OBSTACLE AVOIDANCE FOR AN UNMANNED SURFACE VEHICLE (USV)

Submitted by

Pranay Shirodkar

A0124257J

Department of Mechanical Engineering

National University of Singapore

Session 2016/2017

ME3000 Independent Study Report

Summary

This independent study achieves obstacle avoidance by implementing a path planning algorithm that generates collision free paths for the Unmanned Surface Vehicle (USV) to travel, enabling it to navigate over water bodies independently. The obstacle avoidance algorithm combines Voronoi diagram partitioning with Dijkstra's algorithm to find the best path for the USV to travel from its start position to the desired destination.

Converting the obstacle map into a Voronoi diagram constraints the USV to travel only on specific collision free regions of the obstacle map. This effectively reduces the number of valid nodes that the USV can traverse. Thereafter, the Dijkstra algorithm runs very rapidly and efficiently as it has far fewer nodes to check. The Dijkstra algorithm generates the shortest path from the start node to the goal node under the constraints set by the Voronoi diagram. The path generated by this technique always maintains sufficient distance from obstacles, ensuring a safe path for the USV to navigate.

Simulations were conducted to test the effectiveness of this obstacle avoidance algorithm on three different obstacle maps, for different map sizes. The results were analysed and compared with the results of a straight forward A* path planning algorithm in identical conditions. It is found that the implemented algorithm performs better than the A* algorithm. It is capable of generating the path more rapidly, and generates safer paths that maintain a significant distance from obstacles. Recommendations for further improvement to the algorithm are included at the end of the report.

Table of Contents

List of Figures and Tables

1 Introduction

Recent technological advancements have enabled the development of Unmanned Surface Vehicles (USVs), autonomous crafts that operate on the water surface without on-board crew. USVs are becoming increasingly relevant and important to oceanographers who wish to study the ocean, or to military personnel conducting maritime surveillance and reconnaissance activities [1]. For a USV to operate autonomously, it must be able to avoid determine the most suitable path for getting to its destination without colliding with obstacles. For a USV to operate autonomously, it must be able to determine the most suitable path for getting to a destination without colliding into any obstacles. This would avoid damage to the structure of the USV and ensure that it has a long operational life. Achieving this capability with real-time functionality serves as the motivation for this project.

1.1 Objective

The objective of this project is to develop the necessary software and algorithms that would equip the USV with the capability to plan appropriate paths for it to navigate from its position to a destination.

The algorithm should be robust, ensuring that the path planning capability can be applied on various obstacles maps generated from different environments. The path generated by the algorithm should be safe for the USV to traverse, maintaining a minimum distance from obstacles and avoiding unnecessarily dangerous scenarios or circumstances. Lastly, the path should be generated real-time, ensuring that the

USV can run smoothly, avoiding a situation where the USV fails to react to an obstacle due to a slow avoidance algorithm.

1.2 Scope

Research is conducted into path planning and obstacle avoidance techniques. Popular algorithms are considered and discussed in the literature review section, and the most suitable approach for our application is selected. The methodology is the core of this project, and it will cover how Voronoi diagrams and Dijkstra's algorithm are combined and utilised to achieve a robust obstacle avoidance method for this application.

In the results section, the simulation tests of the algorithm are analysed and compared with the popular A* algorithm to validate the effectiveness of the obstacle avoidance technique employed in this report. Finally, recommendations for further improvement are discussed.

2 Literature Survey

2.1 A* and Dijkstra Algorithm

A literature review was prepared to determine the best way to approach an obstacle avoidance algorithm for the USV.

A* algorithm and Dijkstra's algorithm are two established and popular path planning techniques that are regularly utilised. The A* algorithm begins by examining the start node. Examining a node involves visiting nodes that are adjacent to it and adding these nodes into a list of to-be-examined nodes [2]. The

nodes in this list are ordered from lowest cost to highest cost, according to Equation (1), where f(n) represents the total cost of that particular node [3].

$$f(n) = g(n) + h(n) \quad (1)$$

In every iteration, the node with the lowest cost in the list is examined, by visiting every node that is adjacent to it. This procedure is repeated until the algorithm examines the goal or destination node. In equation (1), g(n) represents the cost of travelling from the start node to the current node, and h(n) estimates the cost of travelling from the current node to the goal node. The value or formula for h(n), the heuristic, depends on the situation or application in question. In the case of our application, the heuristic is defined by the Euclidean distance from the current node to the goal node.

