CSci 5551
Project - Escape Room
Final report

Yashasvi Sriram Patkuri - patku001@umn.edu
Levi Mathwig - mathw011@umn.edu
Yaseen Khan - allar036@umn.edu

Robot: An intelligent machine that can sense and act on the world.

1 Abstract

In this project we propose an algorithm for global planning and navigating in known environments with dynamic obstacles without collision. We use probabilistic road map as the global planning method and use local repulsive forces for local dynamic collision avoidance. We optimize the path using a simple strategy of furthest milestone lookup. We also propose an algorithm for planning and navigating in unknown environment with only static obstacles without collision. We update an environment map as the robot navigates through it and replan only when necessary. Using simple techniques we achieve promising results for both scenarios.

2 Introduction

Goals Given a robot placed in a known environment with at least one exit, the goal is to

1. Plan a path and exit the environment.
2. Avoid colliding with static and dynamic obstacles.

Given a robot with a laser scanner placed in a static unknown environment with at least one exit, the goal is to

1. Sense and make a map of walls.
2. Plan and exit the environment avoiding collision with obstacles.

Purpose This project was designed to earn knowledge and experience in basic sensing and planning techniques, so that they can be composed together later to build more sophisticated and robust pipelines. This type of robot can be used as a scout where it can map out the layout of the environment and find an exit point with no prior knowledge.

Assumptions The simulation is done in ROS/Gazebo. We use the turtlebot3 model for simulation with a laser sensor. For simplicity, we assume no uncertainty in the sensors or control. We assume that we know an upper bound of the room size i.e. a bounding square that encompasses the whole room. We assumed that a new control is applied instantaneously to the robot. The robot motion is all in 2D.

Related work There is a lot of interesting work in the problem of path planning in known and unknown environments, with static and dynamic obstacles. [4] introduces a simple paradigm changing method called probabilistic road map for global path planning that avoids local mininma by discretizing continuous space as a graph. This combined with A* graph search makes a simple path planning algorithm. [5] introduces a simple algorithm called D*Lite for re-planning in dynamic environments which can be used in case of unknown environments too. [3] introduces a simple repulsive force based method for animating bird like agents in a realistic fashion without collisions.

3 Approach

We incrementally present our approach in this section.

3.1 Known environment

Here, we assume we have full information about the environment i.e. positions of both static and dynamic obstacles at any instant. Static obstacles can model walls, chairs, cupboards etc... Dynamic obstacles can model humans, animals or other moving robots. As our motion is in 2D we use line segments to represent static obstacles and circles to represent dynamic obstacles. Even if an obstacle in an arbitrary shape we can always imagine a circumscribing rectangle or circle around it and use the same process for planning. This way there is no loss of generality.

Local planning vs Global planning One of the simple strategies to reach a goal is to exert force on the agent in the direction of goal position from the current position. This is a local planning method. However, this strategy fails to avoid collision when an obstacle comes between the agent and the goal position as illustrated in Figure 1.

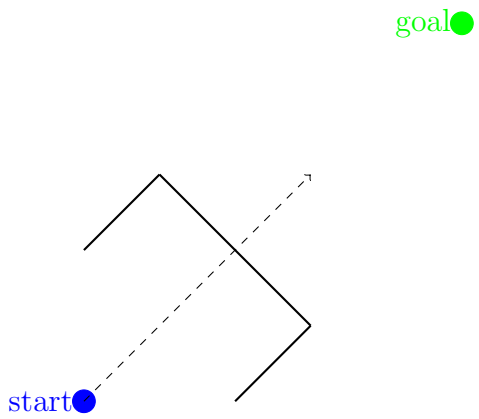

Figure 1: A failing case for local planning

To avoid such paths we need a global planning method like probabilistic road map (PRM) [4] or randomly exploring rapid trees (RRT) [6]. We choose the probabilistic road map method.

