Standard Search Algorithms Report: Jash Mehta

Q-1 Advantages and disadvantages of 4 sampling methods:

Uniform sampling:
Points are taken uniformly and connected. If the same map is sampled multiple times uniformly, the same nodes and edges will be formed giving the same path cost. The disadvantage of this sampling is that if not enough points are sampled between the narrow spaces, no path will be found even if it is sampled multiple times.

Random sampling:
In this type of sampling method, we sample the points randomly within the free space. This random selection of points generates a new number of nodes and edges always. Due to this sometimes, random sampling may take a longer path compared to the optimal path. The connectivity of the sampling is also important as sometimes it may fail to connect to other parts of the map and failing to return a path altogether.

Gaussian sampling:
Samples are found close to the obstacle thus if there are a lot of obstacles, we will find nodes near the obstacles making finding the optimal path easy. But if the obstacles are far away, there won't be enough samples cast in the free space so the connectivity will be less resulting in no path to be found. As the points need to be sampled in the obstacle first and then find another point outside of the obstacle, we need to sample a lot of points to generate enough nodes on the graph. This makes the sampling run for longer during computation. 2000 samples instead of 1000 in uniform and random

Bridge sampling:
This is best used when there are a lot of narrow passages. In other types of sampling methods, there aren't enough nodes that are generated in the narrow passage thus making it difficult to find a path. In bridge sampling, we find most of the points in the narrow region. This is the slowest sampling method of all the above as it requires a maximum number of samples to find enough nodes on the graph. 20000 samples compared to 2000,1000 in the above methods.

Q-2 RRT vs RRT*
RRT stands for Rapidly Exploring Random Trees. In this algorithm, points are randomly generated and connected to the closest node available. This keeps happening until a node is formed very close to the goal or the total number of nodes to be generated are over.
This algorithm is fairly quick compared to many other algorithms but the computation time increases as the number of nodes to check in closest
On the other hand RRT* is an improvement over RRT as it checks the neighboring nodes when a new node is found. If the relative cost to traverse from the neighboring node is lower, the node with the lower cost replaces the original nearest node. This is called rewiring. This creates fan shaped twigs at the end of the branch. Furthermore, if a path has already been found and new nodes are added, we may find a better path as the tree keeps getting rewired.

Q-3 PRM vs RRT
In RRT every time a new point is generated it has to create a new graph corresponding to the new point. If the graph is very large, generating a new graph to find a single path is inefficient. So in such cases, instead of generating a fresh graph, PRM builds a single roadmap that covers most of the free space. Then a shortest path planning algorithm is used to connect all the possible points on the roadmap. A disadvantage of PRM is that it cannot take into account the changes in the environment whereas in RRT it can.

Probabilistic Roadmap

A full explanation of every step of the code is given in the comments of the Code itself.

Functions working:

Query phase:

Check collision-
Find slope between 2 points and increment along the line to see if it passes through an obstacle

Distance-
Calculates the euclidean distance between 2 points

Uniform sampling-
Equally distribute total samples along rows and cols of the graph. Sample points that don't lie within the obstacle

Random sampling-
Select random integer points that lie in the free space and append them to final sampled points

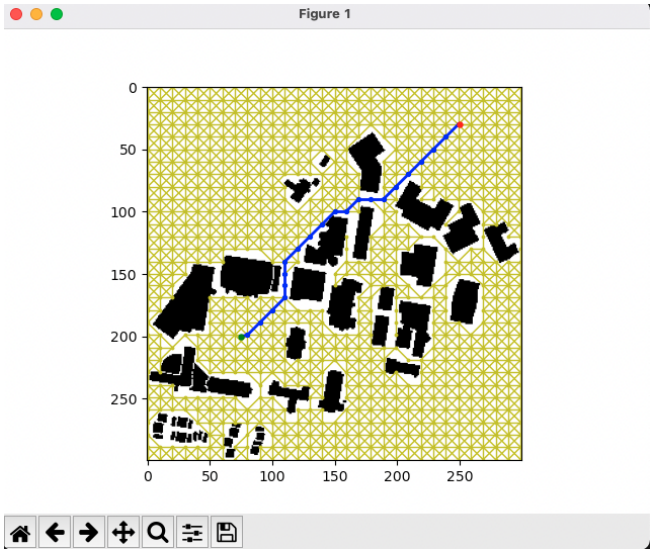
Gaussian sampling-
Sample random point in obstacle. Find another random point that lies in the free space within the gaussian distribution of the first point and append the point in the free space

