Path Planning, Modeling and Control of Autonomous Ships - An Approach based on the Minimum turning radius

Abd ElHakim Khaled Amer
Supervised by: Jeroen Van Stappen

19 October 2019

Contents

1 Summary **3**

2 Introduction **4**

3 Path Planning Algorithms **5**
3.1 Road Map methods Methods 5
3.1.1 Combinatorial Methods 5
3.1.2 Sampling Based methods 6
3.2 Methods Comparison 6

4 Simplified Problem Implementation **8**
4.1 Obstacles definition 8
4.2 Visibility Graph 8
4.3 Dijkstra Algorithm 10
4.4 Simple problem Results 10

5 Full Problem Analysis **12**
5.1 Turning Manoeuvre implementation 12
5.2 Elimination of paths with turns that are too sharp 13
5.3 Obstacles Inflation 13

6 Ship Model and Controller implementation **15**
6.1 Modeling of the ship 15
6.2 Controller Implementation 16
6.3 Test case 18

7 Recommendations for future work **20**

8 Appendix **21**

1 Summary

The aim of this project is to investigate and implement path planning algorithms on vessels for autonomous navigation. In this paper several state of the art path planning algorithms is presented and one is chosen to be implemented. The paper is divided into three parts, in the first the shortest path problem is solved without taking account the physical constraints of the ship. In this part, Visibility Graph method along with Dijkstra's algorithm were chosen to be implemented in order to find the shortest path between the initial and final position. The Visibility graph method is implemented in a robust way by performing two tests on each obstacle in the map. In the second part a method based on the minimum turning radius of the ship is proposed to take into account the physical constraints of the ship. There are two aspects to this problem, first a turning maneuver needs to be performed in order to follow the first segment of the calculated desired path. Moreover, an extra segment needs to be added at the end of the path to adjust the final heading of the ship. The second aspect is making sure that the path chosen doesn't have turns that are too sharp for the ship to perform. This is done by implementing an extra constraint within the Dijkstra algorithm. This constraint is that the wheel over points, which are the the points where the turning of the ship starts and ends, are at a distance at maximum half way from the nodes in both segments connecting to the node. An inflation method used for expanding the obstacles is discussed and implemented. In the final part a simple first order model of a ship is developed and a controller was designed to test the performance of the the proposed method on a realistic test case.

2 Introduction

Path planning is the problem of finding a route, ideally being the best route, connecting two poses (orientation and position). The path should be collision-free and at the same time be within the capabilities of the vehicle. Path planning is an important part of autonomous mobile vehicles as it provides the trajectory that the vehicle will follow to complete its mission.
Autonomous shipping, like all autonomous systems, is a topic under heavy research by major players in the industry and academia. For examples, technology developer Kongsberg and fertilizer Yara are developing the to-be autonomous container ship Yara Birkeland to reduce load on roads ineastern Norway [1].



Figure 1: Konsberg Autonomous vessel

Another example is the Roboat program at the port of Amsterdam, which explores and tests the possibilities for autonomous shipping. The increase in research in this field is due to the obvious benefits that autonomous ships bring. Autonomous ships can improve safety, increase efficiency, and relieve humans from unsafe and repetitive tasks. According to Allianz " between 75 percent and 96 percent of maritime accidents are caused by human error".



Figure 2: Roboat rendering

3 Path Planning Algorithms

There are two main classes in path planning methods, Complete Path path Methods and Road Map methods. Complete Path Methods are methods which rely on minimizing an objective function and takes into account the dynamics of the ship and its motion constraints. The resulting output of this methods is a smooth trajectory that is executable by the ship. On the other hand the Road Map methods discretizes the map into nodes and lines connecting the nodes. The resulting output is a map of interconnected roads, hence its name. After the discretization, a graph search algorithm is used to determine the best route to take, which is usually the shortest route. In this project, Road map methods are chosen to be explored, as they are simple to implement, and provide a good first path for the ship to take.

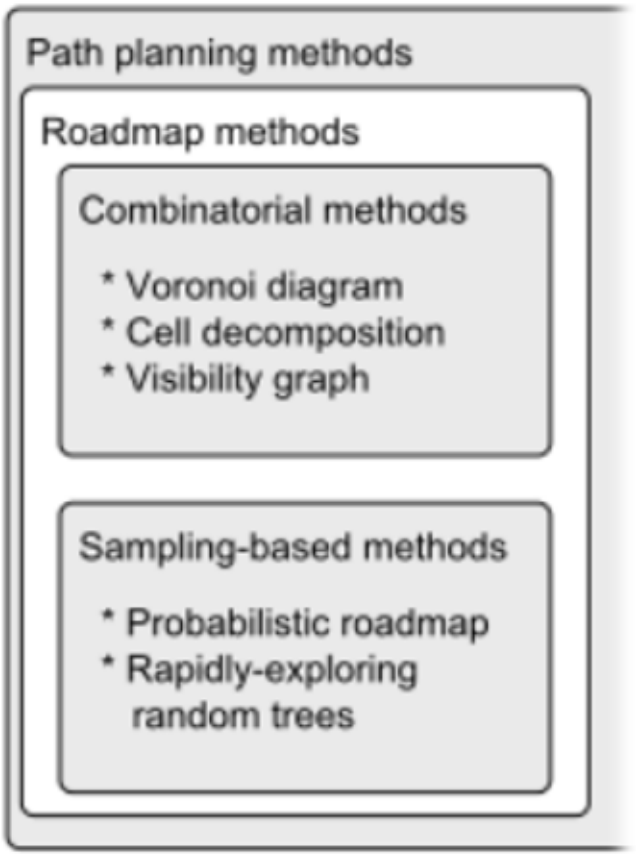


Figure 3: Road Map methods

3.1 Road Map methods Methods

As explained earlier this class of methods are used to find a set of collision free routes connecting nodes on a map. There are two main categories within this class of methods, Combinatorial and Sampling based methods.