Probabilistic Road map A probabilistic road map randomly samples points in a given region of interest which are called milestones. Then each milestone is connected to nearby milestones forming a graph. Then the start and goal positions are connected to the graph using some of the nearest vertices. This way we convert a continuous space planning problem to a discretized space i.e. graph planning problem. Then a graph search like A* is conducted to find a path from start to goal. If the start and goal are not in the same connected component of the graph, we sample more milestones and connect them to the existing graph. At limit, PRM guarantees a path from start to goal. But here we don't have any particular

goal pose at the beginning. To convert this problem into a path finding problem, we take a bounding square over a room that encompasses the environment (with some padding) and set the goal position to be at one corner of the bounding square. A* plans a path from start to goal avoiding obstacles. This way if there is at least one exit, this path is guaranteed to pass through it.

Configuration space To plan in presence of obstacles using PRM, one can just remove the edges that intersect with them. A subtlety here is that we need take the extent of the agent into account. For example if we represent path from the center of the robot, the path in the first sub-figure of Figure 2 will make robot collide whereas the path in second one does not.

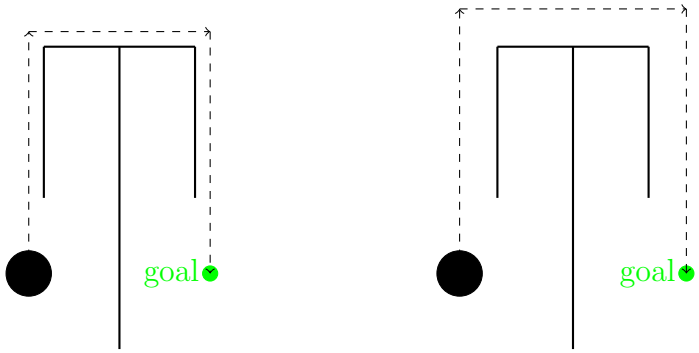

Figure 2: Two possible paths for the center of robot. Left: Collides with obstacle, Right: Does not collide with obstacle

Let the set of possible points in space where center of the robot can exist be called configuration space. Let the set of points where the center the robot cannot exist for it to be not colliding with any obstacle be called a configuration space obstacle. For a circular agent like the turtlebot3 and a line obstacle the configuration space obstacle will be intersection of two circle and a rectangle as illustrated in first sub figure of Figure 3. The exact configuration space obstacle will have rounded corners. To make intersection calculations simpler, we extend the configuration space obstacle to have a rectangular shape that circumscribes the original one. This does not matter in most cases and more importantly does not compromise on collision avoidance.

Figure 3: For a circular agent and line obstacle, Left: Exact configuration space obstacle, Right: Extended configuration space obstacle

Thus for every line segment we have a rectangle around it at a distance of agent's radius that describes the configuration space obstacle of the lines segment and agent duo. To check an intersection b/w a line segment and a rectangle we check intersection b/w line segment and four line segments of rectangle. For detecting intersection b/w line segment AB and CD

