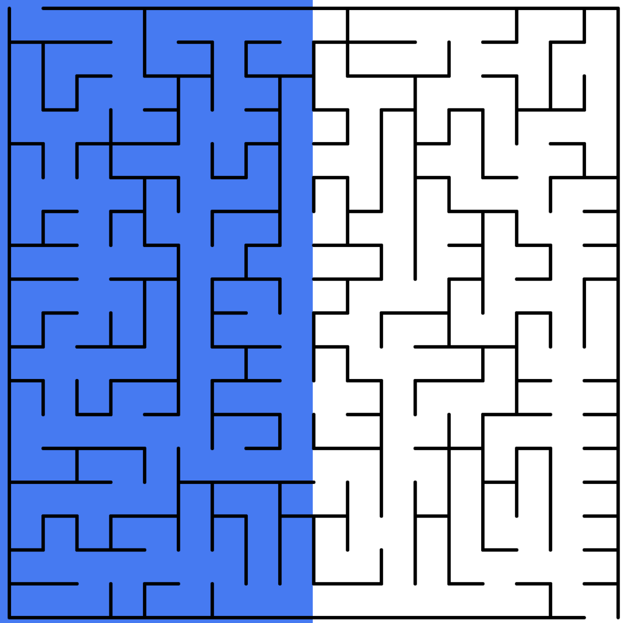
16-745 Optimal Control and Reinforcement Learning

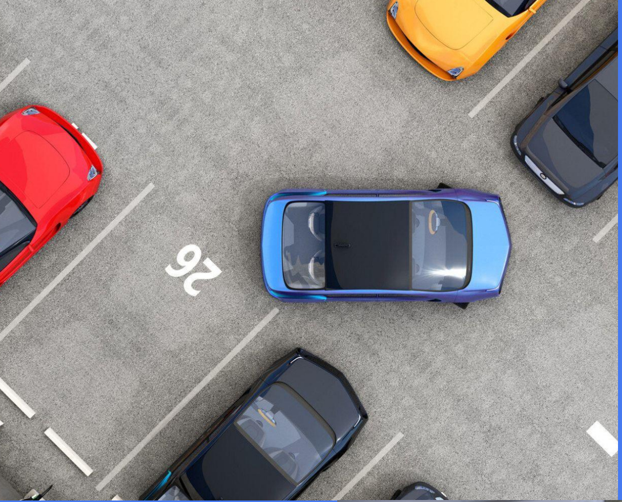
Trajectory Planning with obstacle avoidance

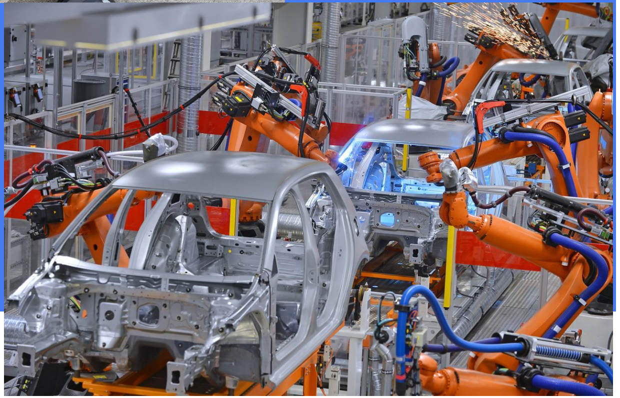


Team: Into the Unknown

John Zucca (jzucca)
Parv Parkhiya (pparkhiy)