Dijkstra's algorithm is in fact a special case of the A* algorithm, when the heuristic, h(n) is equal to zero. This means that Dijkstra's algorithm only computes the cost of a node based on its distance to the start node.

2.2 Voronoi Diagrams

One problem with the conventional A* and Dijkstra path planning algorithms is time complexity. The time complexity of these algorithms is directly proportional to the number of nodes in the map and the total number of edges between nodes. This poses a challenge because the algorithm can take too long to compute the optimal path for the USV to traverse. One way to tackle this problem, is by pre-processing the obstacle map to reduce the number of edges and nodes that are considered by the A* and Dijkstra algorithm. This can be achieved by converting the obstacle map into a Voronoi diagram.

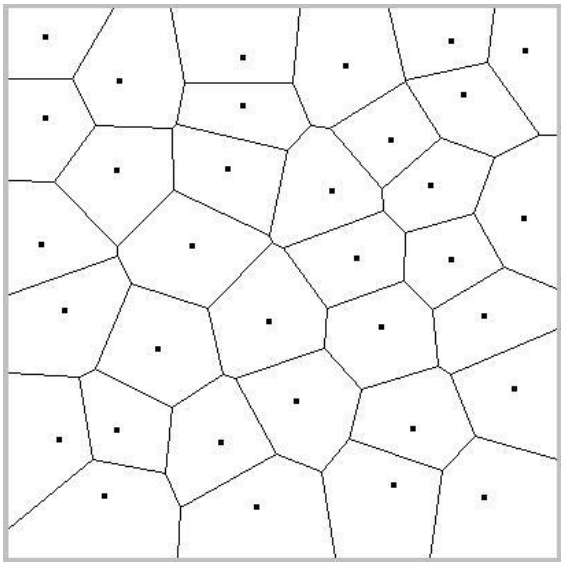

Figure 1: Example of a Voronoi diagram

A Voronoi diagram is a plane that has been partitioned into regions based on the distance from a set of points called seeds on the plane. In Figure 1, these seeds are the black dots. Every seed has a surrounding region of white consisting of all points that are closer to it than any other seed [4]. The region boundaries are the black lines in Figure 1. Using the seeds as obstacles and the region boundaries as the available nodes and edges, the number of nodes and edges in the obstacle map can be significantly reduced, thus minimising the amount of time a path planning algorithm takes to find a path from the start node to the goal node.

In addition, using the region boundaries of a Voronoi diagram as the only paths that are traversable by a USV ensures that the maximum possible distance from all obstacles is maintained throughout navigation, guaranteeing that the USV remains away from obstacles at all times.

As a result of the above investigation, Voronoi diagram partitioning is selected to pre-process the obstacle map in our implementation before path planning is

executed by the Dijkstra's algorithm. This sequence of steps is selected for our particular application and discussed in the methods section [5]. Furthermore, the results of this approach are compared with the results from the A* algorithm, to demonstrate that the selected approach outperforms the A* algorithm. This validates and justifies the obstacle avoidance algorithm recommended in this report.

3 Methodology

In this chapter, the obstacle avoidance algorithm employed will be broken down and presented in the following order: 1) Input Obstacle Map, 2) Voronoi Diagram, and 3) Dijkstra's Algorithm. In each section, the approach will be discussed and explained.

3.1 Input Obstacle Map

To illustrate the methodology, a simple obstacle map is used in this section.

