# Standard Algorithm Implementation: PRM, RRT and RRT*

Shiva Kumar Tekumatla
*Robotics Engineering Department*
*Worcester Polytechnic Institute*
Worcester,MA
stekumatla@wpi.edu

***Abstract*—Many discrete planning methods have limitations and cannot be scaled easily to high-dimensional spaces. Discrete planning is sensitive to branching factor and the overall algorithm can be complex based on the c-space dimensions. Sampling based approaches addresses these problems to a great extent. In this work, the various types of sampling based algorithms are discussed. First, probabilistic road map(PRM) with different sampling methods is presented. For this, uniform sampling , random sampling , Gaussian sampling and bridge sampling are implemented. Further, two more sampling based approaches such as rapidly exploring random tree(RRT) and it's variation RRT* are implemented. Finally , the results between all these algorithms are compared.**

***Index Terms*—PRM, Unifrom Sampling,Gaussian Sampling, Random Sampling, Bridge Sampling, RRT and RRT***

## I. Introduction

Probabilistic Road Map (PRM) is a motion planning method that can solve the path planning problem between a start and goal position for a robot while avoiding the obstacles. The main idea of PRM is to sample the map randomly and connect these samples to other nearest samples. Then these connections are used with an another local planning algorithm to find the final path. There are two main steps in general implementation of PRM, learning phase and query phase. In the learning phase, the map is sampled with different nodes. This step is called construction step. Once the map is constructed, the nodes are connected to each other based on their proximity. This is called expansion step. Once all the neighbors are connected using expansion step, a local planning algorithm such as Dijkstra, A* etc is used to find the best path. This is called query phase. Figure 1 shows the general structure of the PRM implementation.

Rapidly exploring Random Tree(RRT) is an algorithm that can efficiently search non-convex spaces by randomly sampling the maps. The tree is grown from start until the goal point is found using one of the extend or connect methods. As each node is sampled, a connection is made to the nearest available nodes in the already constructed tree. This process is repeated until the max number of predetermined nodes is reached or the goal is reached. Each step in constructing the node is defined beforehand. This step can be determined as simply the euclidean distance. Figure 2 shows a simple RRT implementation on a map with free and obstacles.

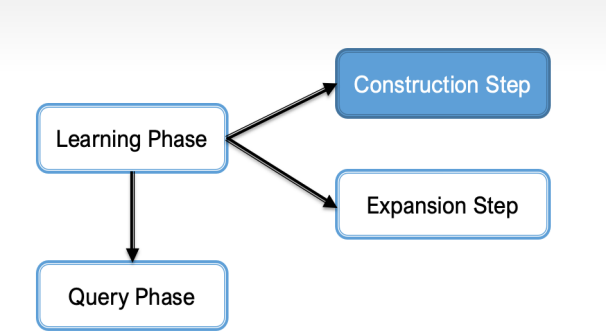


Fig. 1. PRM implementation steps

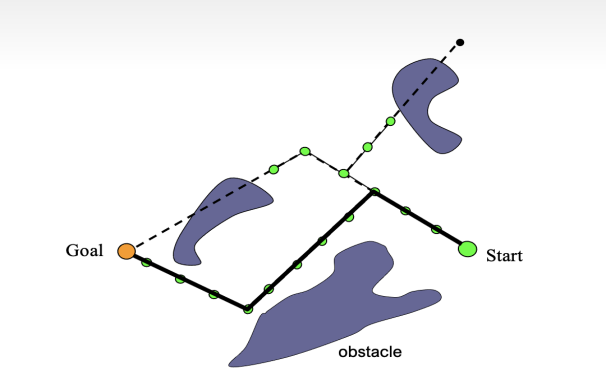


Fig. 2. RRT implementation using connect method

RRT* ,a variation of RRT , on the other hand tries to optimize the RRT path using rewiring method. In RRT*, based on the computation availability and optimization requirement, we can decide when to stop the search process. Mainly, in RRT*, the path is optimized using nearest neighbors. When the number of nodes approaches to infinity, this algorithm can deliver the shortest possible path. This may not be possible in actual implementation , but RRT* can result in better paths compared to RRT. Figure 3 shows the differences between RRT and RRT*.