Bridge sampling-
Sample random point in obstacle. Find another random point that lies in an obstacle within the gaussian distribution of the first point and find the midpoint of the 2 points. If midpoint lies in free space, append midpoint to samples

Find k nearest neighbors or every point and store pairs. The result is the index of the points in the main sample list.
Iterate over all points in pairs list to check for collision between k-nearest neighbors. Calculate the distance of the pairs that don't have collision and append to pairs = [ ]

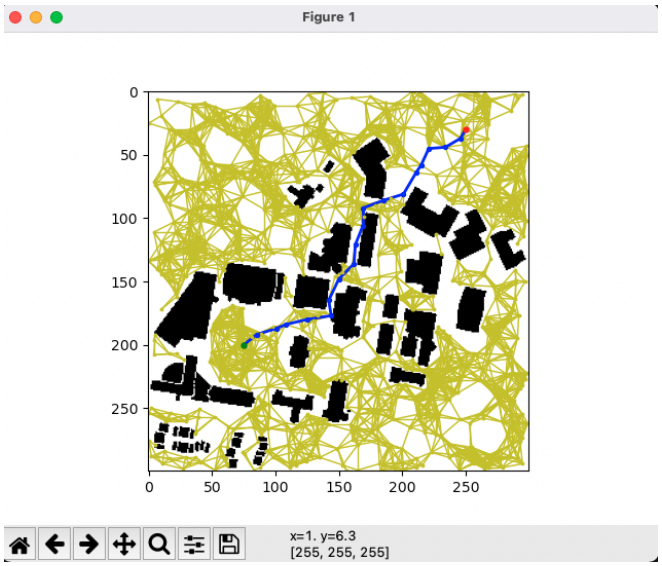
Query phase:
Take start point and iterate over all sampled points. Check for collision between start and all points and similarly the distance. Finally, append to start=[ ]
Similarly, calculate for goal and append to goal=[ ]

Results:



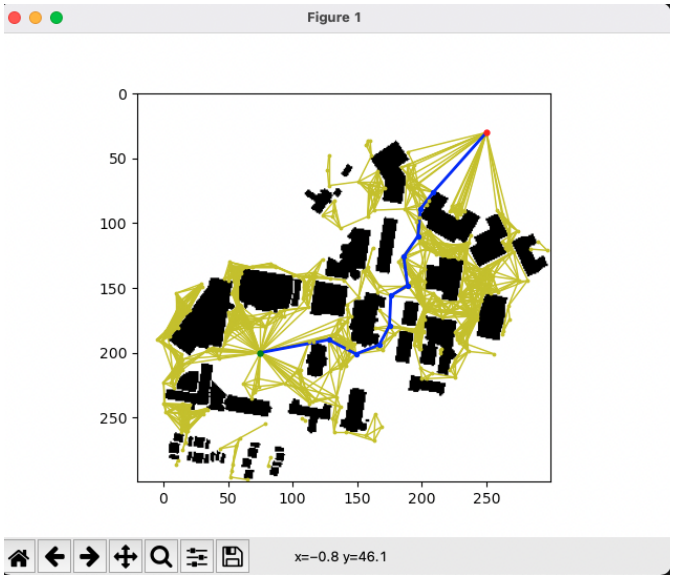

```
The constructed graph has 810 nodes and 2659 edges
The path length is 262.09
```

Uniform sampling always has the same amount of path length every time as it samples the same points.