3.1.1 Combinatorial Methods

Combinatorial methods are deterministic methods, which means that they will give you the same output every time you apply the algorithm for the same map. Examples include Visibility Graph, Vornoi Diagram and Uniform Cell Decomposition. In teh visibility graph you put the nodes at the vertices of the obstacles and you connect the edges between the nodes if there are no obstacles in between. Visibility graph produces many possible paths, among which one is shortest. Vornoi diagram on the other hand tries to maximize the distance between the obstacles and the trajectory,

so the nodes are between the obstacles. Uniform cell decomposition divides the map into equal cells, some are collision free and the rest are not.

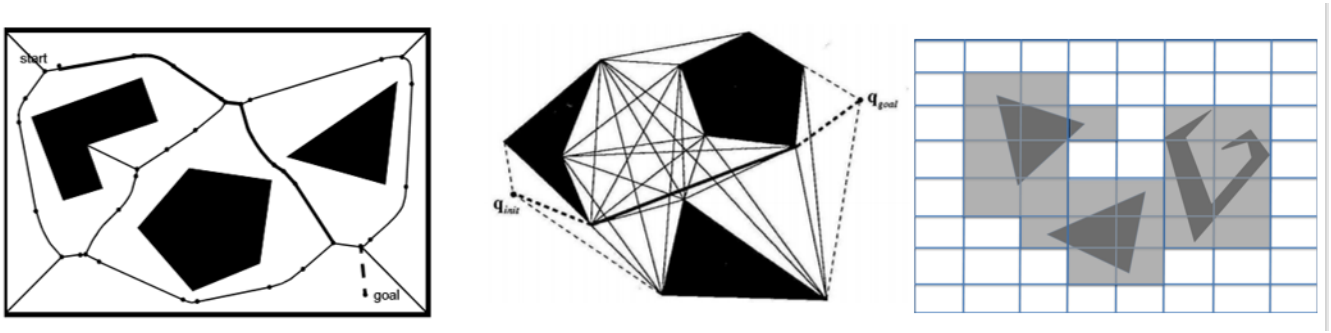


Figure 4: Combinatorial Methods

3.1.2 Sampling Based methods

Sampling based methods are Probabilistic methods, which might give you different outputs every time you run the code. There are two main examples in this category Probabilistic Road Maps (PRM) and Rapidly Exploring Random Trees (RRT).

3.2 Methods Comparison

Three main criteria is usually used when comparing path planning algorithms. Computational Time, Completeness and whether the algorithm produces an optimal path or not. For our application the two most important criteria is optimality and completeness. Computational time is not critical as our environment is the sea environment is usually relatively simple with few obstacles. Moreover, we are only considering static obstacles, so the algorithm will only be run once at the start of the journey. The table below provides a comparison based on the 3 criteria men.

COMBINATORIAL METHOD			
Method	**Optimal Path**	**Computation Time**	**Completeness**
Visibility Graph	√	×	√
Voronoi Diagram	×	√	√
Regular Grid	×	×	×
Adaptive Cell Decomposition	×	√	√
Exact Cell Decomposition	×	×	√
Potential Field	×	√	×
SAMPLING BASED METHOD			
RRT	×	×	√
PRM	×	√	×

Figure 5: Comparison

For each Method the three criteria are evaluated and checked if they are satisfied or not. We can see that only the visibility graph method provides the optimal path and it is complete. Moreover, the output of the path planning algorithm will be used as input to the Dynamic Positioning system in the form of way points. The DP system works best if the way points are fewest in number.It is clear that the visibility graph method produces the fewest way points of the methods we are comparing.Therefore it was chosen to be implemented in our path planning algorithm.

4 Simplified Problem Implementation

In this section a simple version of the algorithm will be implemented, connecting the initial and final positions of the ship. The algorithm takes inputs the initial and final positions as well as the vertices of the obstacles. There are two main steps in the implementation of the algorithm, first applying the visibility graph method followed by Dijkstra's algorithm.

4.1 Obstacles definition

In order to minimize the computation time as well as simplify the coding, the obstacles are divided into three main categories - Triangular obstacles, Rectangular obstacles and shore points. Rectangular and shore point lines are treated as a combination of several triangular obstacles, however, with extra constraints within the path planning algorithms. For example a rectangular obstacle is made of two triangular obstacles which are augmented together, with an extra constraint that there cannot exist a path along the edge joining the two triangles. Two additional algorithms need to be implemented. The first algorithm is used to sort the nodes in clockwise direction in order to plot the obstacle using the patch command on mat lab. The second algorithm is used to transform the rectangular obstacle into two triangular obstacles by replicating the common nodes. For shore points they are considered as triangular obstacles sharing a certain node, with the extra constraint that a line cannot past this line.

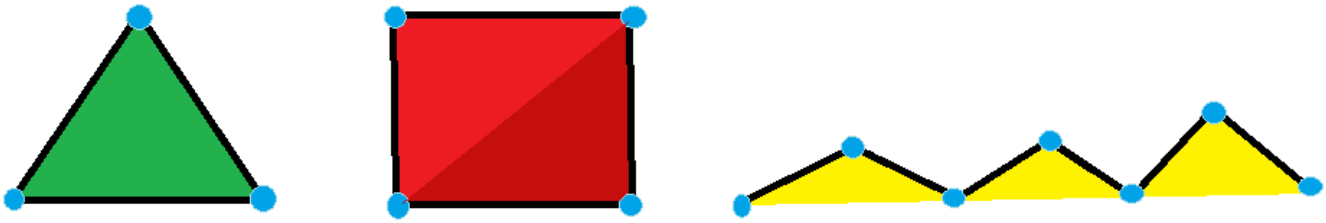


Figure 6: Obstacles Categories: Triangular, Rectangular and Shore Line

4.2 Visibility Graph

In the Visibility graph method (VG), the vertices of the nodes are connected to each other on the condition that the lines that connect them do not intersect with an obstacle. In order to make sure that the path between two nodes is collision free two tests will be performed on each obstacle.
The first test is to check whether the obstacle is between the nodes we are trying to connect or not. For this test the nodes will be checked if they lie on opposite sides of the line connecting the obstacle edges. If the nodes we are trying to connect lie on opposite sides of all the three edges, or on just two of the edges and one of the nodes is a vertex of the obstacle, then the obstacle lies between the nodes and we need to do a second check. Otherwise we move on to the next obstacle and repeat the test. The methodology is explained in the flow chart shown below.