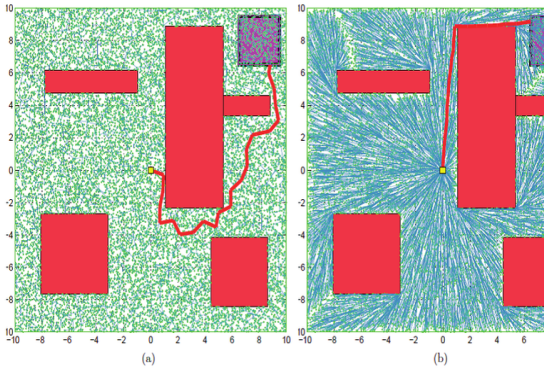


Fig. 3. Difference between path generated by RRT and RR

## II. Implementation of PRM

In this section, the implementation of PRM is discussed. PRM is implemented using four different types of sampling methods. They are , Uniform Sampling, Random Sampling, Gaussian Sampling and Bridge Sampling. RRT and RRT* implementation is discussed in this section as well. implementation of these algorithms, the map of WPI This map is given by figure 4.

Fig. 4. WPI map that is used for standard algorithms implementation

### A. Uniform Sampling

In uniform sampling, the given map is divided into uniform nodes. For this the map is divided into uniform grid like structure. After forming the grid, the sampled points that are in free spaces of the map is considered as the final sampled points. To determine if a sampled point to be considered or not , the location of it is checked with obstacles of the given map. In the given map, the free spaces are represented with 1s and obstacles are represented with 0s. Uniform sampling on the given map is shown in figure 5.

### B. Random Sampling

In random sampling, the nodes are randomly generated in the given map.Similar method that is explained in uniform sampling is used to check if a sampled point is valid or not. Randomly sampled nodes on the given map are shown in figure 6.

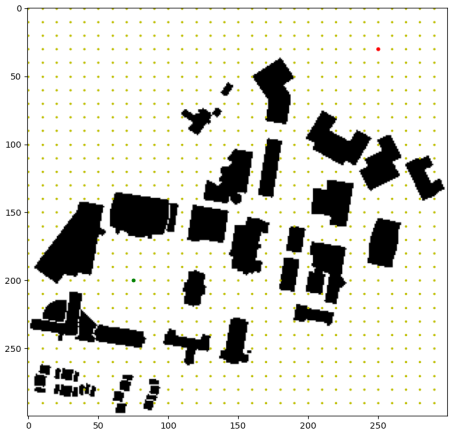

Fig. 5. Uniformly sampled nodes on the given map. Here green node is the start node and the red node is goal node.

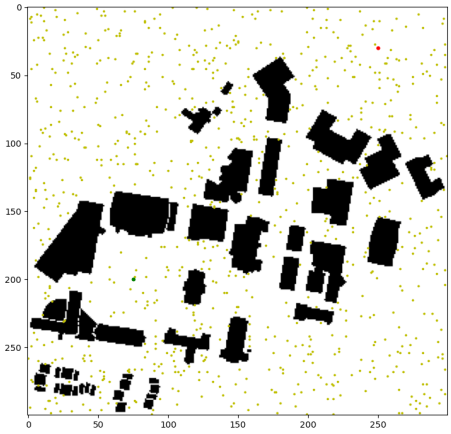

Fig. 6. Randomly sampled nodes on the given map. Here green node is the start node and the red node is goal node.

### C. Gaussian Sampling

In Gaussian sampling, initially a random node is generated. An another node is generated in the Gaussian distribution of the first node. If both of these nodes are in either free space or in the collision , then both of these nodes are discorded. If only of them is in collision , then the node that is collision-free is considered as a valid node. Figure 7 shows the normal/Gaussian distribution curve that is followed in this sampling. For this sampling, we need to provide 2 more parameters that are mean and standard deviation. These parameters define the spread of the nodes.

