Sumeet Shanbhag

RRT Documentation

- **For RRT, what is the main difference between RRT and RRT*? What change does it make in terms of the efficiency of the algorithms and optimality of the search result?**

  Rapidly exploring random trees (RRT) and its variant RRT* are both probabilistic algorithms used for path planning in robotics. The main difference between RRT and RRT* lies in the way they handle the exploration of the search space and the optimality of the resulting path.

  In RRT, the search space is explored by adding random nodes to the tree and connecting them to the nearest existing node. The algorithm continues to explore the search space until it reaches the goal region. However, the resulting path may not be optimal as it only guarantees to find a feasible solution, but not necessarily the shortest path.

  In RRT*, on the other hand, the algorithm introduces an additional step of rewiring the tree after adding new nodes. This means that after a new node is added to the tree, all nodes that can be connected to it with a shorter path are rewired to it. This results in a more efficient exploration of the search space and a more optimal solution as the algorithm continuously improves the path towards the goal.

  The introduction of the rewiring step in RRT* makes it more efficient than RRT, especially in high-dimensional search spaces where the probability of finding a path to the goal region is low. However, the added complexity of the rewiring step also makes RRT* computationally more expensive than RRT. Nonetheless, the optimality of the search result in RRT* makes it a good choice for applications where the cost of failure is high or where finding an optimal path is critical.

- **Compare RRT with the results obtained with PRM in the previous assignment. What are the advantages and disadvantages?**

  RRT (Rapidly exploring random trees) and PRM (Probabilistic Roadmap) are two popular algorithms used for motion planning in robotics. PRM uses a sampling-based approach to construct a roadmap of the free space and then searches for a path on the roadmap, while RRT builds a tree structure by randomly exploring the free space and then connects nodes to form a path. When comparing the sampling methods used in PRM with RRT, there are some advantages and disadvantages to consider:

  1. Uniform Sampling:

  - Advantage: It is easy to implement and is the simplest method of sampling. It can also provide good coverage of the search space.
  - Disadvantage: It may not be efficient in exploring certain parts of the search space, leading to a suboptimal solution.

  2. Gaussian Sampling:

- Advantage: It can provide better coverage of the search space compared to uniform sampling, as it biases the samples towards unexplored areas.
- Disadvantage: It can be computationally expensive, especially when the search space is large.

3. Bridge Sampling:

- Advantage: It can provide better coverage of narrow passages and obstacles, which are common in robotics applications.
- Disadvantage: It requires the knowledge of the environment's topology and structure, which may not always be available or may be difficult to obtain.

4. Random Sampling:

- Advantage: It is simple to implement and provides good coverage of the search space.
- Disadvantage: It may not be efficient in exploring certain parts of the search space, leading to a suboptimal solution.

When comparing PRM and RRT, PRM can be more computationally efficient for finding a global optimal path, especially when the search space is complex or has many narrow passages. This is because PRM builds a roadmap of the free space first and then searches for a path on the roadmap. On the other hand, RRT is more suitable for finding a locally optimal path and is more efficient when the search space is less complex or when the start and goal configurations are close to each other.

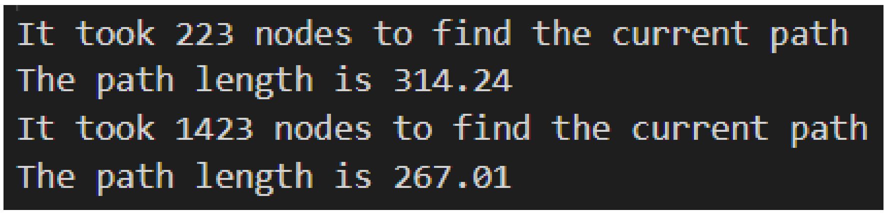
In summary, both RRT and PRM have their advantages and disadvantages, and the choice of algorithm and sampling method depends on the specific application and the requirements of the problem.

