ACCELERON AEROSPACE JOURNAL (AAJ)
E-ISSN: 2583-9942 (Online) | Accessible at www.acceleron.org.in

Volume 5, Issue 2, (pp.1362-1367)
Article ID: AAJ.11.2106-2547
Research Paper
https://doi.org/10.61359/11.2106-2547

Automatic Navigation Map Decomposition for Efficient Reinforcement Learning

Michael Liu*

Granite Bay High School, Granite Bay, CA 95746

Abstract: Safe navigation in an environment with obstacles is a challenging problem. Reinforcement learning (RL) is a promising approach to solve the problem. However, RL often suffers lengthy training time with large maps. This paper presents an algorithm to decompose a map into smaller regions to enable efficient RL. The approach is based on the use of generalized Voronoi Diagrams. The idea is to decompose a large map into a set of regions that are free of crossroads, which greatly reduces the state complexity. In the experiments, the method is applied to decompose six maps. The results show that the approach is effective and efficient.

Table of Contents

1. Introduction

Automatically generating a navigation path that does not collide with any obstacles is a challenging problem. The recent advancements of artificial intelligence (AI) and machine learning (ML) have attracted a lot of interest in using this technology to solve the problem. One promising approach is based on Reinforcement Learning (RL) [1] [2]. The idea is to train a neural network (NN)-based controller by repeatedly simulating the navigation and maximizing a specifically designed reward. For example, if the vehicle collides with an obstacle, a penalty (i.e., negative reward) will be given. If the vehicle reaches its destination, a positive reward will be given. However, the learning process could become lengthy for a large map with many crossroads. This is because the RL explores all possible branches. To make this approach scalable, a hierarchical approach [3] [4] was proposed. The navigation task is decomposed into a number of sub-tasks, which can be learned faster. The task decomposition is usually done by partitioning the map into a set of smaller regions. The map decomposition is often done manually, which is tedious and time-consuming. This paper proposes a method to automate the decomposition.

Contributions

This paper presents an algorithm to automatically decompose a map with obstacles into smaller regions that are free of crossroads to enable efficient RL. The approach is based on the use of generalized Voronoi Diagrams. An example is used to illustrate the algorithm and show that the approach is efficient and can handle complex maps. The paper is organized as follows. In the background section, the relevant background concepts and terminology are introduced. In the next section, the proposed algorithm is described in detail. In the case study section, an example is used to illustrate the steps and intermediate results of the algorithm. In the subsequent section, an overview of related work is presented. Finally, the limitations and conclusion are presented.

2. Background

Navigation planning [5] [6] refers to the process of determining the best route or path to travel from one location to another, considering factors like obstacles (e.g., building, walls), terrain, and environmental conditions, essentially deciding how to get somewhere by considering all relevant information and potential challenges beforehand. Generally, the navigation planning is done in 3D. For the scope of the paper, only 2D maps are considered.

*Granite Bay High School, Granite Bay, CA 95746. **Corresponding Author:** michael.liu.space@gmail.com.

Article History: Received: 04-August-2025 || Revised: 19-August-2025 || Accepted: 19-August-2025 || Published Online: 30-August-2025.

2.1 Deep Reinforcement Learning

Reinforcement learning (RL) [1] [2] is a machine learning technique that trains a machine (e.g., a neural network model) to make decisions through trial and error to maximize cumulative rewards. During RL, the machine interacts with an environment and receives a reward or penalty based on its decision or action. RL algorithms typically require a huge number of interactions with the environment to learn optimal policies.

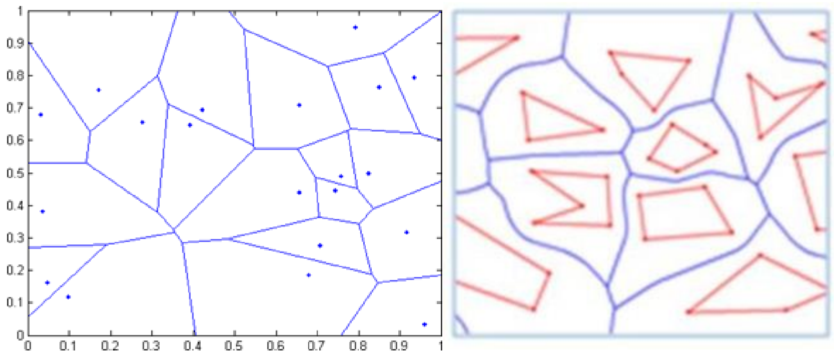
Generalized Voronoi Diagram

The Voronoi diagram [7] [8] is one of the most fundamental structures in computational geometry. It has been used in many fields, including meteorology, social sciences, math, engineering and informatics. Given a finite set of points P on a plane, a Voronoi diagram divides the plane into regions with a set of line segments. Every point on a line has equal distance to the nearest two points in P. Figure 1a shows an example of Voronoi diagram. A generalized Voronoi diagram (GVD) is a Voronoi diagram for a set of polygons, instead of points. Each point on a line of the diagram has equal distance to the two nearest points on the boundary of two polygons. Figure 1b shows an example of GVD. A classic Voronoi diagram consists of straight-line segments, while a GVD consists of curved segments, which are often represented by many vertices. As shown later, we analyze the vertices in a GVD to identify the crossroads in a map.