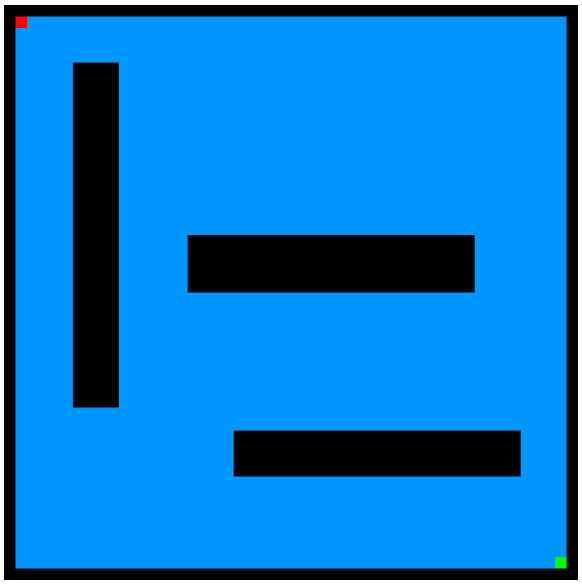

Figure 2: Sparse obstacle map

The map shown in Figure 2 represents a 50 metre by 50 metre grid, containing three big obstacles. The black sections of the Figure are the obstacles and the blue sections are the traversable or available nodes that represent the water surface around the USV. The boundaries of the map have also been marked as an obstacle, as it is desirable to treat this situation as an isolated task and constraint the planned path to remain within the map boundaries. The red point in the top left corner of the map is the starting node of the USV, and the green point in the bottom right corner of the map is the goal node.

3.2 Voronoi Diagram

Figure 3: Voronoi diagram of sparse obstacle map

The Voronoi diagram of the obstacle map is generated by iterating through every non-obstacle node in the map and determining the minimum distance from this node to every obstacle. If this node is equally close to two or more obstacles it is considered a Voronoi edge and is traversable. However, if the node is closest to one obstacle, it is not considered a Voronoi edge, and its colour is converted to grey, as

shown in Figure 3, to represent that it is no longer available for the USV to travel through. An exception is made for the start and goal node. If any node is found to be nearest to the start or goal node only, it is considered to be a Voronoi edge [6].

In addition, a minimum safety distance can be implemented in this step of the algorithm for the safety of the USV during navigation. Any node that is closer than three metres to any obstacle is automatically eliminated from being a Voronoi edge as it would be too dangerous to manoeuvre the USV through that location at such close proximity to obstacles.

The nodes that remain for consideration in the path planning phase are left as blue in Figure 3. It is clear that the number of blue nodes has decreased significantly after the Voronoi diagram implementation. Furthermore, the blue nodes that remain are made up entirely of nodes that will be perfectly safe for the USV to travel through, as they maintain sufficient distance from all obstacles.

3.3 Dijkstra's Algorithm

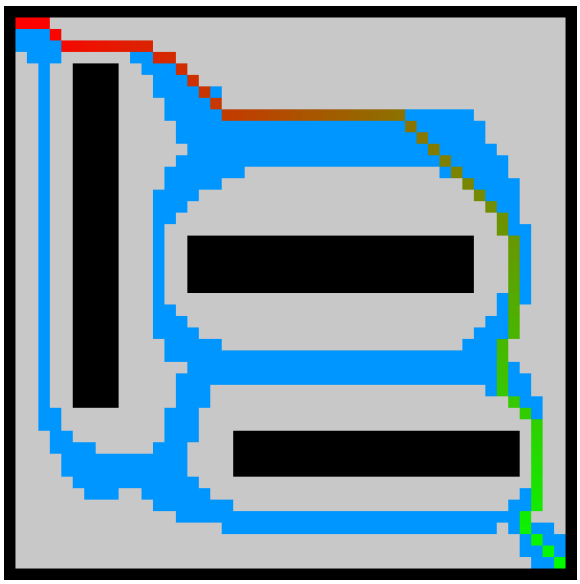

Figure 4: Resultant path after Dijkstra's algorithm

In the final stage, Dijkstra's algorithm can be applied upon the remaining traversable nodes to determine the shortest path from the start to the goal node. The resultant path found can be seen going from the red colour node to the green colour node in Figure 4. The path successfully travels from the start node to the goal node without colliding into any obstacles. This serves as a proof of concept for the effectiveness of the algorithm in achieving obstacle avoidance.

4 Results

4.1 Execution Time Analysis

The results of the implemented Voronoi-Dijkstra algorithm are analysed based upon the execution time of the algorithm, which is the time it takes to generate the resultant path found. The results are analysed for the obstacle map shown above, and for two addition maps shown in Figure 5: a jetty-like map and a dense obstacle map.