Figure 8 shows the nodes that are sampled using Gaussian sampling. As seen in this figure, Gaussian distribution captures the edges of obstacles really well.

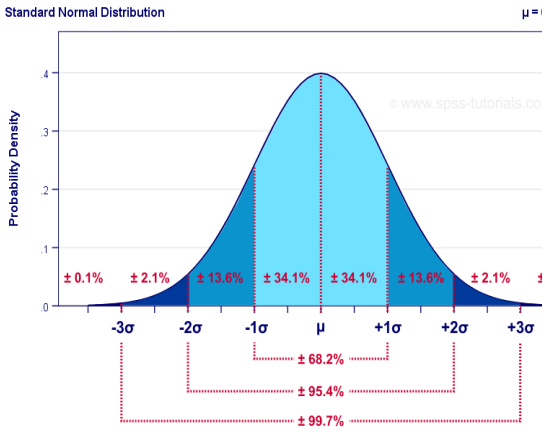


Fig. 7. Probability density in Gaussian distribution

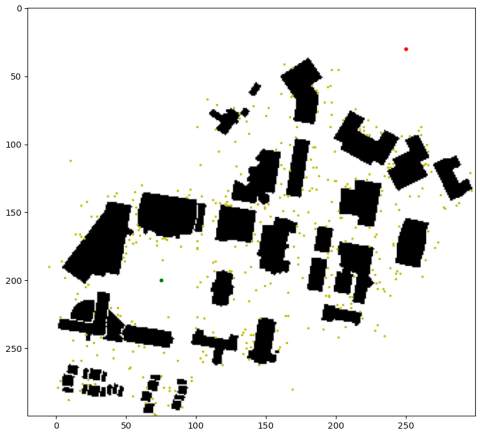

Fig. 8. Nodes that are sampled using Gaussian sampling

### D. Bridge Sampling

In bridge sampling , initially one point that is in collision is selected. Using Gaussian distribution , an another node is selected. If the second node is in collision, then mid point if both of these point is calculated. If the mid point is in collision, then all these points are dropped/ignored. If the mid point is in collision, then it is considered as a valid Node. Bridge sampling has a great feature of capturing the narrow regions in the given map. This can be observed in figure 9.

Once the modes are sampled , the expansion phase is initiated. In this phase, the nearby nodes are connected to each other. KD Trees method is used to find the nearby nodes.

### E. KD Tree

KD tree is a space partitioning data structure that is effective for organizing points in k-dimensional space. In simple terms, KD tree is a binary tree that has each node as k-dimensional point. There are different ways to construct KD trees. In this assignment python's scipy module is used to build the trees

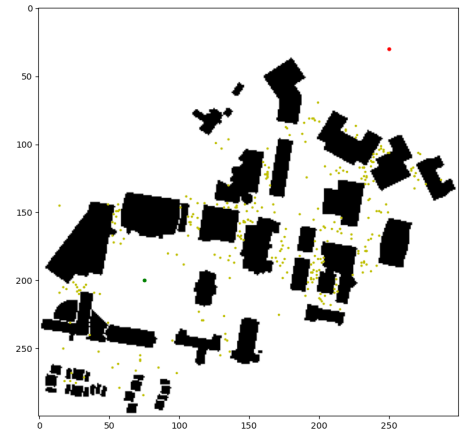

Fig. 9. Nodes that sampled using bridge sampling

and query for the organized pairs. Figure 10 shows a simple KD tree.

