- 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?

The Rapidly-exploring Random Tree (RRT) algorithm and its variant RRT* are both probabilistic algorithms used for motion planning in robotics. The main difference between the two lies in how they handle the exploration of the state space and the optimization of the tree structure.

In RRT, the tree is grown by sampling random configurations in the state space and expanding the tree towards the nearest node in the existing tree. The algorithm keeps track of the path that has been explored so far and returns the first feasible path it finds from the initial state to the goal state. RRT does not guarantee optimality of the returned path, but it is typically faster than RRT*.

In RRT*, the tree is also grown by sampling random configurations and expanding towards the nearest node, but it uses a more sophisticated tree structure and rewiring technique to improve the optimality of the returned path. RRT* adds a cost function to each node in the tree to keep track of the cost of the path from the root to that node. When a new node is added to the tree, RRT* re-wires the tree to minimize the cost of the paths from the root to all nodes. This means that RRT* can often find a more optimal path than RRT, but at the cost of increased computation time.

In terms of efficiency, RRT is typically faster than RRT* because it has a simpler tree structure and does not perform as much rewiring. However, RRT* is more likely to find a more optimal path to the goal state, especially in high-dimensional state spaces with many obstacles.

Overall, the choice between RRT and RRT* depends on the specific motion planning problem at hand and the desired trade-off between efficiency and optimality.

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

From the results, it is evident that the main advantage of RRT over PRM is its ability to efficiently explore the state space by quickly building a tree of feasible paths from the start to the goal. RRT is particularly well-suited for problems with complex,

high-dimensional state spaces or environments with many obstacles, as it can quickly generate paths that are feasible and close to optimal. RRT can also be used for real-time motion planning applications, where quick exploration and planning is critical.

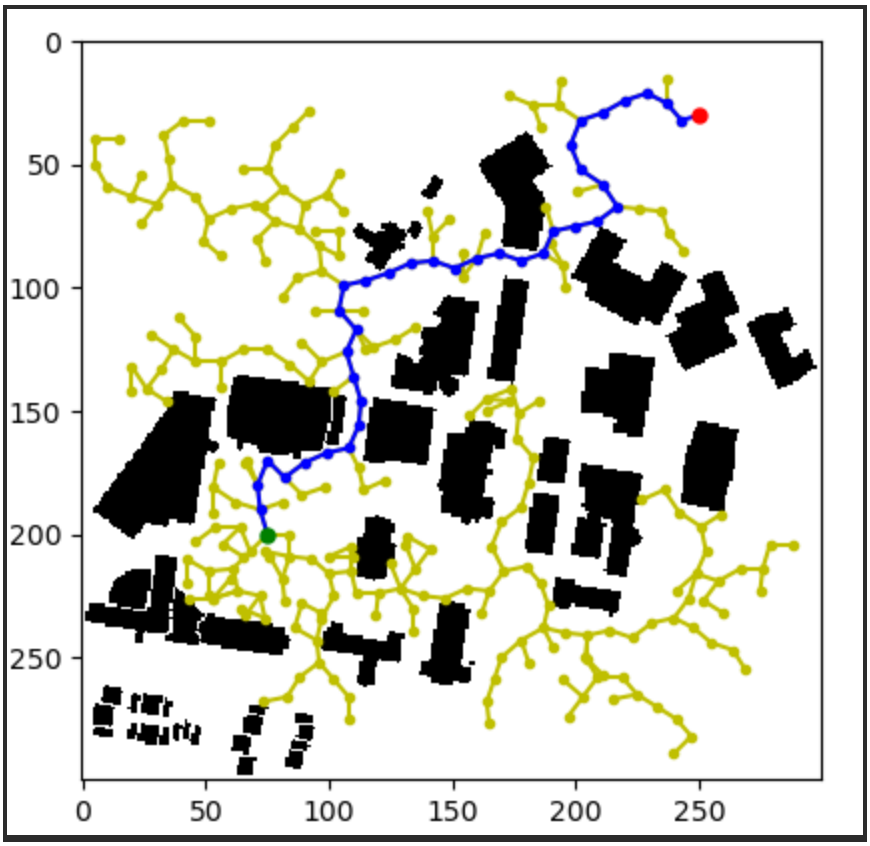
In contrast, PRM is typically more computationally expensive than RRT, but it often finds more optimal paths by constructing a roadmap of the state space and then searching for the shortest path between the start and goal states. PRM can also be more robust than RRT in situations where the environment is relatively static and the start and goal states are known in advance. PRM is particularly useful in problems where the robot needs to find a global solution, rather than a locally optimal one.

