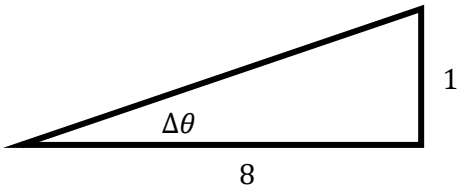
MINI PROJECT – PART 2

Robot Navigation

Table of Contents

1 Choosing the θ resolution

To achieve a valid resolution to make sure that a single rotation of the robot will not result in a greater than 1 unit of distance of any point on the robot, we used a simplistic criterion (as a minimal bound) by simply taking the worst-case scenario and finding the critical value for $\Delta\theta$. Assume the robot (with a length of 8 units) rotates by $\Delta\theta$, and assuming its vertical distance changes by a critical 1-unit value.



$$\Delta\theta \leq \sin^{-1}\left(\frac{1}{8}\right) = 7.18° \quad \Rightarrow \quad N_\theta = 64 > \frac{360°}{7.18°} = 50.14$$

We chose $N_\theta = 64$ to double the initial resolution of 32 and ensure that is greater than the critical value of 50.14 we found. Additionally, we made sure that all algorithms achieve a path plan given this resolution. Thus, the new resolution is given by the grid of size $32x32x64$.

2 Picking a path-planning method

Initially, we implemented the DFS algorithm, and it didn't perform well. This could have been expected because DFS prioritizes "depth" first, meaning that it looks for a solution as fast as possible, but results in a suboptimal path. Then, we tried a similar algorithm Breadth First Search (BFS) which finds all paths from the start to goal and returns the shortest path. This method resulted in an efficient and accurate solution. Additionally, we chose to implement the A* algorithm to investigate the differences between the path finding algorithms, and we wanted to see the performance of adding a heuristic to the path finding process. The A* algorithm also performed well and yielded a very similar solution to that of the BFS.

3 Implement the chosen path-planning method

We implemented the A* algorithm and found a solution taking 107 steps as shown in Figure 3.1.

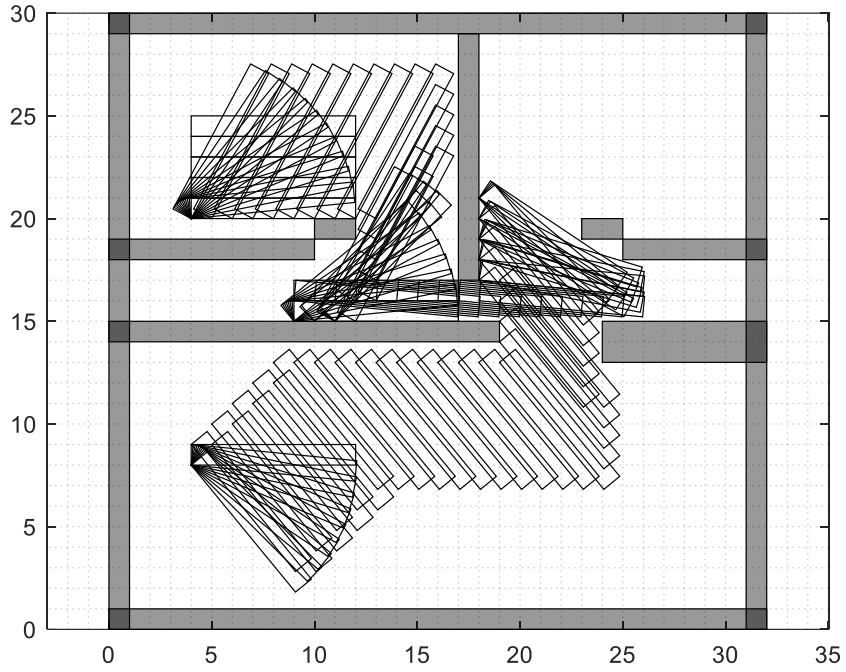


Figure 3.1 - A* algorithm path silhouette. The path was of length of 107 steps.