3. Algorithm

Problem Statement

Given a 2D, top-view, grayscale, area map with obstacles, the problem is to decompose the map into sub-regions, where there is no branches or crossroads within each sub-region.



(a) An example of Voronoi diagram (blue lines) (b) An example of generalized Voronoi di- agram (blue) of a set of polygons (red).

Fig. 1: Examples of Voronoi diagram.

This paper proposes a method to decompose a navigation map with obstacles into a set of regions, where each region contains no crossroads. The method leverages the concept of generalized Voronoi diagram to identify the crossroads formed by the obstacles. Once the crossroads are identified, it creates a set of line segments from each crossroad to partition the map. The partition results in a set of crossroad-free regions.

The detail of the algorithm is described in Algorithm. The input to the algorithm is a map M and two threshold values δ_{th}, h_{th}. The map was pre-processed. During the pre-processing, the obstacles on the map were detected and contour of the obstacles in form of polygons were identified. The first step of the algorithm is to discretize the obstacle contour, and the map boundary lines into a set of points P (Line 2). Second, the Voronoi diagram of the discrete points is generated and optimized (e.g., removing vertices inside polygons) (Line 3). The result Voronoi diagram is represented by a set of connected vertices V. Third, the crossroad vertices $Q \subset V$ are identified. They are Voronoi diagram vertices with 3 or more adjacent vertices (Line 4-9). Finally, the decomposition (partition) is computed. A decomposition is a set ϕ of line segments, whose endpoints are a crossroad vertex and a nearest obstacle (or boundary) point. Thus, the nearest obstacle contour or boundary points of each crossroad vertex need to be identified. This is done through an iteration of finding a new nearest point. Given a point q and a set P of points, the function $NearestPoint$ returns the point $p \in P$ closest to q and the corresponding distance d' (Line 14). Since by definition a Voronoi vertex has equal distance to the nearest obstacle or boundary, the set of nearest points are well-defined and non-empty. In practice, the distance between the crossroad vertex q and the nearest points S may be not the same. The threshold δ_{th} is introduced so that the distance allows a tolerance (Line 18).

Since the obstacle contour is discretized into points, two close points may have similar distance to the crossroad vertex. The distance of the newly identified nearest point p to the previously identified nearest points S is computed using the function $NearestPoint$ (Line 21). To avoid two close points to be included in the set of nearest points, the threshold h_{th} is introduced to make sure that any newly identified nearest point is far enough from any previously identified nearest points (Line 22). The set of lines connecting each of the nearest points and the crossroad vertex forms the final decomposition of the map (Line 24-25).

Algorithm 1. An algorithm to decompose a navigation map

```
1: procedure DECOMPOSE($M, \delta_{th}, h_{th}$)
2:     $P \leftarrow Points(M)$
3:     $V \leftarrow GVD(P)$
4:     $Q \leftarrow \emptyset$
5:     for each $v \in V$ do
6:         if $Neighbor(v) \geq 3$ then
7:             $Q \leftarrow Q \cup \{v\}$
8:         end if
9:     end for
10:    $\phi \leftarrow \emptyset$
11:    for each $q \in Q$ do
12:        $S \leftarrow \emptyset$
13:        $\delta \leftarrow 0$
14:        $(p, d') \leftarrow NearestPoint(q, P)$ 15:
           $S \leftarrow S \cup \{p\}$
16:        $L \leftarrow Line(q, p)$
17:        $\phi \leftarrow \phi \cup L$
18:        while $\delta < \delta_{th}$ do
19:            $(p, d) \leftarrow NearestPoint(q, P \setminus S)$
20:            $\delta \leftarrow d - d'$
21:            $(u, h) \leftarrow NearestPoint(p, S)$
22:            if $h \geq h_{th} \wedge \delta < \delta_{th}$ then
23:                $S \leftarrow S \cup \{p\}$
24:                $L \leftarrow Line(q, p)$
25:                $\phi \leftarrow \phi \cup L$
26:            end if
27:        end while
28:    end for
29:    return $\phi$
30: end procedure
```