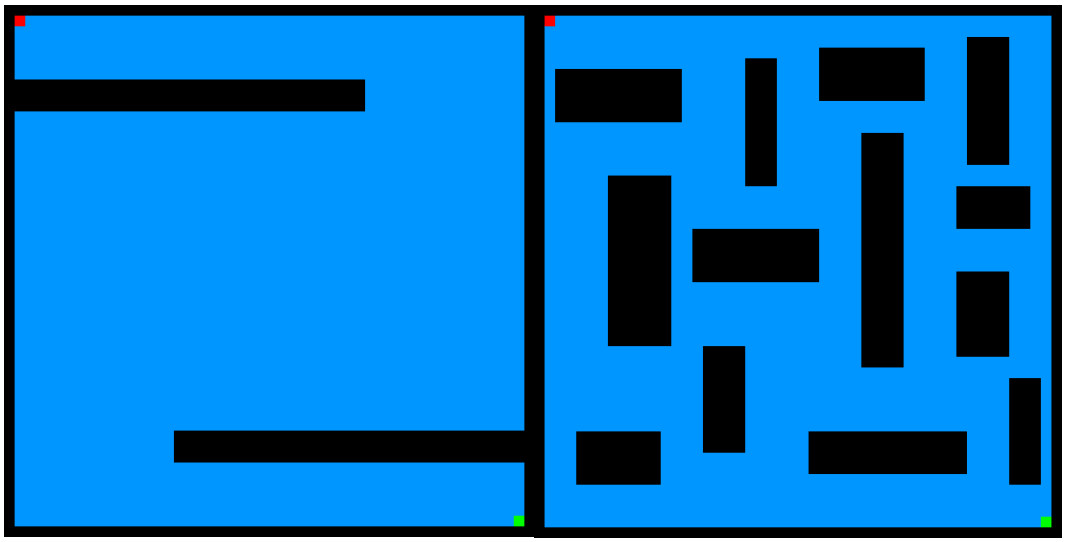

Figure 5: The jetty obstacle map (left) and dense obstacle map (right)

The execution time of the Voronoi-Dijkstra implementation is compared with the execution time of the A* algorithm applied on an identical situation. The results are tabulated into three different tables for the three different map sizes: 50 metres, 75 metres and 100 metres.

Algorithm Used	Execution time, T(s)		
	Sparse map	Jetty map	Dense map
Vor-Dij	0.0310	0.0256	0.0363
A*	0.0425	0.0879	0.0239

Table 1: Results for 50 metre Map

Algorithm Used	Execution time, T(s)		
	Sparse map	Jetty map	Dense map
Vor-Dij	0.1149	0.0941	0.1660
A*	0.1767	0.1219	0.0625

Table 2: Results for 75 metre Map

Algorithm Used	Execution time, T(s)		
	Sparse map	Jetty map	Dense map
Vor-Dij	0.3399	0.2438	0.4953
A*	0.4149	0.3329	0.1636

Table 3: Results for 100 metre Map

One consistent result is that the Voronoi-Dijkstra implementation is always faster than the A* algorithm when applied on the sparse map and jetty map, regardless of the map size. However, when considering the dense map, the A* algorithm performs faster. This result is expected, because the dense map has more obstacle nodes than the other two maps. This means that the A* algorithm has fewer non-obstacle nodes to investigate in the dense map and will thus terminate its search sooner than in the other two maps. This observation explains why the execution time for the A* algorithm is significantly lower in the case of the dense map. In addition, when there are more obstacle nodes, the Voronoi diagram partitioning takes longer because every node in the map has to be checked with more obstacle. This further contributes to A* outperforming Voronoi-Dijkstra in the dense map.

However, the dense map used is an extreme case, and it is highly unlikely that the USV will ever encounter such an environment. The sparse map and jetty map are far more likely scenario that the USV will tackle and the Voronoi-Dijkstra algorithm outperforms the A* algorithm in these important scenarios, thus validating the effectiveness of the implemented algorithm.

4.2 Planned Path Analysis

As a second observation, the paths planned by the A* algorithm and Vonoroi-Dijkstra algorithm are compared.

