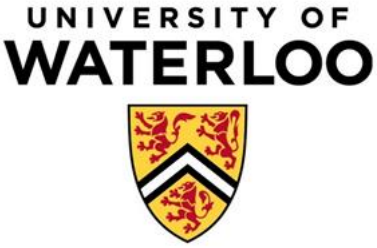

MTE 544 Assignment 2

Prof. M Biglarbegian

MTE 544 – Autonomous Mobile Robots

Due: April 1st, 2020

By:

Reza Rajan – 20599340

Preamble

This assignment focuses on performing path planning using Potential Fields (PF), Probabilistic Roadmaps (PRM) and Rapidly Exploring Random Trees (RRT). A fixed map is used to simulate the path planning techniques on a two-wheel differential-drive robot (Figure 1).

Environment

The environment provided for simulation is shown below:

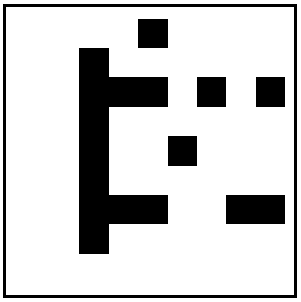

Figure 1 - Environment Map (Blockville)

Potential Fields

Potential Fields (PF) path planning works on the premise of obstacles emitting repulsive forces, while the goal creates an attractive force. These forces are virtual, i.e. only analogous to physical forces, and as such are used to generate a path to the goal as if the forces moved the robot in that direction.

Precautions

i. The radius of influence of an obstacle must be set to a sufficiently high value to generate effective repulsive fields which overcome attractive forces, or else the robot may collide with obstacles;
ii. The timestep must be small enough to effectively simulate every instance of repulsive forces acting on the robot;
iii. Take care to avoid local minima which will cause the robot to collide or move in a loop.

Implementation:

The goal is to move the robot from the top left of the map (5,5) to the bottom right (95,95). *Note that the coordinate convention used here varies from that shown in the results, but corresponds to the same locations.*

The map is also modified for performance reasons – only obstacle edges are considered rather than performing potential field calculations for each pixel corresponding to an obstacle. Since this is done, the scaling factor for the repulsive forces must be increased to compensate for all the pixels which are not considered, i.e. to generate an effective repulsive force for obstacle edges only. A goal tolerance is also set to allow faster convergence to a goal.

Parameters

The following parameters are used:

i. $Obstacle\ Influence\ Region, \rho_o = 40\ (pixels)$
ii. $Attractive\ Constant, K_{att} = 0.1$
iii. $Repulsive\ Constant, K_{rep} = \frac{dist_{infl}}{K_{att}}$
iv. $Grid\ Scaling\ Factor, grd_{scl} = 0.1\ (\frac{meter}{pixel})$
v. $Velocity\ Scaling\ Factor, vel_{scl} = 0.01$
vi. $Timestep = 0.01\ (seconds)$
vii. $Goal\ Tolerance, radius = 5\ (pixels)$,
viii. $Robot\ Radius,\ robt_{rad} = 5\ (pixels)$

Equations

The equations used to calculate these force vectors are described below:

$$F_{att}(q) = -K_{attr} \cdot (q - q_{goal})$$

$If\ \rho(q) \leq \rho_o$:

$$F_{rep}(q) = K_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_o}\right)\left(\frac{1}{\rho^3(q)}\right)(q - q_{obst})$$

$If\ \rho(q) \geq \rho_o$: $F_{rep}(q) = 0$

, where q represents a point's coordinate, and $\rho(q)$ represents its corresponding Euclidean distance.

Results

With the above, the force vectors are calculated, and the path produced is shown below:

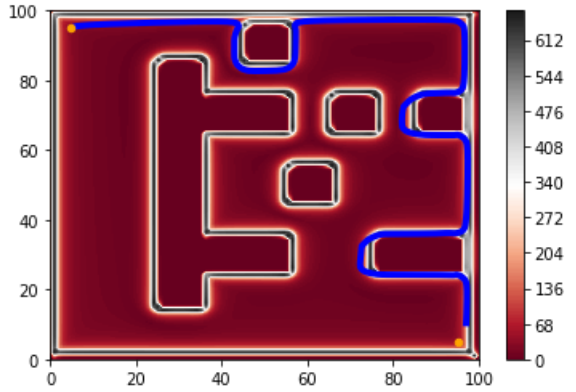

Figure 2 – Contour Map with Potential Fields Path

Figure 2 indicates that the path follows the walls as the repulsive forces from surrounding obstacles "push" it away. When nearby an obstacle, it carefully traces around the obstacle, until it reaches a corner of the map. This behavior continues until the destination is reached. At certain points it traces further away from obstacles, because in this implementation the robot's dimensions are considered – force vectors are calculated from each grid space the robot occupies.

