Autonomous Navigation in Mobile Robots

Ishita Narang

School of electronics engineering
Vellore Institute of Technology

Vellore, India
ishitanarang28@gmail.com

Budhaditya Bhattacharyya

School of electronics engineering
Vellore Institute of Technology

Vellore, India
budhaditya@vit.ac.in

***Abstract*—**

Autonomous navigation[1] is a long-standing field of robotics research, which provides an essential capability for mobile robots to execute a series of tasks on the same environments performed by human every day. It is a challenge in uncontrolled environments because it requires a set of subsystems to work together. It requires building a map of the environment, localizing the robot in that map, making a motion plan according to the map, executing that plan with a controller, and other tasks; all at the same time. Main goal of autonomous robot is to reach the destination by traversing through optimized path defined according to some criteria without any collision with the obstacle which may come in its path. Hence, Path planning and obstacle avoidance is the backbone of Autonomous Navigation in Mobile Robots[3]. Several algorithms have been proposed by various researchers for path planning and obstacle avoidance [3] having several advantages and limitations. In this paper, we have discussed about the PRM[8] and RRT Algorithm for path planning[6] and its implementation and stimulation in MATLAB. We have focused upon Mobile Robot Algorithm Design[11] using the Robotics System's Toolbox. Binary Occupancy Map for mapping, mobileRobotPRM [7] and controllerPurePursuit [9] for path planning and following and lastly the Differential Drive Kinematics model[10] as a Kinematic motion model.

Keywords— Autonomous Navigation, Mobile robots, obstacle avoidance, path planning, PRM and RRT Algorithm, map, motion planning.

I. INTRODUCTION

Autonomous mobile robot navigation plays a vital role in self-driving cars, warehouse robots, personal assistant robots and smart wheelchairs, especially with a shortage of work force and an ever-increasing aging population. Significant progress has been achieved in recent decades advancing the state-of-the-art of mobile robot technologies. These robots are operating more and more in unknown and unstructured environments, which requires a high degree of flexibility, perception, motion and control. Companies such as Google and Uber are developing advanced self-driving cars and expecting to present them into the market in the next few years. Various mobile robots are roaming in factories and warehouses to automate the production lines and inventory, saving workers from walking daily marathons

A mobile robot is a vehicle which is capable of an autonomous motion. With the development of robotic technologies over the years, mobile robots have been widely employed in indoor environments such Cleaning large buildings ,Transportation in industry, Surveillance in large buildings and outdoor environments such as Agriculture ,Forest, Space, Underwater ,Military ,Firefighting Sewage tubes and Mining. The Autonomous mobile robots are being developed for numerous applications where long term capabilities would be beneficial. The main aim of mobile robotics is to create completely autonomous robots in the sense that they must be able to complete their tasks without human intervention.

An autonomous mobile system should be able to plan its route towards the destination, detect and avoid obstacle in the path to reach the destination with an acceptable accuracy. In every mobile robotic system, the skill to avoid obstacles is crucial. The robot must be trusted to complete its tasks without being a hazard to itself or to others. The requirements for better obstacle avoidance algorithm are it should be fast, robust and not dependent on prior information about the environment.

Mobile robots navigating autonomously and safely in uneven and unstructured environments still face great challenges. Fortunately, more and more environments are designed and built for wheelchairs, providing sloped areas for wheeled robots to navigate through. However, little work focuses on an integrated system of autonomous navigation in sloped and unstructured indoor areas, especially narrow sloped areas and cluttered space in many modern buildings. The robots are required to safely navigate in narrow uneven areas while avoiding static and dynamic obstacles such as people and pets.

II. EASE OF USE

A. *Principle*

Sampling based motion planning uses randomization to construct a graph or tree (roadmap) in C-space on which queries (start/goal configurations) may be solved. Arguably, the most influential sampling-based motion planning algorithms to date include Probabilistic Roadmaps (PRMs) (Kavraki et al., 1996, 1998) and Rapidly exploring Random Trees (RRTs) (Kuffner and LaValle, 2000; LaValle and Kuffner, 2001; LaValle, 2006).