- Algorithm results and explanation

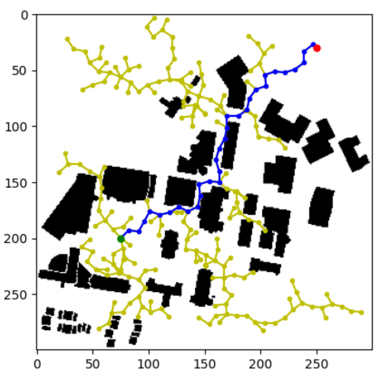
The **RRT** class includes the following functions:

- **__init__(self, map_array, start, goal)**: Initializes the RRT object with a binary array representing the map, the starting point, and the goal point.
- **init_map(self)**: Resets the vertices and found flag to perform a new search.
- **dis(self, node1, node2)**: Calculates the Euclidean distance between two nodes.
- **check_collision(self, node1, node2)**: Determines if there is an obstacle between two nodes.
- **get_new_point(self, goal_bias)**: Chooses a new point to add to the tree with a possibility of choosing the goal.
- **get_nearest_node(self, point)**: Finds the nearest node in the tree to a given point.
- **extend(self, extend_dis=10, goal_bias=0.05)**: Adds a new node to the tree by extending from a random point towards the nearest node.

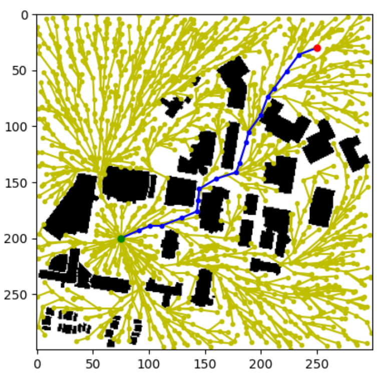
- **get_neighbors(self, new_node, neighbor_size)**: Finds the nodes in the tree that are within a given distance of a new node.
- **rewire(self, new_node, neighbors):** It reassigns the parent node of some existing nodes in the tree to a new node if the path to the new node from the existing node is shorter than the original path to the parent node of the existing node.



```
It took 223 nodes to find the current path
The path length is 314.24
It took 1423 nodes to find the current path
The path length is 267.01
```



RRT



RRT*

- **Why RRT and RRT* result in different trees?**

  They both start from an initial configuration of the robot and iteratively grow a tree of feasible configurations that connect the initial and goal configurations. However, the way they grow the tree is different, which can result in different trees.

  The main difference between RRT and RRT* is the way they handle the tree's growth. RRT grows the tree by extending the tree from a randomly chosen node towards a new random configuration, whereas RRT* extends the tree from the node that has the lowest cost-to-come from the initial configuration towards a new random configuration.

  RRT* is an extension of RRT that aims to improve the quality of the generated tree by considering the cost of the path that connects the nodes. By using cost-to-come information, RRT* tries to steer the growth of the tree towards the optimal path. As a result, RRT* can generate more optimal paths with fewer nodes than RRT.

  In summary, RRT and RRT* use different strategies for tree growth, which can result in different trees. While RRT is generally faster and generates a tree that connects the initial and goal configurations, RRT* can produce a more optimal and shorter path.

- **How different sampling methods lead to different sample sets in the graph?**

  Rapidly-exploring random tree (RRT) and its variant RRT* use different sampling strategies to generate new nodes in the graph.

  RRT uses uniform random sampling, which means that at each iteration, it randomly selects a point in the configuration space and attempts to connect it to the existing tree. This sampling strategy tends to produce a more spread-out tree with nodes uniformly distributed throughout the configuration space.

  On the other hand, RRT* uses a technique called "biased sampling." Biased sampling assigns a probability distribution to the space, such that the probability of sampling a point in a region is proportional to the amount of unexplored space in that region. This means that RRT* tends to focus on unexplored areas of the configuration space, resulting in a more directed and efficient tree.

  Therefore, because the two sampling strategies are different, they will generally produce different sample sets and, consequently, different trees.