Motion Planning Assignment
Standard Search Algorithm
-Sailesh Rajagopalan

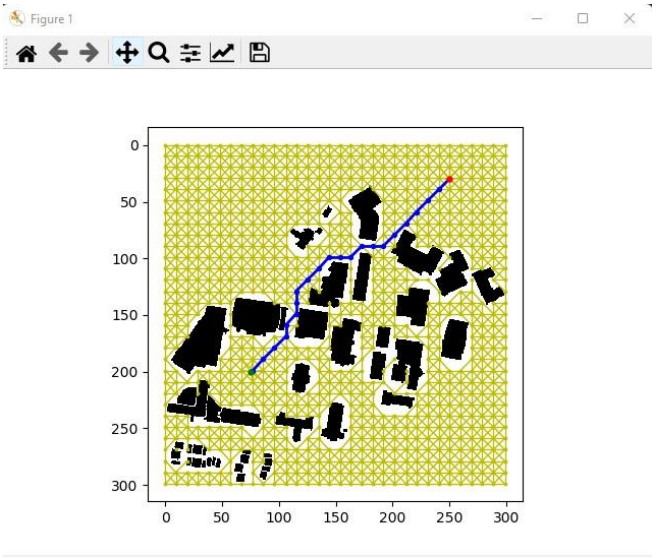
PRM:

The methodology basis of PRM works on a step basis, first the learning of the map which can be done as a roadmap construction, the finding of the path is the second step which involves solving the Query in phase, Computing the short path when there are no obstacles present and performing collision free. For a path between start and goal, to the graph a local planner and then the graph is searched to form a path.

$V \leftarrow \emptyset$; $E \leftarrow \emptyset$;
loop
  $c \leftarrow$ a (random) configuration in $\mathcal{C}_{\text{free}}$
  $V \leftarrow V \cup \{c\}$
  $N_c \leftarrow$ a set of nodes chosen from V
  for all $c' \in N_c$, in order of increasing distance from c **do**
    if c' and c are not connected in G **then**
      if the local planner finds a path between c' and c **then**
        add the edge $c'c$ to E

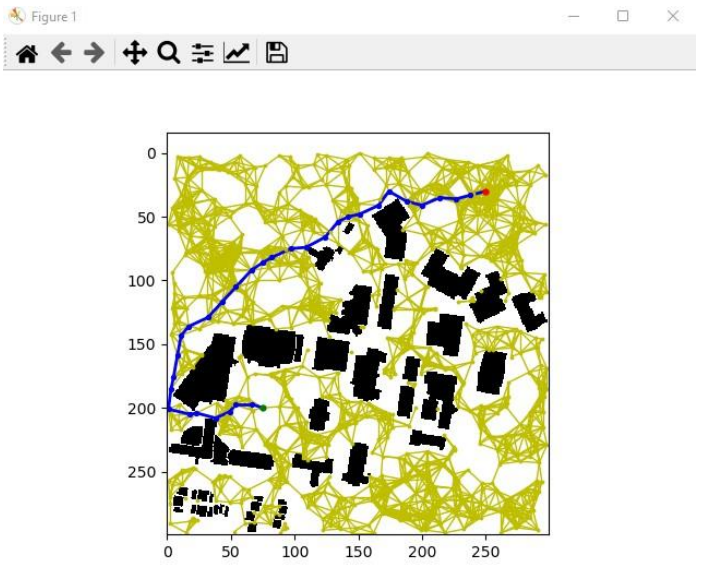
Uniform Sampling:

In uniform sampling, the aspect is to check for collision for every sample, the points to be sampled are chosen from the whole to be created as uniform.