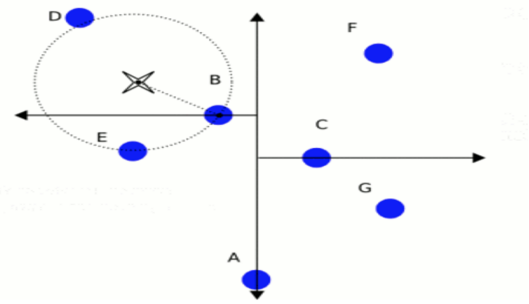


Fig. 10. A simple KD tree is shown here. For finding the closest point to a given point, start at the median in the tree , and then do DFS until the nearest neighbor is found

Once the k-neighbors for a given node is found, they connectivity between each pair is checked using Bresenham's algorithm.

### F. Bresenham's Algorithm

Bresenham's algorithm is a line drawing algorithm that is used extensively in computer graphics. For a given two points, this algorithm can approximately form a line that can be represented using pixels of a computer screen. This algorithm has a great usage with bitmap images. In our case , the map that is used in this assignment is a bitmap image that has obstacle and free spaces data at each pixel. An example of Bresenham's line is shown by figure 11.

Once the Bresenham's line is formed between two neighbors, each pixel between these points are checked for an obstacle that is represented by zero. If a pixel with a value of zero is found , then the connectivity between these points is not formed. Figures 12 to 15 show the connected neighbors in uniform , random , Gaussian and bridge sampling methods respectively.

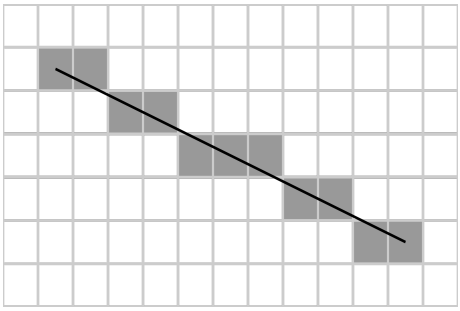

Fig. 11. For a given two points , Bresenham's algorithm can appr
fill the pixel to form a straight line

