**NAVAL POSTGRADUATE SCHOOL**
**Monterey, California**

# ME 3720

# Project 2

## Motion Planning in the BOP Environment

### Names

Jeré Combs

Teddy Herrera

Joseph Young

## TABLE OF CONTENTS

# I. INTRODUCTION

Building on the methods and concepts learned in Project 1, Project 2 had the Fusion UUV navigate a subsea blowout preventer (BOP) structure. The group achieved depth, heading, and waypoint navigation control with the vehicle. Now, the group must display fine control and a robust autopilot algorithm for the vehicle to find and scan AprilTags in the BOP environment. The primary sensor to be used for the vehicle will be its onboard camera. The group will prioritize minimizing the time it takes to navigate the BOP, as well as minimizing the distance the Fusion UUV must travel to find all the AprilTags.

## A. OBJECTIVES

To ensure success, the group will have a series of goals to meet to reach the final objective of having the Fusion UUV using its autopilot system to navigate and find all the AprilTags. The group will build their own map for the vehicle to navigate, then employ an algorithm of their choice to assist the vehicle in waypoint navigation. This is similar to the waypoint navigation of Project 1, but instead of the waypoints being hard coded in, the waypoints will be based on the solution of the path finding algorithm. The team will then incorporate the AprilTags that will provide random instructions to the vehicle telling it where to go.

## B. APPROACH

During the project's initial stages, each group member independently developed their code and analyzed each available UUV. After week 2 of the project, independent development was consolidated and standardized so that each group member had access to the same base code and algorithms.

# II. DEVELOPMENT, ANALYSIS, AND RESULTS

## A. 3D WAYPOINT NAVIGATION WITHIN BOP ENVIRONMENT

This goal was undertaken early to ensure the group knew how to control the Fusion UUV in the BOP environment. While most of the initial development of the motion planning for the Fusion UUV occurred outside of the Gazebo environment, there was some development within Gazebo to evaluate the motion of the UUV. This was initially done using plannerRRT then using plannerRRTStar. The motion planning algorithm plannerRRT was used first to ensure the group members had a handle on the implementation of a motion planner for the UUV’s navigation.

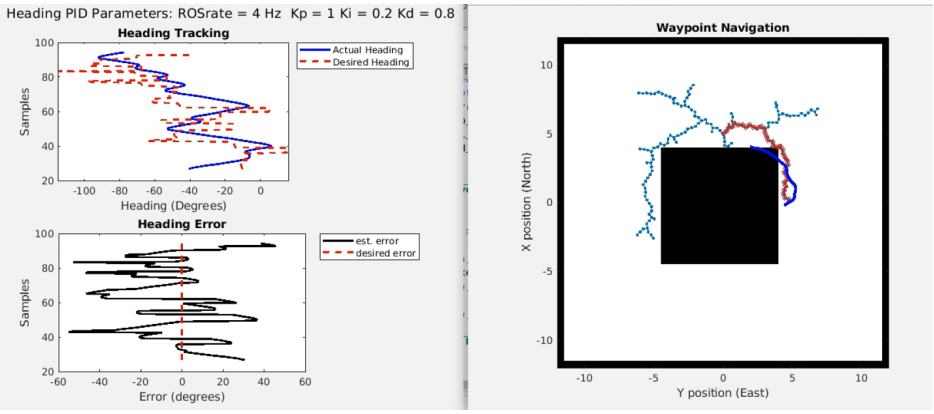


Figure 1. Initial Evaluation of Fusion UUV behavior in Gazebo

In the early implementation of the code using plannerRRT, the desired start position was not provided to the UUV. So, the UUV simply started wherever it was in the simulation. Here in Figure 1, the starting point for the path was (0,5,0) and the goal was (5,0,0). The selected path is in red and the UUV's path is in blue. The heading controller was similar to the controller used in project 1. The difference was that the waypoints were determined by the selected path to reach the goal, rather than a series of waypoints hard coded into the script. The heading controller worked as expected, but there was an oversaturation of waypoints for it to follow. The UUV was constantly turning to drive to all the waypoints. The UUV ended up skipping several waypoints, but it eventually made it to the goal. The UUV did not stop at the goal, however, and continued to coast past it. Also, the UUV cut the corner, indicating that the map's boundaries needed better integration. This preliminary test showed that the UUV could navigate in 3D space as the UUV was also instructed to change and maintain a specific depth, though this is not represented on the 2D occupancy map.

## B. MOTION PLANNING WITHIN SUBSEA BOP ENVIRONMENT

The approach to the motion planning problem started with building a configuration workspace of the subsea BOP environment within MATLAB and then investigating and testing motion planning algorithms from the MATLAB Motion Planning Toolbox. The motion planning algorithms were initially investigated and tested within the MATLAB development environment outside of the ROS and Gazebo simulation environment. When an algorithm meets the performance requirements within MATLAB, it will then be implemented with the Fusion UUV in Gazebo and begin with 2D navigation within the undersea BOP environment. Motion planning will then be expanded to include clockwise and counterclockwise navigation around the BOP. The final milestone in this phase will be the Fusion UUV starting at the top of the BOP and reaching altitude objectives.

Three motion planning algorithms were investigated to determine their efficacy in the problem space. The MATLAB motion planning algorithms plannerHybridAStar, plannerPRM,

and plannerRRTStar will be presented and investigated. The results of the investigation will show that the plannerRRTStar algorithm will be chosen for implementation within the Gazebo simulation environment. The motion planning algorithms will be evaluated based on their ability to generate a path that connects the start and goal states that is free of obstacles and the speed in which the algorithm finds a solution. Algorithms must be able to consistently find a solution from the start and goal state and additional consideration will be given to algorithms that work in both a 2D and 3D state space. Additionally, algorithms that can find solutions faster than the ROSrate used to operate and control the Fusion UUV. In this problem set, the fastest ROSrate to this group will use is 5 Herts, or 1 operation every 200 milliseconds. Therefore, algorithms whose computation time exceeds 200 milliseconds will not be considered for use in this lab.

Figure 2 shows a 2D occupancy map of the BOP environment with white representing the free space and black representing occupied space. The occupied space was artificially inflated by 0.5 meters to improve obstacle avoidance by accounting for the Fusion UUV's size and path following errors from Lab 1.

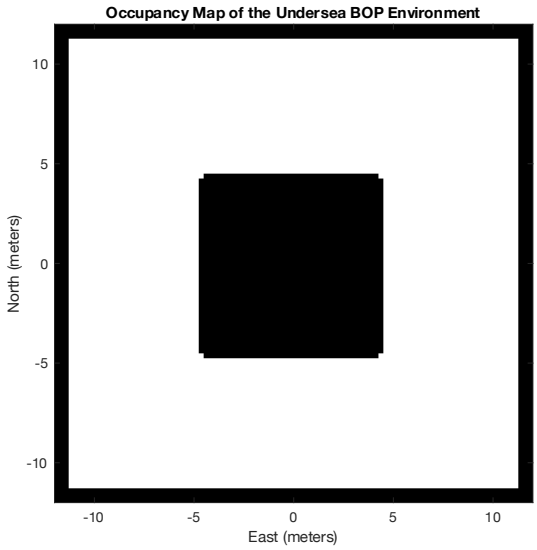


Figure 2. 2D Occupancy Map of the Undersea BOP environment. Obstacle boundaries were inflated by an additional 0.5 meters to account for vehicle size and path following error.

The first algorithm the group investigated was plannerHybridAStar, which uses the A* search algorithm to create a path within a 2D map. The algorithm can be tuned by setting a minimum turn radius of the Fusion UUV, the length of motion primitives generated, a forward cost and reverse cost that penalizes forward or reverse motion. When tuning the algorithm, the forward cost and reverse cost default values provided adequate results in terms of prioritizing forward motion and not reverse motion. A minimum turning radius of 1.5 meters is set, based on the Fusions UUV's performance when navigating a box of waypoints in Lab 1.