We also added the results we obtained from the depth-first search (DFS) and breadth-first search (BFS) algorithms to illustrate their differences and performance. Clearly, the DFS algorithm does not suit this type of problem as it prioritizes finding a solution as fast as possible as opposed to BFS which prioritizes finding the shortest path. It is interesting to see how A* prioritizes to minimize the cost (in this case – distance) between the start and goal in the first few steps as opposed to BFS which just finds a valid shortest path solution. This phenomenon is also seen in the last few steps, where the robot reaches the goal location diagonally in A* as opposed to horizontally and then vertically in BFS.

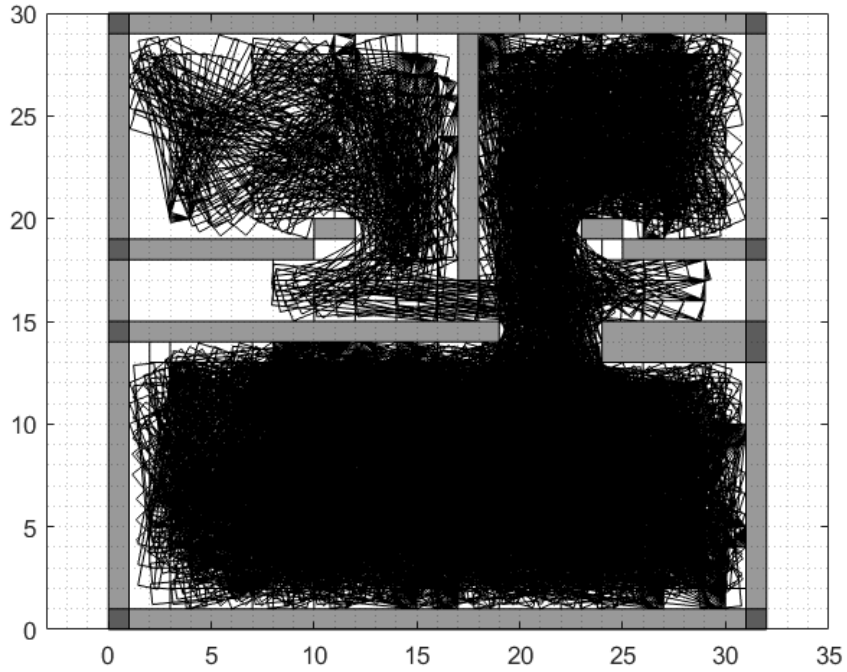


Figure 3.2 - DFS algorithm path silhouette. The path was of length of 3389 steps.

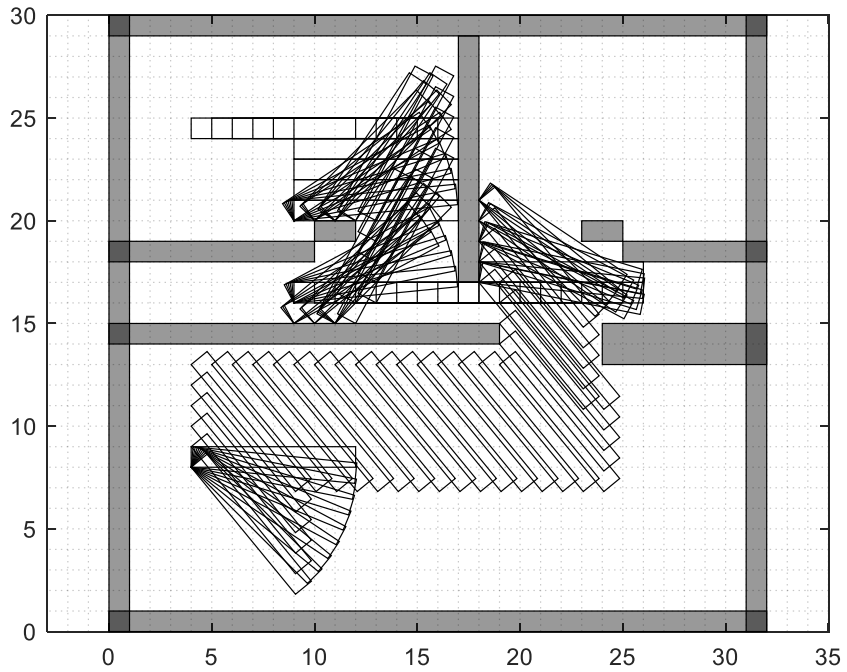


Figure 3.3 – BFS algorithm path silhouette. The path was of length of 107 steps.