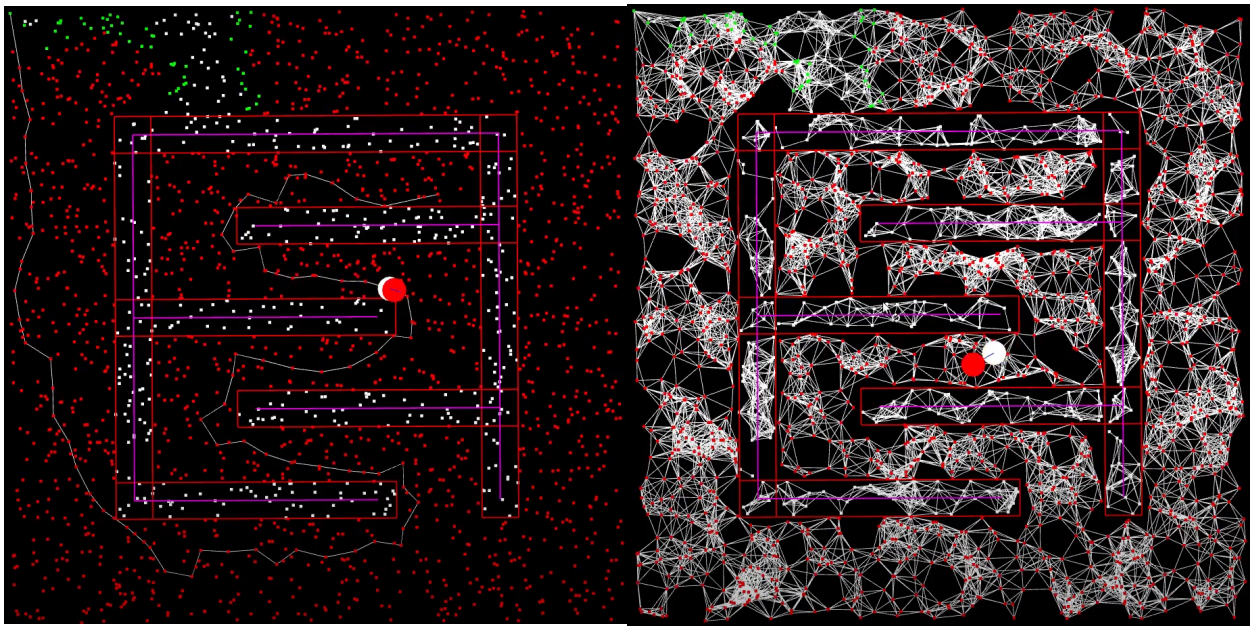

Figure 4: Left: PRM milestones and generated path, Right: PRM edges, Purple lines are obstacles, Red rectangles are configuration space obstacles, Colors of milestone represent search state.

let,

$$A \equiv \begin{bmatrix} b_y - a_y & -(b_x - a_x) \\ d_y - c_y & -(d_x - c_x) \end{bmatrix}$$

$$b \equiv \begin{bmatrix} a_x b_y - a_y b_x \\ c_x d_y - c_y d_x \end{bmatrix}$$

$$x = A^{-1} b$$

x represents the point of intersection of two lines formed by joining points A and B, and C and D respectively. To further ensure that point of intersection of these lines falls in b/w both line segments we check if

$$\|A - B\|_2 \equiv \|A - x\|_2 + \|x - B\|_2$$

$$\|C - D\|_2 \equiv \|C - x\|_2 + \|x - D\|_2$$

If lines are parallel or co-incident $det(A) == 0$ and A^{-1} is undefined. In such cases we define the line segments as not intersecting.

We then check the intersection of each edge in PRM with each rectangular configuration space obstacle and remove edges with at least one intersection. This is illustrated in Figure 4. Observe that there is no edge that intersects configuration space obstacles shown as red rectangles.

Differential drive agent A differential drive robot has two controls linear velocity and angular velocity which can be independently controlled. To follow the path generated by PRM, we follow a simple strategy of orient and translate. Invariantly, after reaching a

milestone on the path the robot orients itself in the direction of next milestone without translating and then translates without rotating.

Trajectory optimization We also perform an impromptu trajectory optimization over the path generated by PRM. At every instant instead of going towards the next milestone the robot goes to the furthest milestone in the path for which the line segment joining the current position and the milestone does not intersect with any configuration space obstacle. This is illustrated in the Figure 5 where we can see the robot going directly to the left most milestone pruning all others in between.

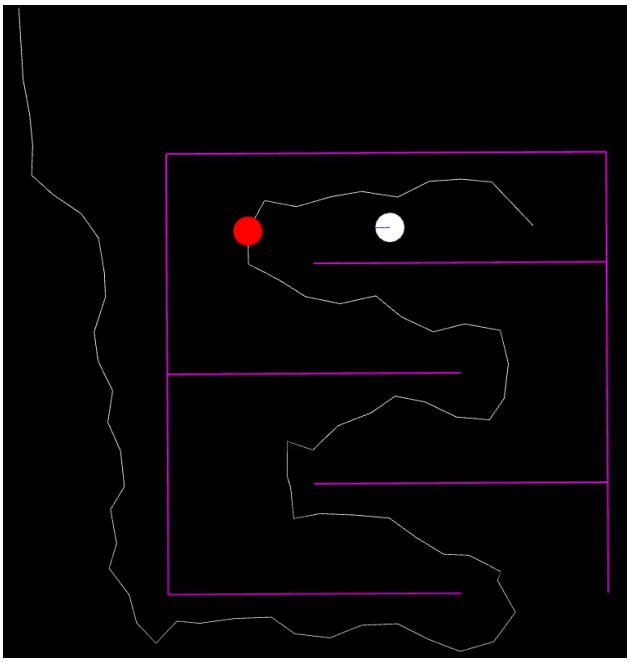

Figure 5: Trajectory optimization. Agent prunes some milestones to go directly to the furthest one in the path without colliding with obstacles.

Handling Humans While following the path generate by PRM, to avoid collision with dynamic obstacle we introduce repulsive forces from them on robot. The robot shall have three types of forces in total.

1. An attractive force towards the next milestone.
2. A repulsive force from each static obstacle.
3. A repulsive force from each dynamic obstacle.

We use forces from static obstacles too because the force along from dynamic obstacle might displace robot from its original collision free path. The repulsive forces follow an inverse power law.

$$F = \frac{-k}{\|P_{or}\|_2^n} * \hat{P_{or}}$$

where P_{or} is the displacement from obstacle to the robot and k, n are tunable parameters. In case of dynamic obstacles which are modeled as circles $P_{or} \equiv P_r - P_o$ where P_o is the center of

circle. In case of static obstacles which are modeled as line segments P_{or} is the perpendicular distance from center of robot to the nearest side of configuration space rectangle obstacle. We use inverse power law forces because they quickly go up if the robot is too close and quickly dissipate once robot is far enough. All these forces are added up and the instantaneous velocity of robot is calculated. Some moments of simulation with dynamic obstacles are shown in Figure 6.

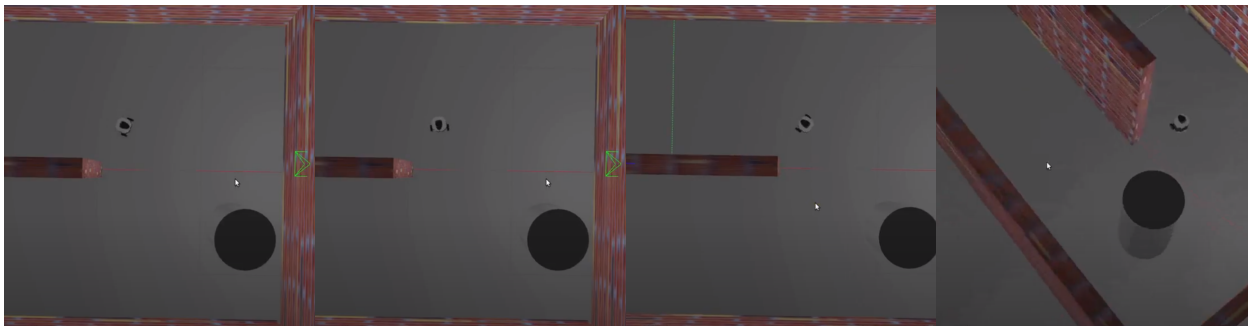

Figure 6: Robot avoiding collision with human using repulsive force method.

3.2 Unknown environment

Here, we assume we do not have any information about the environment initially. We also assume only static obstacles and no dynamic obstacles.

Laser line extraction The robot is equipped with a laser scanner. The scanner just provides distances of objects in a particular direction. Combined with current pose of the robot this implies we get a set of relative position measurements from laser scanner. From these measurements we have to extract lines that represent obstacles. This can be done by first good partitioning the measurements based on distance discontinuity and iterative end point method [7] , applying RANSAC [2] repeatedly to identify lines in individual partitions and then using least squares line fitting [1] to further improve the line extracts. But this is not the focus of our project so we use an external ros package named 'laser line extraction' to extract the lines for us.

Transforming extracted lines into global frame The laser line extractor provides us with a set of pairs of end points each representing a line segment. But these are in the frame of robot. We first convert them into a global frame using the relation

$$p_L^G \equiv p_R^G + C_R^G * p_L^R$$

where p_j^i is the position vector of j in ith frame, C_R^G is the rotation matrix of the robot, the green quantities are known and blue one is measured.

Repeated PRM modification Here we start assuming that there are no obstacles in the environment and plan a path to goal position. Hence no edges will be removed from

PRM initially. Similar to the case of known environment each time we receive a set of line segments from line extractor, we construct configuration space rectangle obstacle. For each such configuration space obstacle we mark PRM milestones that are inside it and remove PRM edges that intersect it. To determine whether a point x is inside an arbitrary rectangle with center c, perpendicular vector from center to smaller edge l, perpendicular vector from center to larger edge b we check if

$$abs(P_{cx}.\hat{l}) \leq \|l\|_2$$

and

$$abs(P_{cx}.\hat{b}) \leq \|b\|_2$$

are true. Here . represents dot product. To determine the intersection of line segment with rectangle we use the same method as earlier. A point to note is that checking if milestone is inside rectangle is much cheaper computationally than checking line segment intersection with rectangle. Therefore we do many of the first checks and do the second one only if significant mount of first checks are true as a heuristic. One problem with this approach is that initial measurements can be quite noisy due to the implementation of laser line extraction package. Therefore after reaching a milestone we take a fixed number of measurements and only cull the milestones if majority of them encompass the milestone. As this is done repeatedly we shall have a set of PRM milestones marked as inside obstacles and a bunch of PRM edges removed. This is the obstacle map that the robot generates in the process. One such map is illustrated in Figure 7.

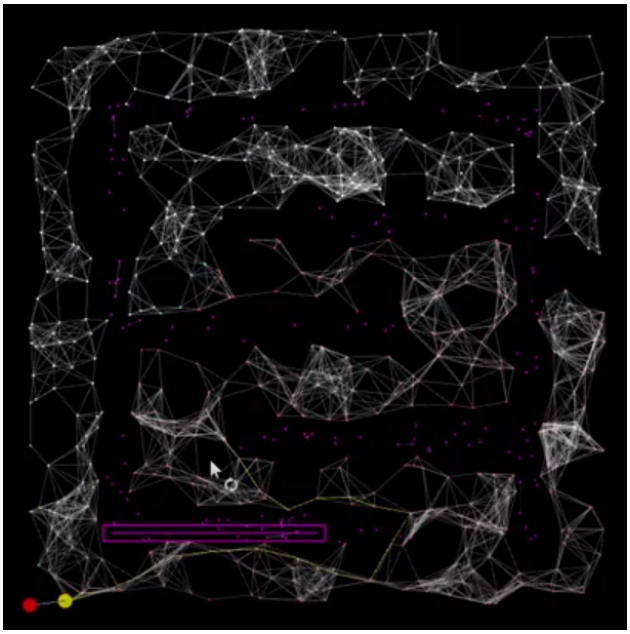

Figure 7: Obstacle map generated by the robot. Pink milestones and lack of PRM edges indicate obstacle.

Replanning As robot discovers new obstacles, some of them might be on its planned path. So after reaching each milestone it checks if its planned path intersects any observed

obstacle and if so it replans the path from the current milestone to the goal position. As the milestones inside obstacles and edges that intersect with obstacles are removed the new path will not collide with already observed obstacles. This is in spirit of D*Lite [5] replanning algorithm. The replanning is illustrated in the Figure 8.