An iterative approach was taken with respect to the length of motion primitives generated. Figure 3 and 4, show the path and computational performance of different motion

primitive lengths for clockwise and counterclockwise navigation, respectively. Computational performance was in the tens of milliseconds and all the paths crossed within two meters of the goal point. The goal point also had a desired heading, θ, that is meant to orient the bow of the Fusion UUV towards the BOP, which is likely why the paths slightly overshoot the goal point. From the figures, it can be extrapolated that a length of 1 or 1.5 meters for the motion primitive yields a path that is well under the algorithm computational time constraint, avoids obstacles in the environment, and gets within 1 meter of the goal point.

While the plannerHybridAStar algorithm yields desirable motion planning results in 2D space, it will not be used to solve the motion planning problem presented in the undersea BOP environment. The undersea BOP environment requires the Fusion UUV to navigate a 3D space, requiring the Fusion UUV to navigate depth independent of the 2D map. In this problem set it is desirable to use an algorithm that can navigate a 3D map of the environment.

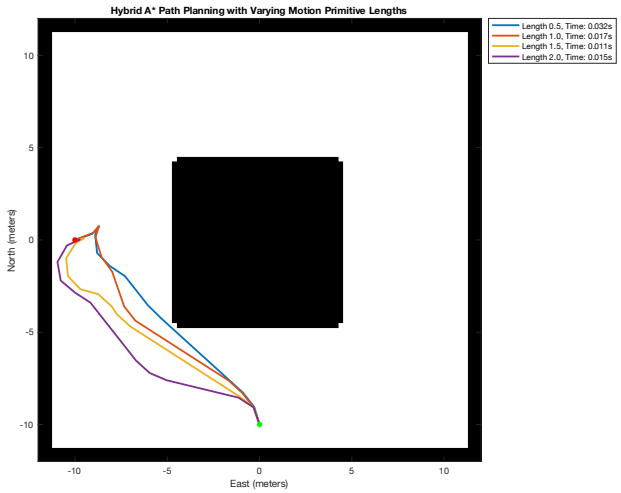


Figure 3. Clockwise navigation of the BOP environment using plannerHybridAStar.

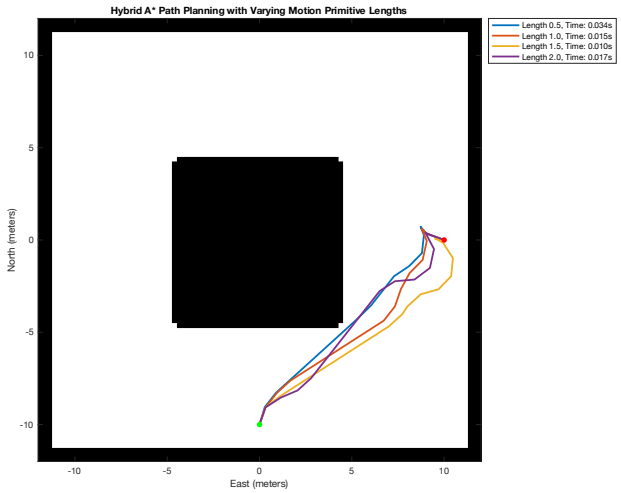


Figure 4. Counterclockwise navigation of the BOP environment using plannerHybridAStar.

The second algorithm to be investigated was plannerPRM which develops a roadmap without start and goal states and uses the MATLAB 'plan' function to find an obstacle free path between a specified start and goal state. This algorithm comes with a caveat that it does not always find a connected path between the start and end goals. The parameters that affect computation time and the likelihood of finding solution are the maximum number of nodes in the graph of the state space and the maximum connection distance between two connected nodes.

The default parameters of plannerPRM will yield a complete solution while sacrificing computational efficiency. Figure 5 shows the resulting path. While a complete solution is achieved, the computation time does not meet the criteria defined at the beginning of this section. Figures 6 and 7 show that the computational time was reduced by almost 75% through increasing the maximum number of nodes and decreasing the maximum connection distance between nodes. The pitfall with this approach is that if not properly tuned the plannerPRM algorithm will not be able find a solution connecting the start and goal states.

While the plannerPRM could be further tuned to yield a faster computation time that would meet the requirements in this problem set, it comes with the increased risk of not finding a motion planning solution. Therefore, within this problem set, the plannerPRM is not an algorithm that is suitable for employing in the simulation environment.

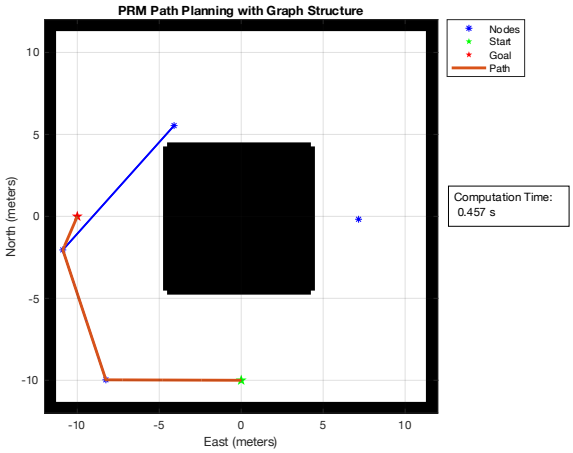


Figure 5. Clockwise navigation using the default parameters of the plannerPRM algorithm.

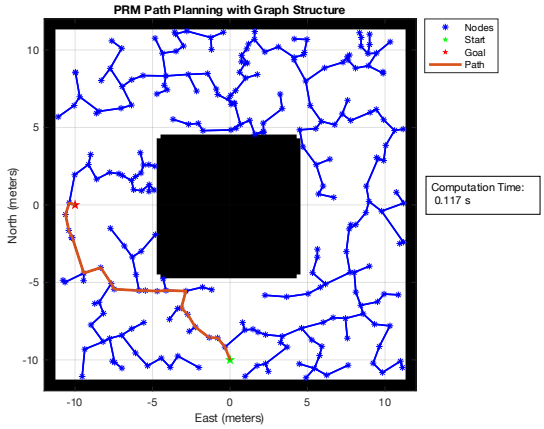


Figure 6. Clockwise navigation using tuned parameters of the plannerPRM algorithm

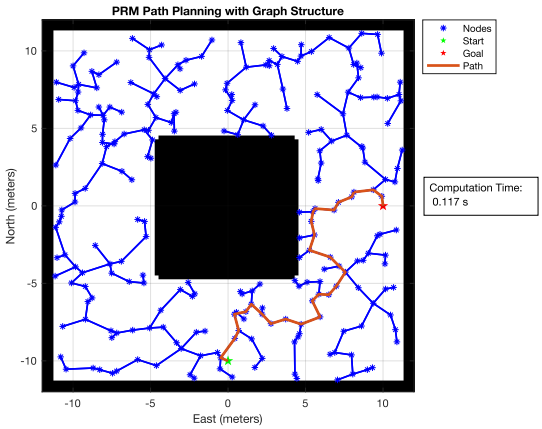


Figure 7. Counterclockwise navigation using tuned parameters of the plannerPRM algorithm

The last algorithm investigated was the MATLAB plannerRRTStar which uses the RRT* search algorithm to solve a geometric planning problem and works within a 2D and 3D space. This is extremely desirable for solving the problem set within this lab. The algorithm was first applied to a 2D state space with a 2D occupancy map to represent the BOP environment. Then a simplified 3D state space of the BOP environment was generated for 3D motion planning. Finally, by importing the model of the BOP from gazebo, a more accurate representation of the environment could be investigated. The algorithm parameters of maximum connection distance and maximum number of iterations were changed to evaluate computational efficiency and path smoothness.

Within the 2D state space, Figures 8 and 9 show the resulting motion planning of the different motion planning parameters with a uniform state sampler. The resulting paths are complete and from inspection it appears that a maximum connection distance of 1 meter and maximum number of iterations of either 5000 or 10000 will yield a complete solution that is relatively smooth with a computation time of less than 10 milliseconds.

Additionally, within the 2D state space it was desirable to see how a different sampling method would affect the computational time and motion planning results. Figure 10 and Figure 11 utilize the same range of differing algorithm parameters with a Gaussian state sampler in place of the uniform state sampler. The Gaussian state sampler significantly increased computation time and with motion planning results that appear comparable to the uniform state sampler. Therefore, the Gaussian state sampler will not be used for the plannerRRTStar algorithms used in the 3D state space investigations.