Probabilistic Roadmaps

Probabilistic Roadmaps (PRM) is a path planning algorithm which uses random sampling to generate waypoints for the robot to traverse. These random samples are checked for obstacle collisions at its coordinate, as well as for any obstacle which exists between two connected randomly sampled points. Then, the A star algorithm, or any other similar path finding algorithm, is used to search connected points for a path from the start to a goal.

Precautions

i. Ensure that each sampled point is connected to at least one other point, as well as its parent point;
ii. Remove any connections which collide with obstacles;
iii. Manually add the start point and any other required waypoint(s) to the set of sampled points, so that they may be connected to other nodes.

Implementation:

The goal is to move the robot through the following waypoints:

Table 1 - Waypoints for PRM

Waypoint	X	Y
Start	5	5
1	70	15
2	90	50
3	30	95
4	5	50

Note that the coordinate convention used here varies from that shown in the results but corresponds to the same locations.

Nearest Neighbours

The Nearest Neighbours algorithm is used to find the N nearest neighbours to each sampled point. There are two ways to use this algorithm:

i. Find the N nearest neighbours to each node, regardless of their distances. In this case, noting Precautions – i, either a sufficiently high number of neighbours, N, must be specified to ensure that the parent node is re-connected, or the parent node must be manually added to the list of connected nodes. This is required for the A star algorithm to work;
ii. Find all the neighbouring nodes within a specified radius, which guarantees

that the parent node is re-connected. However, this may have an impact on performance if there are a large number of sampled points.

A Star

Once a list of connected nodes is generated, the A Star algorithm is used to find the most optimal path through these nodes, from specified start and end points. This algorithm works by traversing a node, then checking the Euclidean distances between the start point and current node, and the current node and the end point. It searches for the path which minimizes the sum of these distances.

Results

With the above, points are sampled on the map, checked for collisions, and the A Star algorithm is used to find the most optimal path from one waypoint to another:

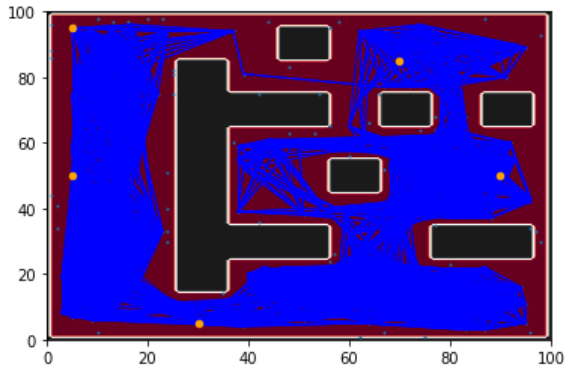

Figure 3 - PRM All Connected Nodes

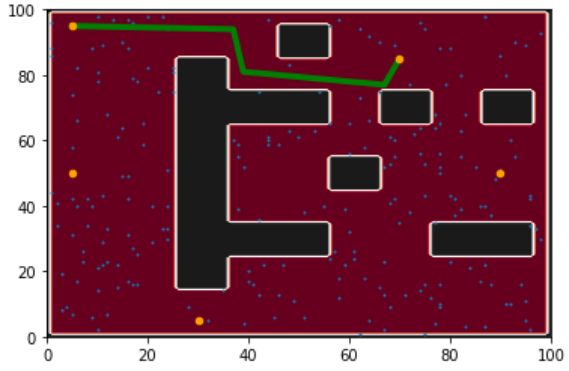

Figure 4 - PRM Route from Start to Waypoint 1

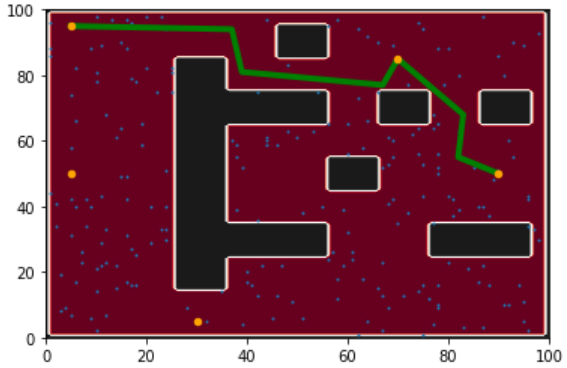

Figure 5 - PRM Route from Start to Waypoint 2

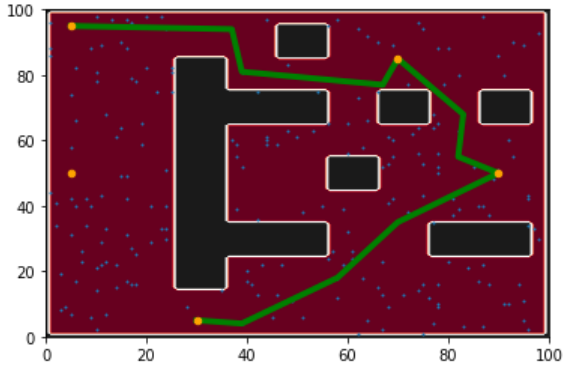

Figure 6 - PRM Route from Start to Waypoint 3

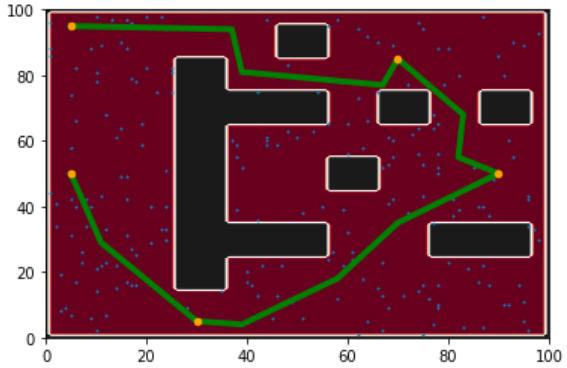

Figure 7 - PRM Route from Start to Waypoint 4

Figure 3 shows that even without considering the entire map, the paths between nodes are sufficient to represent most traversable paths. Note that the number of connected nodes is high, since the method of finding connected nodes follows that outlined in Nearest Neighbours - i. The A Star algorithm is used to generate paths between Figure 4 to Figure 7, successfully connecting all waypoints. This method is generally fast, compared to a scenario which considers each free point on

the graph. Note that the path avoids obstacles within the specified dimensions of the robot (0.45m, plus a 0.05m overall padding).

Rapidly Exploring Random Trees

Rapidly Exploring Random Trees (RRT) is a path planning algorithm which uses random sampling to generate and connect a "tree" of nodes. Once a random sample is generated, the algorithm searches for its closest node on the tree and tries to connect to it by:

i. Checking whether it is within a specified distance to the node and if not, finding the closest point along its connection path;
ii. Using the generated point, checking whether there are obstacles along its connection path to the main tree, considering the robot's dimensions;

Due to the random sampling, this produces a "tree-like" structure. Once the specified end point is found along the path-checking step described in ii, without collision, the path generation is stopped and the A Star algorithm is used to find the most optimal path along the generated tree.

Precautions

i. Set the maximum length of each tree "branch" to a value which is appropriate for the map type:
    a. Smaller if there are many obstacles;
    b. Larger otherwise – this may allow faster searches;

Implementation:

The goal is to move the robot through the following waypoints:

Table 2 - Waypoints for RRT

Waypoint	X	Y
Start	5	5
1	45	50
2	9	90
3	90	10
4	90	90

Note that the coordinate convention used here varies from that shown in the results but corresponds to the same locations.