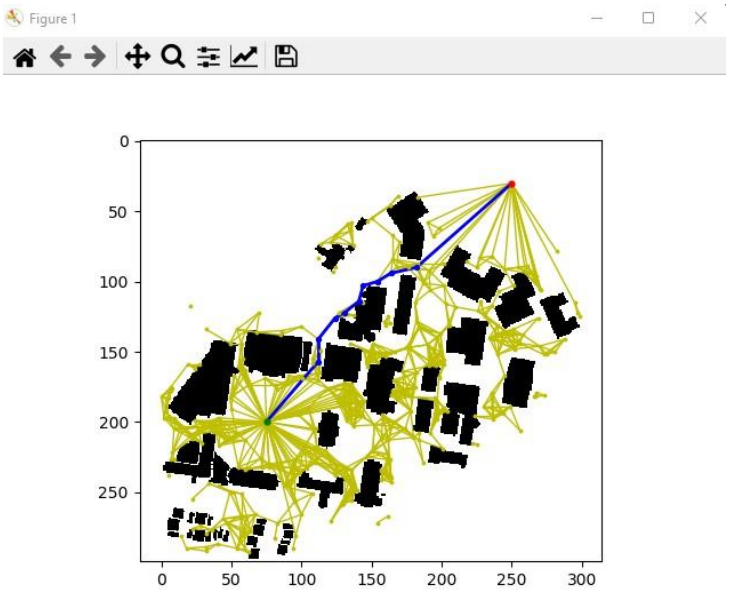


Random Sampling:

With random sampling, samples are chosen at random and we do check if there is an obstacle or not if there isn't any we can proceed, and append them values. We define random points every time which would also create path for the points obtained

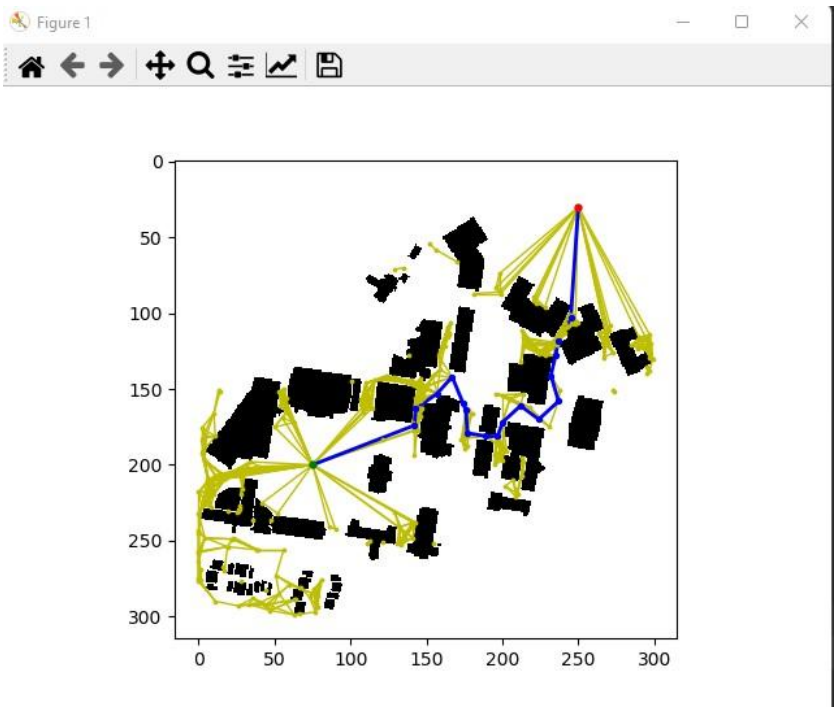


Gaussian Sampling:
Gaussian sampling works on the phenomena by adding more samples near the obstacles, we find the other point by normal distribution



Bridge Sampling:

It works on the concept of getting free space coverage, by taking random samples, the ideology is similar to gaussian sampling

