Introduction to Unmanned Vehicle Systems
Fall 2019

Name: Lalit Goski
UTA ID: 1001728801

AUVSI Student Competition Homework

Deliverable 5:

Competition Name: Intelligent Ground Vehicle Competition (IGVC) with Rulebook 2019.
Challenge: Auto-Nav Course

1. Path Planning needed for the competition

Path planning deals with the problem of finding the optimal path from start state to goal without colliding with obstacles.
We know that a GNC (Guidance-Navigation-Control) system can be used to guide the unmanned vehicle to reach its goal state using the waypoints. But there are some problems with using this approach for the vehicle navigation.

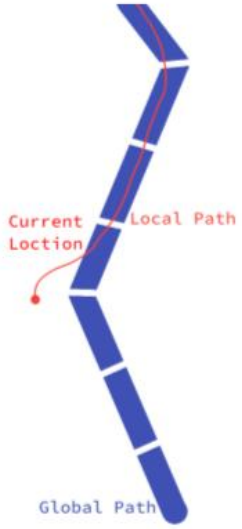
- The waypoints have to lead to the goal state and at a minimal cost.
- The waypoints need to avoid the collisions on its trajectory to the goal state.

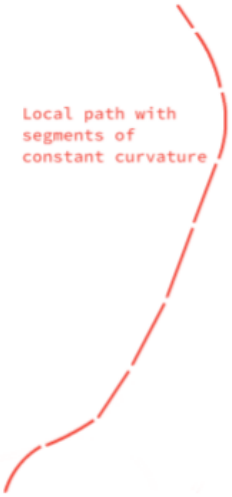
So, rather than manually finding the optimal waypoints for the UGV system by considering all the failure points on the trajectory, the path planning approach computes a safe guidance to reach a specific goal in an environment.
The path planning approach considers the dynamics of the unmanned vehicle which makes it a better approach. But this approach assumes to have the resources of the mapping of its environment.

The best feasible path planning approach for the UGV to successfully complete the Auto-Nav course is to use two planners, Global Path Planner and a Local Path Planner.
Having two planners help to efficiently generate paths that are locally feasible for the UGV system but also to be globally accurate in a computationally efficient manner. For path planning, it is found that the RRT* algorithm to be a robust method of finding the optimal path towards our next goal.

Why not Potential Field Approach
This method was prone to getting stuck in local minima (depending on the obstacle map layout). Plus, the Navigation functions are hard to compute.

Why not Cell Decomposition Approach: it is easy to compute but the precision and completeness is limited by the resolution of the cells, which requires large memory cost.





Global Planner:

The perception module provides the planner with waypoints to traverse through the field. These waypoints are sequentially provided to the planner, which then uses the RRT* algorithm to generate an optimal path from the current position to these way points taking care of the obstacles in between. The global planner plans a course through the entire cost map at a low frequency, typically once per second.

Local Planner:

Local Planners deal with finding optimal path over short distances in order to avoid dynamic obstacles whilst the Global Planner finds the overall optimal path from start to goal. The local planner runs at a higher frequency and is responsible for moving the robot along the global plan. For example, if the rover is a foot off course, then the local planner should correct for it instead of re-running the global planner.

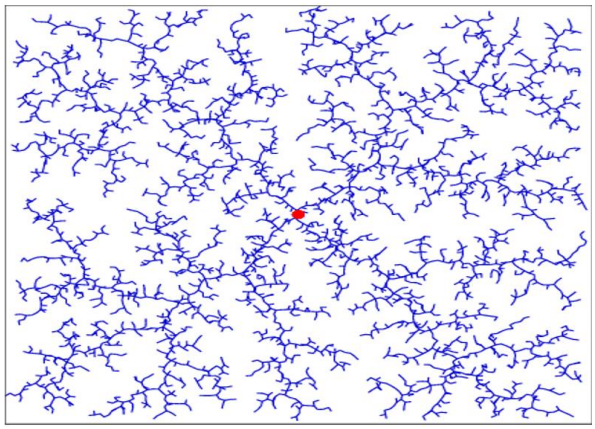
The local planner takes in the pathway points and produces a linear and angular velocity profile which takes care of the kinematics along the generated path.

The Local Planner that is used is integrated with the Pure Pursuit Algorithm based Controller.

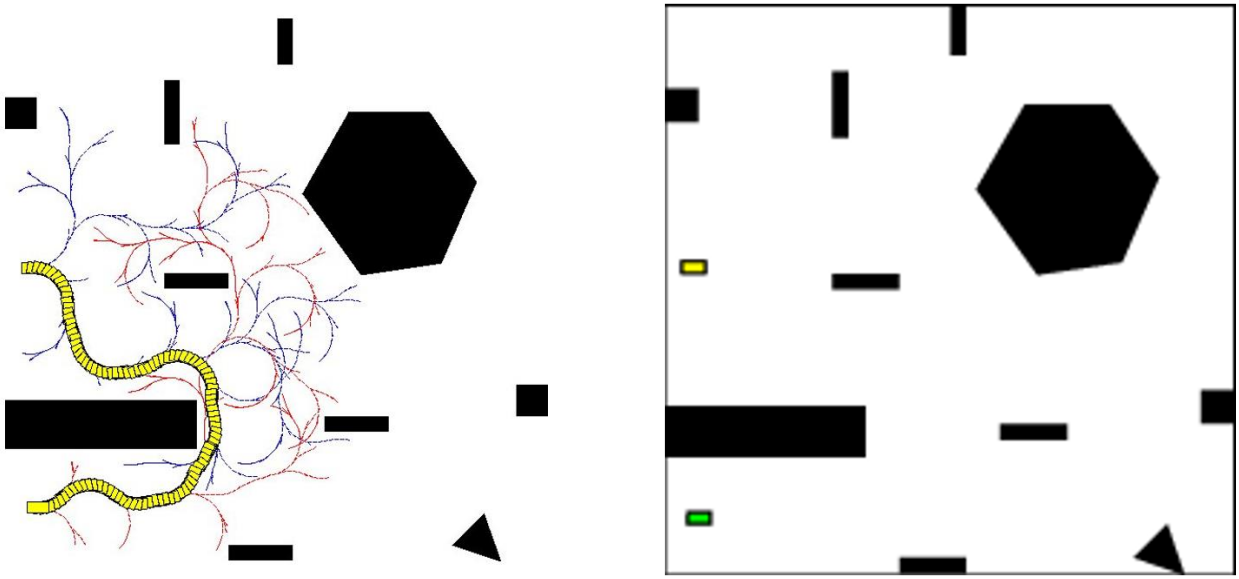
Pure Pursuit Algorithm

The controller gives appropriate control inputs to the robot to ensure that the planned path is followed. Implying a pure-pursuit based controller to move the robot smoothly while taking its non-holonomic constraints into account proves to be effective. The controller essentially tries to mimic the human driving nature by taking a point at a fixed distance in front of the vehicle as the immediate goal. This goal changes continuously as the vehicle moves on the planned path, traversing through a coarse set of waypoints. Once this look-ahead goal point is found, the controller algorithm plans a smooth curvilinear path from the current position of the vehicle to this point which is used to generate velocities to be given to the wheels for following the intended course.

Why RRT(Rapidly exploring Random Trees) works well



It can efficiently deal with unseen obstacles without the need to re-run the algorithm multiple times. RRT takes in cost of each grid given by the cost map and gives the optimal path by growing a tree towards the goal. Only a part of the tree is removed whenever an obstacle is encountered, and the forest of trees thus formed are recombined into one tree by random growth of vertices. RRT finds the solution to single query problems efficiently but RRT* extends RRT to find the optimal path but asymptotically and in doing so finds the optimal path from a given state to every other possible state within the planning domain. This is inefficient for a single-query problem.



Informed RRT* intelligently chooses only a subset of these possible states and finds the optimal path. This subset is chosen such that inclusion of any state outside this set does not significantly improve the solution. The states in this subset are described by a prolate hype-spheroid(heuristic). Also, for the given subset the problem converges at a faster rate towards the optimal solution.

Dijkstra Path Planning

Purple cells indicate the cells visited before finding an optimal path.

Dijkstra's Algorithm finds the shortest path by expanding vertices in the graph starting with its initial pose/vertex. It expands the closest unexpanded vertices in an iterative manner until it reaches the goal. Dijkstra's will always find the shortest path while none of the edges have negative cost. This planner is the easiest to implement and will always find the shortest path.

A* Path Planning

Purple cells indicate the cells visited before finding an optimal path.

A* is very similar to Dijkstra's algorithm, the only difference being that it considers both the exact cost of expanding a vertex as well as the heuristic cost. In each iteration it maintains a balance between the two. It chooses the vertex with the least of the sum of exact cost and heuristic cost. A* like Dijkstra is simple to implement and is very reliable.