One disadvantage of RRT is that it does not guarantee optimality of the solution, especially in high-dimensional state spaces. RRT is also more likely to get stuck in local minima or to generate paths that are longer than necessary. PRM, on the other hand, is more likely to find a globally optimal solution, but it can be slower and more computationally intensive than RRT.

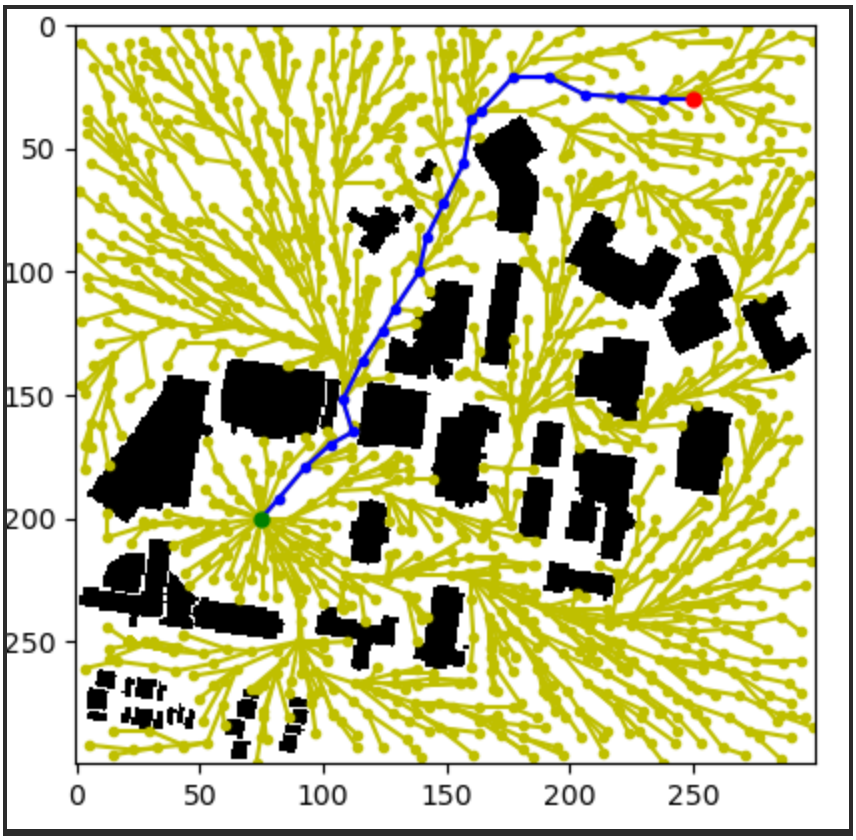
In summary, the choice between RRT and PRM depends on the specific motion planning problem at hand and the desired trade-off between efficiency and optimality. RRT is often preferred for problems with complex, high-dimensional state spaces or when real-time planning is needed, while PRM is often preferred when global optimality is required and the environment is relatively static.

Briefly explain why your algorithms would produce these results. Why RRT and RRT* result in different trees? How different sampling methods lead to different sample sets in the graph?

RRT



RRT*



In RRT, the tree is grown by sampling random configurations in the state space and expanding towards the nearest node in the existing tree. The algorithm adds a new node to the tree and connects it to the nearest existing node, forming a new edge in the tree. This process is repeated until the goal state is reached or a termination criterion is met. Since RRT only focuses on growing the tree towards the goal state, it may result in trees that are not as optimized for the overall cost of the path.

In RRT*, the tree is also grown by sampling random configurations and expanding towards the nearest node, but it adds a cost function to each node in the tree to keep track of the cost of the path from the root to that node. When a new node is added to the tree, RRT* re-wires the tree to minimize the cost of the paths from the root to all nodes. This means that RRT* is able to more efficiently explore the state space and optimize the tree structure to generate a more optimal path to the goal state.

In addition, RRT* uses a more sophisticated tree structure and rewiring technique to improve the optimality of the returned path. RRT* can often find a more optimal path than RRT, but at the cost of increased computation time.

Overall, the differences in the exploration and optimization mechanisms used by RRT and RRT* result in different tree structures, with RRT* typically resulting in a more optimized tree that generates a more optimal path to the goal state.

References

Chat gpt