3.1 Ellipsoid Theorem illustration

To create a visual representation of the ellipsoid, we modified the task by setting an environment including only the 3rd boundary. This ensured the path will be straightforward. We also modified the A* algorithm to let it continue running even if it finds the path, until it reaches its stop criteria of the OPEN list to be empty.

The found path for the modified task is shown in Figure 3.6. We then saved the closed nodes and their associated total costs by order and plotted them in 3D space, where the x and y axes represent the 2D plane and the z axis represents the total costs associated with each closed node (f = g + h). In addition, we added a plot showing the closed nodes costs increasing by the order of nodes closing in the search. Note that due to a limitation in the space grid values of x>0 and y>0, the ellipses are "cut" at those values because the algorithm skips those nodes.

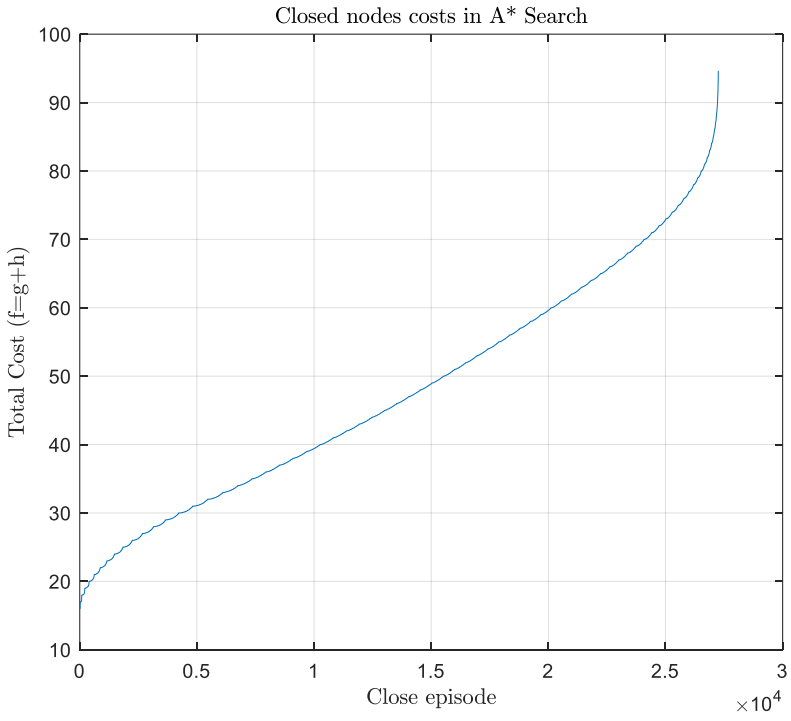


Figure 3.4 - Costs associated with the closed nodes by order in the A* algorithm. It is shown that their values are increasing representing the growing ellipses with the start and goal positions at the loci.