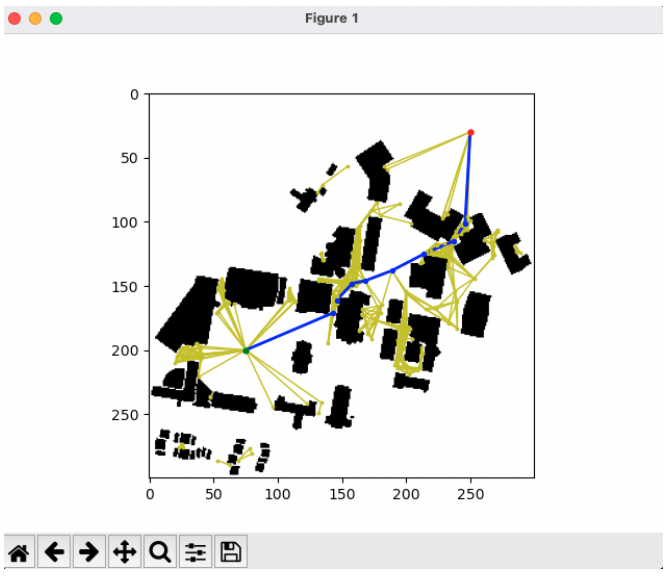
```
The constructed graph has 838 nodes and 4690 edges
The path length is 277.09
```

Random sampling takes points randomly. If the connectivity between nodes is set to low there might not be a final path found.




```
The constructed graph has 336 nodes and 1575 edges
The path length is 292.98
```

Gaussian sampling samples points close to the obstacle boundary. As it only finds points close to the obstacle, the total number of nodes and edges are fewer.




```
The constructed graph has 179 nodes and 774 edges
The path length is 275.74
```

Bridge sampling samples points in the narrow passage as seen from the above image. The number of total samples are huge but the final nodes and edges formed are generally very less.

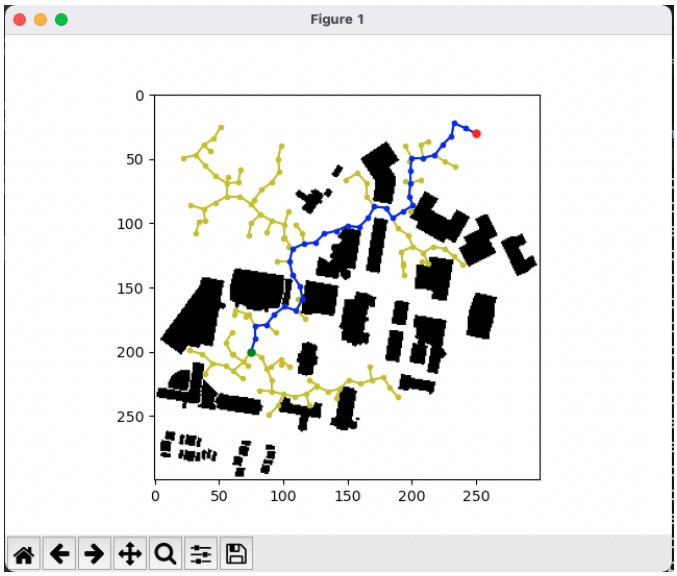
RRT and RRT*

RRT steps

1. Generate random point outside of obstacle
2. Get nearest node to point
3. Get new node in direction of new point
4. Check collision between new node and nearest node
5. Add new node to vertices
6. Update cost of new node
7. Update parent of new node as nearest node
8. If close to goal, new node=goal

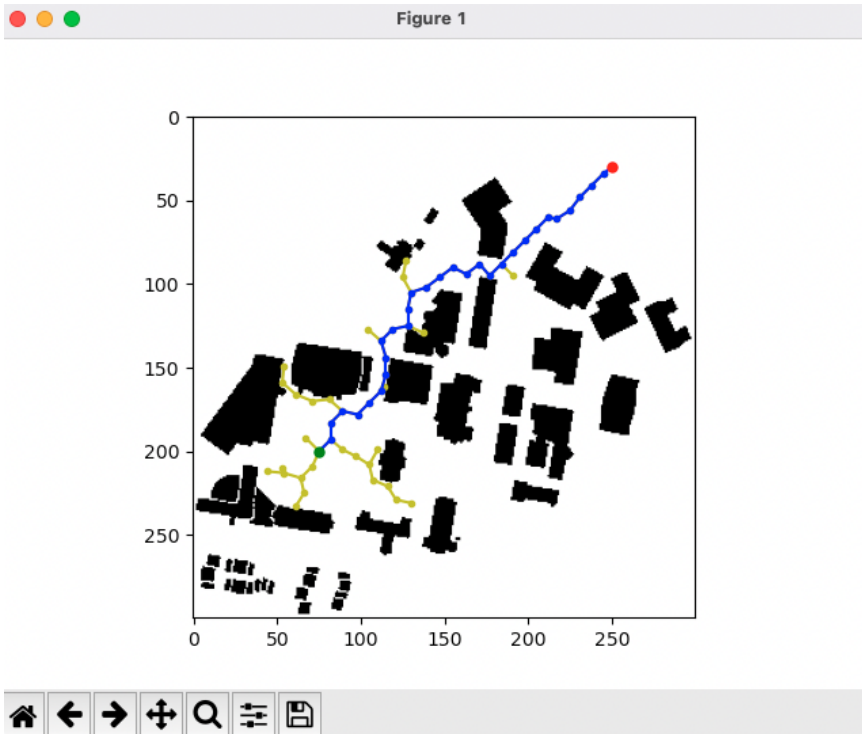
A full explanation of every step of the code is given in the comments of the Code itself.