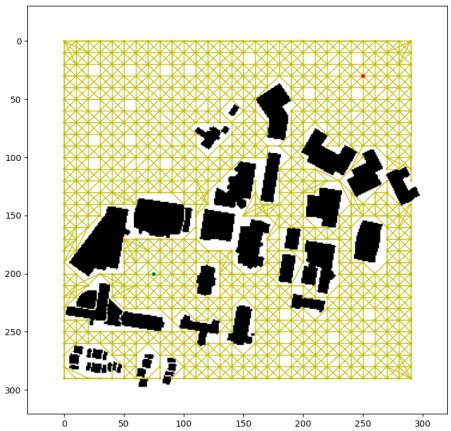


Fig. 12. K-Nearest neighbors are connected to each other in uniform

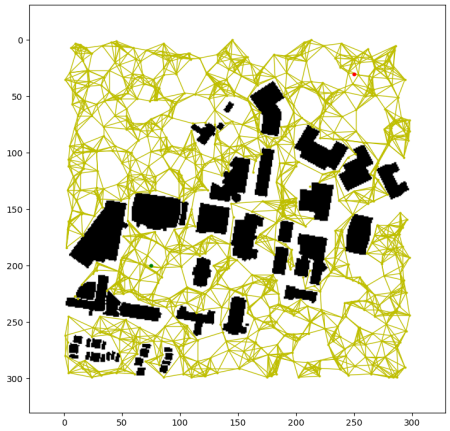


Fig. 13. K-Nearest neighbors are connected to each other in random sampling

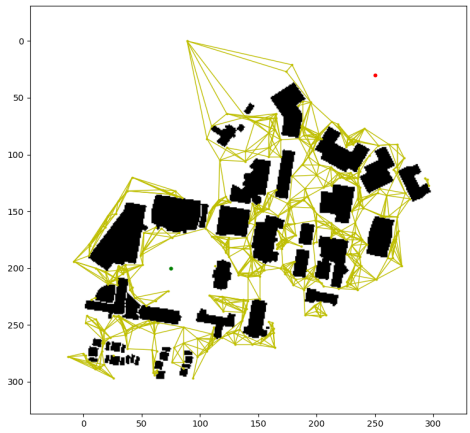


Fig. 14. K-Nearest neighbors are connected to each other in Gaussian sampling

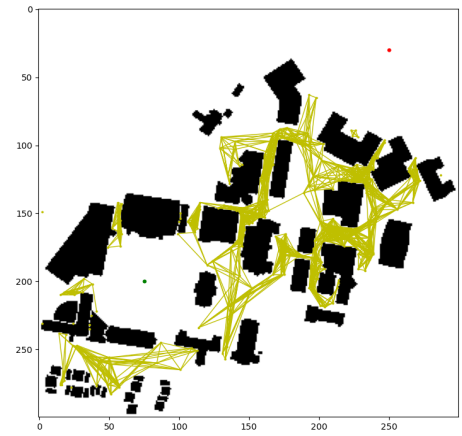


Fig. 15. K-Nearest neighbors are connected to each other in bridge sampling

Here the start and goal nodes are not connected to the nearest point yet. But the same method as explained is used to connect those to the nearest nodes. Once all the nearest nodes are connected to each other, the PRM algorithm is implemented using Dijkstra as the local planner. The Euclidean distance between each node is considered as the cost of the specific edge. Python's netwrokx module is used for this implementation. Eight nearest neighbors are connected to each other for uniform , random and Gaussian sampling whereas twenty nearest neighbors are used for bridge sampling.

Total number of sampled points are different for each sampling method. Sampling is done for different iteration cycles for each sampling method. Thousand iterations are used for uniform and random sampling. Two thousand iterations are used for Gaussian sampling where as Twenty thousand itera-

tions are used for bridge sampling. For uniform distrib Gaussian sampling , mean is set as 1 and standard dev set as 10, whereas in bridge sampling the mean is un but the standard deviation is set to 20. Figures fro 19 show the results produced by PRM with Dijkstra planner for different types of sampling methods.

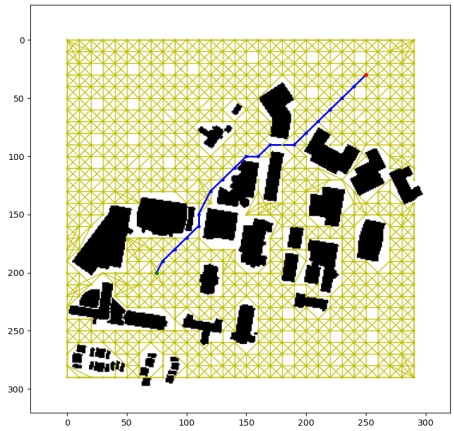

Fig. 16. Path generated by PRM with uniform sampling

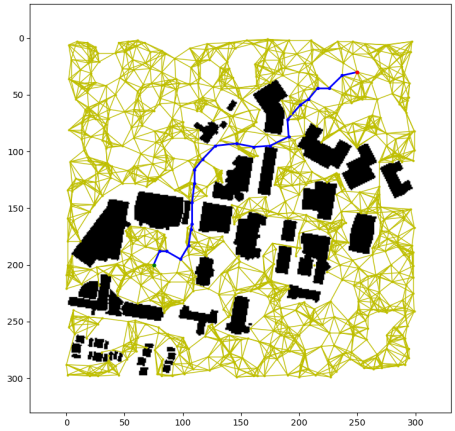

Fig. 17. Path generated by PRM with random sampling

Table 1 shows the comparison of results of the different sampling methods.

| Sampling Method | Nodes | Edges | Distance |
|---|---|---|---|
| Uniform | 754 | 2752 | 257.39 |
| Random | 850 | 3298 | 299.53 |
| Gaussian | 309 | 979 | 265.63 |
| Bridge | 327 | 2310 | 261.08 |