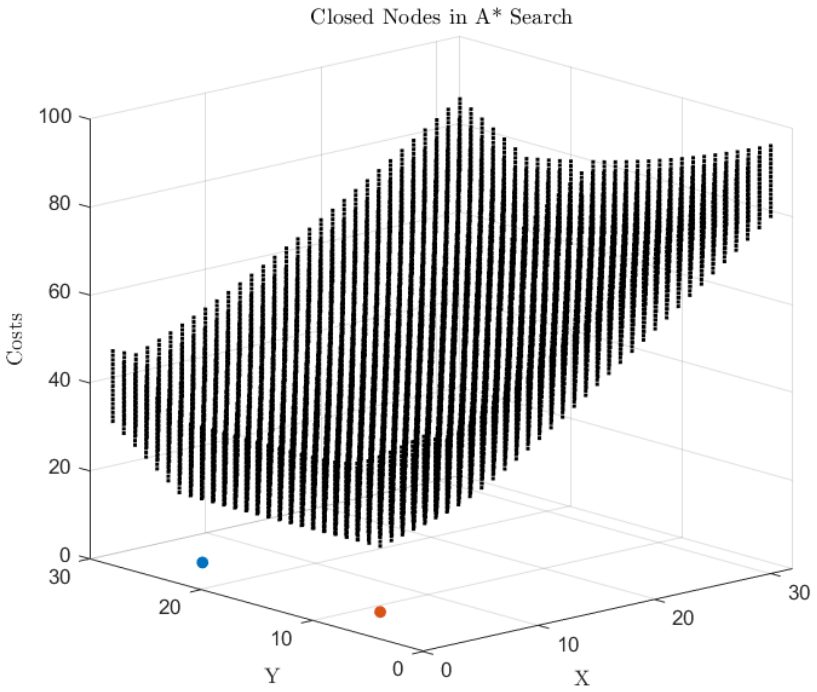


Figure 3.5 - The closed nodes in the modified task in A* search, when neglecting the stop criteria when the path is found. The loci of the ellipses are shown in the start (blue) and goal (red) of the task, and the ellipses are expanding as indicated by the z axis.

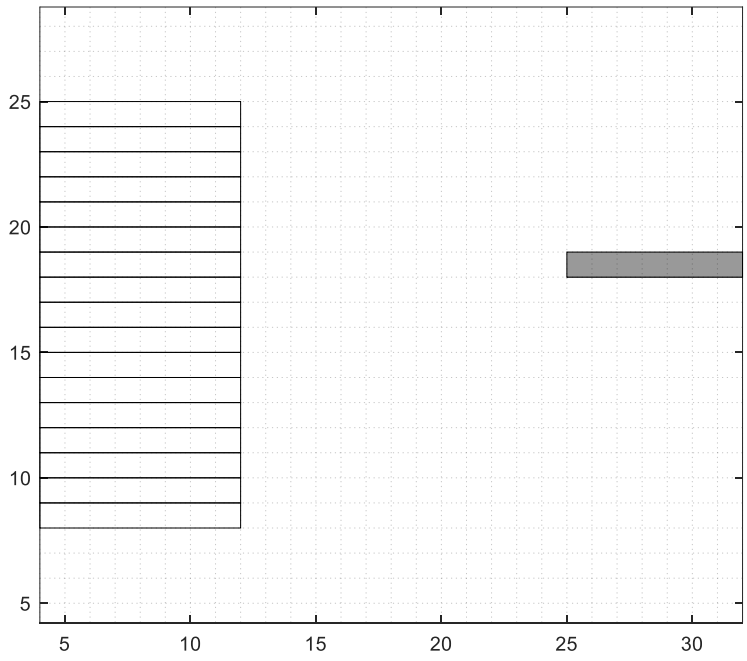


Figure 3.6 - Modified task involving the start and goal with only the third boundary.

4 A* question

By using the A* algorithm neighboring node validations, we can let it run without knowledge of the entire grid, but rather only for the neighboring nodes in the grid and update them in each iteration. We initialized the robot to the start position and checked the neighboring nodes in each iteration. At each step, it calculates the A* cost $f = g + h$ for each accessible neighboring cell when g is the cumulative path length from the start to the neighboring cell and h is the heuristic distance from the neighboring cell to the target. The robot moves to the neighbor with the lowest f cost, updating its position and scanning again. This process continues until the robot reaches the target or exhausts possible paths.

To avoid unnecessary loops of going back and forth to the same nodes, and to make sure the robot reaches the target, we added a penalty for calculating the cost of neighboring visited nodes, and to avoid the revisiting of visited nodes we added a random choice of movement to a valid neighboring node. The resulting path is plotted below, compared to the optimal offline path.