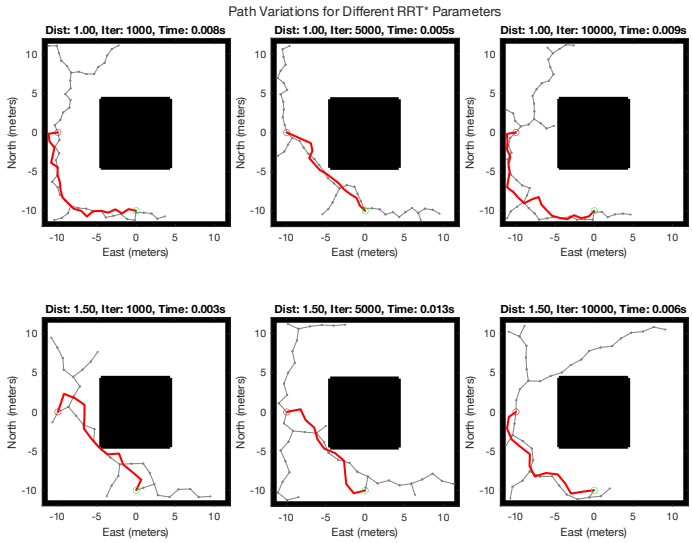


Figure 8. Clockwise motion planning results for different RRT Star Parameters using a Uniform State Sampler.

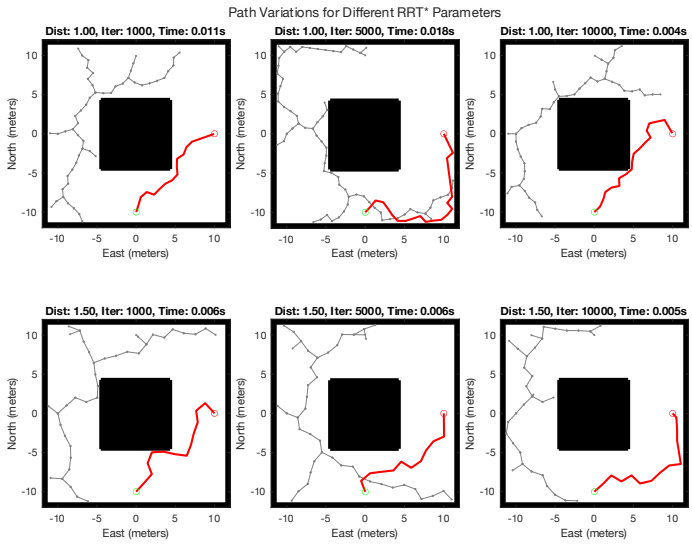


Figure 9. Counterclockwise motion planning results for different RRT Star Parameters using a Uniform State Sampler.

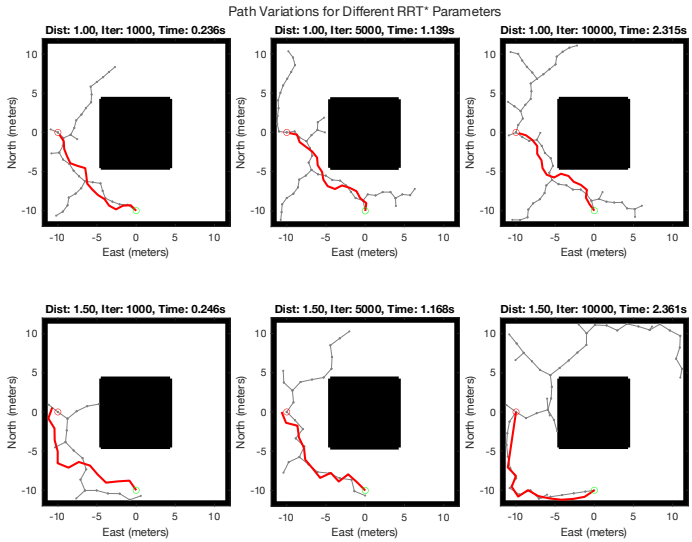


Figure 10. Clockwise motion planning results for different RRT Star Parameters using a Gaussian State Sampler.

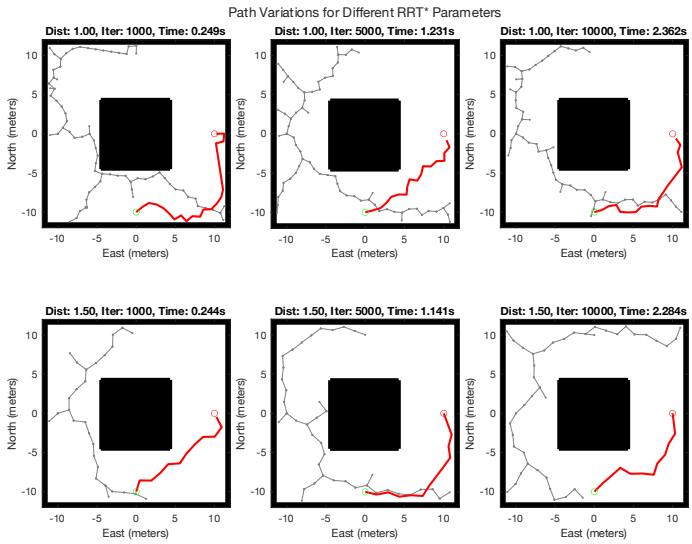


Figure 11. Counterclockwise motion planning results for different RRT Star Parameters using a Gaussian State Sampler.

Due to the methods employed by MATLAB's path planning algorithms, the paths selected are not direct paths between points. This leads to less-than-optimal solutions by increasing the total distance required for the vehicle to navigate and by requiring the vehicle to frequently change heading. A more direct path shortens the distance and minimizes the course change frequency. Figure 12, below, shows optimized paths for several sets of waypoints. This optimized path was found by employing the capabilities of the validatorOccupancyMap function within the MATLAB Motion Planning Toolbox. This function allows for determining whether a valid motion path exists between two points. By iterating through the points on the RRT* determined path, and eliminating any points between two valid states, a shorter, more direct path was determined.

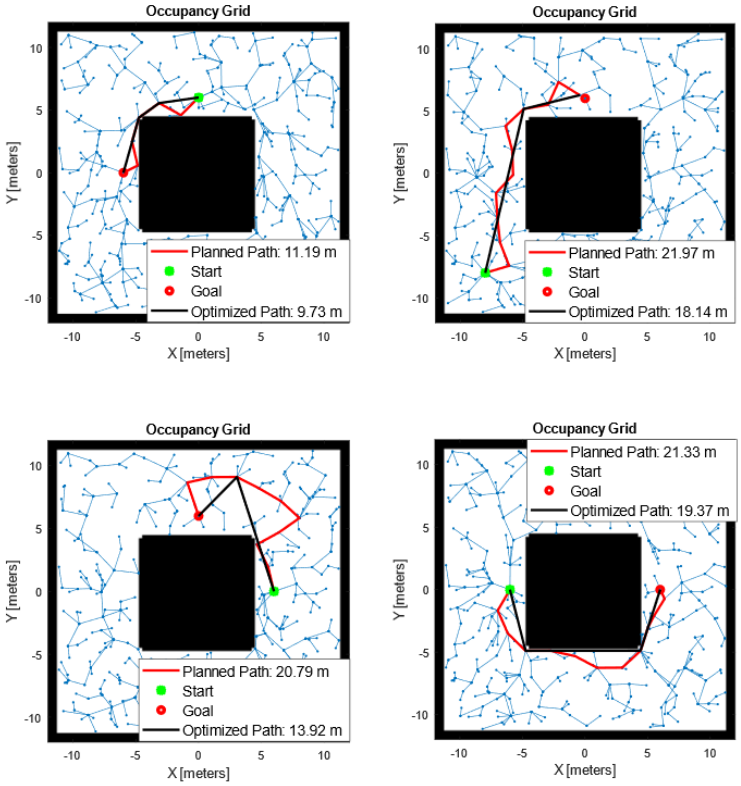


Figure 12: Comparison of MATLAB RRT* planner determined path and optimized path obtained by creating a straight-line path between the farthest points with a valid motion path between them.