B. **Processs**

A **probabilistic roadmap (PRM) is** a network graph of possible paths in a given map based on free and occupied spaces. The PRM algorithm uses the network of connected nodes to find an obstacle-free path from a start to an end location.

III. MOBILE ROBOT ALGORITHM DESIGN

These Robotics System Toolbox™ algorithms focus on mobile robotics or ground vehicle applications. These algorithms help you with the entire mobile robotics workflow from mapping to planning and control. You can create maps of environments using occupancy grids, develop path planning algorithms for robots in a given environment, and tune controllers to follow a set of waypoints.

Functions:

1. **Mapping**

   Binary Occupancy Map: Create occupancy grid with binary values.

   Inflate: Inflate each occupied grid location.

2. **Path Planning and Following**

   Mobile Robot PRM: Create probabilistic roadmap path planner

   findpath: Find path between start and goal points on roadmap

   controllerPurePursuit: Create controller to follow set of waypoints

3. **Kinematic Motion Models**

   Differential Drive Kinematic Model: Compute vehicle motion using differential drive kinematic model.

A. *Differential Drive Vehicle Model*

DifferentialDriveKinematics creates a differential-drive vehicle model [figure 3.1] to simulate simplified vehicle dynamics. This model approximates a vehicle with a single fixed axle and wheels separated by a specified track width. The wheels can be driven independently. Vehicle speed and heading is defined from the axle center. The state of the vehicle is defined as a three-element vector, [x y theta], with a global xy-position, specified in meters, and a vehicle heading, theta, specified in radians. To compute the time derivative states for the model, use the derivative function with input commands and the current robot state.

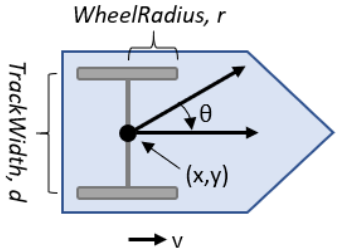

Figure 3.1: Differential Drive Vehicle Model

B. *Pure Pursuit Controller*

- Pure pursuit is a path tracking algorithm. It computes the angular velocity command that moves the robot from its current position to reach some look-ahead point in front of the robot. The linear velocity is assumed constant, hence you can change the linear velocity of the robot at any point. The algorithm then moves the look-ahead point on the path based on the current position of the robot until the last point of the path.
- Reference coordinate frame[figure 3.2] used by the pure pursuit algorithm for its inputs and outputs.
- The robot's pose is input as a pose and orientation (theta) list of points as [x y theta].

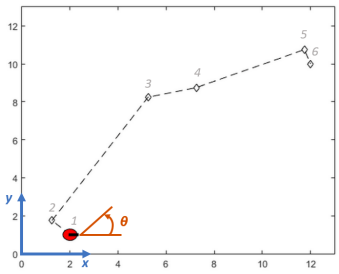

Figure 3.2: Orientation of the mobile robot

➢ Look Ahead Distance

The LookAheadDistance [figure 3.3] property is the main tuning property for the controller. The look ahead distance is how far along the path the robot should look from the current location to compute the angular velocity commands.

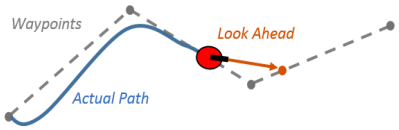

Figure 3.3: Look Ahead Distance

IV. PATH PLANNING ALGORITHMS

Path planning is a computational problem that lets an autonomous mobile robot find the optimal path between two points. In other words, this problem deals with finding the shortest path between point A and point B such that there is no intersection with any configuration space obstacle.

Configuration Space

The set of all configurations or positions that a robot can attain is known as its configuration space. We also refer to it as c-space. We use cartesian coordinates to define a robot's configuration.

Configuration Space Obstacles

It is the region in which either the robot collides with the physical obstacles or some specified links of the robot to collide with each other.

A. *Probabilistic Road Map (PRM)*

PRM is a simplified sample-based algorithm which works by constructing probabilistic road map comprising networks of connected nodes in a given map based on free and occupied spaces to find an obstacle free path from start to end point. Once the roadmap is constructed, a path connecting source to destination is extracted using graph searching techniques. The Algorithm begins road map with a

node at the initial configuration which is then expanded randomly by adding edges and nodes further.

