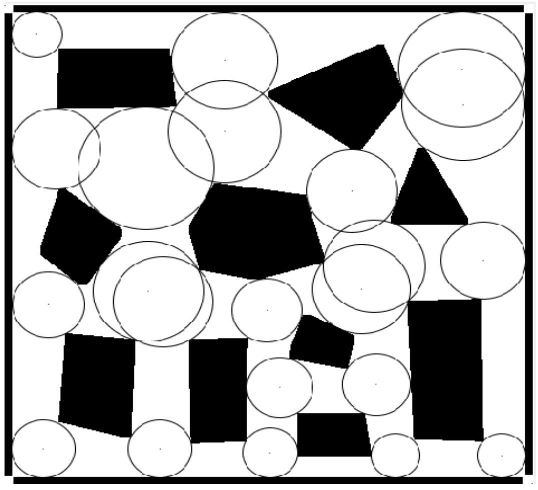
Fig. 7 Witness Circles

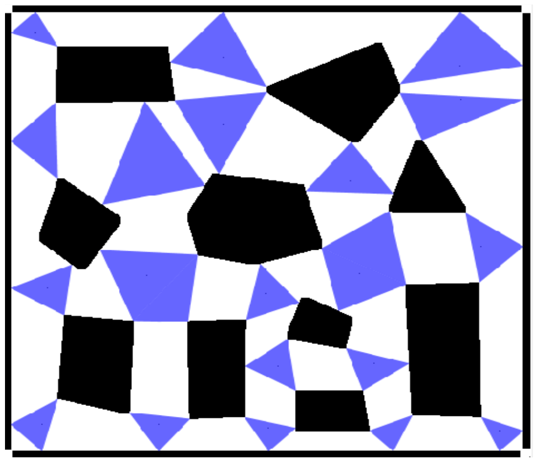
Fig. 8 Delaunay Triangulation

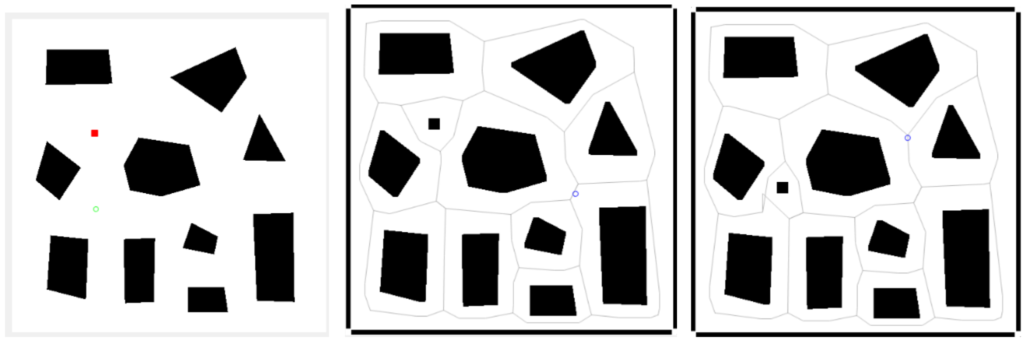

Figure 9. Robot One Progress at start(left), midway(centre) and end(right)

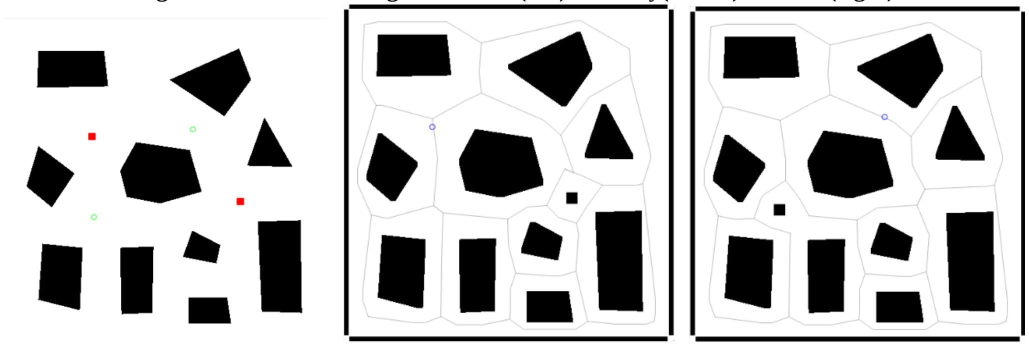

Figure 10. Corresponding progress of Robot two w.r.t Robot one in fig. 9

In general Voronoi diagram, the local certificate involves drawing witness circles and making sure that no other polygon than 3 adjacent polygons coincide with the circle. In compact version, instead of witness circles, junction triangles need to meet this property at all instance.