The plannerRRTStar algorithm was also investigated within the 3D space, this required the occupancy map of the BOP environment to be expanded from a 2D occupancy map to a 3D

occupancy map. Figure 13 presents a simplified 3D occupancy map of the subsea BOP environment that approximates the shape of the BOP. The dimensions of the BOP were obtained by importing the model's data from the stereolithography (.stl) file available within the remus_ws folder on Ubuntu. The approximate 3D occupancy map was primarily used to simplify the learning curve associated with applying motion planning algorithms in a 3D environment. Once that learning curve was overcome the BOP model was inserted into the 3D occupancy as a point cloud with a resolution of 2 cells per meter and 20 cells per meter, shown in Figure 14 and 15 respectively. As the resolution of the of the point cloud increases it results in a more detailed approximation of the BOP, however it is not an exact representation that can be confidently navigated through, there is also the added cost of computation complexity to generate the environment with the higher level of resolution. Therefore, it is advantageous to generate the Occupancy Map in a different .m script and import the map into the path planning script as a .mat file.

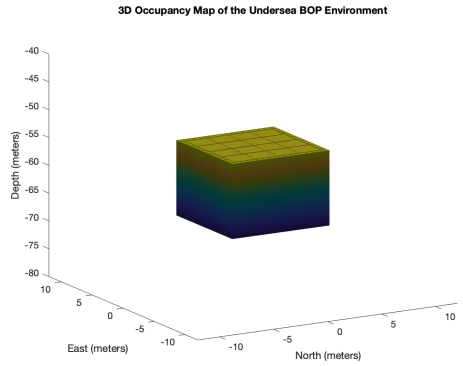


Figure 13. Simplified representation of the BOP in a 3D Occupancy Map.

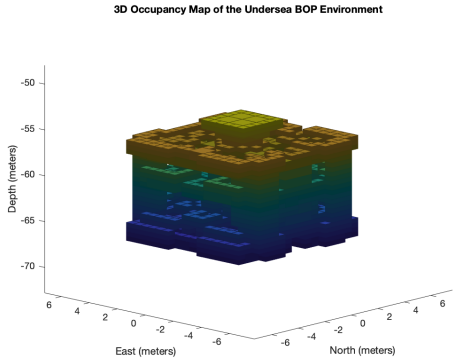


Figure 14. Point cloud representation of the BOP with a 2 cell per meter resolution.

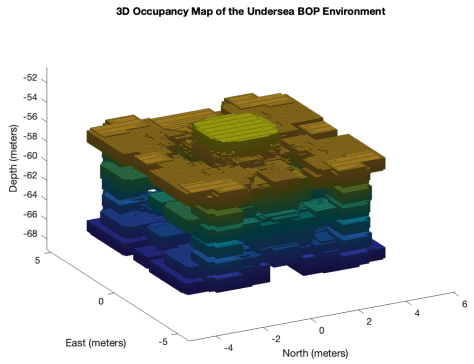


Figure 15. Point Cloud representation of the BOP with a 20 cell per meter resolution

Figure 16 shows the 3D path generated from the plannerRRTStar algorithm for the BOP with a 2 cell per meter resolution. The algorithm provides a complete solution in a respectable amount of time, about 0.047 seconds. Within the context of 3D path planning the MaxConnectionDistance, MaxIterations, GoalBias, and GoalReachedFcn parameters within plannerRRTStar influenced the plannerRRTStar algorithms computation time and ability to find a solution. The GoalReachedFcn parameter uses a callback function to determine if the goal state has been achieved, a smaller threshold will decrease the likelihood of achieving a solution and increase computational time. In this use case, being within a meter of the goal state is considered acceptable.

The MaxConnectionDistance and GoalBias parameters had the largest impact on the plannerRRTStar algorithm's ability to find a solution that is computationally efficient. As discussed earlier the MaxConnectionDistance Maximum influences the length of a motion allowed in the tree. In a 3D environment, the sample space has increased and using too short of a length of motion will yield short, choppy paths that will require further path smoothing and increase the likelihood of not finding a solution and increase computation time. Too long of a connection distance yields extremely computational efficient results but the paths are not optimal and can get close to exceeding the bounds of the environment. Simulations in the 3D environment revealed that a MaxConnectionDistance of five balanced the algorithm's ability to find a relatively smooth path, being computationally efficient, and increasing the likelihood of finding a solution.

The GoalBias parameter is a probability parameter that influences the algorithms state sampling to bias away from or towards the goal state. This is especially important in a 3D space with two points that are at different depths. When the GoalBias is away from the goal state will sample at the same depth as the start state, this is undesirable because it increases the likelihood of not finding solution. The inverse is also true, a GoalBias of 0.9 will yield samples around the same depth as the goal state. When the goal state and start state are at separate depths, biases like these will likely yield no solution. Therefore, a more balanced approach of using a GoalBias between 0.3 and 0.7 yielded computation efficient results that adequately sampled the range between the two depths.

The MaxIterations parameter constrains the algorithm to find a solution or no solution in a finite amount of time running indefinitely. This parameter is useful in determining if a solution

can be found and if the other algorithm parameters discussed are optimized for the path planning environment. Ideally, the adjustment to the other parameters will increase the likelihood of finding a solution while decreasing computation time. Through investigating the results of subsequent simulations, it is found the number iterations required to find a solution decrease into the hundreds. Therefore, this group found that the MaxIterations parameter can be decreased to 1000-2000 iterations, without negatively affecting the algorithms' ability to find a solution.

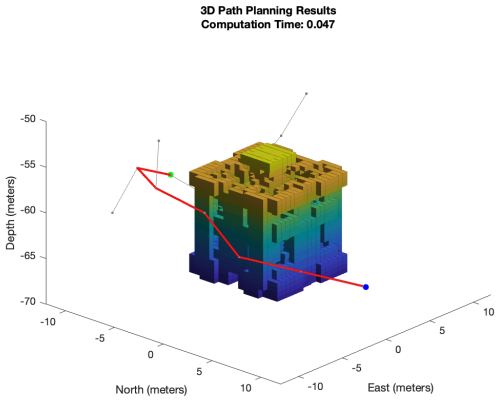


Figure 16. 3D path planning results using the plannerRRTStar algorithm with a BOP resolution of 2 cells per meter.

Figure 17 shows the plannerRRTStar algorithm's performance on a higher resolution BOP. All algorithms remained the same between Figure 16 and 17 simulations, and they found similar paths to the goal state. However, Figure 17 shows to have had better computational efficiency.

The group's hypothesis was that through the use of high-resolution 3D model of the BOP, a path could be found that navigated through the gaps in the BOP structure. This is desirable, because navigation through the BOP structure would yield shorter paths, decreasing the time it takes to accomplish the final run. However, in practice the .STL file and point cloud function used to generate the model of the BOP is not of a high enough fidelity for the plannerRRTStar algorithm to properly navigate through. Additionally, after evaluating the size of the vehicle and the clearances of certain areas of the BOP within gazebo, it was concluded that a more robust thruster control system that allows for translatory movement in the positive and negative North and East planes is required, as well as extremely precise thruster controls in the tight spaces of the BOP. Therefore, attempting to navigate through the BOP was deferred to follow on work after successfully running the required simulations.

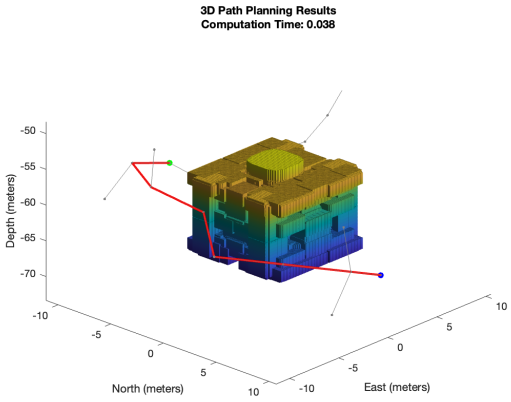


Figure 17. 3D path planning results using the plannerRRTStar algorithm with a BOP resolution of 20 cells per meter.

In this section three motion planning algorithms were investigated they were: plannerHybridAStar, plannerPRM, and plannerRRTStar. All of the algorithms met or exceeded the computational efficiency requirement and were able to find a path between the start and goal states. The plannerHybridAStar yielding the most computationally efficient results in a 2D environment. However, it will not be chosen for future implementation because it does not give the group the option to use as a 3D motion planner. The plannerPRM and plannerRRTStar functions both offer the option for 3D motion planning, but plannerPRM was found to be difficult algorithm to work with in the 3D environment and did not offer the same computational efficiency as the plannerRRTStar algorithm.