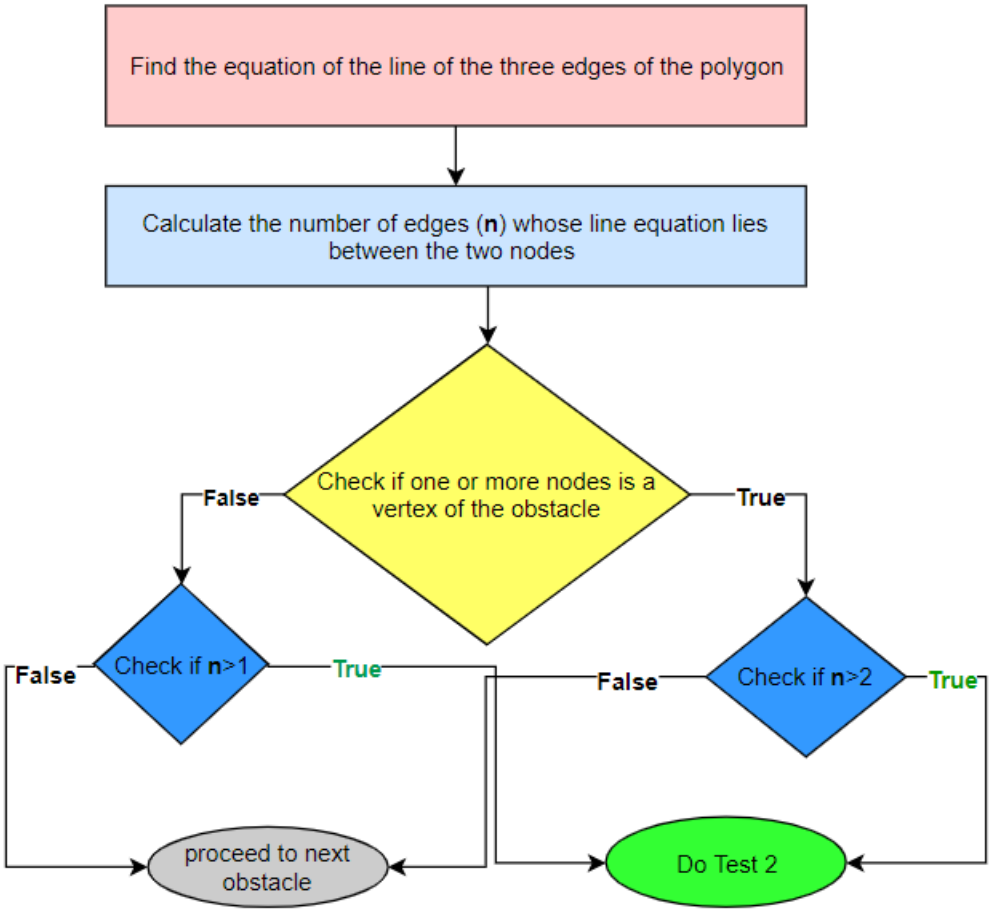


Figure 7: VG First Test Flow Chart

If we pass the first check we proceed to a second test. In the second test we do it the other way round. We find the equation of the line connecting the two nodes we are trying to connect and we check if the 3 vertices of the obstacles are lying on the same side of that line or not. If all the obstacles, fail either the first or second check, we can connect the line between the two nodes. This procedure is repeated for all the nodes.
The figure below shows an example on how the two test are implemented. For route R1 which is the discontinuous blue line connecting the start and end position directly we try to do the collision check for all the obstacles. Obstacle 1 fails the first check making it collision free as it is not between the start and end node. Obstacles 2 and 4 pass the first test, however, they fail the second. Obstacle 3 pass both tests. Because one of the obstacles pass both tests we cannot connect the path between the two nodes. Another option is route R2 shown in red which consists of several segments of collision free paths where all obstacles fail either the first or second test.