Results

With the above, points are sampled on the map, checked for collisions, a path "tree" is generated, and the A Star algorithm is used to find the most optimal path from one waypoint to another:

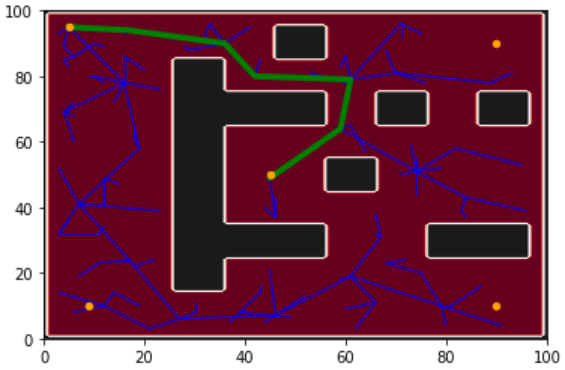

Figure 8 - RRT Route from Start to Waypoint 1

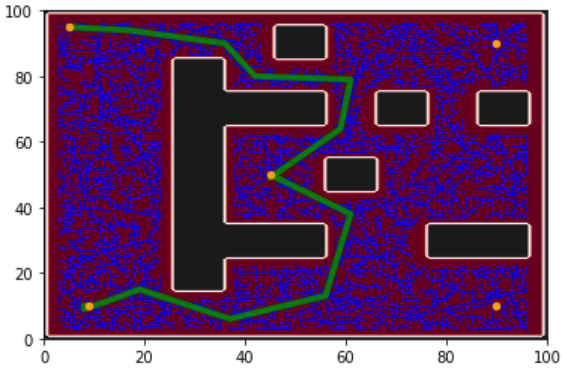

Figure 9 - RRT Route Added between Waypoint 1 and Waypoint 2

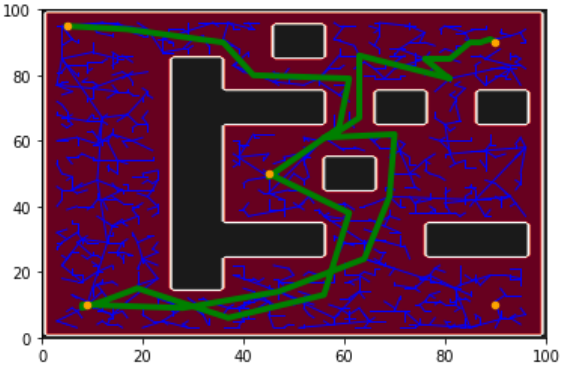

Figure 10 - RRT Route Added between Waypoint 2 and Waypoint 3

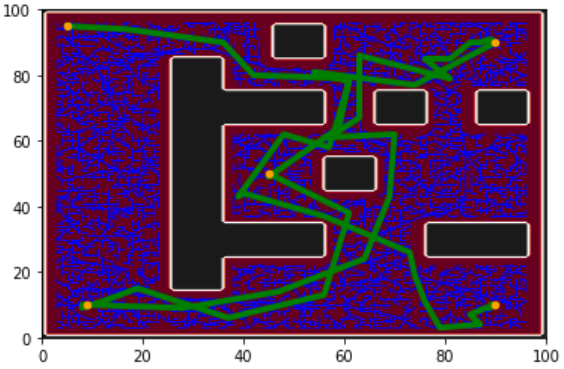

Figure 11 - RRT Route Added between Waypoint 3 and Waypoint 4

From these results it can be seen how the tree structure is generated – branches extending toward its nearest node. Furthermore, there is a distinction between the trees generated in Figure 8 and Figure 10 to those generated in Figure 9 and Figure 11 – depending on the number of samples which must be generated to find a suitable connection from start to end, the trees can become fairly large, hence the name "rapidly expanding". In the end, however, once a suitable connection is found, the A Star algorithm finds the most optimal path across the tree. Note that in Figure 9 and Figure 11 there appears to be a boundary around the obstacles – this is due to the path finding algorithm considering collisions based on the robot's dimensions (0.45m, plus a 0.05m overall padding).

Recommendations & Conclusion

The PF, PRM and RRT path planning algorithms have been successfully implemented to navigate a robot between waypoints without collision (Figure 2, Figure 7, Figure 11). The performances of each algorithm are touched upon, with precautions and implementation methods provided for further exploration. For PRM and RRT, the path finding algorithm used was A Star, since it is optimal and complete. However, for further exploration it is recommended that other path finding algorithms be tested, such as Dijkstra's algorithm. Other constraints may be incorporated in the implementation of PRM and RRT, such as constraining the path generation to an ellipse between the two waypoints to reduce the search space and computational expense. Overall, PRM is the least computationally expensive, and is the most performant of the three path planning methods, for the given environment (Figure 1).