- A random node is generated in the configuration space.
- The system checks whether this node lies in free space or not i.e, whether this configuration intersects with an obstacle or not.
- If the node is in free space, it is added to the graph.
- This newly generated node is then connected to the closest nodes through a straight line.
- The system then checks if the connection between two nodes lies in free space or not.
- If it lies in free space, the connection is added to the graph. Let's refer to theses connection as edges now.

B. *Rapidly Exploring Random Trees (RRT)*

A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree.

An RRT grows a tree rooted at the starting configuration by using random samples from the search space. As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the connection is feasible (passes entirely through free space and obeys any constraints), this results in the addition of the new state to the tree.

RRT Algorithm (x_{start}, x_{goal}, step, n)

```
1   G.initialize(x_start)
2   for i = 1 to n do
3       x_rand = Sample()
4       x_near = near(x_rand, G)
5       x_new = steer(x_rand, x_near, step_size)
6       G.add_node(x_new)
7       G.add_edge(x_new, x_near)
8       if x_new = x_goal
9           success()
```

Figure 4.1: RRT Algorithm

V. METHODOLOGY AND CODE

The Robotics System's Toolbox of MATLAB is utilized for mapping, path following, designing the controller, and stimulating the differential drive model. This toolbox provides utilities for robot simulation and algorithm development.

A. *Path Planning in Environments of Different Complexity*

1) *Import Example Maps for Planning a Path*

Figure 5.1 shows how to import a binary occupancy map, here we have utilized a simple map [Figure 5.2].

```
load exampleMaps
map = binaryOccupancyMap(simpleMap);
figure
show(map)
```

Figure 5.1: Loading a Simple Map

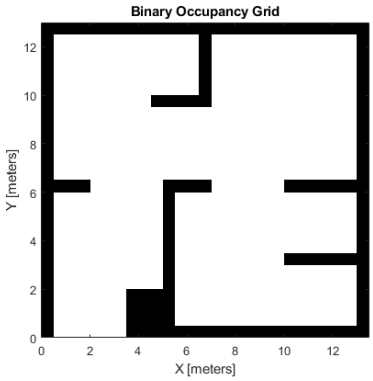

Figure 5.2: Simple Map

2) *Inflate the Map Based on Robot Dimension*

For path planning, first we need to load the Map [Figure 5.2]and inflate it by a safety distance. Inflation increases the size of the occupied locations in the map [Figure 5.3] by the radius specified in meters.

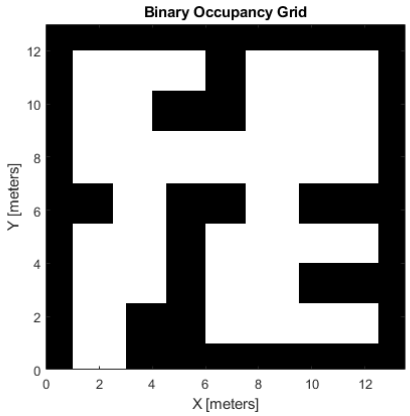

Figure 5.3: Inflated Map

3) Construct PRM and Set Parameters: Next we create a mobileRobotPRM object and define the associated attributes. Then define the number of **nodes and connection distance**[Figure 5.4].

```
prm = mobileRobotPRM;
prm.Map = mapInflated;
prm.NumNodes = 50;
prm.ConnectionDistance = 5;
```

Figure 5.4: Setting PRM parameters

4) Find a Feasible Path on the Constructed PRM: Search for a path between start and end locations using the findpath function[Figure 5.5]. The solution is a set of waypoints from start location to the end location.

```
startLocation = [4.0 2.0];
endLocation = [20.0 3.0];
path = findpath(planner, startLocation, endLocation)
```

Figure 5.5: Finding a feasible path

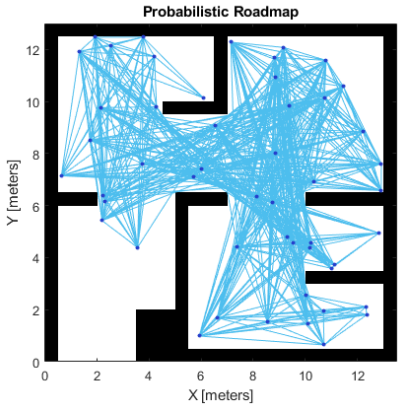

Figure 5.6: Probabilistic RoadMap

B. *Path Following for a Differential Drive Robot*

1) Define Waypoints

Now we need to find a path from the start point to a specified goal point called waypoints.
Define the current pose for the robot [x y theta]