Results:




```
It took 148 nodes to find the current path
The path length is 318.00
```

The above image shows the RRT tree when the goal bias =0. The tree randomly explores and finds the shortest path. As soon a path to the goal is found, the algorithm terminates.




```
It took 56 nodes to find the current path
The path length is 272.00
```

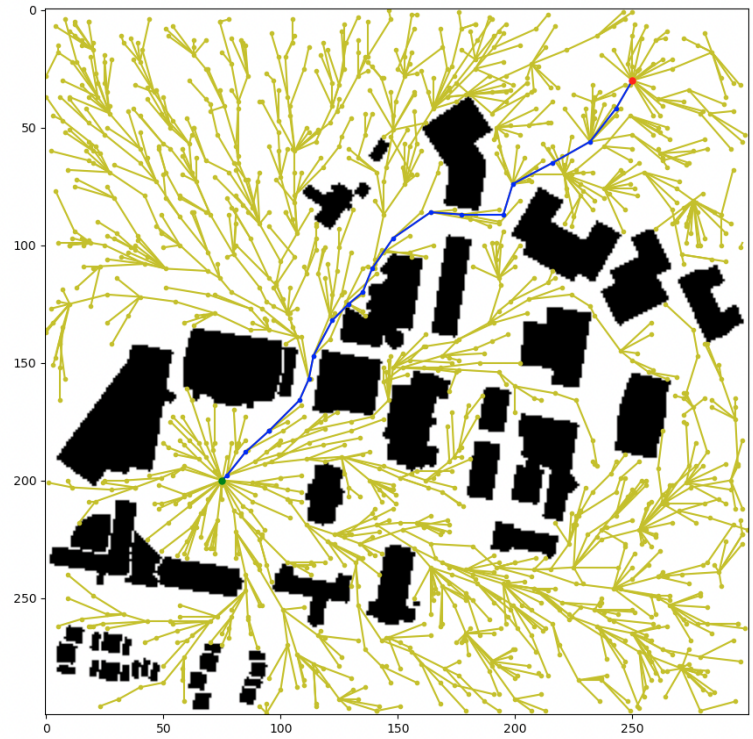
The above RRT is generated when the goal bias is set to 30 i.e. 30% chance of picking goal as a new point. If the new point is set as goal the exploration happens more towards the goal. The results clearly show the bias working as it only took 56 nodes to find a path to the goal compared to 148 nodes. The total path length is also drastically reduced.

RRT* steps

1. Generate random point outside of obstacle
2. Get nearest node to point
3. Get new node in direction of new point
4. Check collision between new node and nearest node
5. Get neighbors of new node
6. Rewire all neighbors
7. Update cost of new node
8. Update parent of new node as neighbor node with lowest cost
9. If close to goal, new node=goal

A full explanation of every step of the code is given in the comments of the Code itself.

Results:




```
It took 1513 nodes to find the current path
The path length is 258.00
```

We can see that the RRT* performs better than RRT as the path length is significantly shorter. This is due to the rewiring process and finding new paths by still adding new nodes even when the original path is found.

The Tree structures of RRT and RRT* differ mainly because of the rewiring process. The rewiring creates fan-shaped ends on the branches of the tree.

References:

1. https://theclassytim.medium.com/robotic-path-planning-prm-prm-b4c64b1f5acb
2. https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378
3. https://www.youtube.com/watch?v=Ob3BIJkQJEw&t=302s
4. http://www.cs.columbia.edu/~allen/F15/NOTES/Probabilisticpath.pdf