### C. APRIL TAGS SCENARIO

The group chose to move forward with the motion planner plannerRRTStar due to its ability to be used in as a 2D and 3D motion planner, and its simplicity in implementation. The plannerRRTStar function provided with the group with the ability to start with a 2D motion planner that could scale into the 3D space.

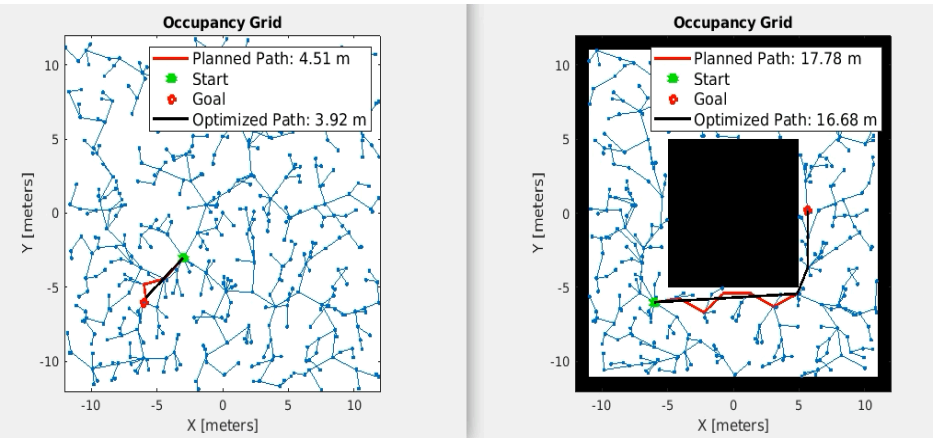


Figure 18. UUV moving from starting point to initial point to AprilTag #0.

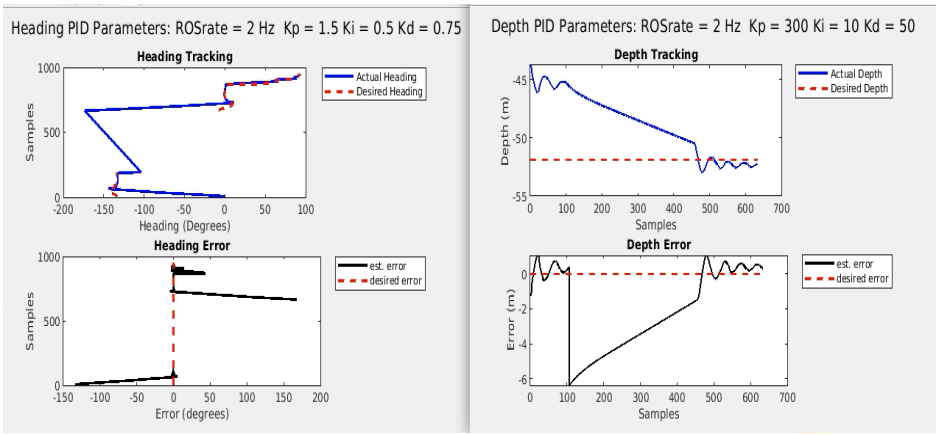


Figure 19. Heading and depth tracking of UUV moving from initial point to AprilTag #0

The group started by having the Fusion UUV move between from its initial starting point to AprilTag #0, as shown in Figures 18 and 19. The vehicle was able to hover at this initial stating position until it received the command to proceed to AprilTag #0. The vehicle then proceeded to that AprilTag, and it was able to hover at it. These are all good metrics for the UUV to meet, however, the UUV moves quite slowly in this process. The group prioritized ensuring the UUV could perform these tasks accurately, as it was not deemed worth having the UUV move between waypoints quickly while also having issues with it drifting past the AprilTags. The next step is to have the Fusion UUV read an AprilTag, then move to the next waypoint that it receives.

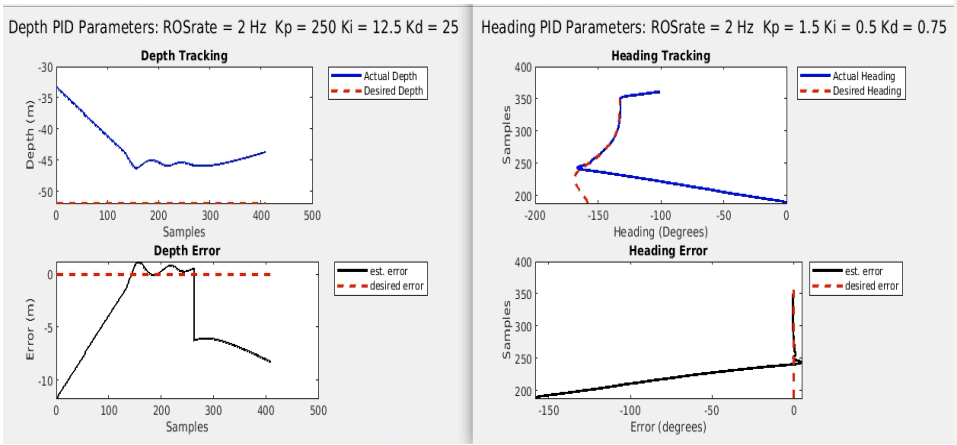


Figure 20. Example of the Fusion UUV not attempting to reach desired depth.

While developing the code for the final run, difficulties were encountered with the Fusion UUV's depth. This ranged from the vehicle not attempting to reach desired depth, as shown in Figure 20, in the middle of a run to the UUV sinking down through the BOP on startup, thus not being able to perform a run. The group attempted to refine the code to prevent the former issue, but it is uncertain if this was an issue with the code or Gazebo itself. The issue was not consistent, and the group did not need to modify gains for the depth controller. Upon startup of the Gazebo environment, the group noticed that the Fusion UUV sunk downward at a higher than usual rate. By the time the code would run, the vehicle would be inside the BOP and unable to perform a run. This was remedied by resetting the UUV's position to somewhere above the BOP. The UUV would then float down at a rate slower than the original, giving the group time to run the code. An alternative solution was to ensure the UUV was resting on top of the BOP prior to starting a run.

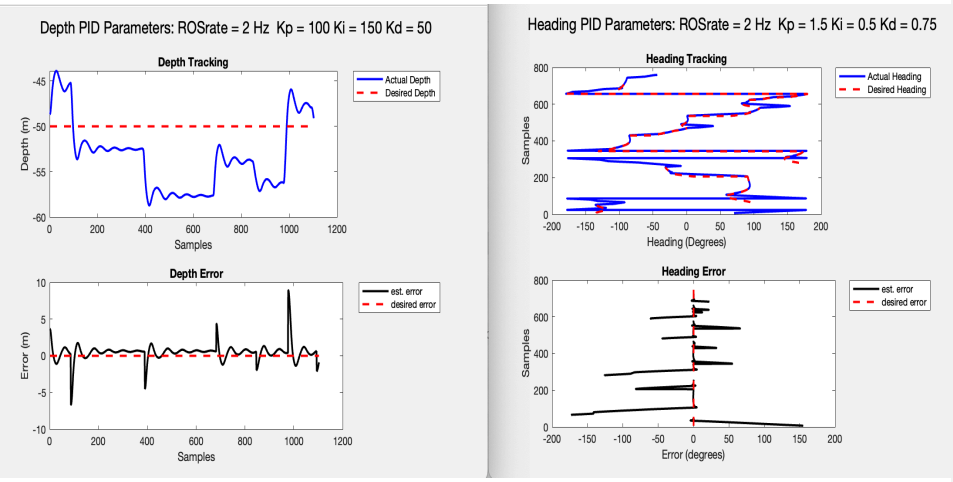


Figure 21. Final run in the BOP environment.