2) Create a Kinematic Robot Model

Initialize the robot model and assign an initial pose. The simulated robot has kinematic equations for the motion of a two-wheeled differential drive robot. The inputs to this simulated robot are linear and angular velocities.

```
robot = differentialDriveKinematics("TrackWidth", 1, ...
    "VehicleInputs", "VehicleSpeedHeadingRate");
```

Figure 5.7: Initializing a differential drive model

```
figure
plot(path(:,1), path(:,2),'k--d')
xlim([0 13])
ylim([0 13])
```

Figure 5.8: Visualize the desired path

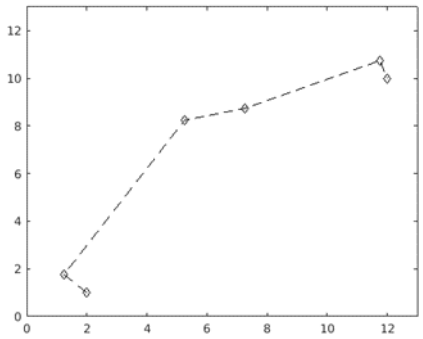

Figure 5.9: Waypoints and the desired path

3) Define the Path Following Controller

Define the algorithm for pure pursuit controller [Figure 5.10], a path tracking algorithm which computes the angular velocity command that moves the robot from its current position to reach some look-ahead point in front of the robot.

```
controller = controllerPurePursuit;
controller.Waypoints = path;
controller.DesiredLinearVelocity = 1;
controller.MaxAngularVelocity = 2;
controller.LookaheadDistance = 0.3;
```

Figure 5.10: Assigning controller parameters

4) Using the Path Following Controller, Drive the Robot over the Desired Waypoints

Next part is the simulation loop. Here we will run the pure pursuit controller and convert output to wheel speeds. The rateControl object helps us to run a loop at a fixed frequency [Figure 5.11]. It also collects statistics about the timing of the loop iterations.

Now we will compute the velocity and convert it from body (i.e., vehicle) coordinates to world coordinates.

```
sampleTime = 0.1;
vizRate = rateControl(1/sampleTime);
```

Figure 5.11: Initialize the simulation loop

```
goalRadius = 0.1;
distanceToGoal = norm(robotInitialLocation - robotGoal);
```

Figure 5.12: Computing distance to goal

Further to make the robot move over the desired waypoints we define the goal radius and compute the distance between the current location and goal location of the robot [Figure 5.12].

```
while( distanceToGoal > goalRadius )

    [v, omega] = controller(robotCurrentPose);
    vel = derivative(robot, robotCurrentPose, [v omega]);
    robotCurrentPose = robotCurrentPose + vel*sampleTime;
    distanceToGoal = norm(robotCurrentPose(1:2) - robotGoal(:));
```

Figure 5.13: Computing control commands for the robot

Drive the robot using the controller output on the given map until it reaches the goal. The controller runs at 10 Hz. Commands for the same are depicted in [figure 5.13].

C. Plan Mobile Robot Paths Using RRT

1. Load Occupancy Map
   First, we load an existing occupancy map of a small office space [figure 5.14 and 5.15], then plot the start and goal poses of the vehicle on top of the map.

```
load("office_area_gridmap.mat","occGrid")
show(occGrid)

% Set start and goal poses.
start = [-1.0,0.0,-pi];
goal = [14,-2.25,0];
```

Figure 5.14: Loading an occupancy map

Figure 5.15: Office occupancy map

2. Define State Space
   Next we specify the state space of the vehicle using a stateSpaceDubins object and specifying the state bounds.A turning radius of 0.4 meters allows for tight turns in this small environment.[figure 5.16]

```
bounds = [occGrid.XWorldLimits; occGrid.YWorldLimits; [-pi pi]];

ss = stateSpaceDubins(bounds);
ss.MinTurningRadius = 0.4;
```

Figure 5.16: Defining the state space

3. Plan The Path

   The RRT algorithm samples random states within the state space and attempts to connect a path. These states and connections need to be validated or excluded based on the map constraints. The vehicle must not collide with obstacles defined in the map.