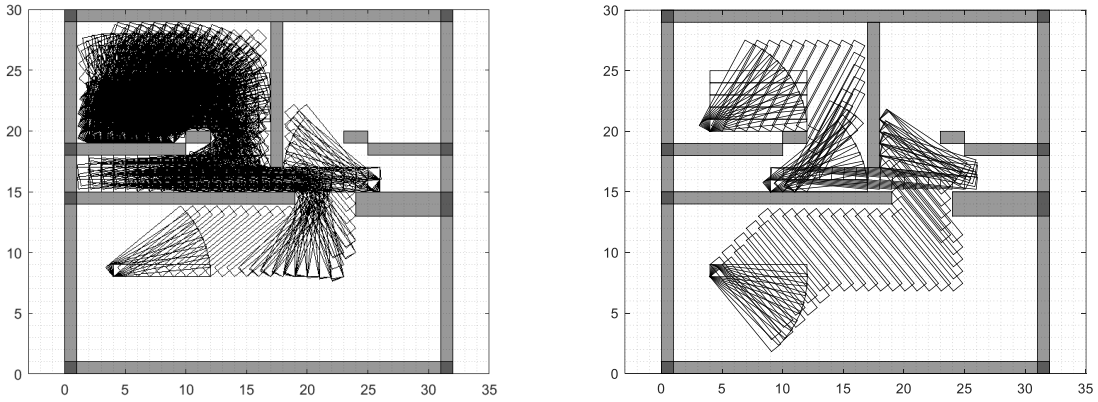
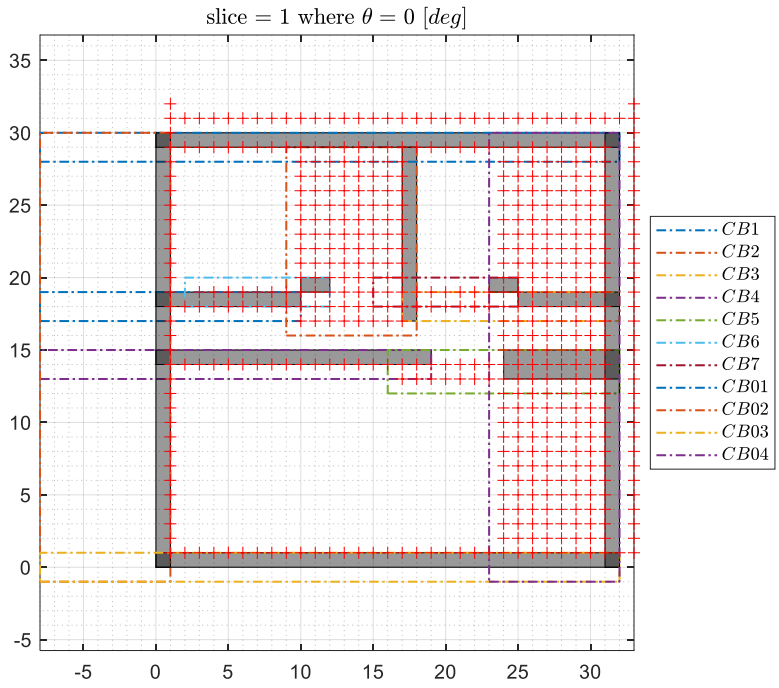
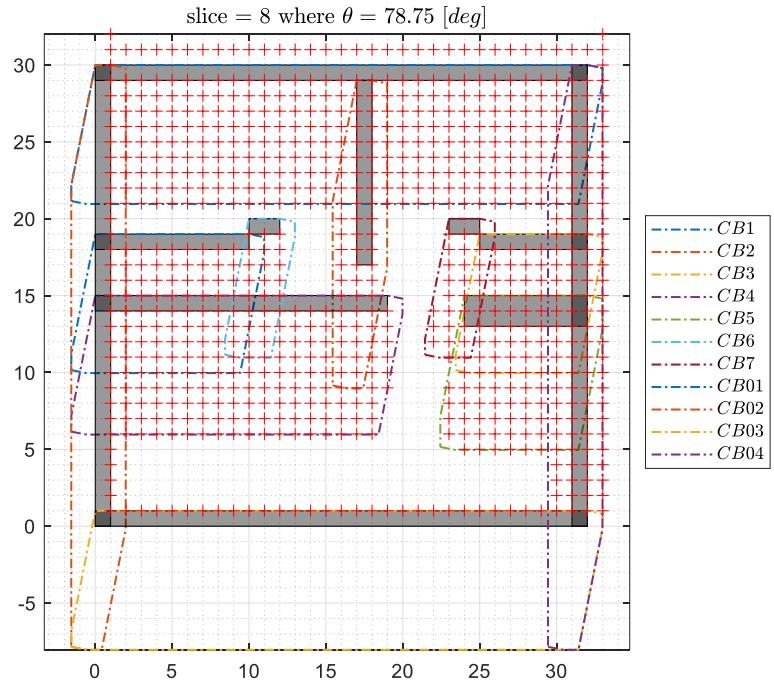


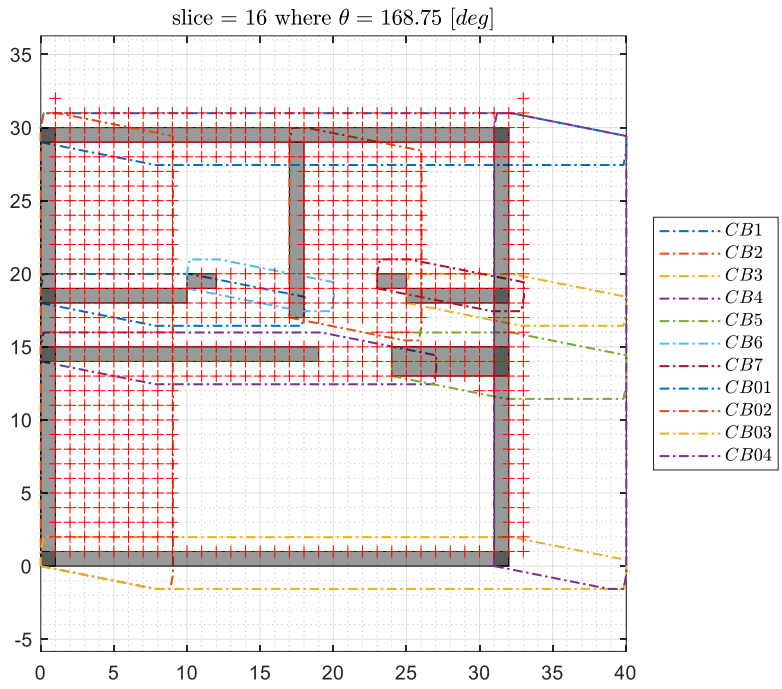
Figure 7 - Online (left) vs Offline (right) A* path finder. The online version took 897 steps, whereas the optimal offline version was 107 steps.

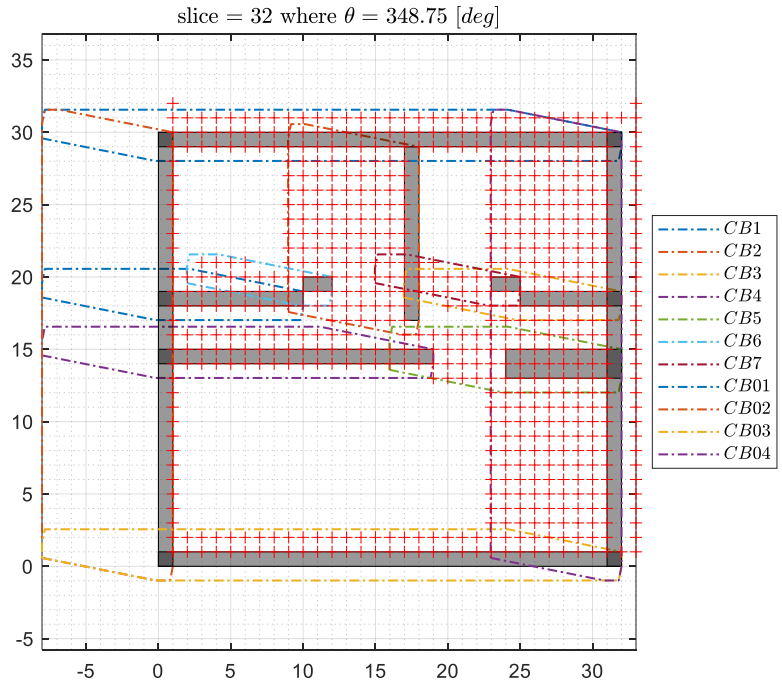
5 Appendix – Fixed grids for Part 1

The following are plots showing a fixed space grid for part 1 of the project.






slice = 16 where θ = 168.75 [deg]
CB1
CB2
CB3
CB4
CB5
CB6
CB7
CB01
CB02
CB03
CB04


slice = 32 where θ = 348.75 [deg]
CB1
CB2
CB3
CB4
CB5
CB6
CB7
CB01
CB02
CB03
CB04