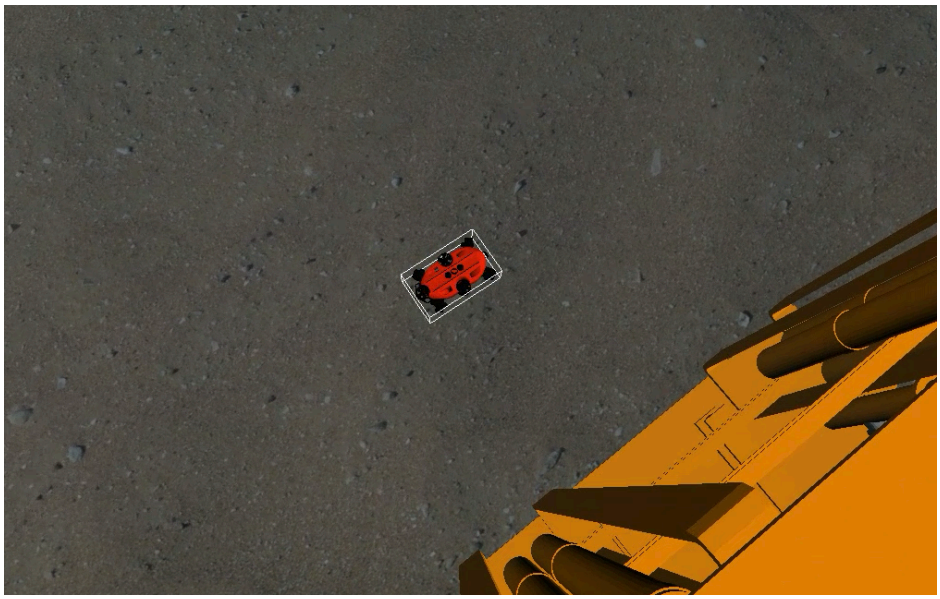

Figure 22. The UUV maneuvering around the BOP.