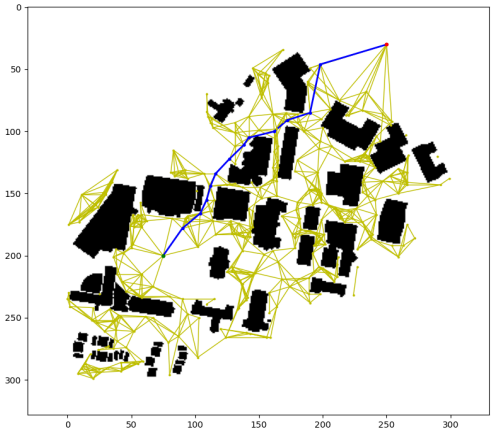

Fig. 18. Path generated by PRM with Gaussian sampling

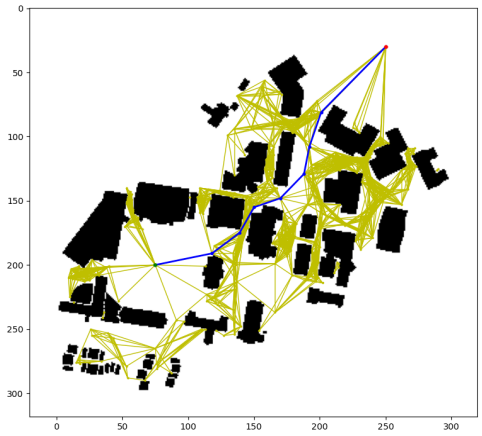

Fig. 19. Path generated by PRM with bridge sampling

## III. Implementation of RRT and RRT*

### A. RRT

For the implementation of RRT, nodes are generated randomly on the map. Once a node is generated, nearest node from already connected nodes is found using KD trees method that is described previously. For the first iteration, starting point is the nearest node.From this neighbor, the tree can be grown in the node direction using two methods. They are extend method and connect method. In the extend method, one step is taken towards the random sample and the collision is checked using the Besenham's algorithm. If there is no collision is detected, then this node is added to the samples.In connect method, the step is taken until the goal or an obstacle is reached. For this assignment, extend method is used. Figure 20 shows the difference between extend and connect methods. To check if the goal is reached, the whole area around the

goal within a certain distance is considered as goal region. If the extended node is in this region , then the RRT search is terminated.

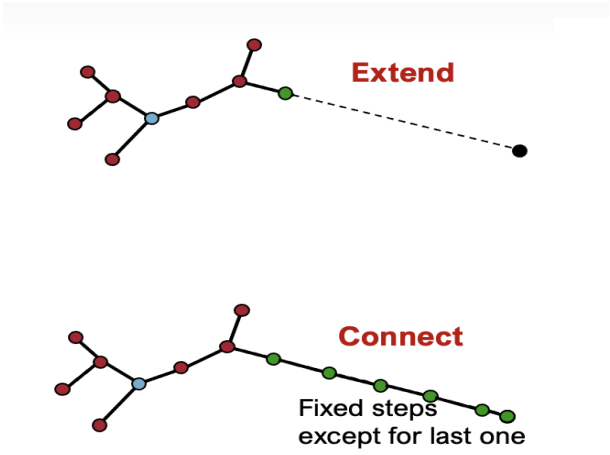


Fig. 20. RRT-Extend vs RRT-Connect. For this assignment , R method is used

After the goal region is reached, the parent of each extended node is traced back to form the path. The result of RRT is shown in Figure 21.

