MTE544 Lab 4

Group 29

Friday 3:00 PM

Kevin Xue (20814292)

Hunter McCullagh (20899434)

Justin Zhu (20879375)

Station Number: 162

Robot Number: 12

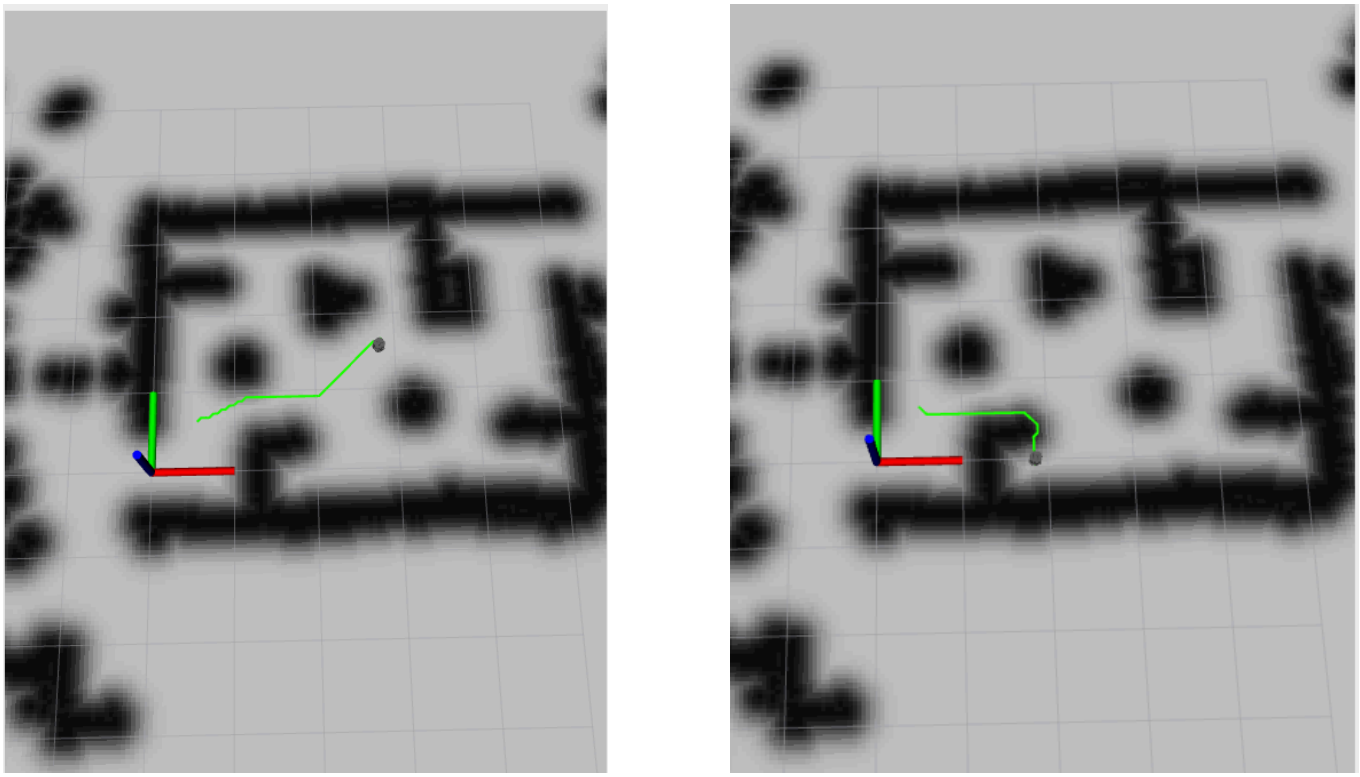
Path Planning Implementation

The A* path planning algorithm is implemented to calculate the shortest path between the starting and end nodes. To obtain the optimal pathing, nodes are progressively assessed through a cost map, based on the actual cost (g) from the starting node to the assessed node and a heuristic cost (h) from the assessed node to the end node. At each step, the algorithm evaluates nodes by minimizing the combined actual and heuristic cost (f), determining which node to evaluate next and making the search more efficient.

In the code, two dictionaries are initialized to track the explored and unexplored nodes. In the main loop, the node with the smallest f-cost is considered the parent node and evaluated to generate children for all adjacent squares in the cost map (max 8 possible moves), and then transferred to the explored dictionary. Children that are not within bounds or are an obstacle are filtered out, leaving the valid children converted into nodes and added to the unexplored dictionary. However, one exception that needs to be checked is if the children node happens to be on a previously explored node, in which case the g, h, and f costs need to be updated to reflect the optimal cost for that specific node. Once the goal is reached (checked in the loop), a path is reconstructed by tracing back through each node's parent attribute, resulting in a list of nodes for the robot to interpret as poses to move towards.

This algorithm was put in practice during the lab, which based on the mapping obtained in Figure 1 with a laser_sig value of 0.01, allows the user to identify any goal pose on the map as the targetted end node. Figure 2 demonstrates the A* path planner in obtaining an optimal trajectory to the specified goal pose. Two cases were ran to compare the effects of utilizing the Euclidean vs Manhattan heuristics. The laser_sig value was also adjusted to 0.2 in the actual test, causing the obstacles to be more "blurred" so that the algorithm would be more conservative in finding a path that was less likely to come in contact with the physical obstacles.

Figure 1: LiDAR map (laser_sig = 0.01)



(a) (b)

Figure 2: Path trajectory with endpoint denoted by black dot utilizing different heuretics (a) Euclidean and (b) Manhattan (laser_sig = 0.2)

The difference between Euclidean and Manhattan is that Euclidean distance calculates the direct distance between two points (hypotenuse), whereas Manhattan calculates the sum of the x and y distance respectively as formulated below:

$$Euclidean\text{: } h = \sqrt{(x_1 - x_2)^2 - (y_1 - y_2)^2}$$
$$Manhattan\text{: } h = |x_1 - x_2| + |y_1 - y_2|$$

The advantage of utilizing the Euclidean over the Manhattan is it allows the robot to move in a straight line diagonally, at the cost of more computations. It is still possible to achieve diagonal pathing with the Manhattan, however due to its limitations in just horizontal and vertical movements, the robot would have to make a horizontal and a vertical movement. For example, a robot with a starting and end node at (0,0) and (1,1) respectively would have an $h_{Euclidean}$ value of 1.41 ($\sqrt{2}$) vs an $h_{Manhattan}$ value of 2.

As can be observed in Figure 2 above, the Euclidean pathing has minimal turns and straight diagonal stretches, while most of the Manhattan pathing is inline with the gridlines. There is a brief diagonal path with the Manhattan, indicating that the robot here is making many 90 degree turns in series to replicate a straight diagonal line. Consequently, utilizing the Manhattan is likely to result in the robot making more PID control adjustments to compensate. Overall, Euclidean is generally better as it produces a more efficient path, particularly in situations where the robot has the capability to move diagonally (8-point connectivity), such as the Turtlebots used in the lab. In cases where the system only moves horizontally and vertically (4-point connectivity), Manhattan may be more suitable with its faster and simpler computation.