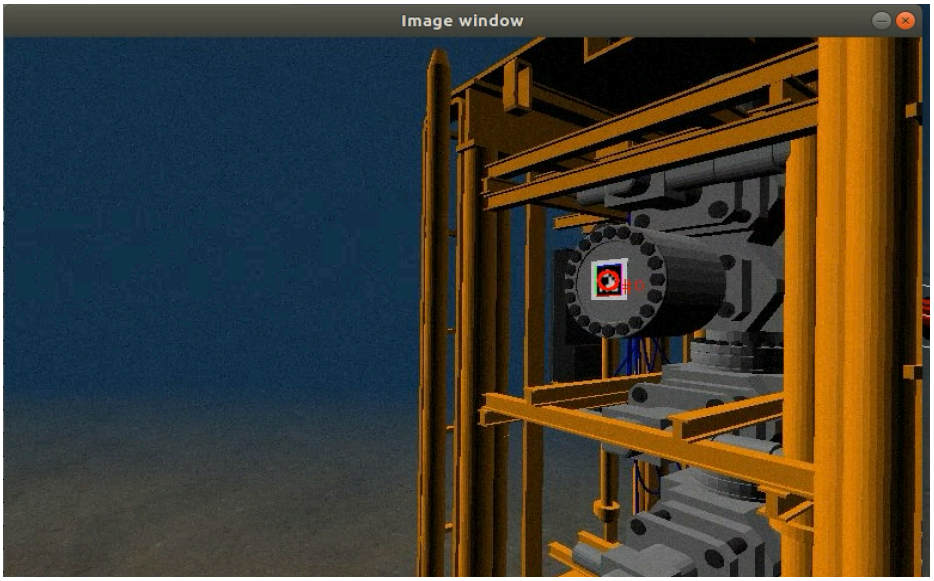

Figure 23. The UUV approaches AprilTag #0

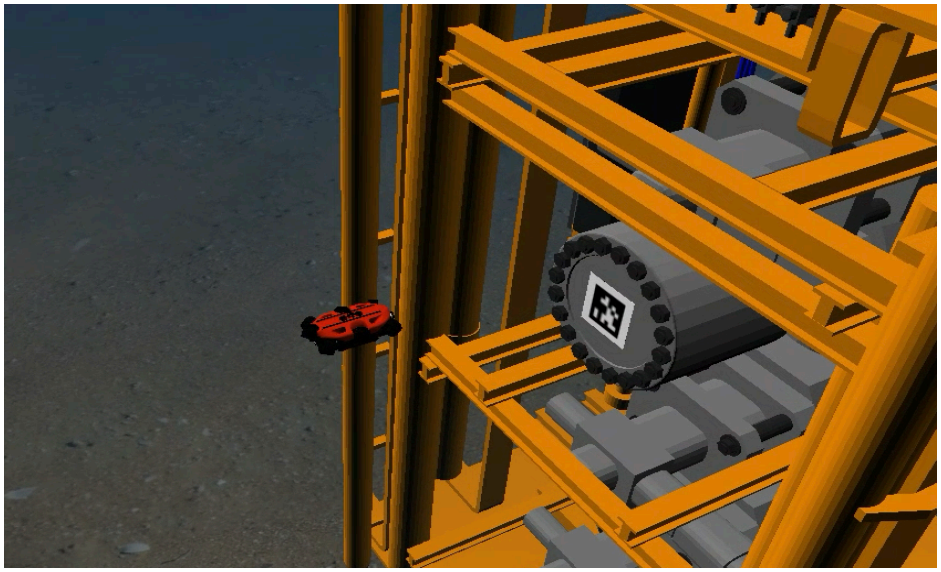
Figure 24. The UUV reading AprilTag #0

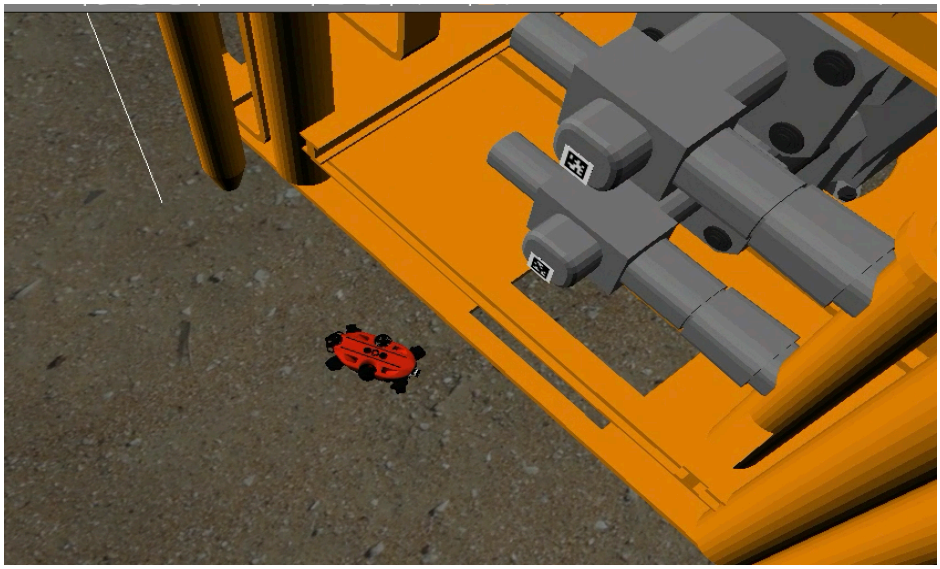

Figure 25. The UUV near AprilTag #1

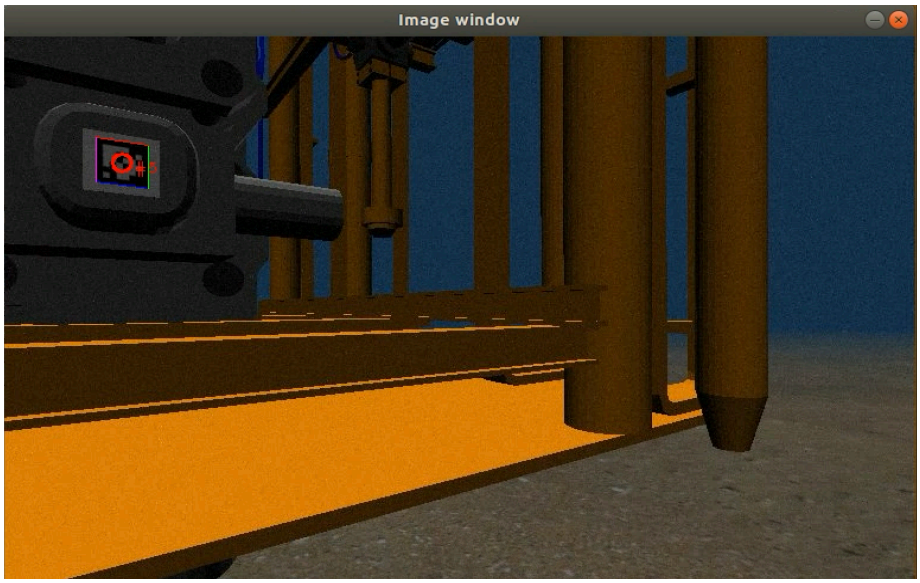


Figure 26. The UUV reading AprilTag #5

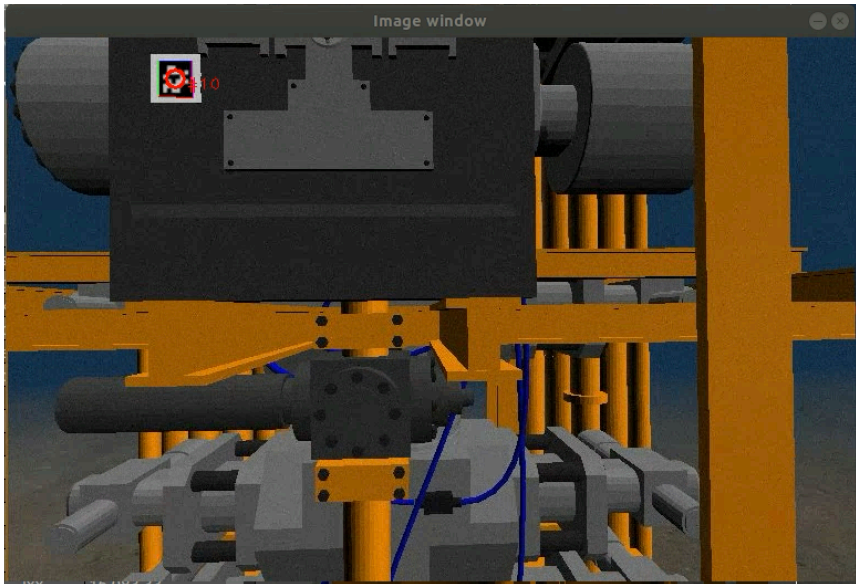

Figure 27. The UUV reading AprilTag #10

Figure 28. The UUV reading AprilTag #1

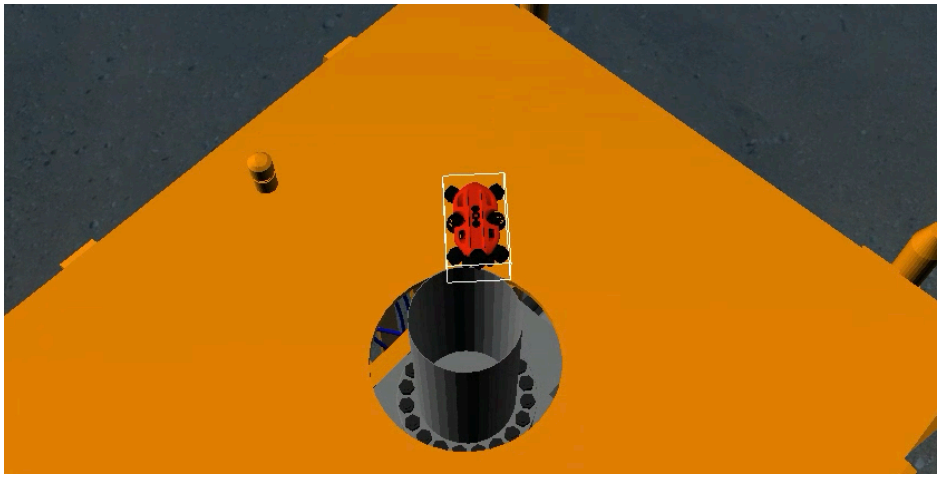

Figure 29. The UUV at the oil pipe.

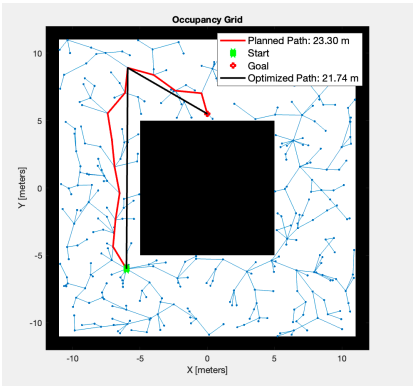


Figure 30. From Initial to AprilTag #0.

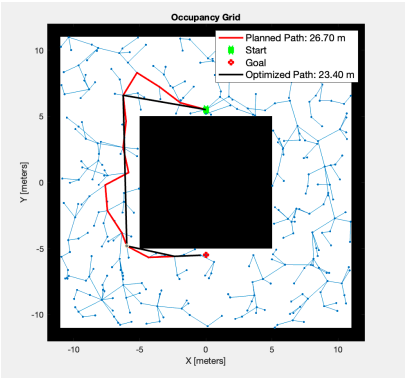


Figure 31. From AprilTag #0 to AprilTag #5.

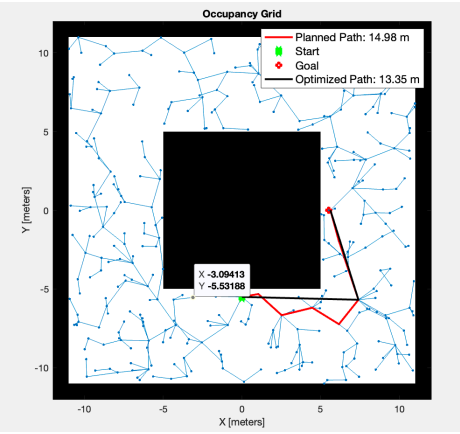


Figure 32. From AprilTag #5 to AprilTag #10

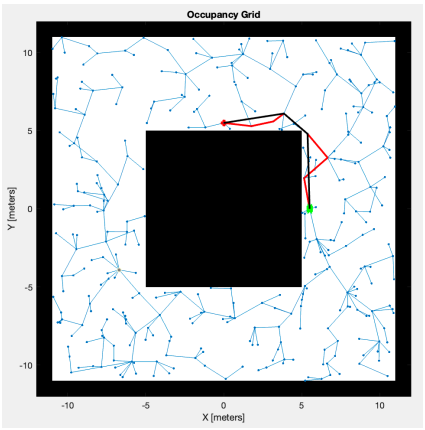


Figure 33. From AprilTag #10 to AprilTag #1.

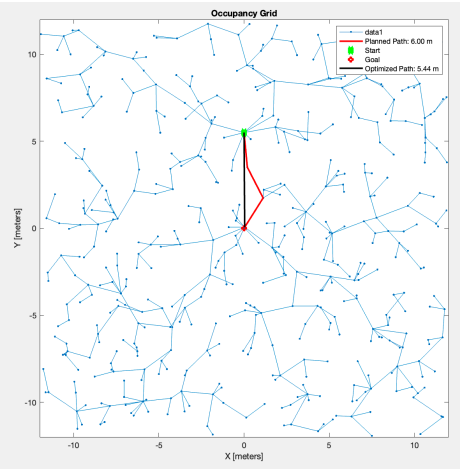


Figure 34. From AprilTag #1 to above oil pipe.

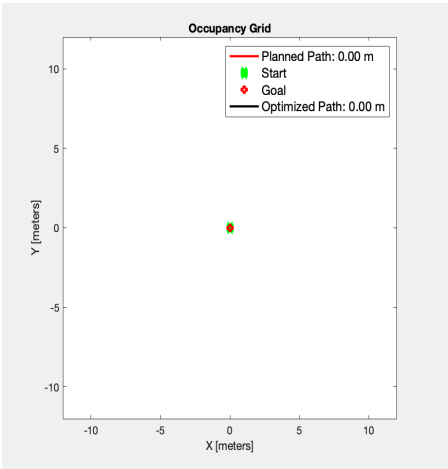


Figure 35. The UUV lowers into the oil pipe.