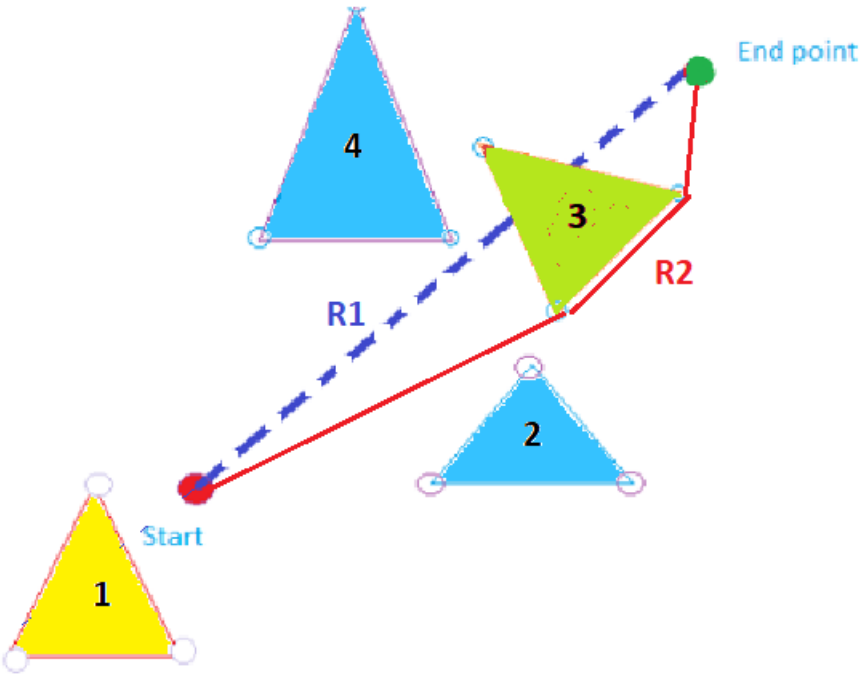


Figure 8: VG obstacles scenario

4.3 Dijkstra Algorithm

Dijkstra Algorithm is used to give the best route given weights assigned to each line segment in the road map. In our case the length of each line segment is the weight. For each iteration we have two types of nodes, a centre node and neighbouring nodes. We start at initial position as centre node and in each iteration we calculate distance of neighbour nodes from centre node and compare the distance with current distance and update distance if necessary. We then move to another centre node in the next iteration and iterations ends when all nodes have become a centre node.

4.4 Simple problem Results

The algorithms described in this section was tested on a test scenario which was coded on matlab and gave the output shown below. There are four 3 node elements in green and two 4 node elements in red and a shore at the bottom. We can see that all possible collision free path in the form of dashed lines, which is the output of the visibility graph algorithm. Then the Dijkstra algorithm works to find the shortest possible route from the start point to the target point out of a all possible routes. The result is a two segment path in blue from node one to 12 and from 12 to 38.

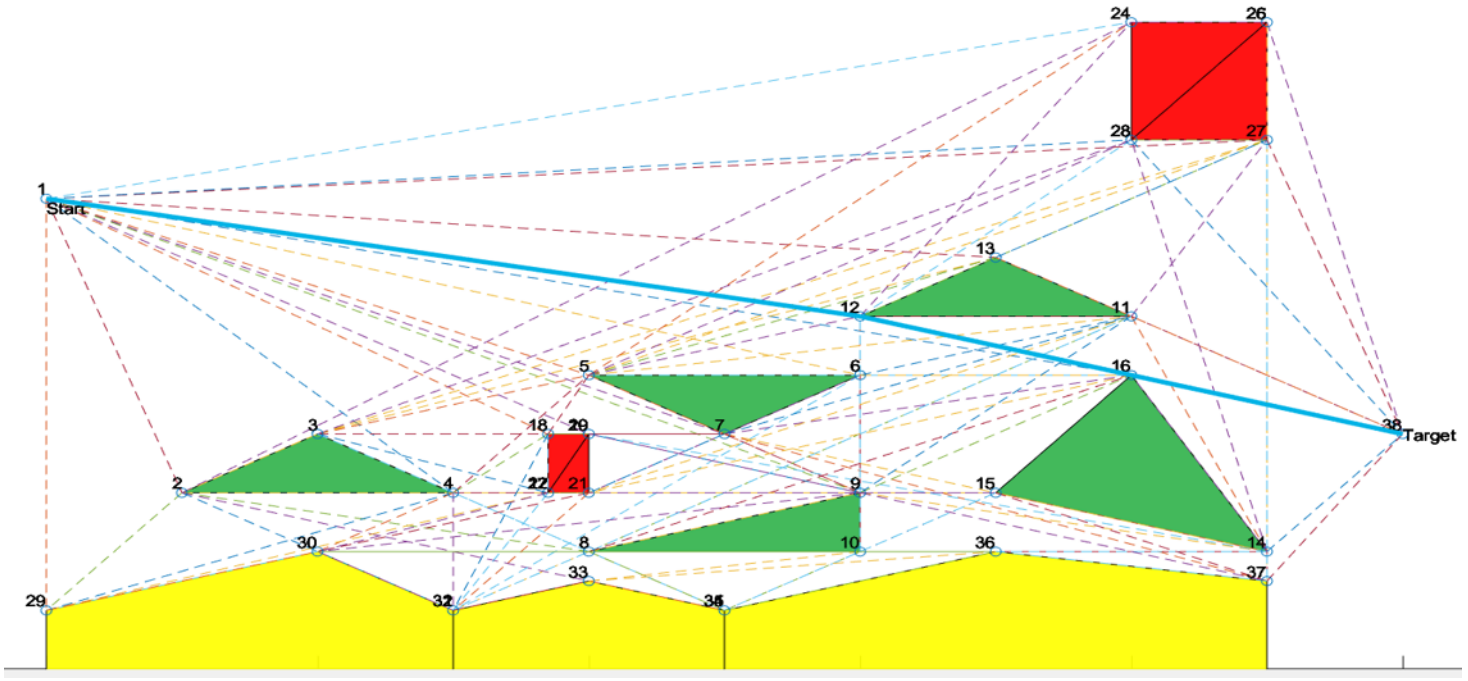


Figure 9: Simple problem Matlab code output with the three obstacles types

5 Full Problem Analysis

In the previous section we achieved the shortest collision free path between an initial and a final position. However, this path doesn't take into account the non holonomic constraints of the ship, i.e that the ship has certain minimum turning radius. Also it doesn't take into account the initial heading and final desired heading of the ship. Finally we need to apply a certain inflation as a safety margin, taking into account the dimensions and the uncertainty of the ship following the path.

5.1 Turning Manoeuvre implementation

We first tackle the problem of adjusting the path to include the heading of the ship. For this we apply a sharp turning manoeuvre algorithm after finding the shortest path. The algorithm consists of 2 sections a right turn then a left turn or vice versa, depending on the line connecting the initial node and the node connecting to it is closer in which direction.

