Centralized Multi-Robot Planning using dRRT Algorithm

Manoj Velmurugan
Robotics Engineering
Worcester Polytechnic Institute
v.manoj1996@gmail.com

Ashwin Disa
Robotics Engineering
Worcester Polytechnic Institute
ashwin.disa@gmail.com

***Abstract*—This project aims to plan a path for multiple robots simultaneously by treating the robots as a single composite system with many degrees of freedom, known as centralized multi-robot planning. The plan is to utilize dRRT (discrete Rapidly Exploring Random Tree) algorithm over preplanned single robot PRM (Probabilistic Roadmap) graphs. Instead of doing a tree search over the entire cartesian workspace and performing expensive collision checks over and over again, we are planning over a environment-collision free state subset obtained via the tensor product of individual robot PRMs.**

***Index Terms*—PRM, dRRT, multi-robot, centralized path-planning**

I. Problem Statement

Compute a plan for all robots simultaneously by treating the robots as a single composite system with many degrees of freedom, known as centralized multi-robot planning.

A naive approach to solving this problem constructs a PRM for each robot individually, and then plans a path using a typical graph search in the composite PRM (the product of each PRM). Unfortunately, composite PRM becomes prohibitively expensive to store, let alone search. If there are k robots, each with a PRM of n nodes, the composite PRM has n^k vertices!

More specifically, the project must try to solve the planning scenarios provided in Figure 1. In the first scenario, robot 1, robot 2 and robot 3, robot4 should exchange positions. And in the second scenario, each robot should reach the diametrically opposite point on the circle without colliding into each other.

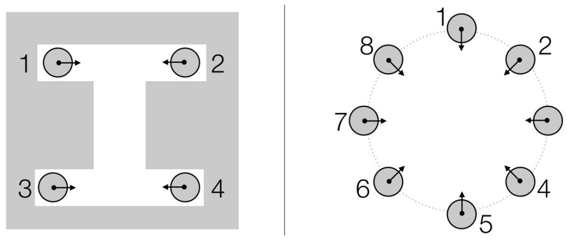


Fig. 1: Problem Statement scenarios

II. Related Work

Searching the exponentially large composite roadmap for a valid multi-robot path is a significant computational challenge. Recent work to solve this problem suggests implicitly searching the composite roadmap using a discrete version of the RRT algorithm (dRRT). At its core, dRRT grows a tree over the (implicit) composite roadmap, rooted at the start state, with the objective of connecting the start state to the goal state. Our project is based on the work on Multi Robot Discrete RRT (MRdRRT) in [1].

III. Methodology

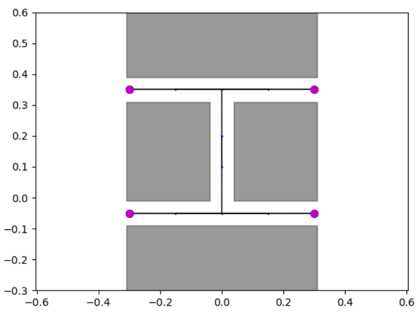
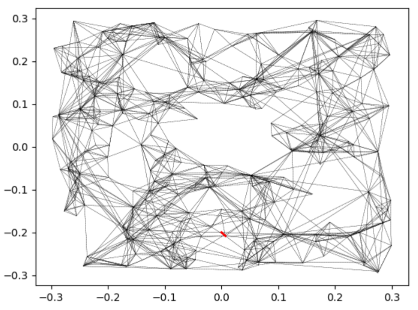



Fig. 2: PRMs for required scenarios.

For n robots moving in 2D cartesian space, the state space is 2*n dimensional. Searching for a collision free path is computationally expensive and it is equivalent to looking for a needle in haystack. One way to simplify the computation is to first obtain the PRM graph for individual robots. This PRM graph already contains collision free (against environment) path for a single robot. The Cartesian product of the individual PRMs would give a composite graph that contains the potential collision free coordinates. Then, a graph search algorithm like

A* can be implemented to plan a path around. PRMs for the given scenarios are shown in Figure 2.

But given that we have n robots, the memory cost for storing such composite roadmap graph is extremely high. One workaround is to not represent the composite graph explicitly and use an algorithm to sample points from the implicit composite roadmap graph. This is done using the dRRT algorithm as follows.

- Sample a (composite) configuration q_{rand} uniformly at random.
- Find the state q_{near} in the dRRT nearest to the random sample in the existing RRT graph.
- Using an $expansion oracle$, find the state q_{new} in the composite roadmap that is connected to q_{near} in the closest direction of q_{rand}. This is done using the individual PRM edges. Whichever edge is in the closest direction of q_{rand} is chosen for an individual robot.
- Perform a robot to robot collision check with the q_{new} point. If there is no collision, add it to the tree.
- Repeat until the goal is successfully added to the tree.

This is shown in Figure 3.

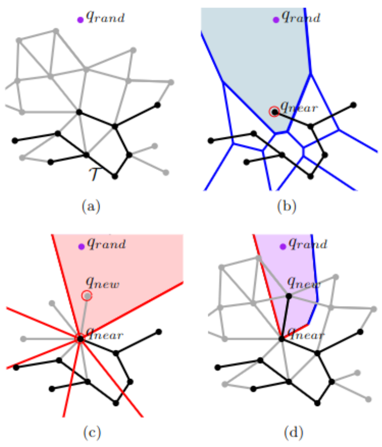


Fig. 3: Expansion Oracle.

IV. Discrete RRT

Since the graph serves as an approximation of some relevant portion of the Euclidean space, traversal of the graph can be viewed as a process of exploring the subspace. The dRRT algorithm rapidly explores the graph by biasing the search towards vertices embedded in unexplored regions of the space.

Let $G = (V, E)$ be a graph where every $v \in V$ is embedded in a point in Euclidean space and every edge $(v, v') \in E$ is a line segment connecting the points. Given two vertices $s, t \in V$, dRRT searches for a path in G from s to t. As G is represented implicitly, the algorithm uses an *oracle* to retrieve information regarding neighbors of visited vertices.

A. Expansion Oracle

In order to retrieve partial information regarding the neighbors of visited vertices, dRRT consults an oracle described below, Given two points $(v, v') \in [0, 1]^d$, denote by $\rho(v, v')$ the ray that starts in v and goes through v' . Given three points $v, v', v'' \in [0, 1]^d$, denote by $\angle_v(v', v'')$ the (smaller) angle between $\rho(v, v')$ and $\rho(v, v'')$. In other words, the direction oracle returns the neighbor v' of v such that the direction from v to v' is closest to the direction from v to u. This is also shown in Figure. 3.

B. Description of dRRT

At a high level, dRRT proceeds similar to the RRT algorithm, and we repeat it here for completeness. The dRRT algorithm grows a trees which is a subgraph of G and is rooted in s. The growth of the tree is achieved by an expansion towards random samples. Additionally, an attempt to connect Tree with t is made. The algorithm terminates when this operation succeeds and a solution path is generated. Here G is the roadmap, s and t are the start and goal configurations respectively. Expansion of T is performed by the $EXPAND$ operation which performs N iterations that consist of the steps given above.

Algorithm 1 dRRT Planner

```
Require: $s, t$
1: $T.init(s)$
2: while $true$ do
3:     $EXPAND(T)$
4:     $\pi \leftarrow CONNECT\_TO\_TARGET(T, t)$
5:     if $not\_empty(\pi)$ then
6:         return $RETRIEVE\_PATH(T, \pi)$
7:     end if
8: end while
```

Algorithm 2 EXPAND

```
Require: $T$
1: for $i = 1 \rightarrow N$ do
2:     $q_{rand} \leftarrow RANDOM\_SAMPLE()$
3:     $q_{near} \leftarrow NEAREST\_NEIIGHBOR(T, q_{rand})$
4:     $q_{new} \leftarrow ORACLE(q_{near}, q_{rand})$
5:     if $q_{new} \notin T$ then
6:         $T.add\_vertex(q_{new})$
7:         $T.add\_edge(q_{near}, q_{new})$
8:     end if
9: end for
```

After the expansion, dRRT attempts to connect the tree T with t using the $CONNECT\_TO\_TARGET$ operation. For every vertex q of T , with one of the K nearest neighbors of t in T, an attempt is made to connect q to t using the method $LOCAL\_CONNECTOR$ which is a crucial part of the dRRT algorithm. A local connector is a mechanism that connects the paths generated by individual robots at a local level. It addresses the coordination challenges between robots by ensuring that the paths are compatible and collision-free in the shared environment. Finally, given a path from some node q of T to t the method $RETRIEVE\_PATH$ returns the concatenation of the path from s to q, with π.

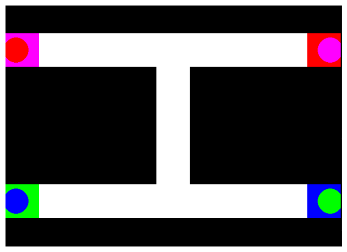
Algorithm 3 CONNECT TO TARGET

```
Require: $T, t$
1: for $q \in NEAREST\_NEIGHBOR(T, t, K)$ do
2:     $\pi \leftarrow LOCAL\_CONNECTOR(q, t)$
3:     if $not\_empty(\pi)$ then
4:         return $\pi$
5:     end if
6: end for
7: return $\phi$
```

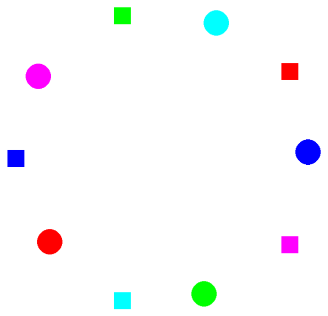
C. Local Connector

In the general dRRT algorithm the local connector is used for connecting two given vertices of a graph. Given two vertices $V = (v_1, v_2, .., v_m)$, $V' = (v'_1, v'_2, .., v'_m)$ of G we find for each robot i a path π_i on G_i from v_i to v'_i. The connector attempts to find an ordering of the robots such that robot i does not leave its start position on π_i until robots with higher priority reached their target positions on their respective path, and of course that it also avoids collisions. When these robots reach their destination robot i moves along π_i from $\pi_i(0)$ to $\pi_i(1)$. During the movement of this robot the other robots stay put.

V. Results



(a) H scenario



(b) Circle scenario

Fig. 4: Path visualization

We implemented MRdRRT for the two case scenarios shown above using the Matplotlib python library to visualize the path given by the algorithm. The robots are denoted by a unique color and their respective goal positions by a rectangle of the same color. Black region are the obstacles and white is the collision free region.

The algorithm returned a feasible path for the first scenario at all times. The second scenario was tested with number of robots starting with 2 and gradually increasing the number. The algorithm returned successful paths for 5 robots at all times. As the robots were increased more than 5, the algorithm was not able to find a solution in finite time in most cases. This issue could be solved by either of the following ways: decreasing the size of the robot, increasing the number of nodes in the PRM or increasing the wait time. The algorithm ran for 5000 iterations but it took more than finite time to run all iterations. The issue was solved by decreasing the size of the robot.

A simulation in MuJoCo was built as a further step. The visualization snapshot can be seen in Figure. 5. A state flow chart was implemented in Simulink as shown in Figure 6. It takes in the current position and heading, provides a linear velocity and turn rate as output. A Proportional controller was used for heading. If the robot gets too close to a waypoint, it will switch over to the next waypoint, provided all the other robots have reached their respective waypoints.

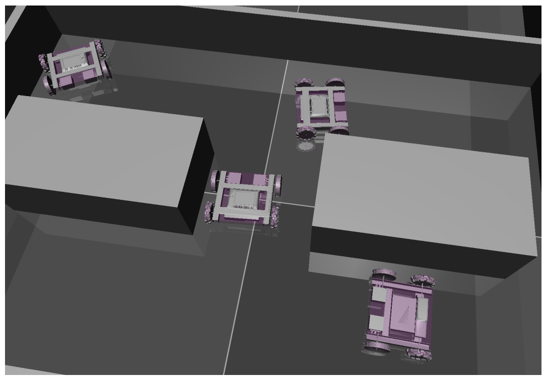


Fig. 5: MuJoCo Visualization

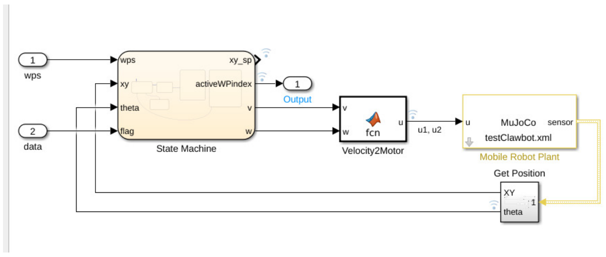


Fig. 6: State Flow in Simulink

References

[1] Solovey, Kiril, Oren Salzman, and Dan Halperin. "Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning." In Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics, pp. 591-607. Springer International Publishing, 2015.

[2] Multi-Robot Discrete RRT planner. https://github.com/mrsd16teamd/MRdRRT.