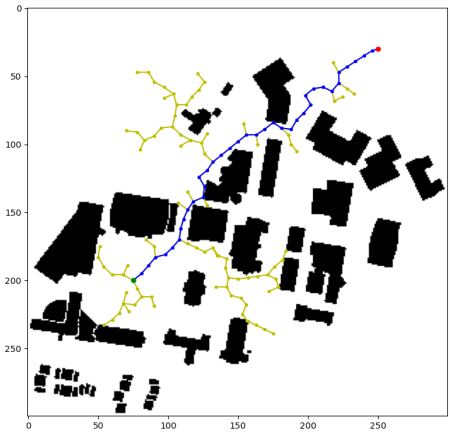

Fig. 21. Path generated by RRT

### *B. RRT**

RRT* is similar to RRT,but it can give more optimized path. There is mainly one extra step in RRT* implementation, that is rewiring. Once a new node is sampled, all the nearest nodes to it are rewired to reduce the cost of their paths. This step results in more refined path. However, for RRT* the terminating condition must be defined. usually it is the maximum number of sampling iterations. Figure 22 shows the path produced by RRT*. Table 2 shows the results comparison of RRT and RRT*.

| Algorithm | Nodes | Distance |
|---|---|---|
| RRT | 123 | 283.44 |
| RRT* | 2000 | 258.07 |

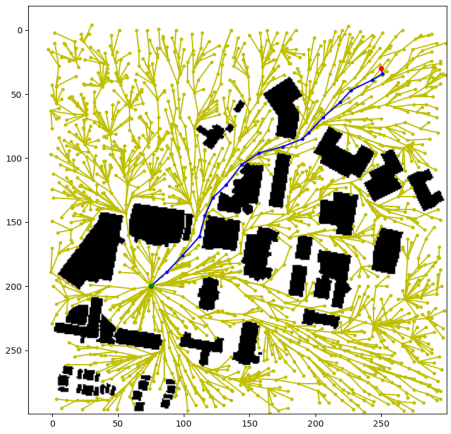

Fig. 22. Path generated by RRT

## IV. Conclusion

Both PRM and RRT algorithms have many advantages and disadvantages. In PRM , each of the four sampling methods gave different paths , and took different number of nodes/edges for finding the path. Uniform sampling performed better when compared to other three sampling methods. Since we are sampling the whole map uniformly, connecting the nearest nodes and finally the start and goal was easy. However, in this sampling we are sampling in spaces that are not really required.In random sampling , since we are placing nodes randomly , this may not sometime lead to an optimal path or even may not give a path. Main advantage of random sampling is there is no bias in sampling. Gaussian sampling is really good in forming the edges of the obstacles , whereas the bridge sampling can find the narrow paths really easily. But, these two sampling methods can not guarantee the solution. Further, we need to tune adhoc parameters such as standard deviation and number of nearest neighbors to make optimal solutions with these two methods.

When compared to RRT , RRT* requires an extra step called rewiring. This step can make the RRT* path more optimal compared to RRT path. Both RRT and RRT* are probabilistic complete , but if RRT* is performed for infinite iteration , it can guarantee the optimal path. In comparison RRT is faster than RRT* but path generated by RRT* is shorter. RRT is not asymptotically optimal , but RRT* is. Both RRT and RRT* are monotonically convergent.

In PRM , the connectivity is formed among all the nearest neighbors , but in RRT or RRT* only one connection is made. IN PRM , because of its best connectivity we can extend the connections to start and goal easily and then perform local planner to get the path. But in RRT , since only one connection

is made at once, sometimes connectivity to goal cannot be guaranteed. RRTs also produce notoriously bad paths, this is not surprising given path quality is not considered.There are many existing methods that can be used for smoothing the path. If there are bug traps , RRT and RRT* may not perform well. To solve this problem , we can use bidirectional RRTs.

Overall, the sampling-based planning has many pros and cons. This type of planning can provide fast and feasible solution, and most of the popularly used methods have few parameters to tune. This method is well known for working with practical problems, and easy to implement in high dimensions. However, quality of the path is not guaranteed by this method.Termination of the algorithm is not guaranteed if there is no solution. Finally , probabilistic completeness is a weak property. To over come these problems, we can smooth or optimize the path afterwards, and set an arbitrary time for termination(This is followed in this assignment). Asymptotically optimal is a desired feature for any sampling based planning algorithm, but standard PRM and RRT are not asymptotically optimal.