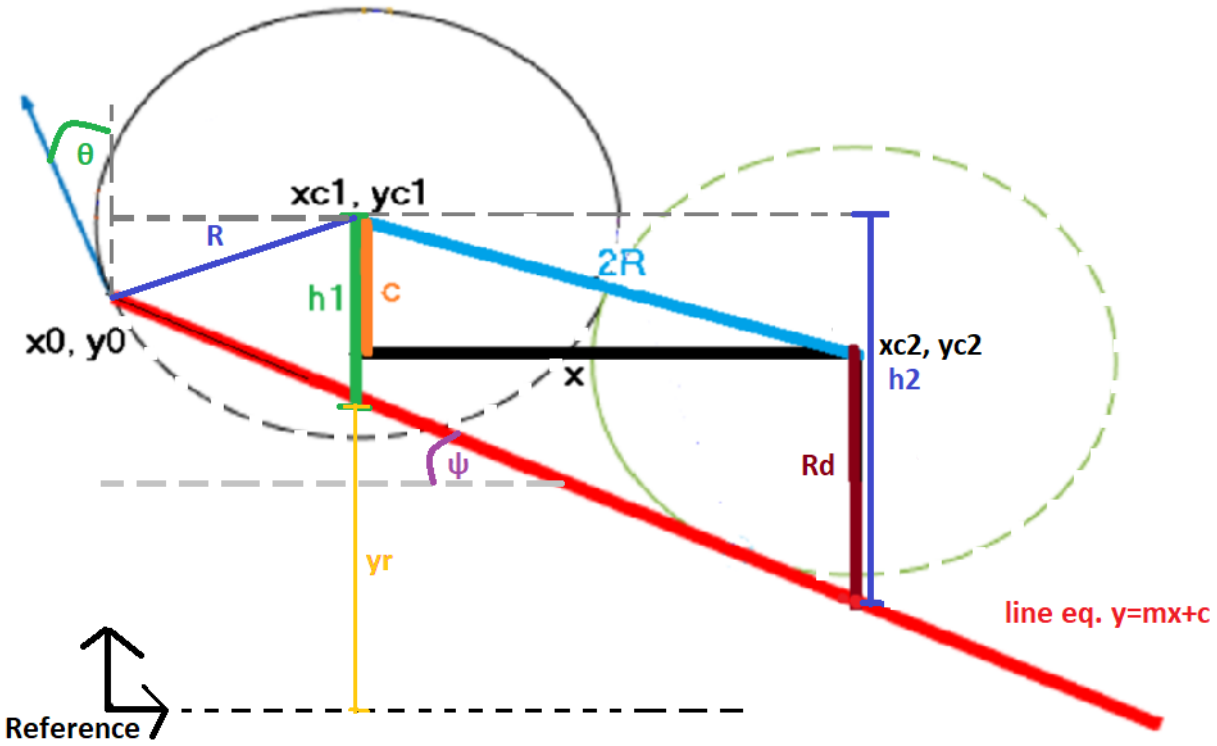


Figure 10: Turning manoeuvre geometrical relations for a certain case

We try to find the centres of the circles that satisfy the following conditions:

1. First circle must be tangent to the initial heading direction.

2. If it is a right turn then then the circle must be tangent to the heading direction to its right side and vice versa.

3. The second circle must be tangent to both the first line segment of the planned path as well as the first circle.

In order to find the path that we need to follow, we have to find the centres of the circles that satisfy the conditions mentioned. It could be found by the following equations which stem from the geometrical relations shown in the figure above.

5.2 Elimination of paths with turns that are too sharp

Another thing we need to make sure is that the path chosen doesn't have turns that are too sharp for the ship to perform. This is done by implementing an extra constraint within the Dijkstra algorithm. This constraint is that the wheel over points (WOP), which are the points where the turning of the ship starts, are at a distance at maximum half way from the nodes in both segments connecting to the node. The reason why we need to make sure that the wop distance does not exceed the centre is to always give enough turning space for approaching the new line segment.
The geometrical constraints are depicted in the figure below.

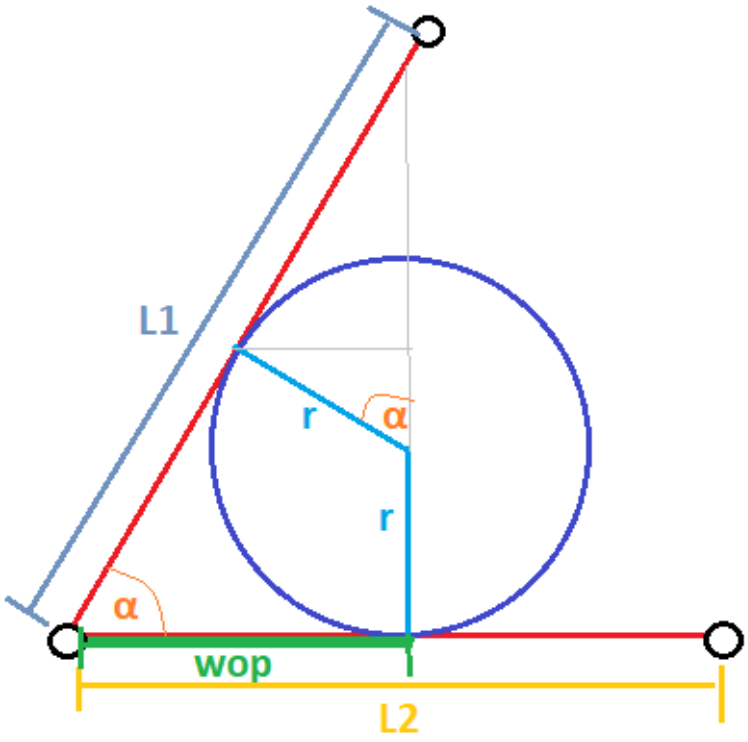


Figure 11: WOP calculation using geomterical relations **?**

5.3 Obstacles Inflation

Two approaches for obstacles inflation where experimented, the first approach that was tried was that the vertices of the obstacles are expanded outwards along the line connecting them to the centre node. This approach worked well for symmetric obstacles, however, not as good for non symmetric ones as shown in the figure. After failing of the first approach, a second approach was implemented, which expands the the edges of the obstacles along the direction perpendicular to the edges. The intersection points of the expanded obstacles is the new vertices coordinates. The second method which is shown below showed reasonable inflation for both symmetric and non symmetric obstacles.