In case of local certification property not being met, i.e. a co-circularity event arises, we delete the edge that is common to both triangles and join the diagonal formed by other two points on quadrangle. This is more clear from the figure 11.[1] This operation is known as Edge-Flip.

Throughout the motion of robots, at every timestep, we make sure that if any local certificate is compromised, we do edge-flip operation(fig. 11) and maintain the quality of the compact Voronoi diagrams. Using Deformation retract, robots trace the whole map without collision.

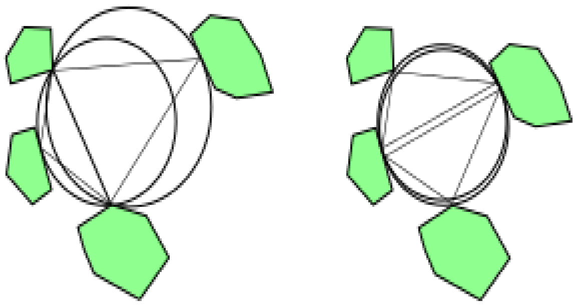

Fig 11. Co-circularity event(Left) can be bypassed by edge-flip operation(Right)[1]

IV. RESULTS

For multi-robot path planning, each robot maintains its copy of Delaunay triangulation and corresponding roadmap. Greedy approach is employed by each robot while traversing the roadmap, thus reducing the go-to-cost. As other robots move, their position is updated on its robot map and collision-free path is found. At every time-step, each map is updated for every robot. This way each robot reaches its destination without collision (fig. 9 and fig. 10). It is important to note that this approach guarantees a feasible path (if it exists) but not necessarily an optimal path.

V. CONCLUSION

In this implementation, we achieved multi-robot path planning using Delaunay triangulations through Compact Voronoi diagrams. There exists many ways for multiagent path planning, with Generalized Voronoi diagrams being one of them. Initially we show path planning using GVD but we have to update the GVDs at each time step for complete path tracing. In the second section we look at path planning using CVDs. They are as efficient as finding routes for each robot but in addition, complete CVDs does not need to be updated at each time, instead a portion of the CVDs is updated when robots move close to each other. This requires less time and size in the memory for path planning than conventional Voronoi diagrams.

Currently we demonstrated path planning in 2D world using CVDs. In future, this model can be used for path planning for 3D world as well.[1]

VI. REFERENCES

1. Guibas L.J., Snoeyink J., Zhang L. (2000) Compact Voronoi Diagrams for Moving Convex Polygons. In: Algorithm Theory - SWAT 2000. SWAT 2000. Lecture Notes in Computer Science, vol 1851. Springer, Berlin, Heidelberg.
2. Pokojski, Wojciech & Pokojska, Paulina. (2018). Voronoi diagrams – inventor, method, applications. Polish Cartographical Review. 50. 141-150. 10.2478/pcr-2018-0009.
3. M. McAllister, D. Kirkpatrick, and J. Snoeyink. A compact piecewise-linear Voronoi diagram for convex sites in the plane. Discrete Comput. Geom., 15:73–105, 1996.
4. Garrido S., Abderrahim M., Moreno L. Path Planning and Navigation using Voronoi Diagram and Fast Marching. IFCA Proceedings Volumes, Volume 39, Issue 15, 2006, Pages 346-351.
5. Drysdale, Robert & Lee, D. (1978). Generalized Voronoi Diagram in the Plane. Proceedings - Annual Allerton Conference on Communication, Control, and Computing. 833-842.
6. Pokojski, Wojciech & Pokojska, Paulina. (2018). Voronoi diagrams – inventor, method, applications. Polish Cartographical Review. 50. 141-150. 10.2478/pcr-2018-0009.J. Teller, Seth. (1993). Visualizing Fortune's Sweepline Algorithm for Planar Voronoi Diagrams.. 393. 10.1145/160985.161169.
7. F. P. Preparata and M. I. Shamos. Computational Geometry: An Introduction. Springer-Verlag, 3rd edition, October 1990.
8. L. J. Guibas, J. S. B. Mitchell, and T. Roos. Voronoi diagrams of moving points in the plane. In 17th Internat. Workshop Graph-Theoret. Concepts Comput. Sci., volume 570 of Lecture Notes Comput. Sci., pages 113–125. Springer-Verlag, 1992.
9. Wagner, Glenn & Choset, Howie. (2011). M*: A Complete Multirobot Path Planning Algorithm with Performance. IEEE International Conference on Intelligent Robots and Systems. 3260-3267. 10.1109/IROS.2011.6095022.
10. Schuster, Megan. (2008). The Largest Empty Circle Problem.
11. https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm