NAVIGATION OF MOBILE ROBOTS IN DYNAMIC ENVIRONMENTS

Team **NASA**

Research Question

How to efficiently and effectively plan paths for mobile robots in complex dynamic environments?

(HINT: Explore using RRTx, OP-PRM, Risk-DTRRT,)

What is the Application Impact?

Application Impact

Autonomous vehicles	Robotics	Industrial Automation
Service Robots	Drones	Logistics

Problem Description

- Offline path-planning algorithms may not be suitable for navigating tight spaces
- Search-based algorithms can be computationally and temporally expensive
- Reactive path planning is needed to navigate through narrow conditions
- The proposed solution is to use OP-PRM, Risk-DTRRT, RRTx

Goal Plan

Compare the performance of the following algorithms:

Baseline- OP-PRM

Intermediary Goal- RRT^X

Stretch/Final Goal - Risk-DTRRT

Obstacle Potential - Probabilistic RoadMap

- OP-PRM makes use of two algorithms, one of which is Probabilistic Roadmap (PRM) which uses random samples from the robot's configuration space to check if they lie in free space.
- Obstacle potential field helps in constructing a potential linked roadmap to tackle online path planning

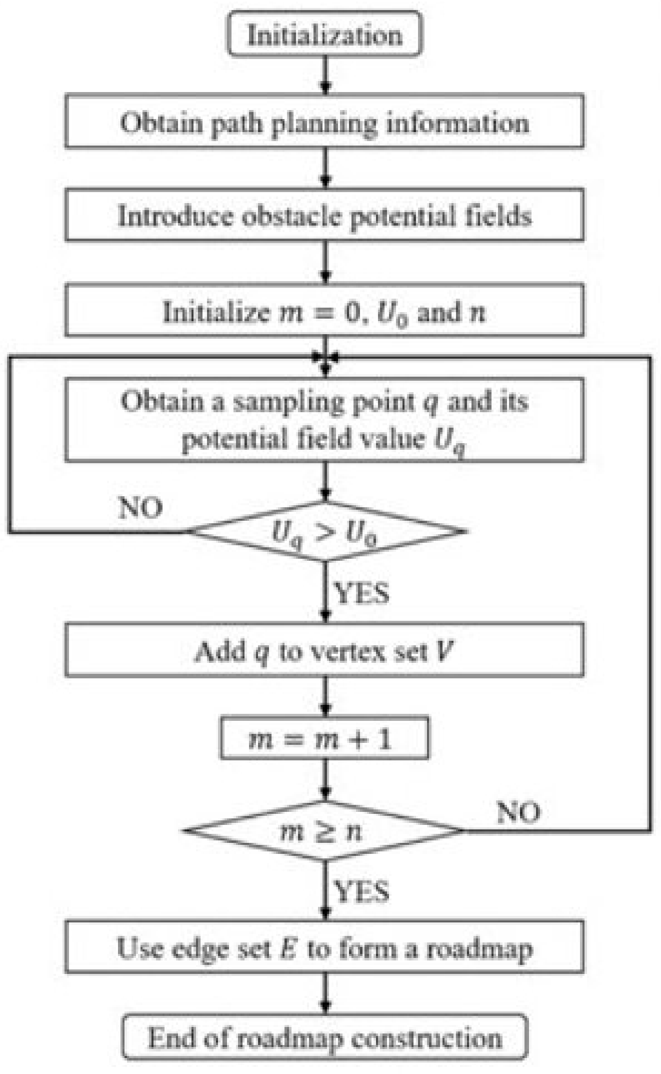


FIGURE 2 | Roadmap construction based on obstacle potential field.

https://www.frontiersin.org/articles/10.3389/fnbot.2022.910859/full

RRT^X

RRT^X (Rapidly-Exploring Random Trees^X) is a sampling-based algorithm. designed for real-time applications where pre-computation is not possible. It is optimized for use in dynamic environments where obstacles can appear, move, or disappear unpredictably, allowing for real-time kinodynamic navigation.

Pseudocode for RRT^X

Algorithm 1. $\mathrm{RRT}^X(\mathscr{X}, S)$.

```
V ← {v_goal};
v_bot ← v_start;
while v_bot ≠ v_goal do
  r ← shrinkingBallRadius(|V|);
  if obstacles have changed then
    updateObstacles();
  if robot is moving then
    v_bot ← updateRobot(v_bot);
  v ← randomNode(S);
  v_nearest ← nearest(v);
  if d(v, v_nearest) > δ then
    v ← saturate(v, v_nearest);
  if v ∉ 𝒳_obs then
    extend(v, r);
  if v ∈ V then
    rewireNeighbors(v);
    reduceInconsistency();
```

https://journals.sagepub.com/doi/full/10.1177/027836491559467
9

Risk DTRRT

The Risk-DTRRT (Risk-based Dual-Tree RRT) algorithm retains the RRT's guarantee of completeness, and the resulting trajectory is optimal within the homotopy class of the heuristic trajectory.

Pseudocode for DTRRT

```
Input: The goal and the map
Output: The trajectory
trajectory = empty
rrt_tree = empty
rewired_tree = empty
goal = read()
map = load()
t0 = clock()
while goal not reached do
    observe(x_r)
    delete unreachable nodes(rrt_tree, x_r, t0)
    observe(moving_obs)
    t0 = clock()
    predict moving_obs at time t0, ··· , t0 + N * Δt
    if environment different then
        update trajectories(rrt_tree, x_r, moving_obs)
    while clock() < t0 + Δt do
        grow the rrt_tree with node depth ≤ N
    rrt_trajectory = choose best trajectory in rrt_tree
    rewired_tree = rewire the rrt_trajectory
    trajectory = choose best trajectory in rewired_tree
    t0 = clock()
    if trajectory is empty then
        brake
    else
        move along trajectory for one step
```

https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=8554117

Approach for the project

- Implementation of OP-PRM as baseline
- RRT^X
- Risk DTRRT
- Path planning comparision Dynamic Environment.

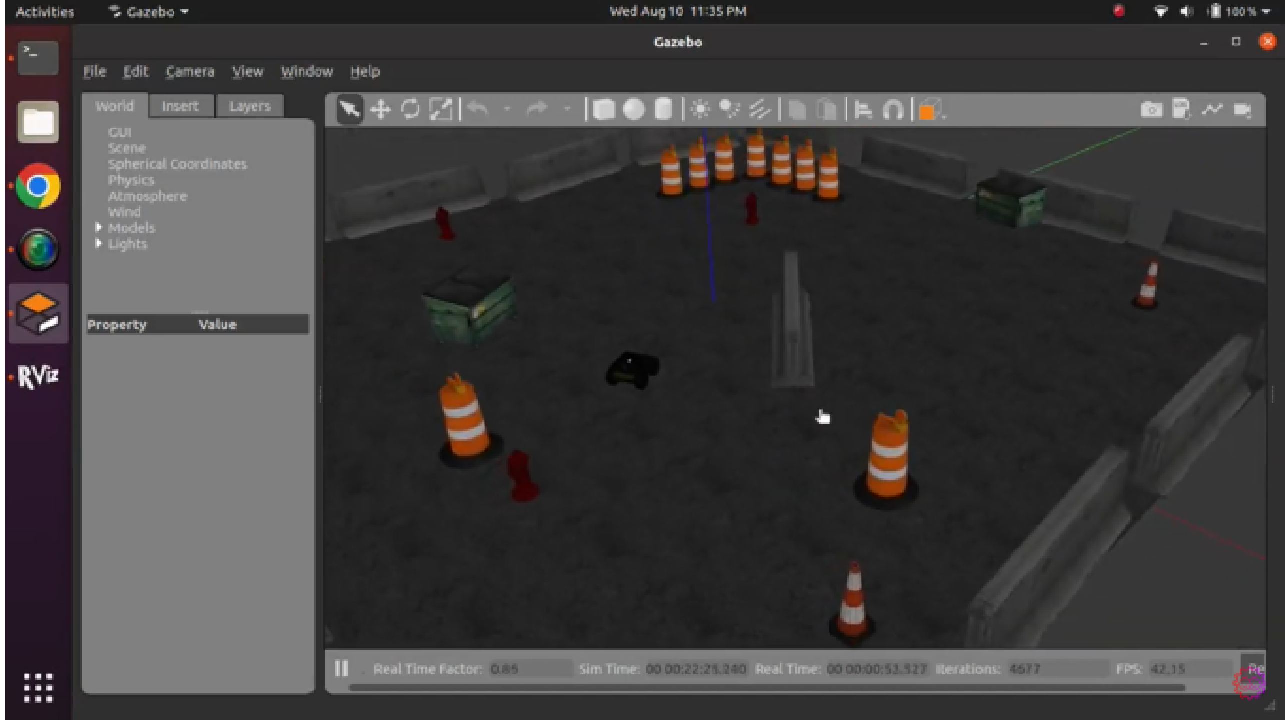
Platform Used

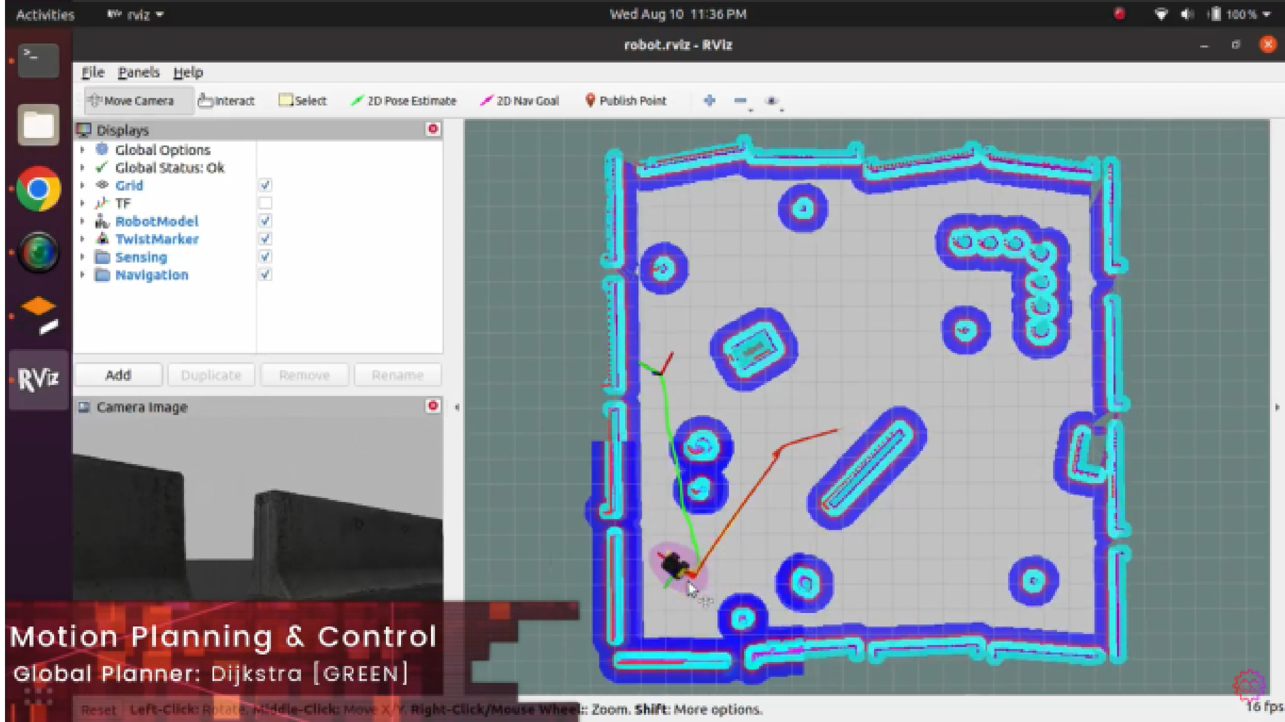
Husky is an all-terrain robotic platform developed by Clearpath Robotics.

Husky is equipped with a variety of sensors, including a LIDAR, GPS, and IMU, allowing it to perform tasks such as autonomous navigation, mapping, and object detection.

Simulation - [husky_gazebo/Tutorials/Simulating Husky - ROS Wiki]

Sample Demo/Tutorial - [(1) ROS | Husky Map-Based Navigation [Tutorial] - YouTube]





Related Work

•Ye, Lingjian, Jinbao Chen, and Yimin Zhou. "Real-Time Path Planning for Robot Using OP-PRM in Complex Dynamic Environment." *Frontiers in Neurorobotics* 16 (2022).

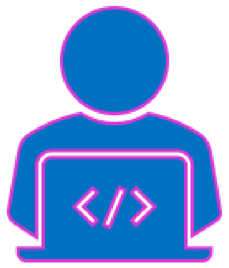
•Otte, Michael, and Emilio Frazzoli. "RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning." *The International Journal of Robotics Research* 35.7 (2016): 797-822.

•Chi, Wenzheng, et al. "Risk-DTRRT-based optimal motion planning algorithm for mobile robots." *IEEE Transactions on Automation Science and Engineering* 16.3 (2018): 1271-1288.

Task List

- Algorithm Understanding
- Code Development
- Code Optimization
- Integration
- Testing and Evaluation
- Functionality Enhancement
- Reporting

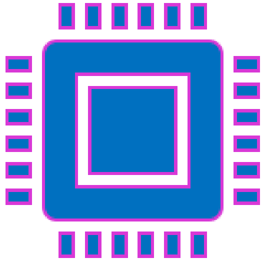
Task Division



Anuj and Nikunj :

Start with understanding the basic concepts and the algorithm in the paper and implement a simple version of the algorithm in Python.

Work on testing and evaluating the performance of the algorithm on various scenarios.



Soham and Ankush :

Work on improving the efficiency and performance of the algorithm. Optimize the code and work on the integration of the algorithm.

Focus on enhancing the functionality of the algorithm to handle more complex and dynamic environments.

Timeline

Week 1:

- Literature Review and finding similar algorithms for baseline and

Week 2 & 3:

- Setup of ROS Husky in Gazebo
- Study and implement OP-PRM algorithm.

Week 4 & 5:

- Study and implement RRT^X algorithm.

Timeline

Week 6 to 7:

- Study and implement Risk-DTRRT in ROS

Week 8 to 9:

- Further optimization of all three algorithm and improvement based on the results.

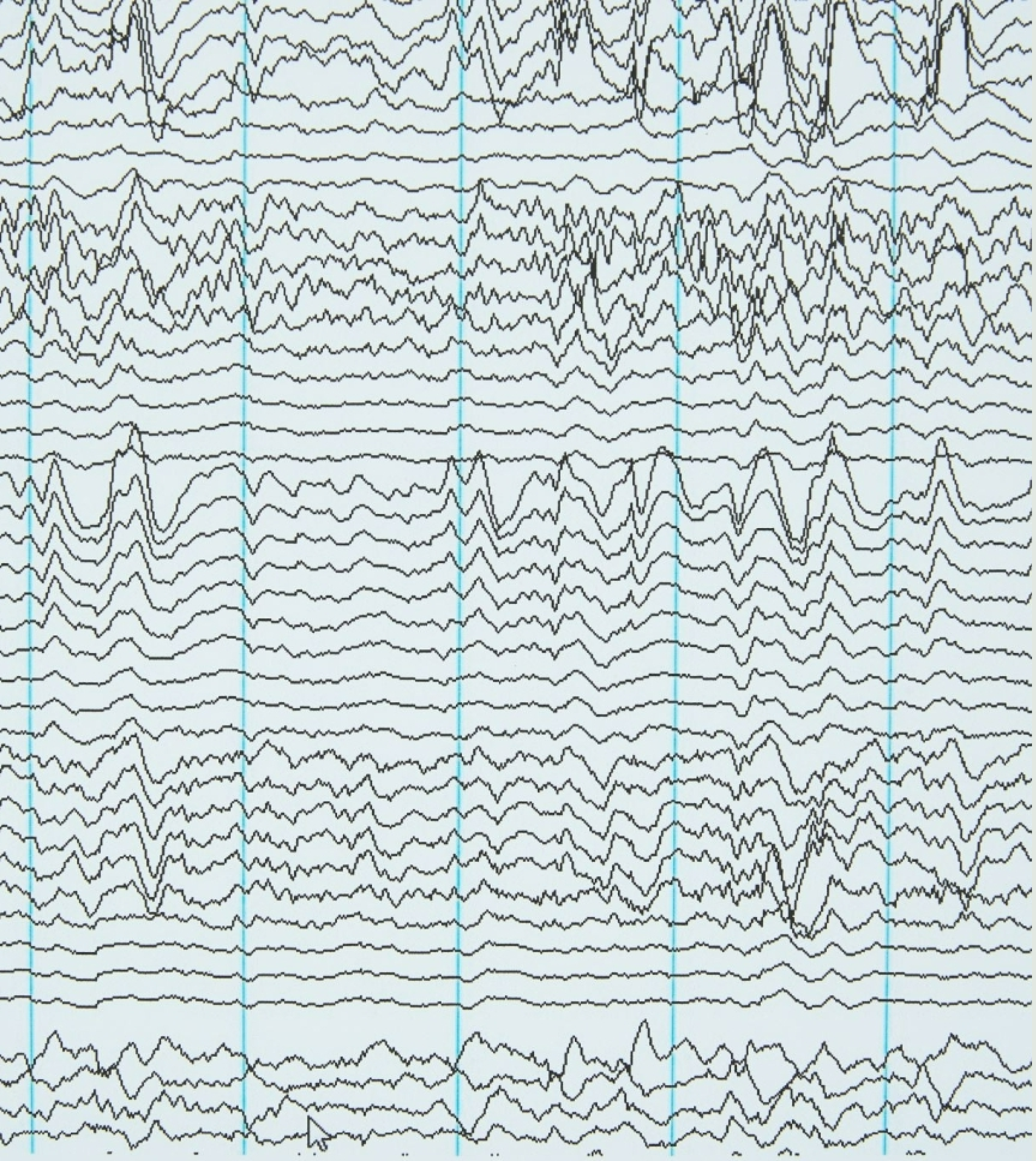
Timeline

Week 10 to 11:

- Final testing and validation of the results

Week 12:

- Preparation of the final report and presentation of the results.



Evaluation

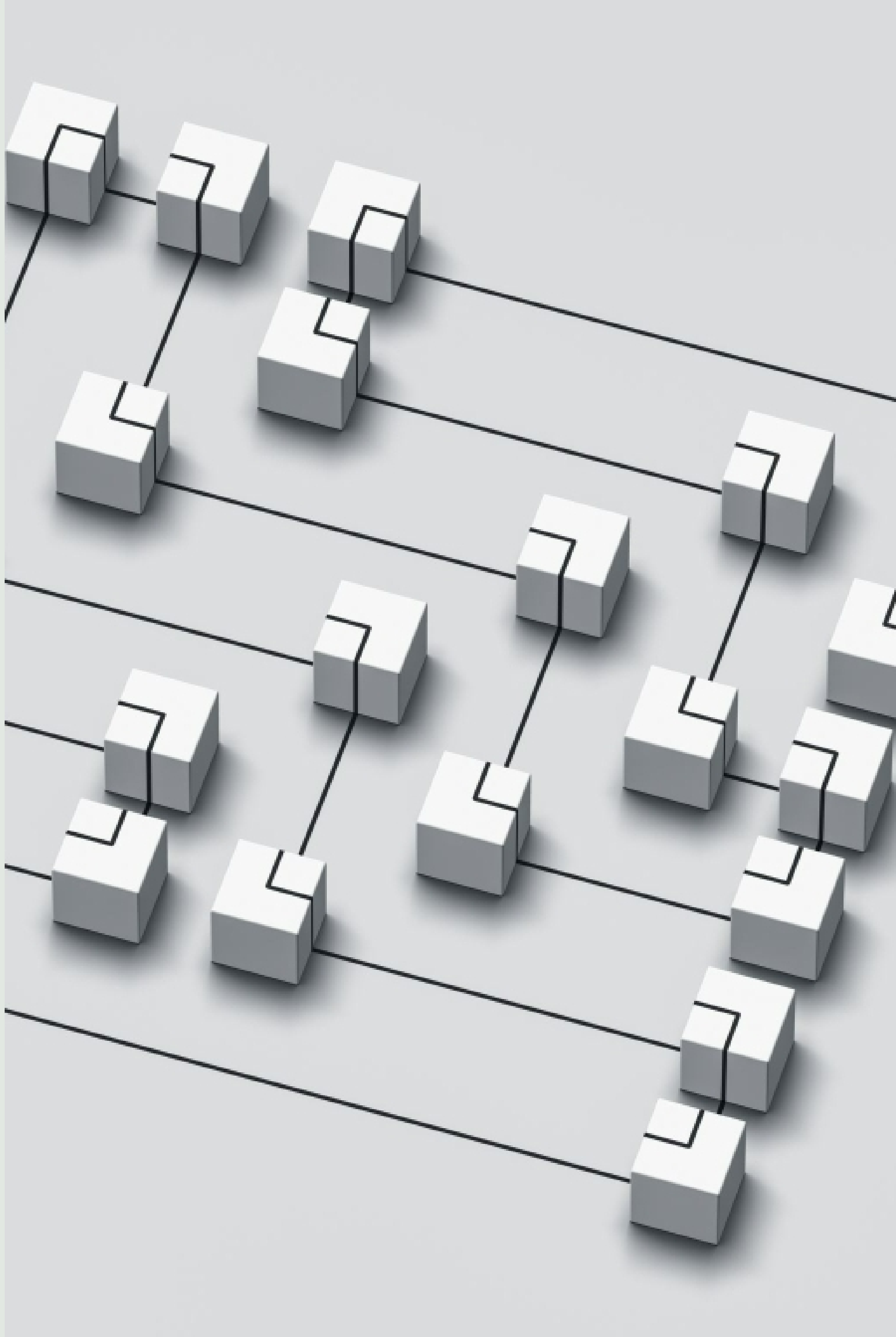
Similar experiments will be carried out for each of the algorithms, with varying number of obstacles and grid size.

Therefore, we can leverage the data as metrics to evaluate our path planning algorithms:

- **Average Path Lengths:** Compare the lengths of the path across the selected algorithms given all other variables are constant
- **Scalability:** How well does the algorithm perform with changes to the size of the grid and/or increased obstacle density?
- **Efficiency:** For individual algorithms, we find the Time taken to goal vs Size of the graph, Time taken to goal vs Number of obstacles

Deliverables

- Implementation of the OP-PRM, RRTx algorithm for path planning in complex dynamic environments
- Stretch Goal --> +Risk-DTRRT
- Evaluation of the algorithms within ROS/Gazebo environment for Husky based on metrics provided



Project Summary

Project components	**Description**
Problem description	Navigating Dynamic Environments for Mobile Robots
Application impacts	Autonomous Vehicles, Robotics , Industrial Automation, Service Robots, Drones, Logistics
Related work	OP-PRM, Risk-RRT^{x},DTRRT
Method	Implementation in Python, ROS-Gazebo
Evaluation	Average Path Lengths, Scalabality to larger workspace, Time taken to goal vs Size of the graph, Time taken to goal vs Number of obstacles
Platform	ROS Noetic-Gazebo for Husky - Robots/Husky - ROS Wiki
Team Members	**A**nkush Singh Bhardwaj, - abhardwaj@wpi.edu **S**oham Shantanu Aserkar – ssaserkar@wpi.edu **A**nuj Pai Raikar – apraikar@wpi.edu **N**ikunj Reddy Polasani – nrpolasani@wpi.edu
Team Leader	Soham Shantanu Aserkar

Team Members

Team NASA:

Nikunj

Anuj

Soham

Ankush