

MEC559 - Final Project Report

**Motion Planning Algorithms to Enhance the RRT Motion Planner**

Group Members

[Prince-David Malendele](mailto:prince-david.malendele@stonybrook.edu)

Sen Qiao

Faculty: Prof. [Nilanjan Chakraborty](mailto:nilanjan.chakraborty@stonybrook.edu)

Department of Mechanical Engineering

Stony Brook University, Spring 2022

May 18, 2022

1. **Introduction**

In order to ensure that mobile robots perform various tasks correctly, it is necessary to study efficient and practical path planning algorithms. Many path planning algorithms such as raster method, visual graph method, A\*, D\*, etc. have their own advantages, but it is difficult to consider the incomplete constraint limitation of mobile robots. The Rapidly-exploring Random Tree algorithm can effectively take into account the incomplete constraints of the mobile robot and can efficiently search the entire solution space to obtain the paths quickly.

The definition of path planning algorithm is: a mobile robot finds a collision-free path from the start state to the goal state in an environment with obstacles according to certain evaluation criteria. One of the classifications of path planning algorithms is into global path planning and local path planning.

Global path planning is based on information that is global to the environment, which includes information that the robot cannot detect in its current state. Global planning stores information about the environment in a graph, and uses this graph to find feasible paths. Global algorithms tend to take a lot of computation time and are not suitable for fast-changing dynamic environments, and they are also not suitable for planning tasks in unknown environments because global path planning requires prior access to global environmental information. Local path planning only considers the instantaneous environment information of the robot, so the computational effort is reduced and the speed is greatly improved. However, local path planning algorithms sometimes do not always allow the robot to reach the target point, resulting in global non-convergence of the algorithm. For mobile robots, path planning considering incomplete differential constraints is one of the challenges in this field. Incomplete differential constraints limit the motion speed of the robot system and the constraints are not integrable, making it impossible to translate such constraints into simple geometric constraints. Random sampling-based path planning algorithms, especially fast random search tree algorithms, can integrate various constraints into the algorithm itself, and thus can solve path planning problems with incomplete differential constraints very effectively.

Rapidly-exploring Random Tree (RRT), proposed by Lavalle, is a random sampling algorithm that uses incremental growth to solve high-dimensional spatial problems with algebraic constraints (due to obstacles) and differential constraints (due to incompleteness and dynamic environments).The advantages of the RRT algorithm are that it does not require modeling of the system, it does not require geometric partitioning of the search area, it has high coverage in the search space, and it can explore as many unknown regions as possible. However, the algorithm is also computationally expensive. Researchers have proposed various improved forms of RRT to solve such problems. For example, Goal-bias RRT algorithm, Bi-RRT algorithm, RRT-Connect algorithm, Extend RRT algorithm, Local-tree-RRT algorithm, Dynamic-RRT algorithm, etc. Among them, Goal-Bias algorithm appears the target node as a sampling point and the probability of the appearance of the target point can be controlled in the algorithm.Extend RRT algorithm, which introduces the set of path points, speeds up the convergence and improves the stability of the path. Bi-RRT algorithm and RRT-Connect algorithm generate two trees from the initial point and the target point until the two trees are connected together algorithm Local-tree-RRT algorithm addresses the problem of narrow passages that are difficult to pass quickly by random sampling algorithm, and proposes a local tree method to solve it. a dynamic RRT algorithm proposes pruning and merging operations to remove invalid nodes before continuing the search.

The random nature of RRT algorithm sampling leads to the final generated paths are often only feasible paths rather than optimal paths.The RRT\* algorithm is also an improved version of the RRT algorithm, which was proposed by S. Karaman and E. Frazzoli in 2011. the main difference between RRT\* and the basic RRT algorithm is that the RRT\* algorithm introduces a search for the neighboring nodes of the newly generated nodes, with the aim of selecting the low cost parent node The RRT\* algorithm is asymptotically optimal and always converges to the optimal solution if given enough running time. Although the RRT\* algorithm solves the optimization problem of the RRT algorithm to some extent, the search for new parents and the rewiring process also make the algorithm much less efficient.

Heuristically Search, also known as Informed Search, uses the heuristic information possessed by the problem to guide the search and achieve the purpose of reducing the search scope and the complexity of the problem. Heuristic strategies can reduce the complexity by guiding the search in the most promising direction. By removing certain states and their extensions, heuristic algorithms can eliminate combinatorial explosion and obtain acceptable solutions (usually not necessarily the best ones).

However, heuristic strategies are extremely error-prone. The heuristic in the problem solving process is merely a guess of the next step to be taken, often based on experience and intuition. Since a heuristic search has only limited information (e.g., a description of the current state), it is difficult to predict the specific behavior of the state space during further searches. A heuristic search may yield a suboptimal solution or nothing at all. This is an inherent limitation of heuristic search. This limitation cannot be eliminated by so-called better heuristic strategies or more efficient search algorithms. In general, the stronger the heuristic information, the fewer useless nodes are extended. The introduction of strong heuristic information has the potential to significantly reduce the search effort, but it does not guarantee to find the solution path (optimal path) with the minimum dissipation value. Therefore, in practical applications, it is desirable to introduce heuristic information that reduces the search effort without sacrificing the guarantee of finding the best path.

1. **Problem Statement**

In this paper, we created 5 environments to run our algorithms, environment 1-5 from simple to complicated. The x and y axis in the results’ graphs means that length of the environments, and the unit of length is meter.

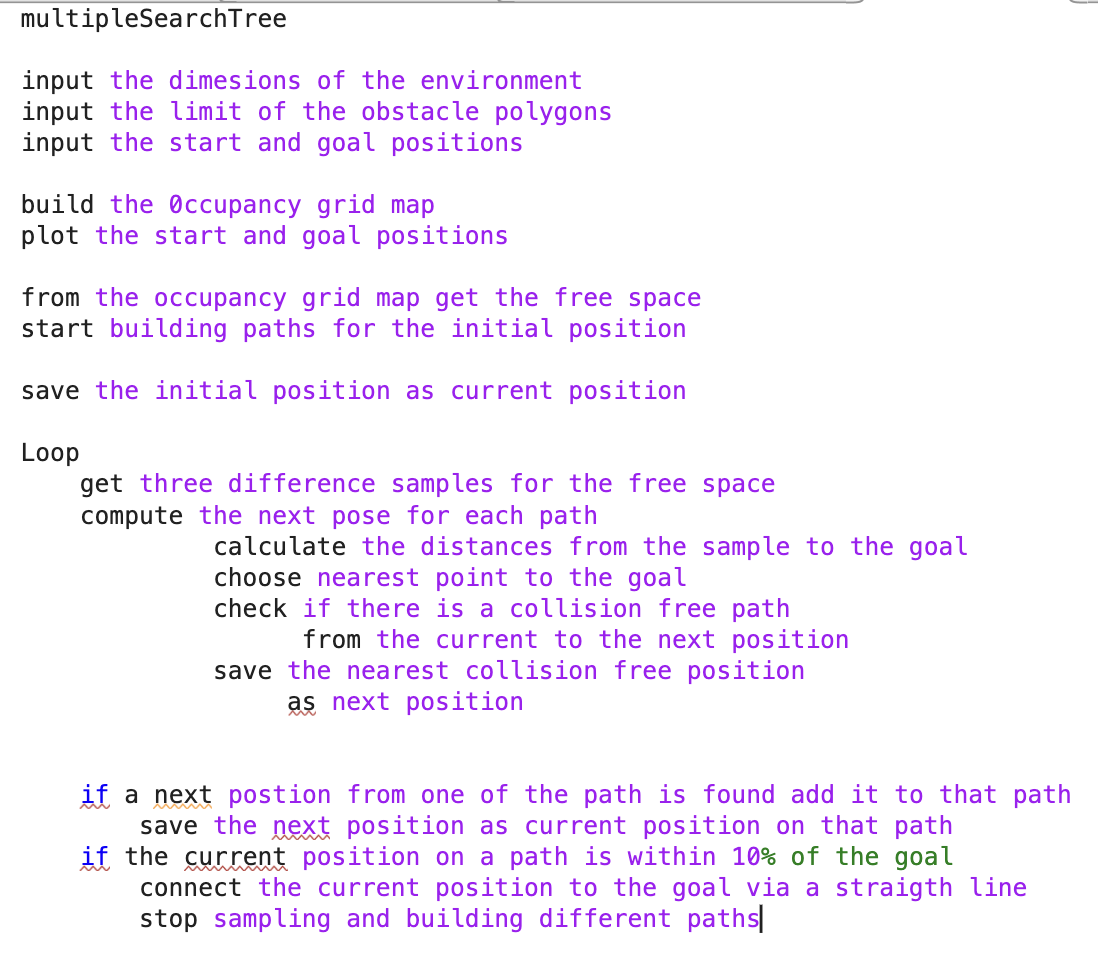
Heuristic sampling: a method for predicting the performance of tree searching programs.

Node expansion: when applied to a node, produces the entire set of nodes that can be produced by applying all of the operators that can be applied to that node. Each application of a successor function to a node is called expanding the node.

Multiple search tree: it can grow trees from start position and goal position at the same time, and when the tree of start and tree of goal meet each other, the right path is created.

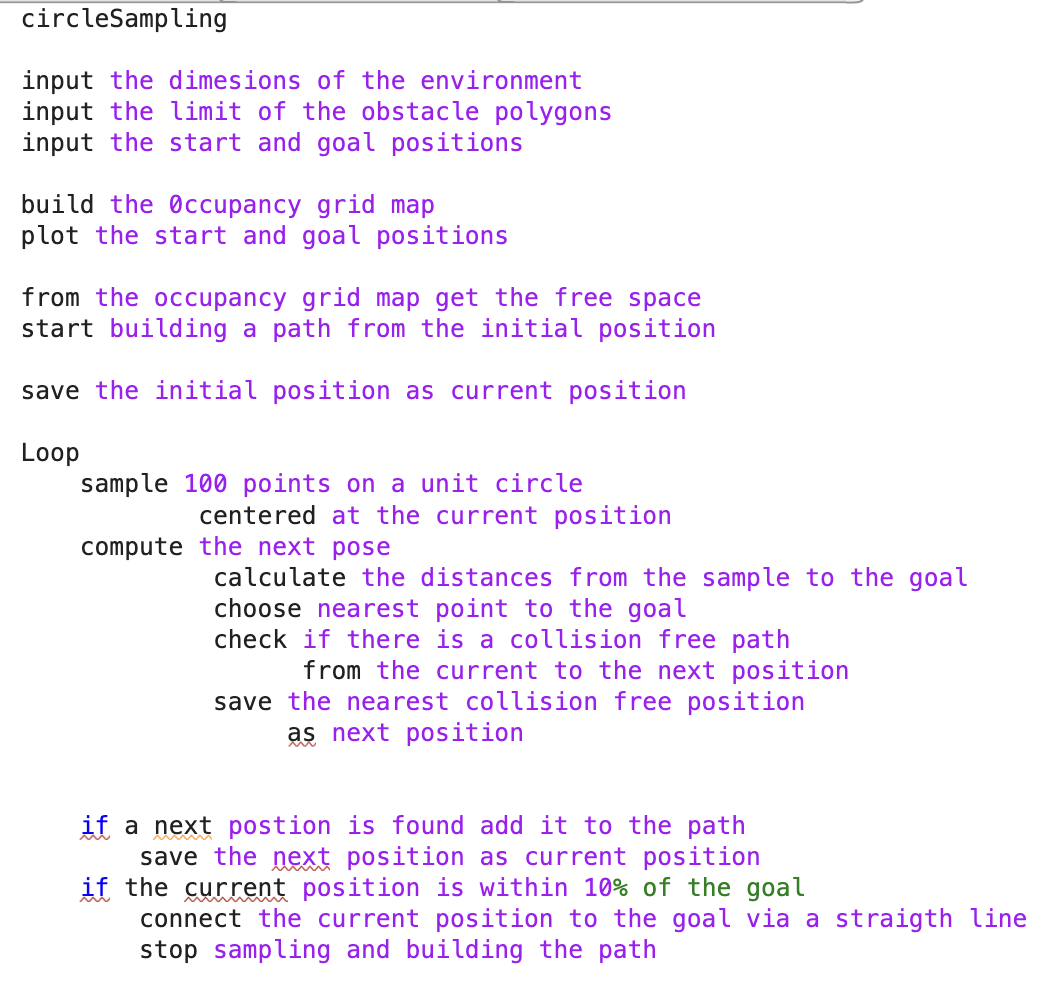
1. **Description of the algorithms**

In this project three different types of enhancements were implemented on the unidirectional RRT planner created in HW3. The first enhancement was using multiple search trees to increase the chances of reaching the goal destination. To do so, three different randoming were performed, and the next pose from each sample was chosen by picking the point on the tree that was the closest to the goal. Once one of three paths created from the tree is close enough to the desired goal, the sampling stops on all paths.

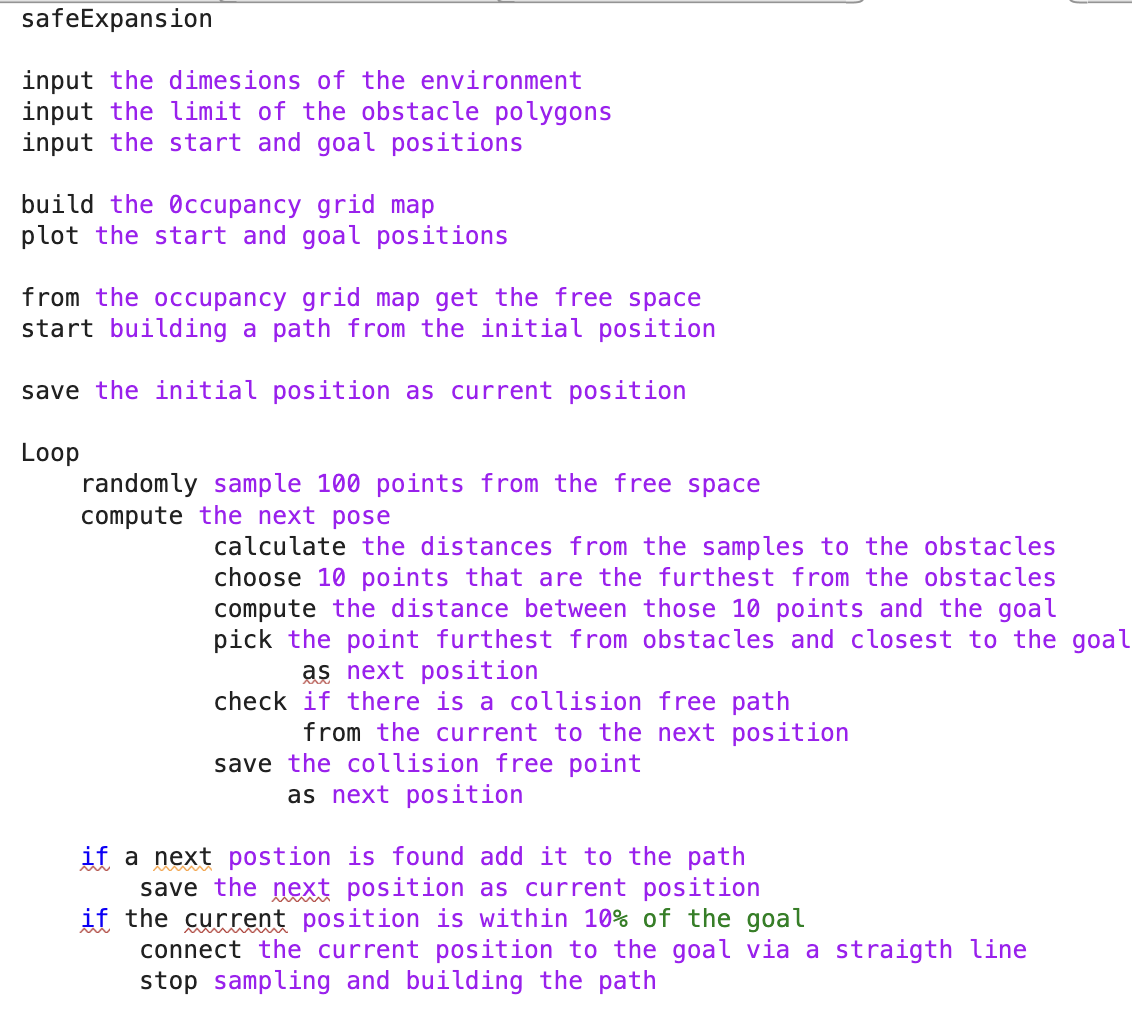
****

*Figure 3.1: PseudoCode for the Multiple Search Tree Algorithm*

The second enhancement was implemented by using a different sampling technique. Instead of randomly selecting possible positions from the free space, the points were sampled by building a unit circle centered at the current position of the robot. This algorithm was implemented to ensure that the points from all directions around the robot are being taken into consideration while sampling for the next position of the robot.

**** *Figure 3.2: PseudoCode for the Circle Sampling Algorithm*

The last algorithm used to improve the unidirectional RRT motion planner used a different heuristic for the node expansion. The multiple tree search and circle sampling algorithms only use the nearest point to the goal from the sample to determine the next position. This may not always guarantee a safe and achievable path. Therefore, the SafeExpansion algorithm consisted of a random sampling from the free space, but the minimal distance between each sample point and the closest obstacle was computed instead. Based on this, the 10 points that were the furthest from the obstacle were selected. Form this subset, the closest point to the goal without obstacle collision was chosen as the next position.

**** *Figure 3.3: PseudoCode for the Safe Expansion Algorithm*

1. **Result and Discussion**

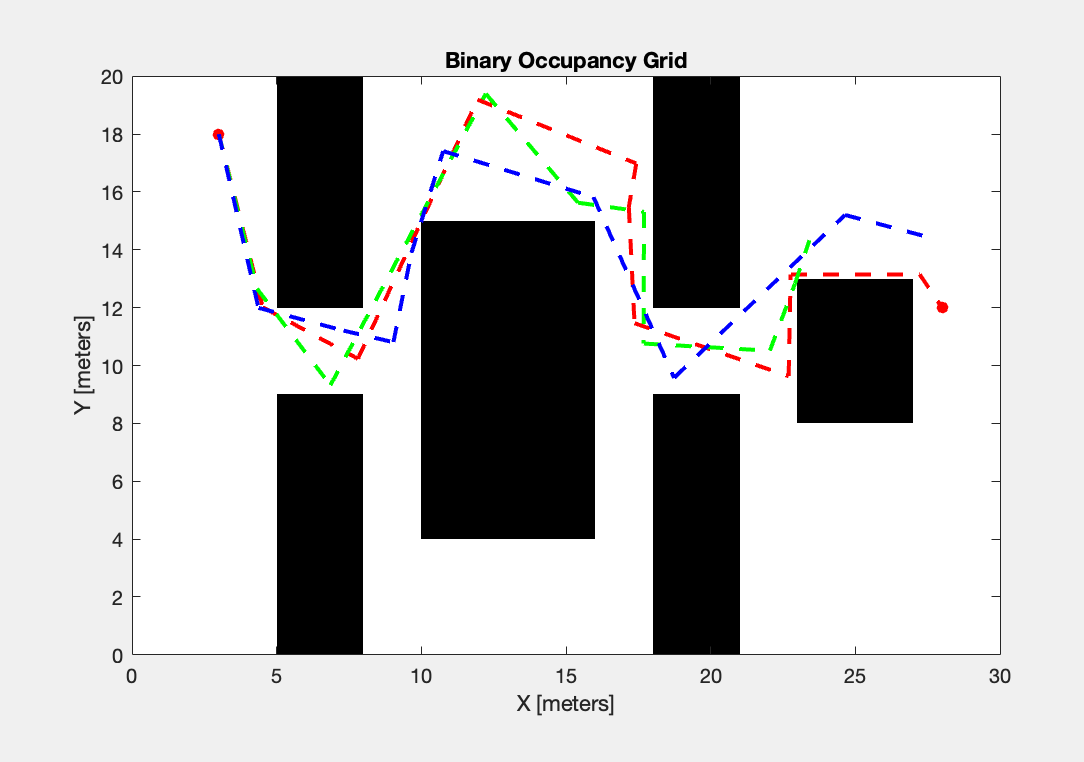
None of the algorithms implemented in this project was perfect. Whenever the code for each algorithm is runned multiple times with the same environment, the algorithm fails to find a path from the start position to the goal position. Environment 1 and 5, the first and last ones, were the post problematic. The algorithms had to be runned 10 to 20 times before one solution was found. In some cases even 20 trials were not enough to find a path from the motion planning algorithm.

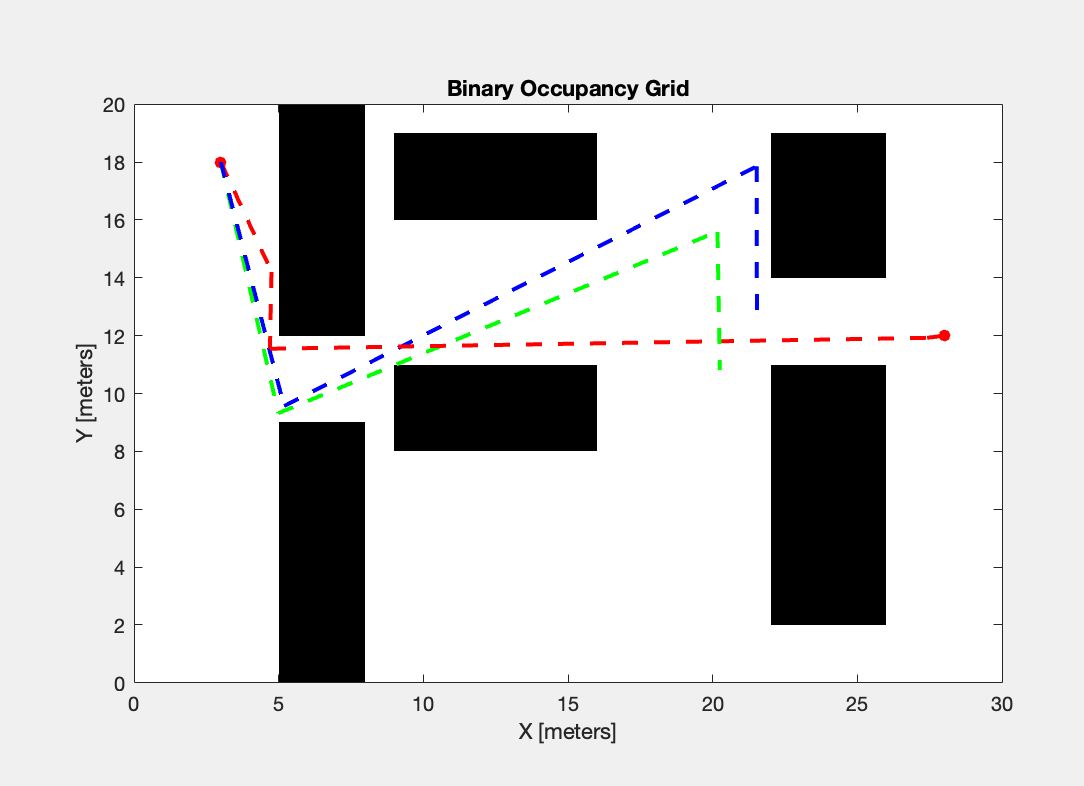
1. **Multiple Search Trees**

Although this algorithm is a search tree algorithm, the plot was made to only the path without much regard to the trees themselves because the efficiency of the algorithm is in its ability to find the best possible path for the robot. The plots are shown from figure 4.1 to 4.5.

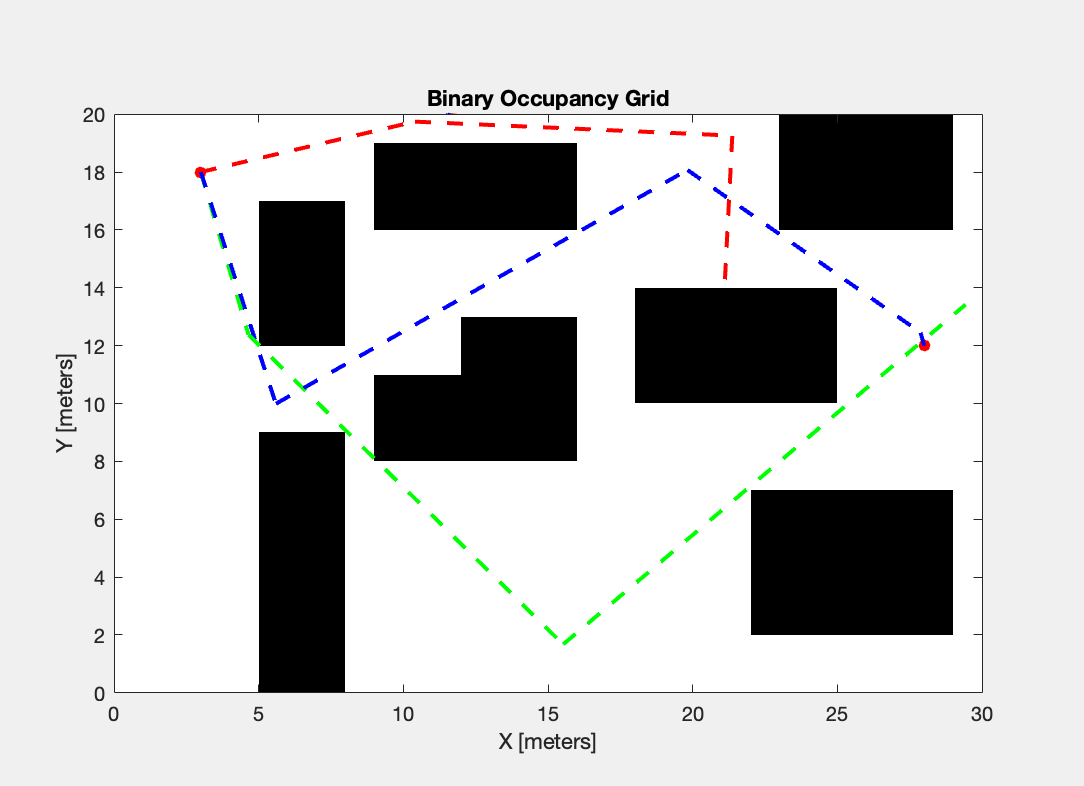
This algorithm worked seamlessly with environment 2, 3, and 4. More than 90% of the time, the algorithm was able to find a path from the start to the goal positions. Out of three paths obtained from the search trees, the shortest path was the designated one. The algorithm was also able to find a path for environment 1 after every 4 to 5 trials, but in environment 5 even 20 trials were not enough to find a single path to the goal. No further iterations were performed, and the path in which the robot was the closest to the goal was presented in figure 4.5.

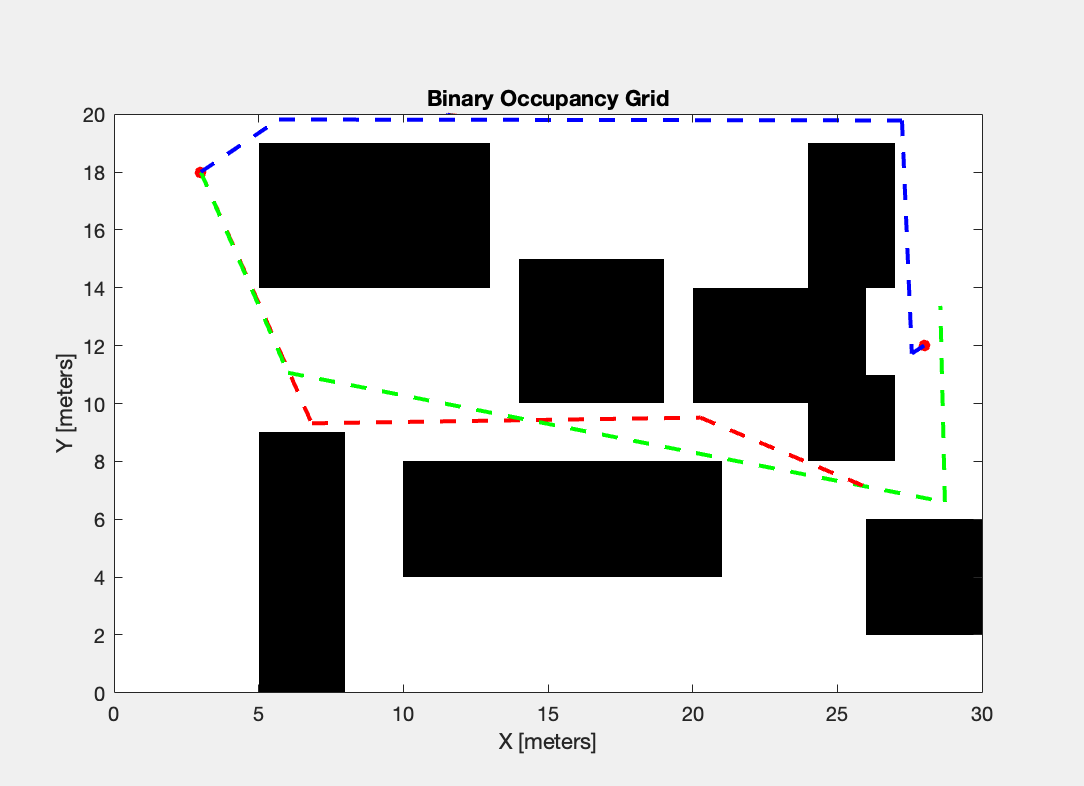
When compared to the basic algorithm used in Homework 3, this algorithm is able to find better paths more efficiently and frequently. This makes sense because using three search trees instead of one improved the probability of success of the algorithm.