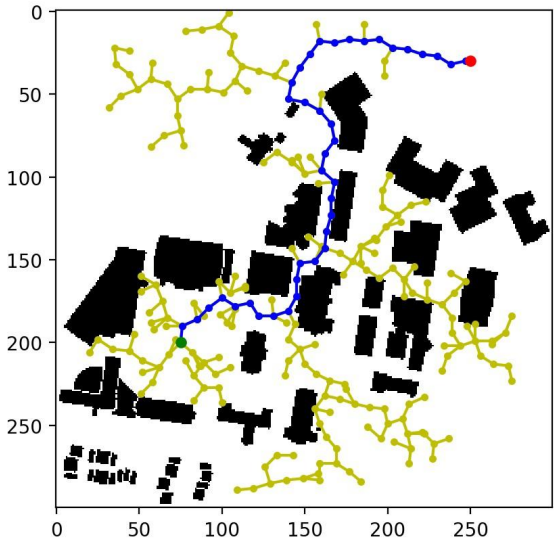



```
"C:\Users\Sailesh Rajagopalan\anaconda3.7\python.exe" "(

Bad key "text.kerning_factor" on line 4 in
C:\Users\Sailesh Rajagopalan\anaconda3.7\lib\site-packa(
You probably need to get an updated matplotlibrc file fi
https://github.com/matplotlib/matplotlib/blob/v3.1.3/ma
or from the matplotlib source distribution
The constructed graph has 838 nodes and 2735 edges
The path length is 264.06
The constructed graph has 832 nodes and 3857 edges
The path length is 423.32
The constructed graph has 341 nodes and 1097 edges
The path length is 255.43
The constructed graph has 288 nodes and 1315 edges
The path length is 363.84
```

RRT:

The ideology behind RRT is it is a sampling based algorithm which follows step by procedure, the exploration with empty area and filling the ith nodes, By taking a star point the distance of the sample maximum will be defined and then finding the nearest node,. For each sample we connect it to the nearest node of the tree and add if it is a success or fail and discard the sample.



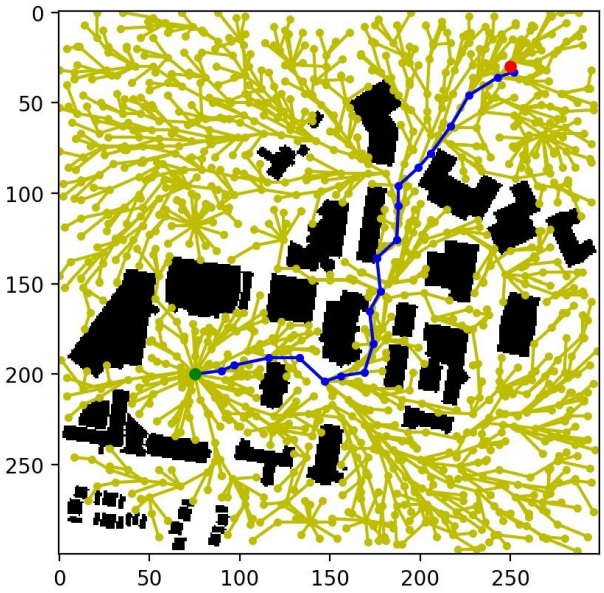
Algorithm 1: RRT

Data: Given initial state X_{init}
Result: An RRT, T, with K vertices is constructed

```
1  begin
2      for K = 1 to K do
3          x_rand ← RANDOM_STATE();
4          x_near ← NEAREST_NEIGHBOUR(X_rand,T);
5          u ← SELECT_INPUT(x_rand,x_near);
6          x_new ← NEW_STATE(x_near,u,Δt);
7          T.add_vertex(x_new);
8          T.add_edge(x_near,x_new,u);
9      end
10     Return T;
11 end
```

RRT*

RRT* is a variant of RRT, We can also infer the trees produced are different in both, as the concept in this is its ability to find the shortest path to reach the goal.




```
"C:\Users\Sailesh Rajagopalan\anaconda3.7\python.exe" "

Bad key "text.kerning_factor" on line 4 in
C:\Users\Sailesh Rajagopalan\anaconda3.7\lib\site-packa
You probably need to get an updated matplotlibrc file f
https://github.com/matplotlib/matplotlib/blob/v3.1.3/ma
or from the matplotlib source distribution
It took 151 nodes to find the current path
The path length is 353.69
It took 1444 nodes to find the current path
The path length is 324.36
```

We can hence infer that RRT* enables in optimal path to be found shortest path.