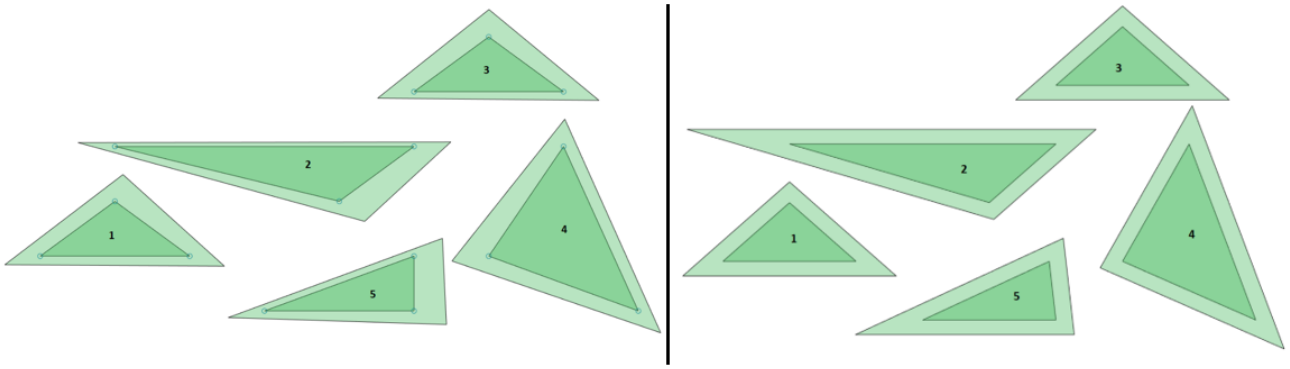


Figure 12: Comparison between two inflation methods,

6 Ship Model and Controller implementation

In order to test the performance of the path planning algorithm before being implemented on the Dynamic positioning system of the ship, we developed a simple model and PD controller on simulink. The simulink model takes input parameters from the output trajectory and outputs the dynamic position of the ship. Then a matlab script is run to plot the animation of the ship as it moves along the curve.

6.1 Modeling of the ship

The vessel is modelled as a ship which is propelled by an aft thruster and a controllable rudder. The dynamical model is based on the the first order response of the velocity towards an input velocity from the thruster. As well as a first order response of the Rate Of Turn of the ship to a given rudder angle and ship velocity. The equations describing the dynamics of the ship are given below.
The variables in the equations are defined as follows. v is the ship speed vth is the input velocity from the thrusters, Tsh is the ship linear time constant. ϕ_{sh} is the angle of the ship from the vertical, ϕ_{d} is the desired angle of the ship. α is the rudder angle. L is the distance of the centre of gravity of the ship from the rudder. The error term which is used for our controller is the perpendicular distance from the desired line segment.

$\dot{v} = \frac{v_{\mathrm{th}} - v}{T_{\mathrm{sh}}}$

$\dot{\phi_{\mathrm{sh}}} = ROT$

$\dot{ROT} = \frac{\frac{v * \mathrm{th} * \alpha}{L} - ROT}{T_{\mathrm{r}}}$

$\dot{error} = v * (\phi_{\mathrm{d}} - \phi_{\mathrm{sh}})$

Where $\mathrm{T_{sh}}$ and $\mathrm{T_r}$ are time constants of the ships linear and angular acceleration respectively. A realistic value for a ship of 100 m length is $\mathrm{T_{sh}}$=100 and $\mathrm{T_{sh}}$= 50. The controller was chosen to be a PD controller with P=0.001 and D=20. $\mathrm{v_{th}}$ was chosen to be 5 m/s.

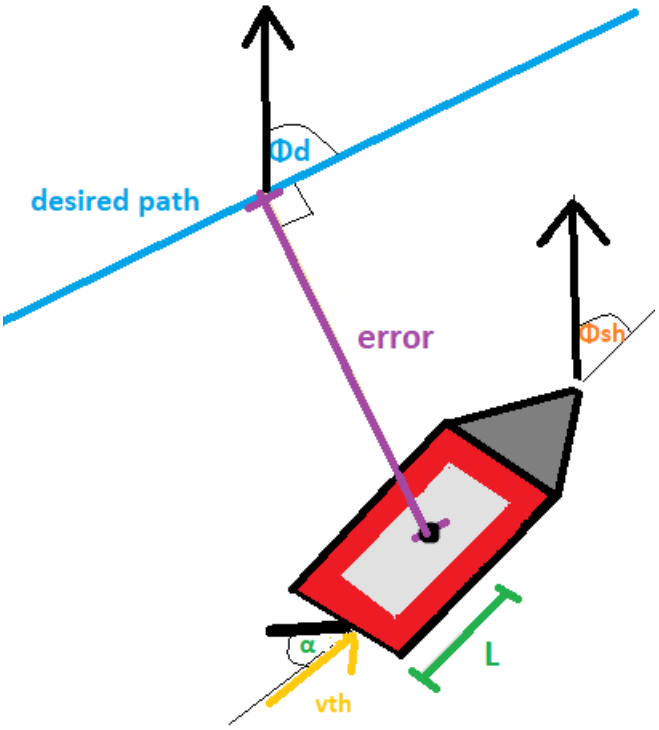


Figure 13: Ship 3 DOF mathematical Model

6.2 Controller Implementation

The model described the the previous section was implemented in Simulink, the block diagram of is shown below. A step input of the the vth is provided from Signal port 1. A PD controller was implemented to control the rudder angle alpha. A saturation was added after the PID controller to make sure that the rudder angle is within acceptable rudder angle range which is between -20 to 20 degrees. The model receives input the input from the matlab script in the form of a look up table containing the desired angle and corresponding travelled distance at several points along the desired path. The initial conditions are also input to the simulink model. The simulink model outputs the actual coordinates of the ship centre as a function of time.

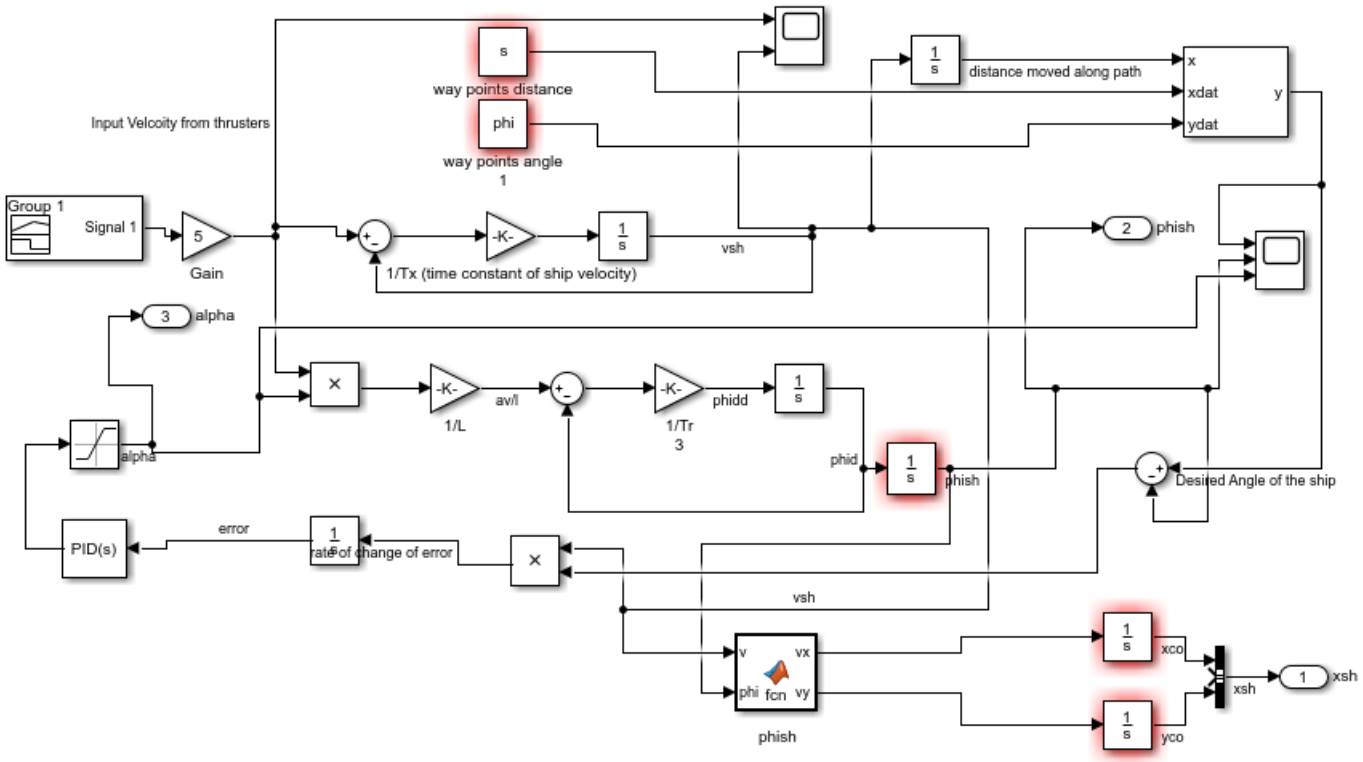


Figure 14: Simulink Model

6.3 Test case

An operational scenario of the ship was created to test the perfomance of the algorithm. The goal is to reach a wind turbine site in order to do some operations on the site and to go back to the initial position. This map was chosen because it has many static obstacles and few dynamic ones, making it within the the capabilities of the algorithm to execute.
Only one obstacle class was used in the test which is triangular obstacles, the discrtiztaion is shown is the figure below.