To demonstrate the proposed method, it is applied to decompose a map. It is a 1000x1000 image with obstacles, as shown in Figure 11. The map contains 13 disconnected irregular-shaped obstacles (black blocks). During the pre-processing, the contours of the obstacles and the map boundary were identified. The discretized obstacles contours and boundary are shown in Figure 12. Note that all obstacle shapes are approximated by triangles. Then the optimized generalized Voronoi diagram were created (Figure 13). Note that all vertices of the Voronoi diagram are connected. The vertices with 3 or more neighbors are identified as the crossroad vertices (Figure 14). Then for each crossroad vertex, the nearest contour or boundary points are identified. The partition is formed by connecting the crossroad vertex and its corresponding nearest points (Figure 15). Finally, the decomposition is mapped to the original map (Figure 16).

4. Experiments

The proposed method was tested on six maps, shown in Figure 17. They include obstacles of different shapes, such as pentagon and circle. Note that some decomposition lines may seem not reach the map boundary. This is because the drawing area is slightly larger than the actual map. The statistics of the experiments is presented in Table. In the obstacle counting, each rectangular block in the connected blocks is considered as one obstacle. The vertex count refers to the number of crossroad vertices. The region count refers to the number of decomposed regions. The CPU time is in seconds. The experiments were performed on an AMD EPYC 7601 2.7GHz 32-Core processor.

Table.1 The Experiment Statistics

Maps	Obstacles	Vertices	Regions	CPU Time
1	27	13	27	1.402
2	11	16	52	0.392
3	14	19	58	0.284
4	15	17	52	0.363
5	13	17	60	0.377
6	11	20	62	0.495

Overall, the experiments show that the proposed method can often complete the decomposition of the maps in less than a second. The final decomposed regions do not contain any crossroads.

5. Related Work

The exact or approximate cell decomposition techniques used in robotic path planning divide the map into simple connected regions based on the shape of the obstacles and boundary. This often results in many small regions. This is not ideal for RL. The DEACCON (Decomposition of Environments for the Creation of Convex-region Navigation-meshes) algorithm decomposes a map into a set of convex polygons. It starts with a set of squares in the navigable areas. Then each square can grow to its maximum convex shape before encountering an obstacle. This approach works for convex regions. It does not handle non-convex regions. For game maps, a decomposition method was proposed to accelerate the pathfinding of real-time game AI. It uses a flooding algorithm to identify the *choke points* of the map, which are pairs of closest obstacle points. However, the crossroads are not necessarily narrower than the pathways. The resulting regions may still contain crossroads. This will complicate RL.

6. Limitations

The approach has some limitations. Obstacle detection relies on enclosed areas, which effectively captures large obstacles but may fail to detect small, tall structures such as flagpoles. Conversely, a fenced ranch might be misclassified as a large building due to its large, enclosed boundary. Additionally, the irregular shapes of obstacles may lead to the generation of multiple closely spaced crossroad vertices in the Voronoi diagram. In reality, these vertices correspond to a single actual crossroad on the map. This may result in unnecessarily small sub-regions.

Check for updates

7. Future Work

Although some preliminary experiments are performed to verify the proposed method, more diverse and realistic maps are needed to demonstrate its general applicability. It is an interesting direction to study whether the method has any limitation in map geometry. Another potential direction is to automate the determination of the parameters (e.g., the threshold values) used in the algorithm. Currently, they are manually defined and often require a few iterations to set them properly.

8. Conclusion

This paper presented a VD-based approach to decompose a navigation map into crossroad-free regions to accelerate RL. The preliminary experiments show that the method is effective and efficient.