   Next we create the path planner and increase the max connection distance to connect more states and set the maximum number of iterations for sampling states.

```
stateValidator = validatorOccupancyMap(ss);
stateValidator.Map = occGrid;
stateValidator.ValidationDistance = 0.05;

planner = plannerRRT(ss,stateValidator);
planner.MaxConnectionDistance = 2.0;
planner.MaxIterations = 30000;
rng default

[pthObj, solnInfo] = plan(planner,start,goal);
```

Figure 5.17: Creating a RRT Planner

4. Plot the path
   Visualise the occupancy map and plot the search tree from the solnInfo. Interpolate and overlay the final path.

```
show(occGrid)
hold on

% Plot entire search tree.
plot(solnInfo.TreeData(:,1),solnInfo.TreeData(:,2),'.-');

% Interpolate and plot path.
interpolate(pthObj,300)
plot(pthObj.States(:,1),pthObj.States(:,2),'r-','LineWidth',2)

% Show start and goal in grid map.
plot(start(1),start(2),'ro')
plot(goal(1),goal(2),'mo')
hold off
```

Figure 5.18: Plotting the path on the map

VI. RESULT AND DISCUSSION

This output corresponds to the MATLAB code we executed.

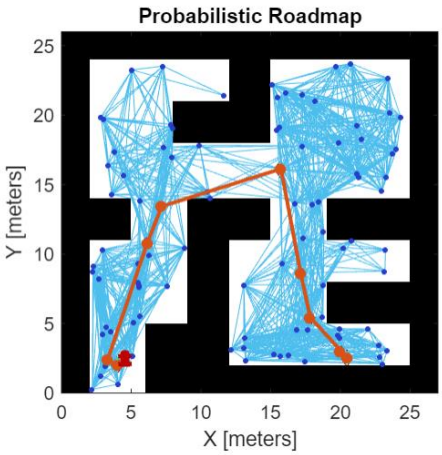

Figure 6.1: Probabilistic Roadmap

In the **Figure 6.1** we realized the possible nodes in the sample space and their interconnected network(graph). Then using PRM (Probabilistic RoadMap Planner) a path is identified avoiding all the obstacles as per the given representation by the binary occupancy map. At the end we find an optimal path given by the start and goal location.

```
path = 9×2
    4.0000    2.0000
    3.3233    2.4276
    6.1518   10.7776
    7.1868   13.4305
   15.7460   16.1375
   17.1503    8.6344
   17.8455    5.4326
   20.4687    2.5615
   20.0000    3.0000
```

Figure 6.2: Waypoints

Figure 6.2 shows the wave points identified by the robot helpful in its path planning.

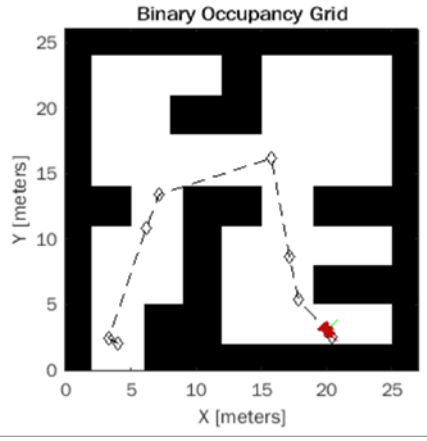

Figure 6.3: Path planning by Robot

Figure 6.3 depicts the motion planning by the robot following the pure pursuit controller algorithm. It is a depiction of the drive kinematics model in MATLAB. The mobile robot traverses along the waypoints towards the goal position.
It follows the path and avoids obstacle defined by the PRM.