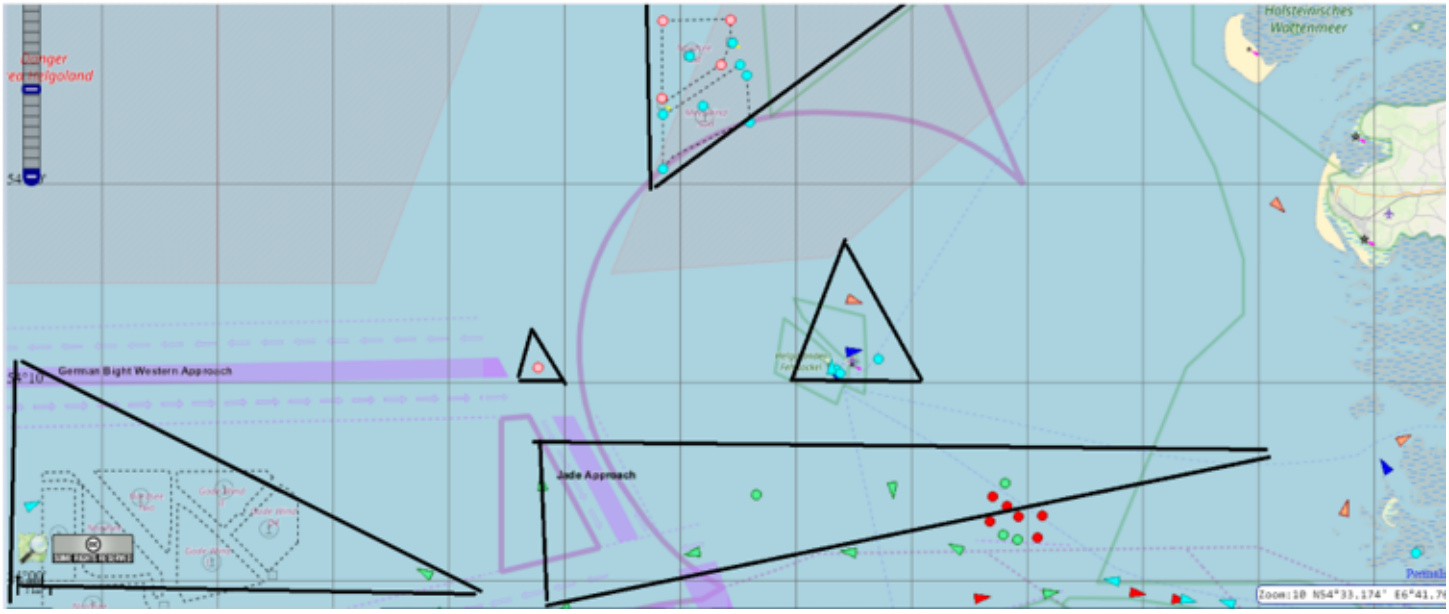


Figure 15: Selected map near the coast of Germany for testing and discretization of obstacles into triangles

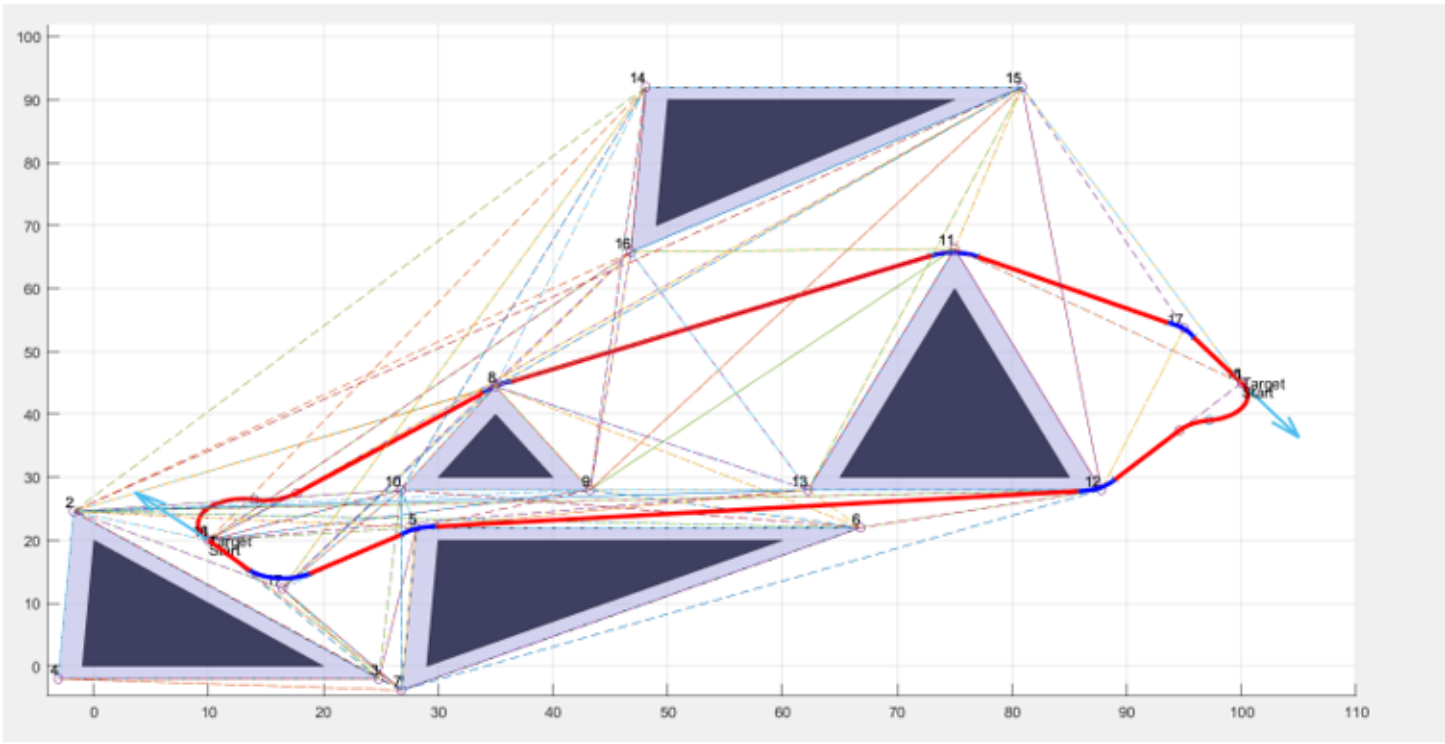


Figure 16: Full problem test matlab code output

An inflation of 2 km of the obstacles was used to make a safe path for the ship. The radius of turn of the ship was taken to be also 2 km. The code is run twice to make a full circuit from the initial position to the inspection position and then back to the initial position. The path taken by the ship is different than the initial path taken by it. That is because in order to return to the initial position with same orientation the ship cannot take the same route as the last segment will contain a very sharp turn. A smoother transition would be through the route taken as shown in the figure.
An animation was performed using the simulated model and it shown that the ship can follow the designated path. However, better control input need to be provided, as error in measurement of the velocity, or deviation from the original path for any reason will cause the ship to divert away from the desired trajectory.

7 Recommendations for future work

Regarding the path planning, which is the main focus of this project, the path planning algorithm can be further developed to incorporate the resistance of wind and current to ship in selecting the optimum path, as well as the sharpness of turn as it would lead to sailing in a mode that is not optimal in terms of energy usage. The turning manoeuvre could be Incorporated in the shortest path, first of all it might be a collision path if there are obstacles near by the starting position. Also the extra point that is added for the final heading needs to be checked for collision. Inflation needs to be applied to different obstacle types and shore points need to be further developed. Moreover, dynamic obstacles need to be considered as most obstacles that are present in the environment are other moving vessels. Also a collision avoidance algorithm needs to be implemented as a backup for unexpected obstacles.

8 Appendix

All the plots and animations are performed using the Matlab file provided.