Algorithm 2: RRT*

Data: Given initial state x_{init}
Result: An RRT*, T, with K vertices is constructed

begin
  for $i = 1,\ldots,n$ **do**
    $x_{rand} \leftarrow$ **SampleFree**$_i$;
    $x_{nearest} \leftarrow$ **Nearest**(G=(V,E),x_{rand});
    $x_{new} \leftarrow$ **Steer**($x_{nearest}$,x_{rand});
    if ***ObstacleFree***($x_{nearest}$,x_{new}) **then**
      $X_{near} \leftarrow$ **Near**(G=(V,E),x_{new},**min**{γ_{RRT^*}(**log**(**card**(V))/**card**(V))$^{1/d}$,η}];
      V $\leftarrow$ V$\cup${x_{new}};
      $x_{min} \leftarrow x_{nearest}$;
      $c_{min} \leftarrow$ **Cost**($x_{nearest}$)+c(**Line**($x_{nearest}$,x_{new}));
      foreach $x_{near} \in X_{near}$ **do**
        if ***CollisionFree***(x_{near},x_{new})$\wedge$***Cost***(x_{near})+c(***Line***(x_{near},x_{new}))<c_{min} **then**
          $x_{min} \leftarrow x_{near}$;
          $c_{min} \leftarrow$ **Cost**(x_{near})+c(**Line**(x_{near},x_{new}));
        end
      end
      E $\leftarrow$ E $\cup$ {(x_{min},x_{new})};
      foreach $x_{near} \in X_{near}$ **do**
        if ***CollisionFree***(x_{new},x_{near})$\wedge$***Cost***(x_{new})+c(***Line***(x_{new},x_{near})) < ***Cost***(x_{near}) **then**
          $x_{parent} \leftarrow$ **Parent**(x_{near});
          E $\leftarrow$ (E$\cup${(x_{new},x_{near})}
        end
      end
    end
  end
  Return G = (V,E);
end

Q1. For PRM, what are the advantages and disadvantages of the four sampling methods in comparison to each other?

Uniform Sampling:

Advantages	Disadvantages
Easy implementation (Low complexity)	Computationally inefficient to find paths as there are a large number of nodes.
Always finds a path	Samples large open areas which are not required.

Random Sampling:

Advantages	Disadvantages
Low Complexity	Does not always find a path not ensuring connectivity

Gaussian Sampling:

Advantages	Disadvantages
Low Complexity	Does not always find a path not ensuring connectivity

Bridge Sampling

Advantages	Disadvantages
Ensures better pathfinding connectivity, does not explore open spaces lesser number of nodes it will explore	Threshold set to explore nearest neighbour is dependent on it, causing failure in finding path

Q2. For RRT, what is the main difference between RRT and RRT*? What change does it make in terms of efficiency of the algorithms and optimality of the search result?

The difference between RRT and RRT* is the new sampled vertex connects only the vertex which has the shortest path to the root vertex of the tree. RRT* records the distance each vertex has travelled with respect to the parent vertex which is known as cost of the vertex. With respect to optimality, RRT* is efficient as it is considered to be asymptotically optimal wherein RRT is not, with respect to efficient RRT has an ability with respect to time taken to find a path since RRT* has to rewire this is a shortcoming.

Q3. Comparing PRM and RRT, What are the advantages and disadvantages ?

PRM:
PRM is probabilistically complete, it will produce a solution alongside a path when it exists, PRM uses a local planner which is fast and saves memory as it does not save the paths but rather works by recalculating the path, The disadvantages or PRM include, it struggles to get a uniform covering of the free configuration space, building graph over state spcar not generating the path, it can also increase the nodes in a region if we increase the iterations.

RRT:
RRT provides uniform covering as it is biassed towards the unexplored state space, alongside fast convergence. The tree is always connected when using RRT. The disadvantages of RRT include, the path generates is random and it may not be the optimal path for the scenario,