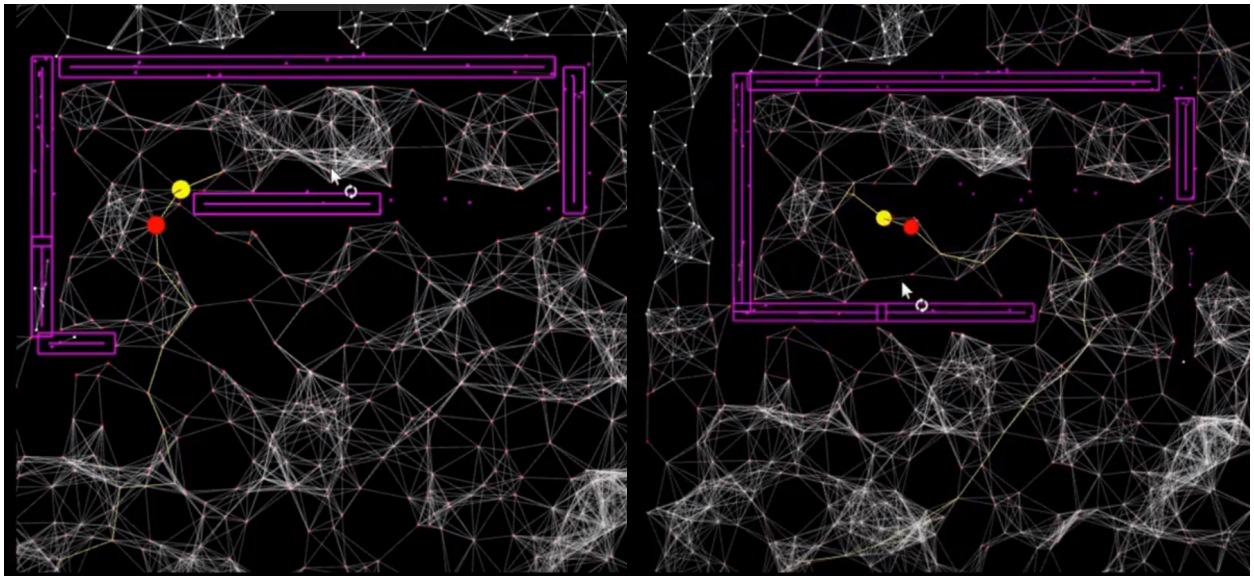

Figure 8: Robot replanning in initially unknown environment as it detects new obstacles. The yellow line is the path planned. The pink lines and rectangles are observed obstacles and configuration space obstacles from that measurement.

4 Simulations

We used ROS/Gazebo for simulation. We built a zig-zag room as our test bed for both known and unknown environment algorithms. A top view of the environement is shown in Figure 9. The robot is placed the far left corner in all cases and the environment has one door to the far right. We used parameters as described by Table 1 for the known environment and Table 2 for the unknown environment.

5 Results

Our robot was able to plan and navigate known environments with dynamic obstacles without colliding with them, even in tight spaces. Due to repulsive force method it behaviour is a bit adaptive to the position of dynamic obstacle. For example if there is significant gap b/w itself and dynamic obstacle it stays on its course whereas if there will be an eventual collision it pauses and even turns back for the obstacle to go away.

In unknown environments, our robot was able to escape the environment without colliding with any obstacle. In the process it updated its map of environment by regularly taking laser scan measurements.