9. References

[1] Sutton, R., & Barto, A. (1998). Reinforcement learning: An introduction. IEEE Transactions on Neural Networks, 9(5), 1054–1054.
[2] Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. IEEE Signal Processing Magazine, 34(6), 26–38.
[3] Dietterich, T. G. (2000). Hierarchical reinforcement learning with the MAXQ value function decomposition. Journal of Artificial Intelligence Research, 13(1), 227–303.
[4] Jeni, L. A., Istenes, Z., Korondi, P., & Hashimoto, H. (2007). Hierarchical reinforcement learning for robot navigation using the intelligent space concept. In Proceedings of the 2007 11th International Conference on Intelligent Engineering Systems (pp. 149–153). IEEE.
[5] LaValle, S. M. (2006). Bibliography. Cambridge University Press.
[6] Latombe, J.-C. (1991). Robot motion planning. Kluwer Academic Publishers.
[7] Aurenhammer, F. (1991). Voronoi diagrams—A survey of a fundamental geometric data structure. ACM Computing Surveys, 23(3), 345–405. https://doi.org/10.1145/116873.116880
[8] Okabe, A., Boots, B., & Sugihara, K. (1992). Spatial tessellations: Concepts and applications of Voronoi diagrams. John Wiley & Sons, Inc.
[9] Zhu, D., & Latombe, J.-C. (1991). New heuristic algorithms for efficient hierarchical path planning. IEEE Transactions on Robotics and Automation, 7(1), 9–20.
[10] Kedem, K., & Sharir, M. (1990). An efficient motion-planning algorithm for a convex polygonal object in two-dimensional polygonal space. Discrete & Computational Geometry, 5(1), 43–75.
[11] Avnaim, F., Boissonnat, J., & Faverjon, B. (1988). A practical exact motion planning algorithm for polygonal objects amidst polygonal obstacles. In Proceedings of the 1988 IEEE International Conference on Robotics and Automation (Vol. 3, pp. 1656–1661). IEEE.
[12] Hale, D. H., Youngblood, G. M., & Dixit, P. N. (2008). Automatically-generated convex region decomposition for real-time spatial agent navigation in virtual worlds. In Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE'08) (pp. 173–178). AAAI Press.
[13] Halldórsson, K., & Björnsson, Y. (2015). Automated decomposition of game maps. In Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 11(1), 122–127.

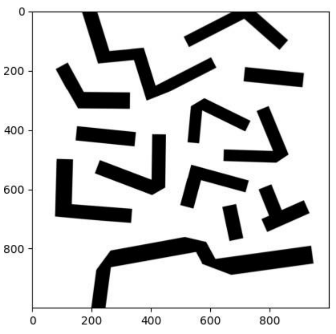
10. Conflict of Interest

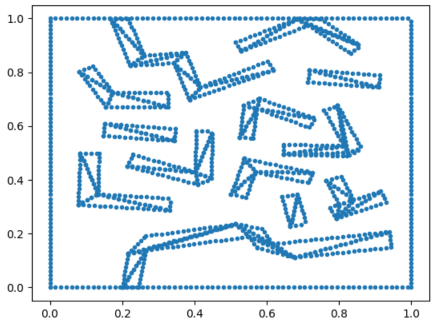
The author declares no competing conflict of interest.

11. Funding

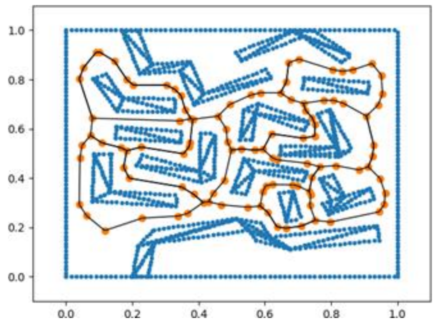
No funding was issued for this research.

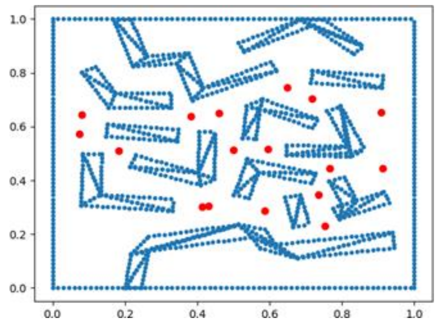
12. Appendix



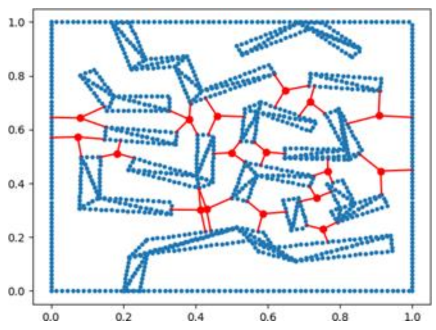


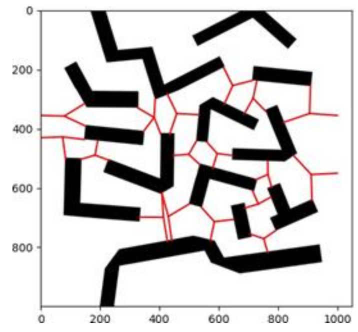
(a) The navigation map with obstacles (black blocks). (b) The discretized obstacle contours and boundary.





(c) The generalized Voronoi diagram after optimization. (b)The crossroad vertices of the generalized Voronoi diagram.





(e) The decomposition of the generalized Voronoi diagram. (f) The final decomposition of the map

Figure 2: Ilustration of the key steps of the algorithm applied to the use case.