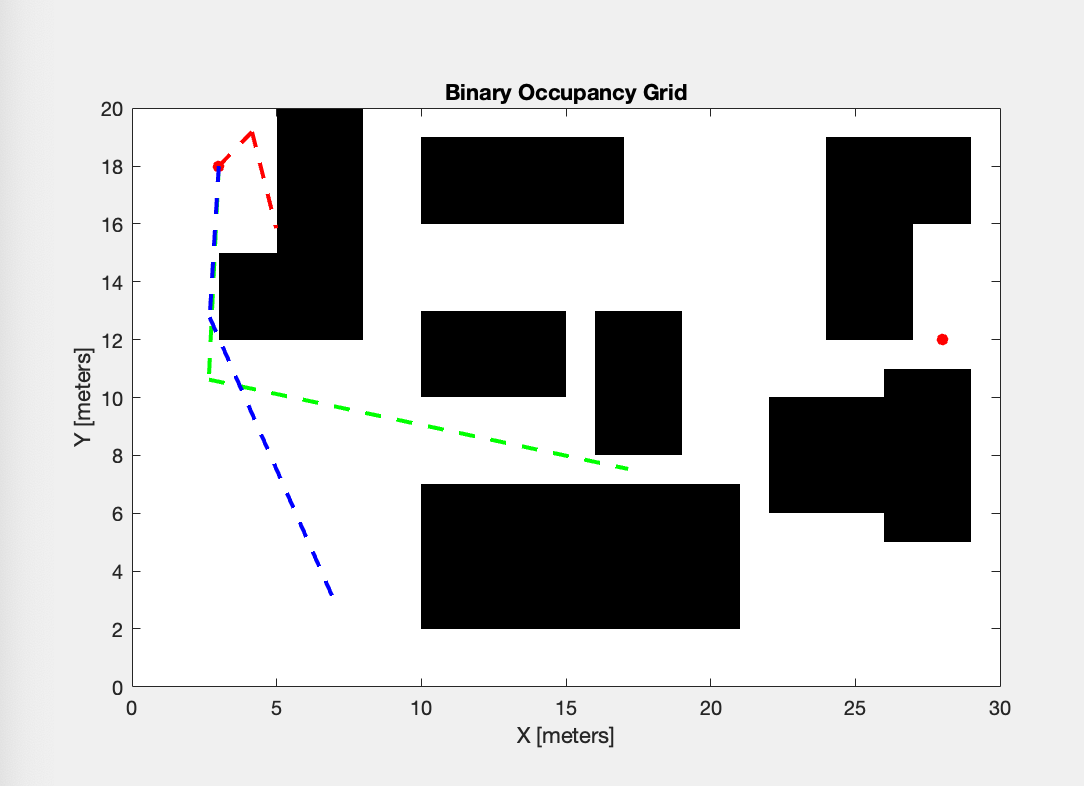
****

*Figure 4.1: Multiple Search Algorithm and Environment 1*****

*Figure 4.2: Multiple Search Algorithm and Environment 2*

**** *Figure 4.3: Multiple Search Algorithm and Environment 3*

**** *Figure 4.4: Multiple Search Algorithm and Environment 4*

****

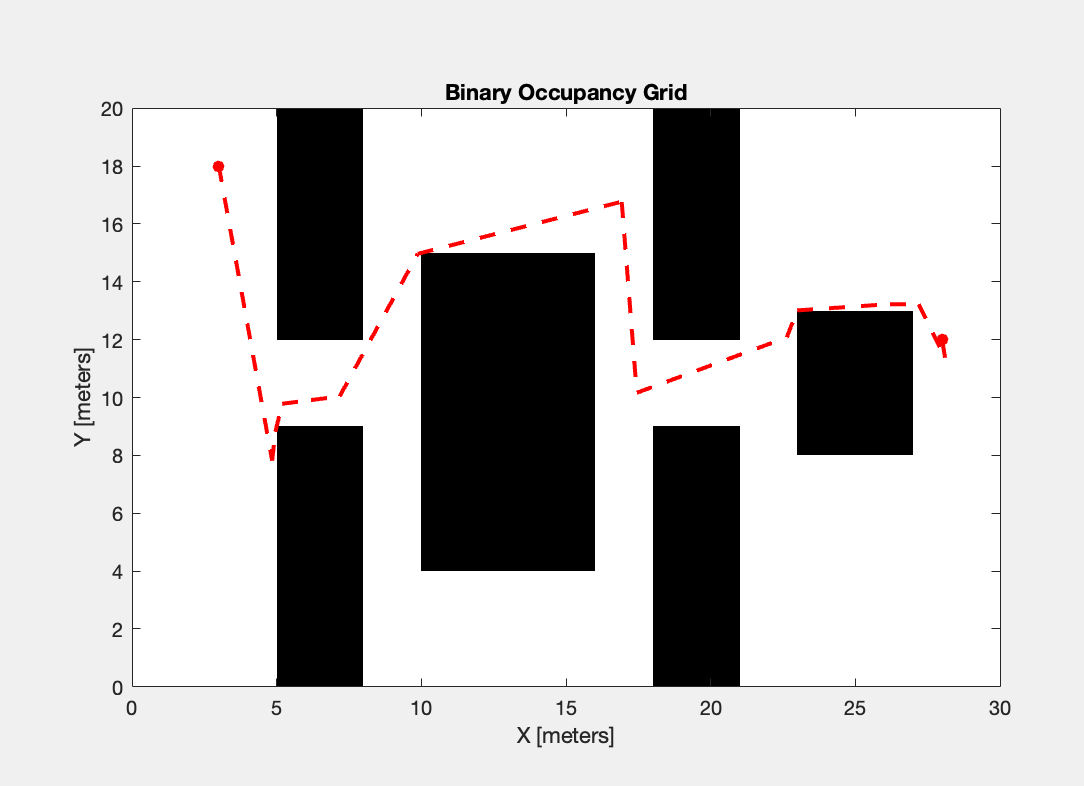
*Figure 4.5: Multiple Search Algorithm and Environment 5*

1. **Sampling Heuristics - Sampling in Circles**

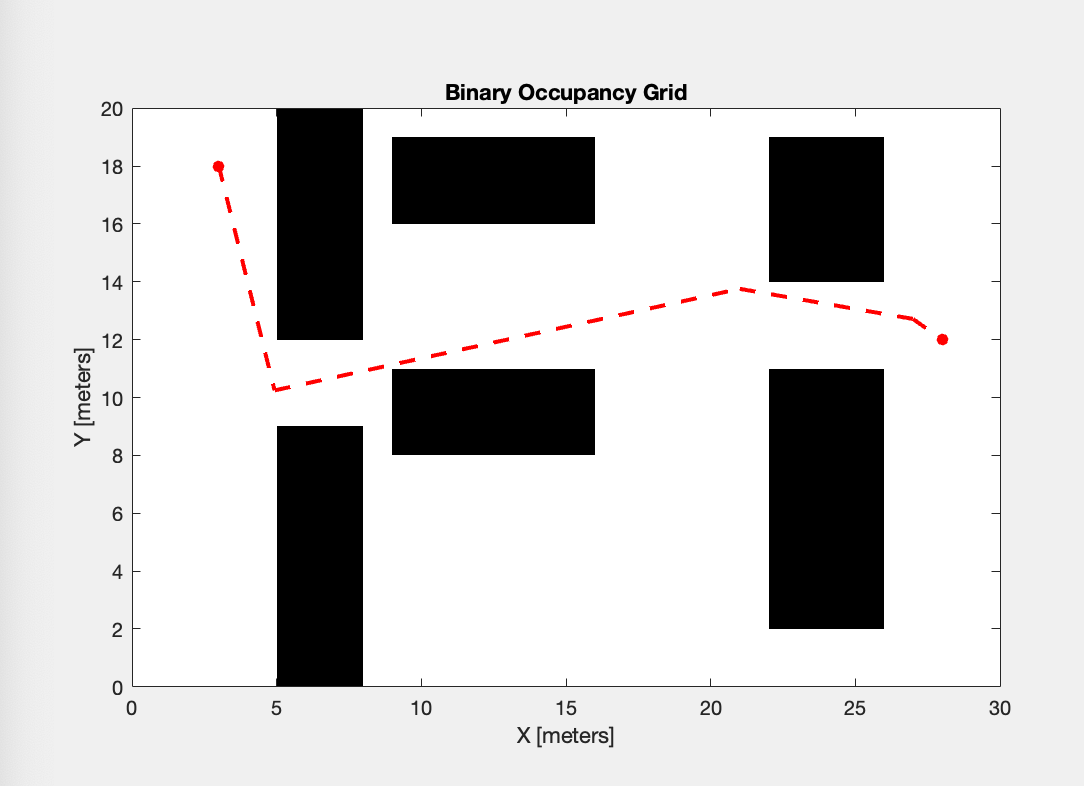
The plots for this motion planning algorithm are shown from figure 4.6 to figure 4.10.

Similar to the multiple search algorithm, the circle sampling algorithm also performed extremely well in environ 2, 3, 4. It had some difficulties in environment 1, and failed to find a path even after 20 trials in environment 5. Figure 4.10 shows the closest this algorithm was to reaching the goal. Since positions were sampled with a unit circle without consideration to the free space, the robot once went beyond the limits of the environment. Therefore, this algorithm should be improved by adding constraints regarding the boundaries of the environment.