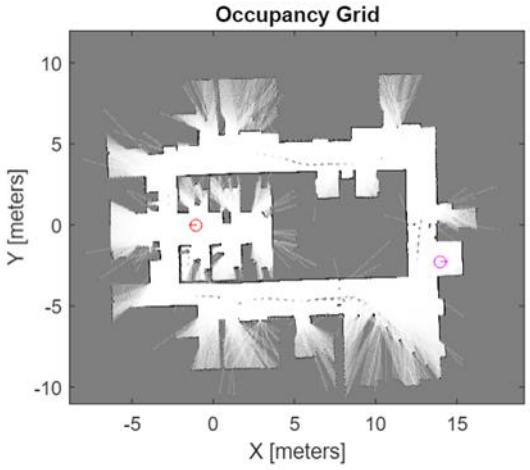

Figure 6.4: Occupancy Grid

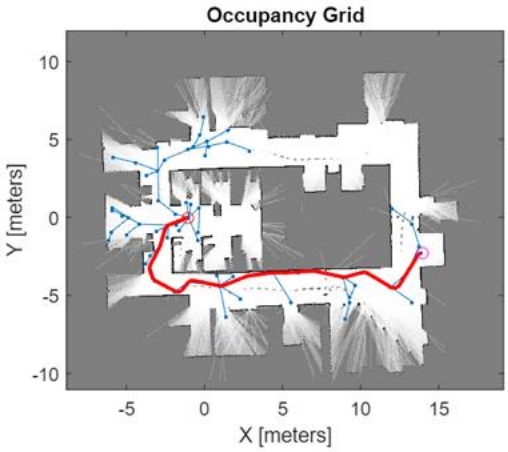

Figure 6.5: Path Planning using RRT

We have successfully planned a path for the vehicle using the RRT Algorithm as we can infer from the **Figure 6.5.**
It is the shortest and an obstacle free path between the start and goal position.

Figure 6.4 is the representation of an office map. It is shown by an occupancy grid. The start and goal position is specified on the map.

VII. CONCLUSION

An overview of 2 different path planning algorithms PRM and RRT and their implementation in MATLAB is discussed in this paper. This review paper discusses different the robot path planning algorithms and their simulation results are also shown in this paper giving an insight into the positive and negative points of every algorithm. From study of different algorithms, it can be concluded that the algorithm should be generic in respect to different maps and it should be capable of resulting a collision free path in less computation time and less path length thereby saving cost and energy.
The mobile robot path planning techniques based on predefined mapping becomes more challenging due to unpredictable. The proposed methodology is successfully implemented, and simulation is done. There are certain limitations such as the controller cannot exactly follow direct paths between waypoints and pure pursuit algorithm does not stabilize the robot at a point. The future work can be such that a program can use onboard sensors such as LIDAR and use SLAM algorithm to locally generate and update the road map to resolve new solutions on the fly.

VIII. ACKNOWLEDGMENT

We would like to thank everyone who has, directly or indirectly, helped us with this project.
First, we would like to thank our professor, Professor Budhaditya Bhattacharyya, for all the help he has provided us with, for this project. Next, we would like to thank all the staff and faculty of Vellore Institute of Technology, in providing us with all the necessities that we required for the project. It would not have been possible had it not been for all the help we have received. We would also like to thank our parents, friends, and seniors, for all that they have done.

IX. REFERENCES

[1].https://www.researchgate.net/publication/321811453_Autonomous_mobile_robot_navigation_in_uneven_and_unstructured_indoor_environments

[2].https://www.researchgate.net/publication/342118760_An_Overview_of_Path_Planning_and_Obstacle_Avoidance_Algorithms_in_Mobile_Robots

[3].https://www.researchgate.net/publication/221019514_Path_Planning_and_Obstacle_Avoidance_for_Autonomous_Mobile_Robots_A_Review

[4].https://medium.com/acm-juit/probabilistic-roadmap-prm-for-path-planning-in-robotics-d4f4b69475ea

[5].https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378

[6].https://in.mathworks.com/help/robotics/ug/path-planning-in-environments-of-difference-complexity.html

[7].https://in.mathworks.com/help/robotics/ref/mobilerobotprm.html

[8].https://in.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html

[9]. https://in.mathworks.com/help/robotics/ug/pure-pursuit-controller.html

[10].https://in.mathworks.com/help/robotics/ref/differentialdrivekinematics.html

[11].https://in.mathworks.com/help/robotics/ground-vehicle-algorithms.html?s_tid=CRUX_lftnav