The Fusion UUV executes the final run satisfactorily. The run was conducted at least three times, and all runs were similar. The heading and depth controllers were implemented similarly to Project 1 with minimal changes to the gains. The Fusion UUV used the motion planner plannerRRTStar to navigate its way around the BOP to the desired AprilTags. The final run took about 13 minutes.

The group encountered difficulties towards the end of the final run, because the UUV would approach the top of the BOP, but it would not go to the oil pipe. The vehicle either became stuck on something and unable to free itself, it would land next to the pipe but not on it, or the code would encounter an error during the approach to the oil pipe and the vehicle would

drift away. This was remedied by accounting for the vehicle's rpm as it approached the oil pipe, the watch circle radius for the vehicle, as well as the height the vehicle should be at when it approached the pipe. The rpm of the vehicle was designed to lower as it approached the oil pipe, and the watch circle radius was decreased for when the vehicle approached the oil pipe. The group also had the vehicle approach at a shallower depth to prevent it from getting stuck on something. These three adjustments allowed the vehicle to safely navigate to the oil pipe, as shown in Figures 29, 34, and 35.

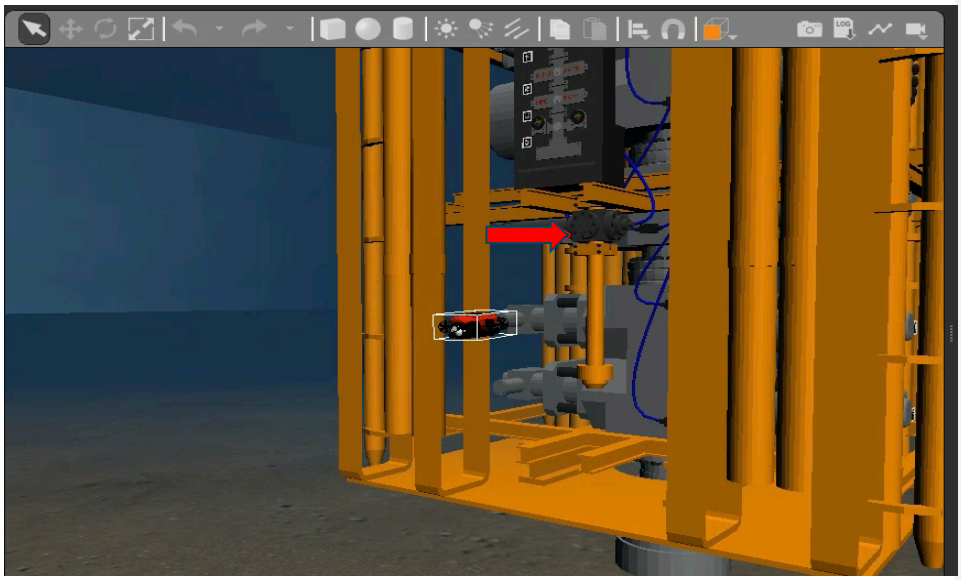

Figure 36. The missing AprilTag #11

The group ran into some difficulty when attempting to perform a random run. AprilTag #11 does not seem to appear on the BOP. The red arrow in Figure 36 points to where the group believes AprilTag #11 should be. The group is unsure why the tag does not appear, but if AprilTag #11 appears as a required waypoint in the random run, the run will be unsuccessful. However, a random run was completed successfully as shown in the figures below. The sequence of the random run was inital position, AprilTag #0, AprilTag #3, AprilTag # 9, oil pipe.

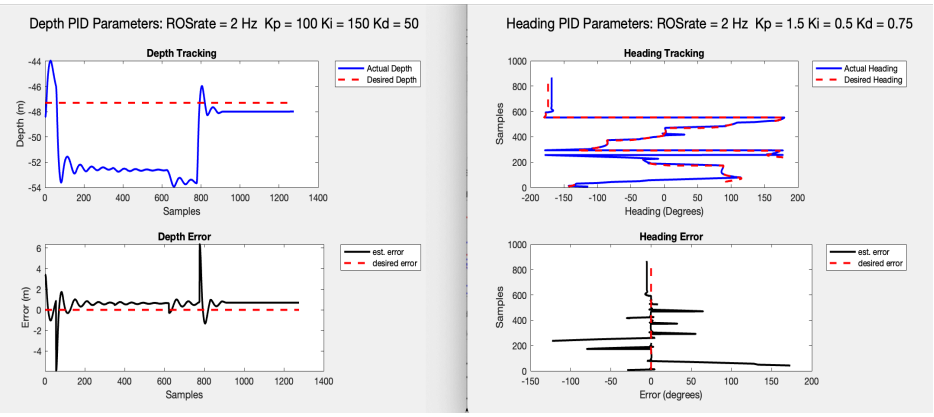


Figure 37. Random run depth and heading graphs.

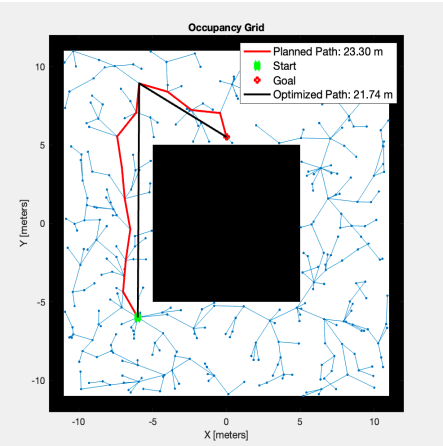


Figure 38. Initial position to AprilTag #0.

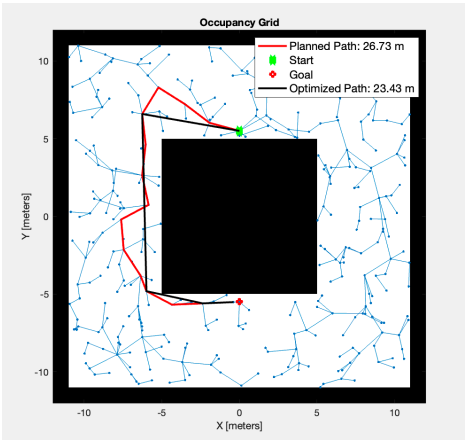


Figure 39. AprilTag #0 to AprilTag #3.

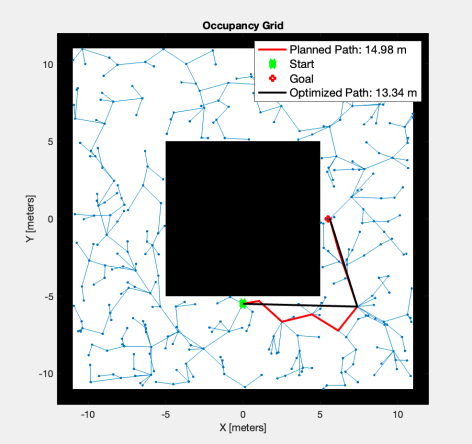


Figure 40. AprilTag #3 to AprilTag #9.

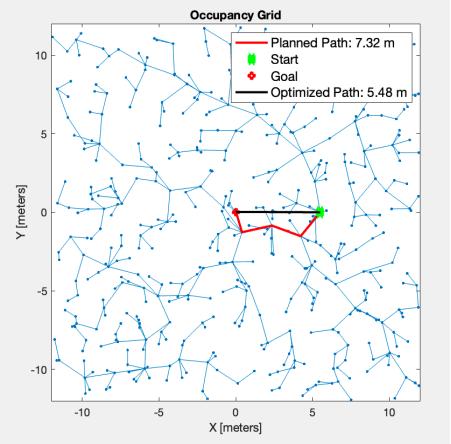


Figure 41. AprilTag #9 to the oil pipe.

## III. Conclusion and Future Work

This project was built on the code the group developed in project 1. However, there was still significant effort required to allow the Fusion UUV to safely and efficiently navigate the BOP and scan the AprilTags. Just as for project 1, the group opted to work together and share individual developments throughout the process. Joseph Young supplied most of the core code for project 2, and he developed and refined many of the core functions needed. Teddy Herrera evaluated many of the available motion planners and suggested the group use plannerRRTStar. Jeré Combs provided preliminary development and analysis of applying a motion planner to the Fusion UUV in a 3D space. As Joseph was working on the code, Teddy and Jeré did much of the report write-up and analysis.

**Formatted:** Line spacing: single