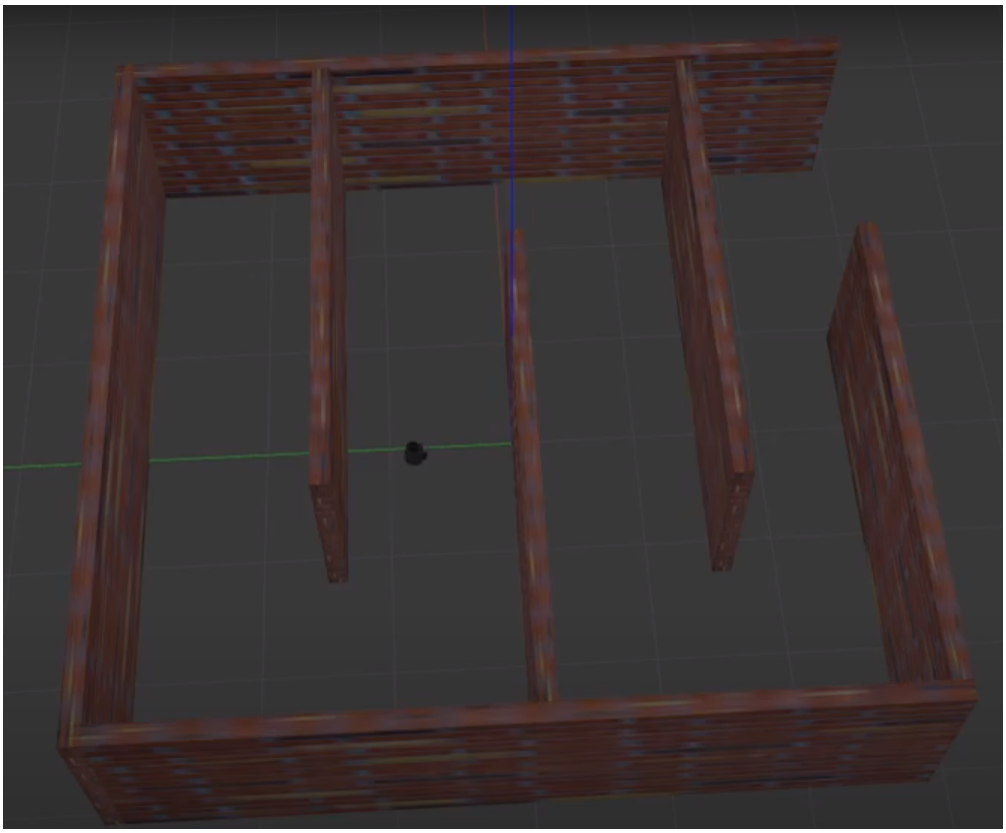

Figure 9: Top view of the environment used as a test bed for both algorithms.

Parameter	Value
Region of interest	6mx6m
Num. PRM milestones	2000
Max edge length	0.5m
Robot radius	0.2m
Dynamic obstacle radius	0.3m
k for wall repulsion	0.5
n for wall repulsion	2
k for dynamic obstacle repulsion	3
n for dynamic obstacle repulsion	2

Table 1: Parameters used for the known environment.

Parameter	Value
Region of interest	8mx8m
Num. PRM milestones	800
Max edge length	0.6m
Robot radius	0.1m

Table 2: Parameters used for the unknown environment.

6 Conclusion

We started out with a goal to plan and navigate in known environments with static and dynamic obstacles or in unknown environments with static obstacles. We achieved both of our goals with good consistency.

The main challenges we faced were

1. To figure out and implement intersection checking algorithms.
2. To maintain a global state to work with ROS callbacks.
3. To control the rate of measurements from laser scanner.
4. To come up with a good heuristic for reliable sensing in unknown environments.
5. Tuning the parameters of repulsive forces.
6. To do anything else in computer with a running Gazebo instance.

The major limitation of our algorithm for dynamic obstacle avoidance is its heavy dependence on tuning parameters. Usually tuning them manually takes a long time and is very context specific. Therefore for each new significantly different environment, parameters have to be re-tuned.

The major bottleneck in algorithm for unknown environment sensing is the intersection check b/w every PRM edge and every configuration space rectangle obstacle. This is the prime reason for not handling dynamic obstacles in unknown environment. The robot is too slow for it to react to moving obstacles promptly. This can be mitigated by using spatial data structures like K-D tree. If this step is sped up well, then dynamic obstacles can also be handled in unknown environments. This is a good direction for further refinements.

7 Launch Files

To demonstrate our findings we created launch files, one showing the agent path planning in an unknown environment, and another showing the path finding in a known environment that contains dynamic obstacles. The unknown environment can be run using *roslaunch turtlebot3_gazebo turtlebot3_zigzag.launch* The known environment containing moving cylinders can be run using *roslaunch turtlebot3_gazebo turtlebot3_zigzag_cylinder.launch* Please use the readme included in source code for full instructions.

References

[1] Å. Björck. *Numerical Methods for Least Squares Problems*. 1996.

[2] Martin A. Fischler and Robert C. Bolles. "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". In: *Commun. ACM* 24.6 (June 1981), pp. 381–395. ISSN: 0001-0782. DOI: 10.1145/358669.358692. URL: https://doi.org/10.1145/358669.358692.

[3] Christopher Hartman and Bedrich Benes. "Autonomous Boids: Research Articles". In: *Comput. Animat. Virtual Worlds* 17.3–4 (July 2006), pp. 199–206. ISSN: 1546-4261.

[4] Lydia Kavraki et al. *Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces*. Tech. rep. Stanford, CA, USA, 1994.

[5] Sven Koenig and Maxim Likhachev. "D*lite". In: *Eighteenth National Conference on Artificial Intelligence*. Edmonton, Alberta, Canada: American Association for Artificial Intelligence, 2002, pp. 476–483. ISBN: 0262511290.

[6] Steven M. Lavalle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. 1998.

[7] Shin Ting, Mercedes, and Rocío Gonzales Márquez. *A non-self-intersection Douglas-Peucker Algorithm*.