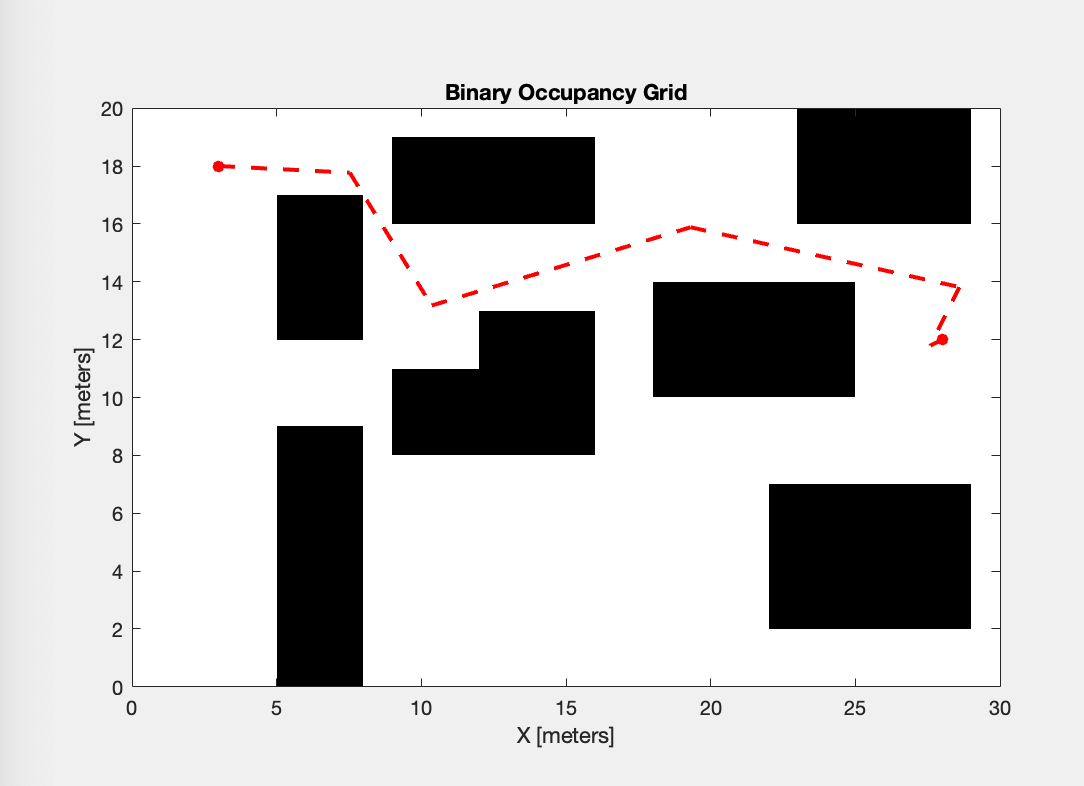
In terms of performances, the paths found in this algorithm were quite similar to the one used in Homework 3. Although this algorithm provides a good way to sample within the surroundings of the robot, it requires further constraints and modifications to be a more efficient way to reach the goal position.

****

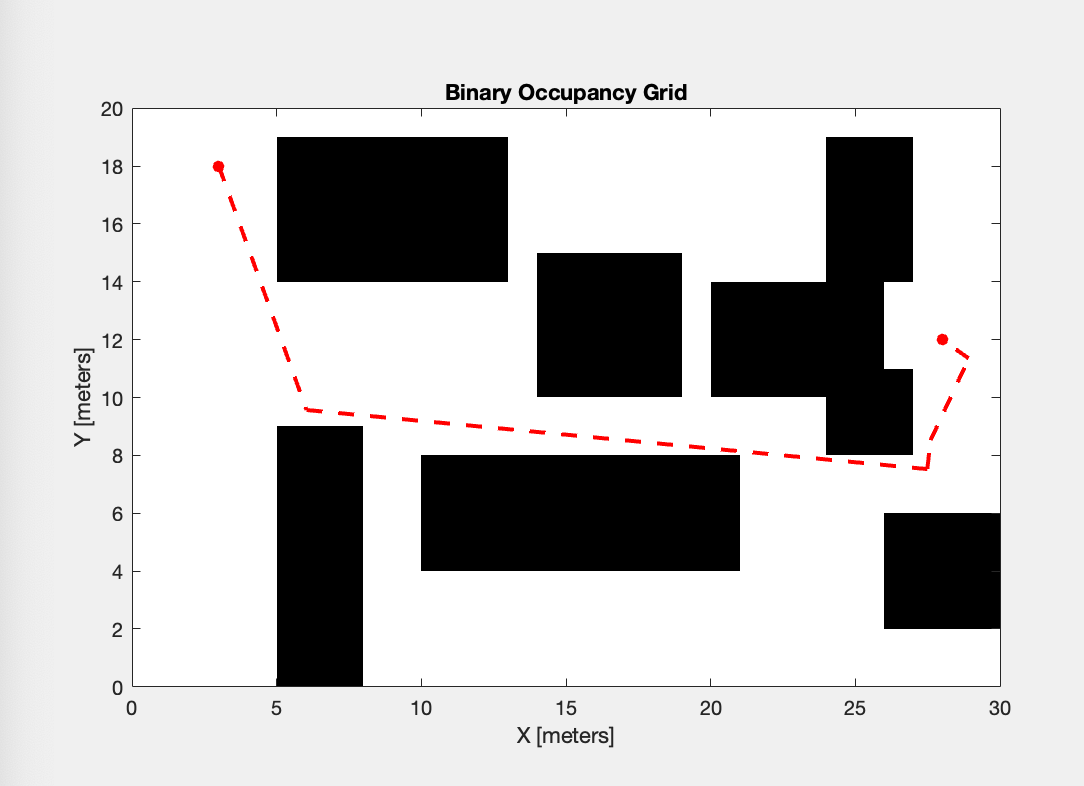
*Figure 4.6: Circle Sampling Algorithm and Environment 1*

****

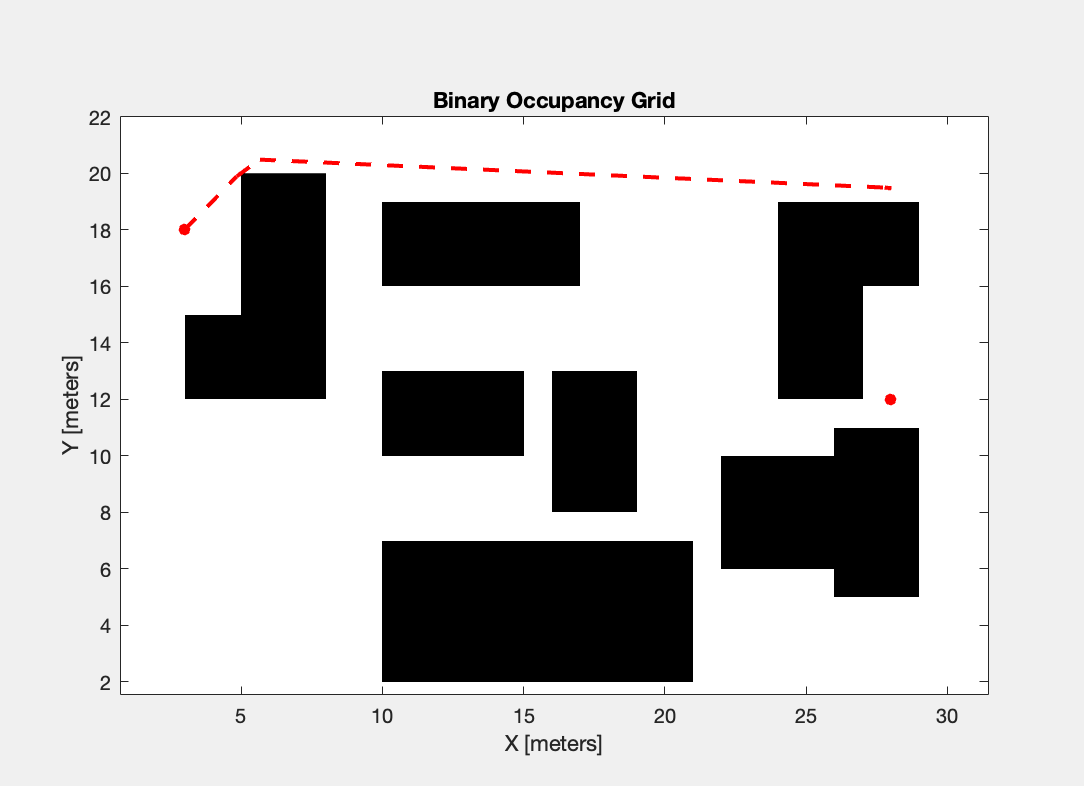
*Figure 4.7: Circle Sampling Algorithm and Environment 2*