Content





Motivation

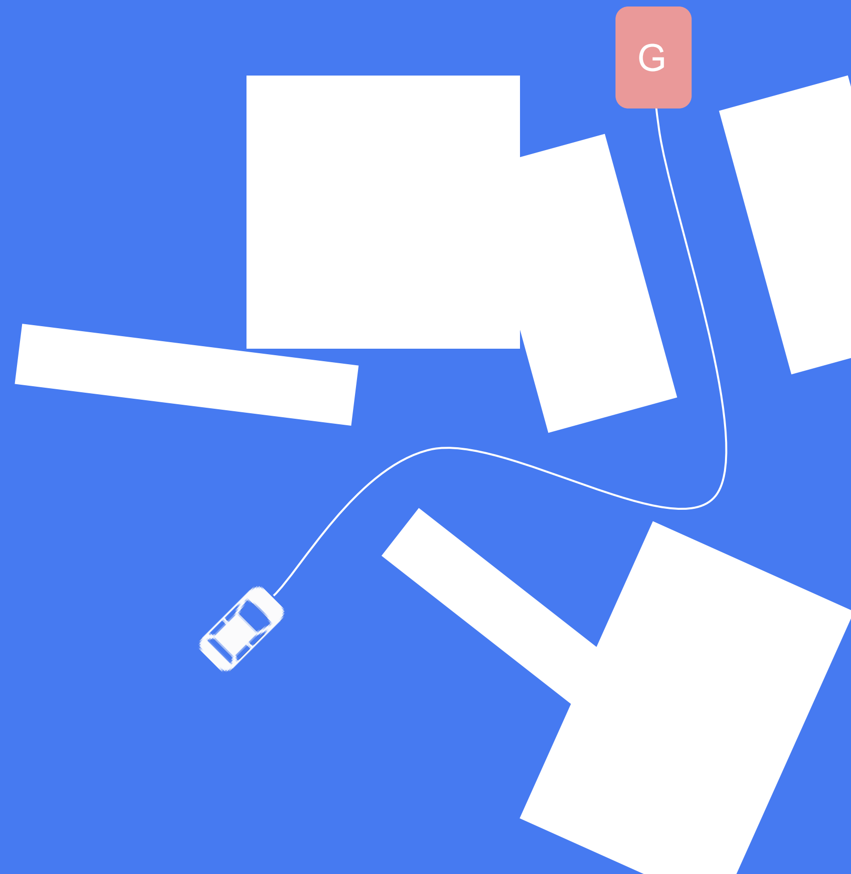
Robots operating in real world will always have plan and replan around the obstacles.

- Parking a Car
- Manufacturing using manipulator arm

Problem Statement

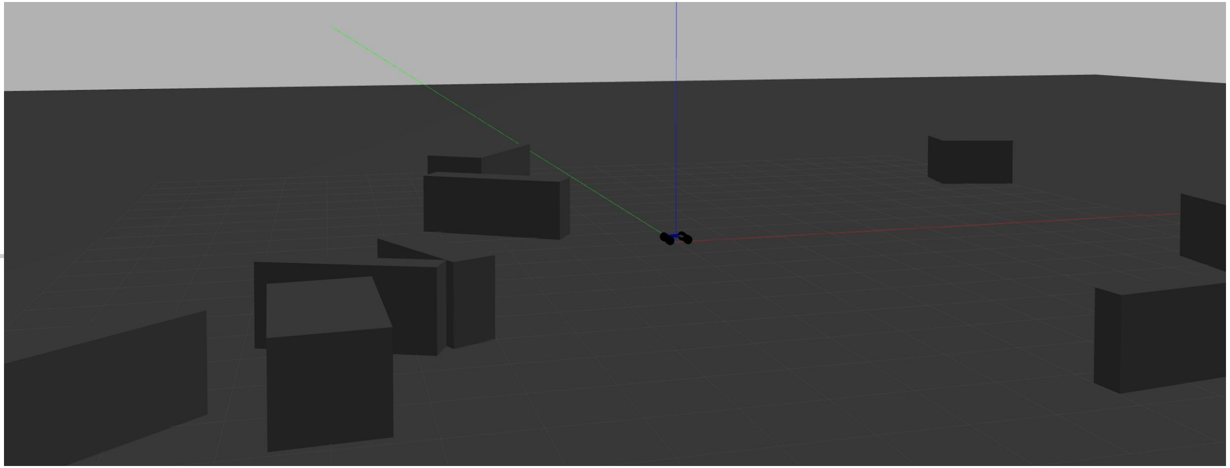
Traverse an Ackermann Vehicle from starting position to goal position in presence of obstacles

Goal:
Implement and compare classic approaches and propose an extension



Overview of the project

Gazebo-ROS Environment

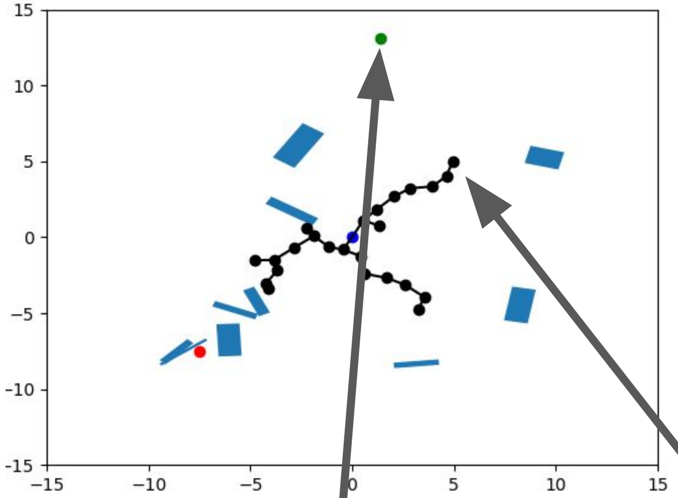


Script to generate a gazebo world with random obstacles

Approaches

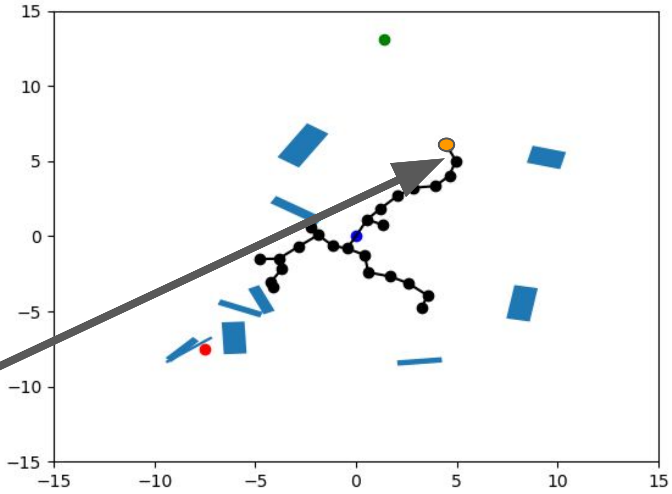
RRT — **A*** — **R*** — **Proposed Extension**

RRT
Rapidly Exploring Random Tree

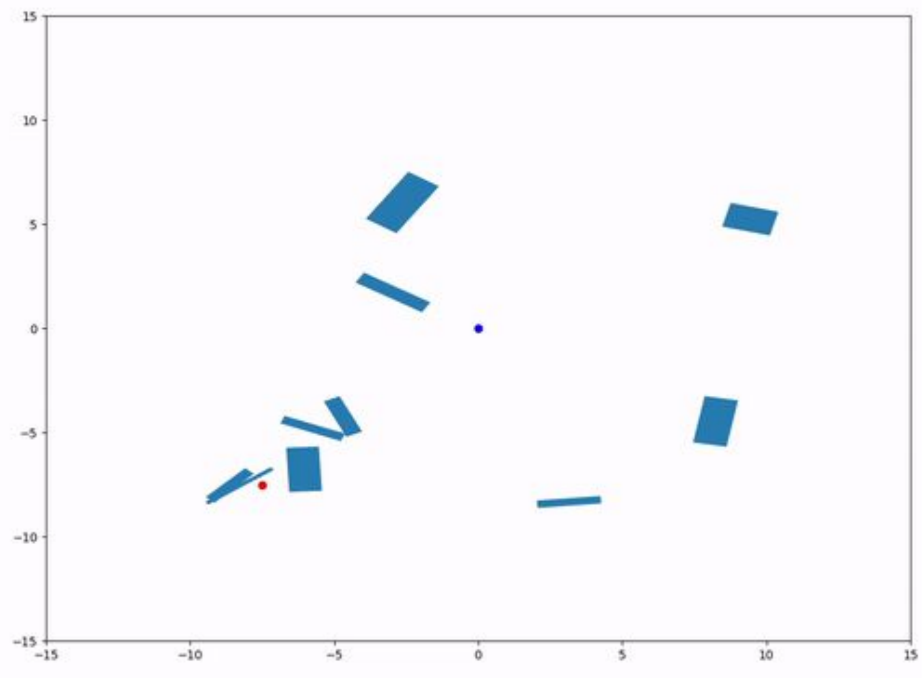


Growing a tree:

- Sample a *random point* in the space
- From the existing tree structure find the *closest point*
- Consider the *new edge* from *closest point* in the direction of the *sampled point*
- If edge's intersection with obstacle is null, add the *edge* and corresponding *node* in the tree



RRT (Implementation)

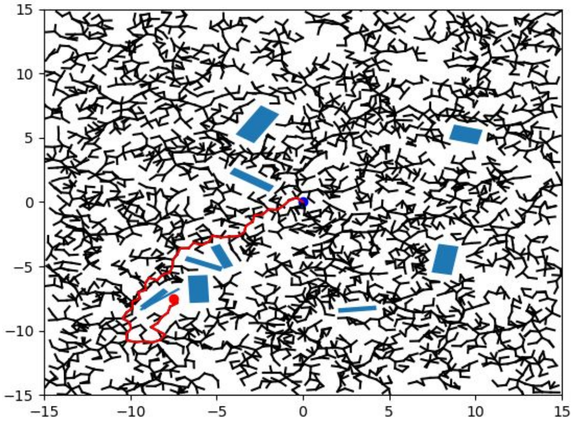


RRT in action

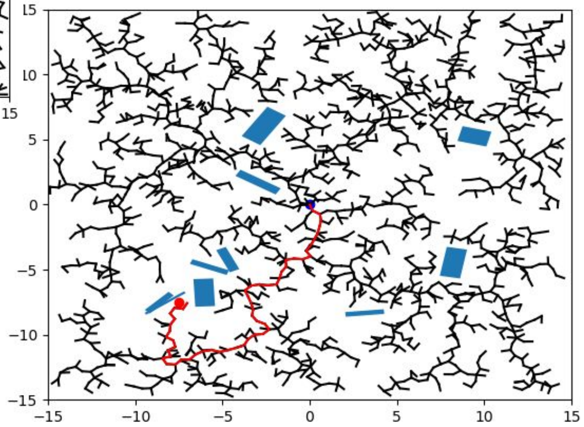
Key Details:

- Implemented from scratch
- A sampled node is sampled near goal region with some probability (Let's call it goal bias)
- Step length is also randomly sampled each time
- Kd-Tree for efficiently finding the closest node to sampled node
- Occupancy grid for validating new nodes and edge

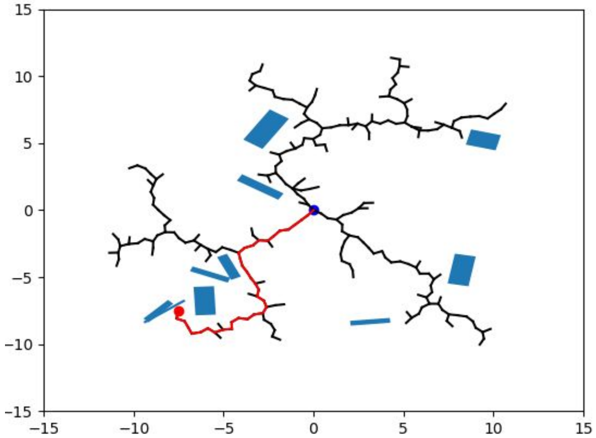
RRT (Results)



Without Goal Bias



Goal Bias 10%

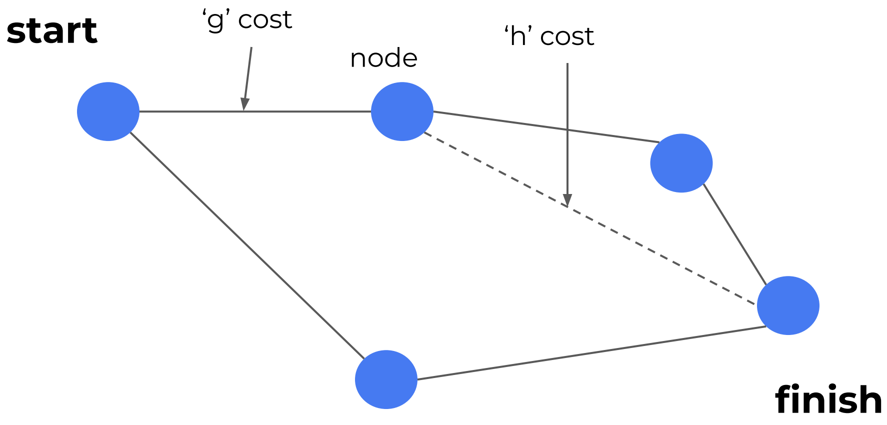


Goal Bias 25%

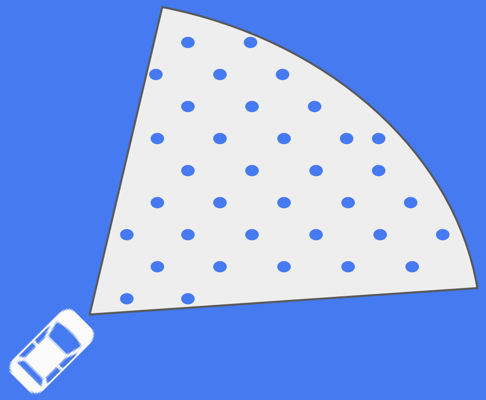
Creating a trajectory with A*:

- Maintain two lists of nodes
  - Nodes that have not been explored (open list)
  - Nodes that have been explored (closed list)
- Each node consists of its (x,y) coordinate values and direction theta.
- Each node has two associated costs:
  - From the start node to the current node (‘g’ cost)
  - Estimated cost (heuristic) to the goal (‘h’ cost)
- Chose the node from the open list with lowest total cost (g+h) to explore
- Generate a list of successors to this node based on system constraints (e.g. steering angle and obstacles)
- Evaluate the cost functions for these new nodes and add each node to the open list.
- If one of these nodes already exists in the open list and the stored ‘g’ cost is greater, update this node in the open list, as a cheaper path to this node has been discovered
- Iterate until the goal has been reached

A*



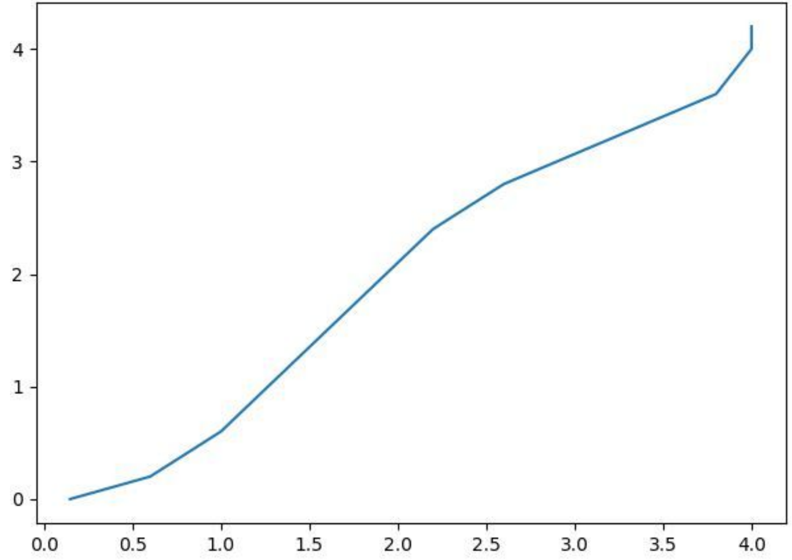
A*



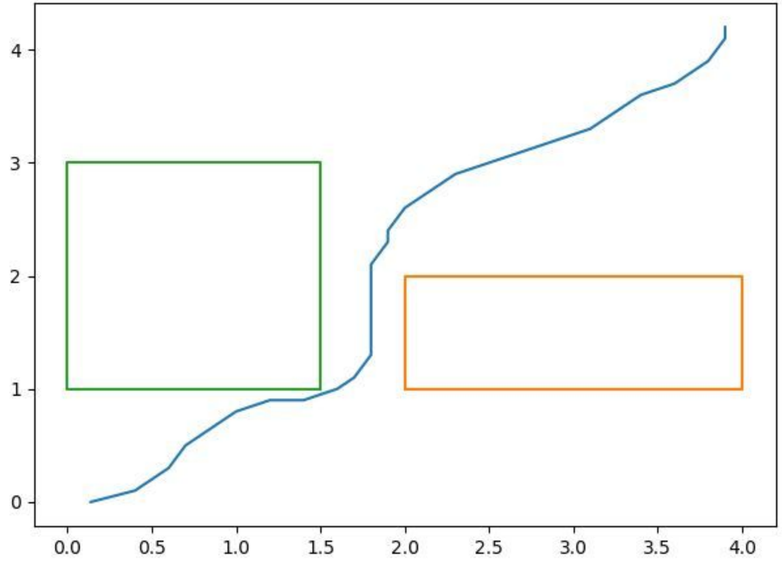
Key Implementation Details:

- Implemented from scratch
- (X,Y) grid defined in .1 meter increments
- ‘g’ cost is defined as euclidean distance
- ‘h’ cost is defined as $\sqrt{((\Delta x)^2+(\Delta y)^2+(\Delta \theta)^2)}$
- Allow successor nodes to be up to .5 meters away
- Each node also store its previous node, allowing for simple reconstruction of the trajectory

A*
(Results)

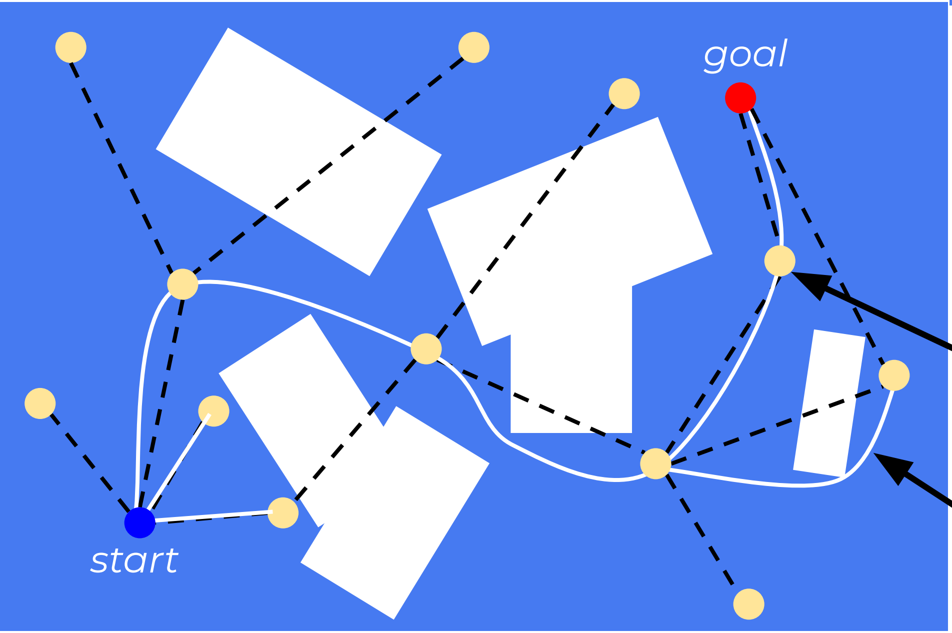


No Obstacles



Multiple Obstacles

R*
(Randomized A*)



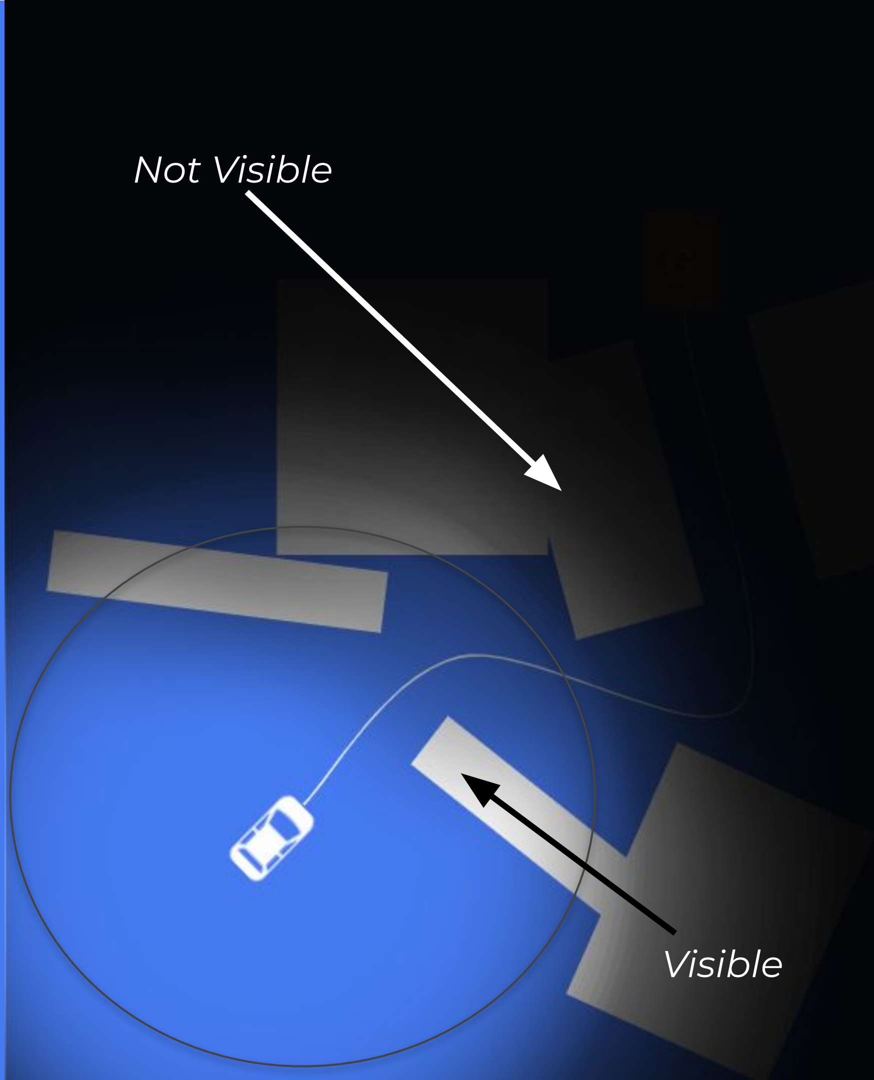
Key Idea:

- Weighted A*
  - *f = g + w*h* (*where weight(w) > 1.0*)
- Bounded suboptimal solution (*w*optimalCost*) but can convergence quite early
- Searching at 2 spatial scale level
- *High level graph* can quickly cover large area but feasible path between them may or may not exist.
- If certain node in the high level graph is found to be promising based on heuristic and cost, then only *Low level path* is attempted to be calculated using time-bound weighted A*

Proposed Extension

Real Time Trajectory Modification

- In practice, not all obstacles are known prior to driving.
- We must be able to adapt and change our desired trajectories in real time, as new information presents itself
- We aim to limit the obstacles that the vehicle can see during initial trajectory planning, and reveal these obstacles when they are within a defined distance of the vehicle.
- We are likely to use the algorithm that yields the quickest results, as computation time is crucial for this task



Limitations

RRT:

- Not Optimal At All
- Not Smooth
- Convergence not consistent

A*:

- Slow(current implementation)
- Smoother than RRT, but still not Smooth
- Consumes A Lot of Memory
- Local Minima takes long
- Lack of precision(fixed grid)

R*:

- Sub Optimal

Applicable to all:

- Can not react to new obstacles introduced at runtime

Future Work

A*:
- Optimization of code for faster runtime
- Further refine cost functions
- Explore utilizing a non-fixed grid

RRT:
- Implementing RRT* (if time permits)

R*:
- Full implementation

Proposed Extension:
- Recompute trajectories while vehicle is moving as obstacles are sensed by the vehicle

Applicable to all:
- Integrate with Pure Pursuit controller
- Add width of the vehicle into consideration
- Compare accuracy, trajectory feasibility, and compute time of each implementation

Questions?