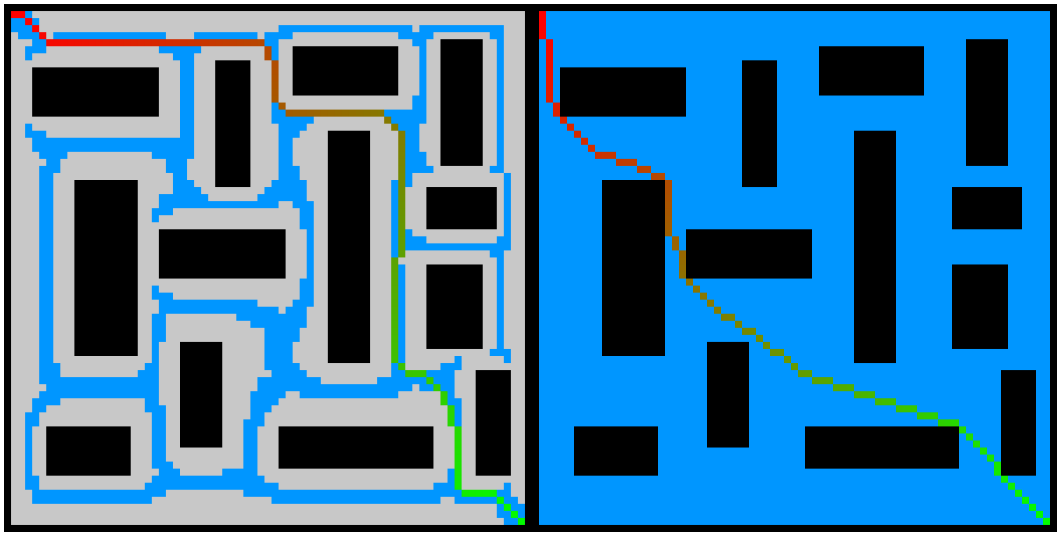

Figure 6: Path planned by Vor-Dij algorithm (left) and A* algorithm (right)

In Figure 6, the path generated by the Voronoi-Dijkstra algorithm is 131.6 metres, whereas the path generated by the A* algorithm is shorter, at 115.2 metres. Although, the A* algorithm generates the shorter and thus more optimal path, a more nuanced comparison is necessary. The A* algorithm fails to keep distance from obstacles and in this regards, it does not ensure a safe navigation path for the USV. Slight perturbations in the USV or obstacle positions could be catastrophic, making such paths unreasonable. In contrast, the Voronoi-Dijkstra algorithm only allows the planned path to exist on nodes that are between obstacles and at safe distances from any obstacles. This ensures collision free paths without any uncertainty. This further validates and verifies the effectiveness of the implemented Voronoi-Dijkstra algorithm in generating appropriate paths for the USV's navigational purposes.

5 Conclusions

The focus of this project was to develop a robust obstacle avoidance algorithm that would enable the USV to autonomously navigate on the water surface on a collision free path. The proposed approach of Voronoi diagram partitioning and Dijkstra's algorithm was developed and implemented to enable the USV to achieve independent path planning. The theory and important techniques have been discussed and analysed in this report thoroughly. The obstacle avoidance capability of the algorithm was tested on three different maps and compared with results from an A* algorithm, and the effectiveness of the Voronoi-Dijkstra algorithm was validated through this comparison. The algorithm has established its reliability in performing efficient path planning. The favourable results conclude the success of the project.

6 Recommendations for Future Work

One area for future work is to improve the time complexity of the Voronoi diagram partitioning step. The time complexity can be reduced by using a more elegant implementation of the algorithm and this may allow the Voronoi-Dijkstra algorithm to consistently outperform the A* algorithm when path planning for the dense map.

Another area for future work is to add a post processing step after Dijkstra's algorithm called the visibility algorithm. This could reduce the path length of the final path. However, care should be exercised as this a computationally expensive algorithm that may make the implementation unreliable for real-time usage.

References

1. Stein, A. D. (2012, January 17). *NPS Acquires Two USVs, Opens Sea Web Lab for Expanded Undersea Warfare Research*. Retrieved from America's Navy: http://www.navy.mil/submit/display.asp?story_id=64803/

2. Patel, A. (n.d.). *Introduction to A**. Retrieved from redblobgames: http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html

3. Al-Arif, S. M., Ferdous, A. H., & Nijami, S. H. (2012). Comparative Study of Different Path Planning Algorithms: A Water based Rescue System. *International Journal of Computer Applications*, 25-29.

4. Wager, M. L. (2000). *Making Roadmaps Using Voronoi Diagrams.*

5. Hanlin Niu, Y. L. (2016). Efficient Path Planning Algorithms for Unmanned Surface Vehicle. *ScienceDirect.*

6. Mamdouh, T. (2014, January 14). *Robot motion planning based on Voronoi*. Retrieved from https://tarekmamdouh.wordpress.com/: https://tarekmamdouh.wordpress.com/2014/01/24/robot-motion-planning-based-on-voronoi/