****

*Figure 4.8: Circle Sampling Algorithm and Environment 3*

****

*Figure 4.9: Circle Sampling Algorithm and Environment 4*

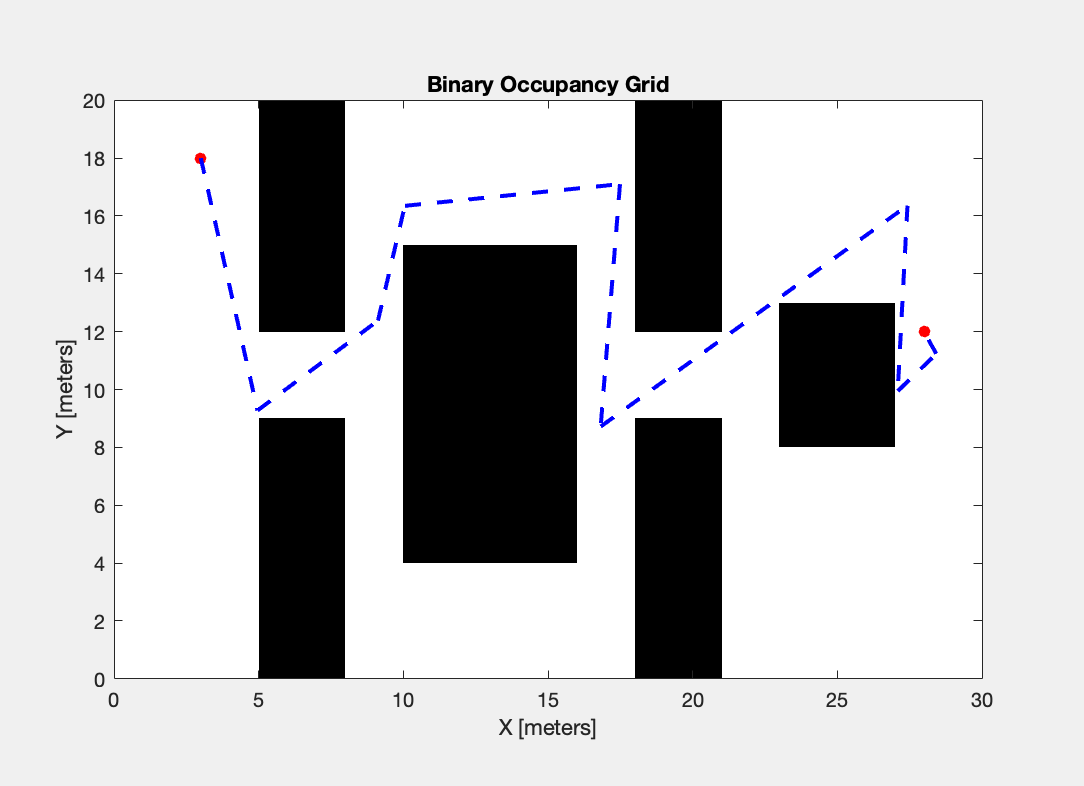
****

*Figure 4.10: Circle Sampling Algorithm and Environment 5*

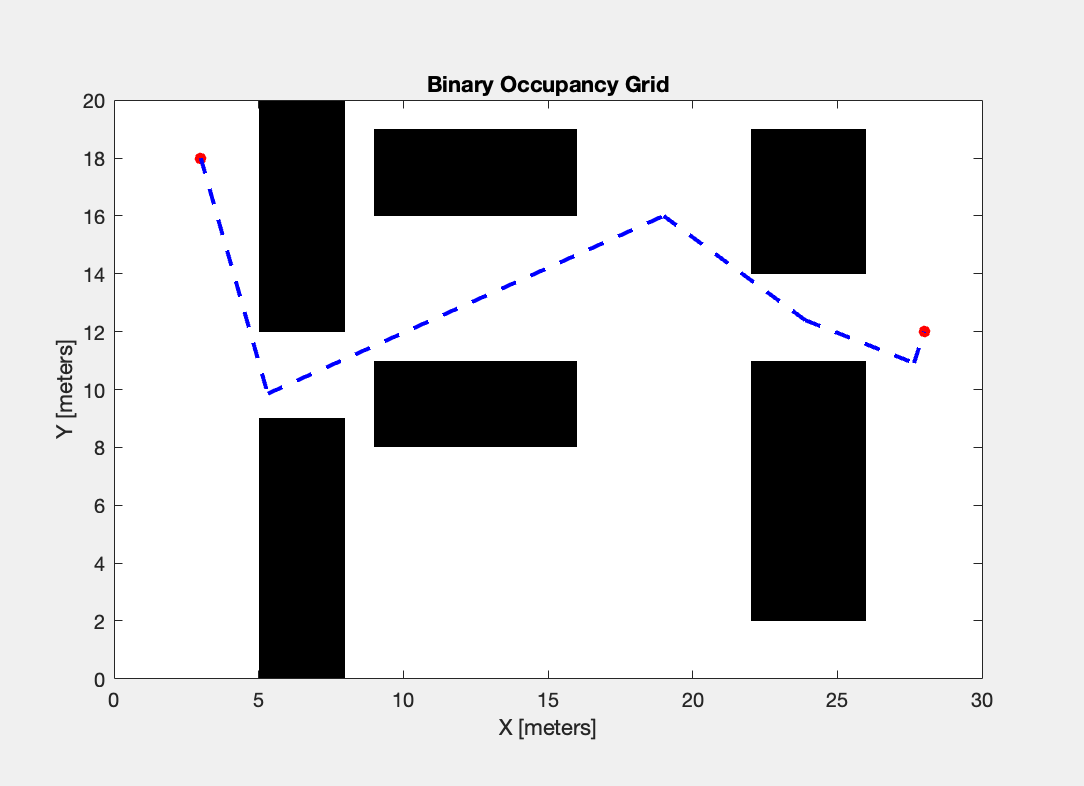
1. **Heuristics for Node Expansion - Safe Expansion**

This motion planning algorithm was mostly implemented to successfully find a path in environment 5. It differs from the basic algorithm implemented in Homework 3 because it adds an extra safety constraint by sampling from a subset of points that are the furthest away from obstacles. The plots for this motion planning algorithm are shown from figure 4.11 to figure 4.15.

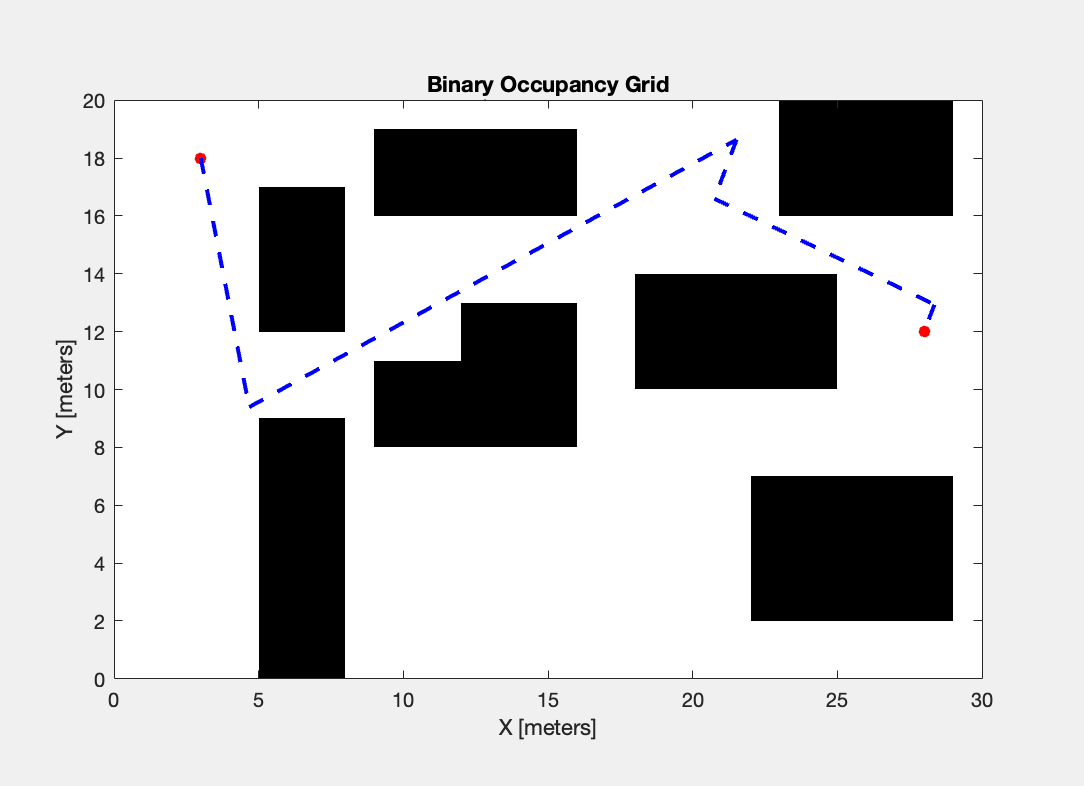
This algorithm was the most successful in all the five environments that were used in this project. It worked more than 80% of the time in environments 1, 2, 3 and 4. In environment 5, it was able to find a path in at least 3 out of 20 trials. Compared to the paths found in Homework 3 and with the other two algorithms used in this project, the found with the safe node expansion were the least likely to cause any obstacle collision.

****

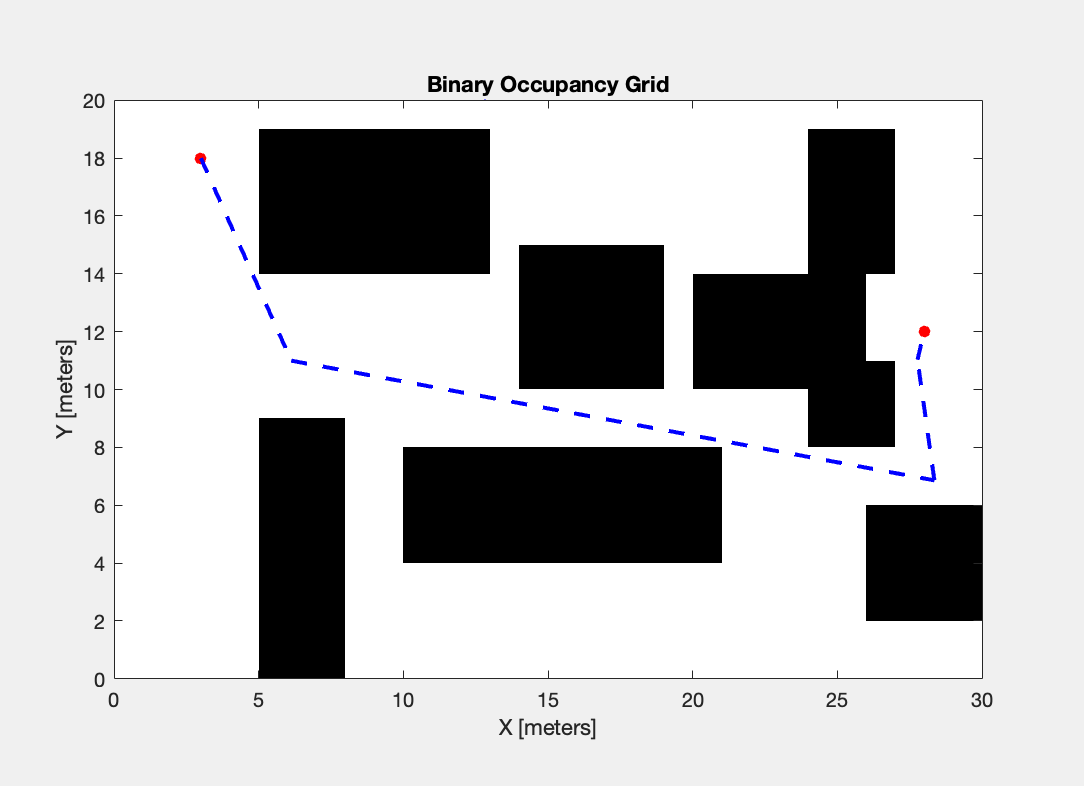
*Figure 4.11: Safe Expansion Algorithm and Environment 1*

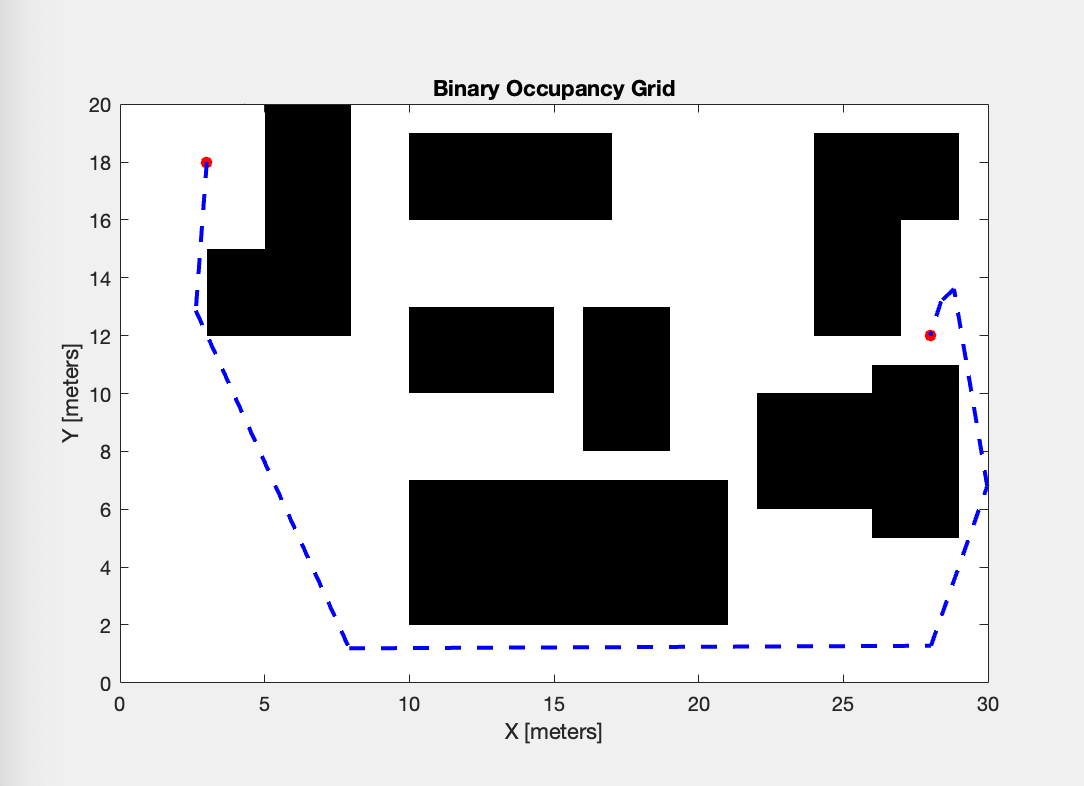
****

*Figure 4.12: Safe Expansion Algorithm and Environment 2*

****

*Figure 4.13: Safe Expansion Algorithm and Environment 3*

****

*Figure 4.14: Safe Expansion Algorithm and Environment 4*****

*Figure 4.15: Safe Expansion Algorithm and Environment 5*

1. **Conclusion**

In short, every algorithm has its own advantages and disadvantages. For the multiple search trees method, it can find many ways to get the goal position, and no need to model the system and geometric partitioning of the search area, it can explore unknown areas. But every time run the basic RRT algorithm, it will take so long to find the right path to the goal, even not the best way to the goal, so that’s why this algorithm needs to be improved. For the heuristic sampling algorithm, it can find the path to the goal more quickly than the basic RRT, because of sampling from the environment, it needs more constraints to limit the robot. And for the node expansion algorithm, it can finish the job in all environments, especially in more complicated environments, but may need more time to run the algorithm. So we can choose an algorithm that based on our major requirements, every algorithm has its limitations and strength.