RBE 550 Motion Planning

Assignment 3
Probabilistic Roadmap

Aadesh Varude
avarude@wpi.edu

Q1 . Sampling methods

- **Uniform Sampling**
  In Uniform sampling, we basically divide the grid into an equal number of rows and columns for the desired number of sampling points.

```
    j=0
    i=0
    for i in range(0,self.size_row,int(self.size_row//np.sqrt(n_pts))):
        for j in range(0,self.size_col,int(self.size_col//np.sqrt(n_pts))):
                if self.map_array[i][j]==1:
                    self.samples.append((i, j))
```

- **Random Sampling**
  In random sampling, we pick up random integer points from the grid and pick only those who are in the free space.

```
    for i in range(n_pts):
        row=random.randint(0, self.size_row-1)
        col=random.randint(0, self.size_col-1)
        if self.map_array[row][col]==1 and (row,col) not in self.samples:
            self.samples.append((row, col))
```

- **Gaussian Sampling**
  In Gaussian sampling, we take a random point and then using that point as the centre for gaussian we randomly pick another point in that gaussian distribution. If both of the points are in free space or collision space we reject them and pick only those pairs where one is in obstacle and the other in free space, we finally add the point in the free space to the samples list.

```
        for i in range(n_pts):
            row1=random.randint(0, self.size_row-1)
            col1=random.randint(0, self.size_col-1)
            #pick point using gaussian
            row2=np.random.normal(row1,10,1)
            col2=np.random.normal(col1,10,1)
            row2=round(abs(row2[0]))
            col2=round(abs(col2[0]))

            if(row2<=self.size_row-1 and col2<=self.size_col-1):

                if (self.map_array[row1][col1]==1 and self.map_array[row2][col2]==1) or
                    (self.map_array[row1][col1]==0 and self.map_array[row2][col2]==0):
                    continue
                # Adding the one with collision free
                elif(self.map_array[row1][col1]==1):
                    self.samples.append((row1, col1))
                else:
                    self.samples.append((row2, col2))
```

- **Bridge Sampling**
  In this method we pick up random points in the obstacle or collision space, and then we pick up another point that is gaussian centric to the previous point. Then we take the midpoint of these points and add it to the sample only if it is in the free space.

```

        for i in range(n_pts):
            row1=random.randint(0, self.size_row-1)
            col1=random.randint(0, self.size_col-1)
            if self.map_array[row1][col1]==0:
                # print("in the collision stuff")
                #pick point using gaussian
                row2=np.random.normal(row1,20,1)
                col2=np.random.normal(col1,20,1)
                row2=round(abs(row2[0]))
                col2=round(abs(col2[0]))
                if(row2<=self.size_row-1 and col2<=self.size_col-1):
                    if self.map_array[row2][col2]==0:
```

```
                    m_row,m_col=self.get_midpoint(row1,col1,row2,col2)
                    m_row=round(m_row)
                    m_col=round(m_col)
                    if(m_row<=self.size_row-1 and m_col<=self.size_col-1):
                        if self.map_array[m_row][m_col]==1:
                            self.samples.append((m_row,m_col))
```

2 . Learning Phase

- First we create a Kd tree of the samples.
- Then we iterate in the samples and all the neighbours of that specific point and add the point pairs if there is no collision, or the point pair is not already in the pairs list.

```
    pairs = []
       self.kdtree=KDTree(self.samples)

       for point_idx in range(len(self.samples)):
           _,idx=self.kdtree.query(self.samples[point_idx],10)
           for i in idx:
               if self.samples[point_idx]==self.samples[i]:
                   continue
               if self.check_collision(self.samples[point_idx],self.samples[i]):
                   continue
               if (point_idx,i,self.dis(self.samples[point_idx], self.samples[i])) not in pairs and
                    (i,point_idx,self.dis(self.samples[point_idx], self.samples[i])) not in pairs :
                   pairs.append((point_idx,i,self.dis(self.samples[point_idx], self.samples[i])))
```

3. Query Phase

- We gather 20 neighbouring points of the start and goal.
- Check the validity of the point pairs and add them to respective pair lists.

```
self.path = []

      # Temporarily add start and goal to the graph
      self.samples.append(start)
      self.samples.append(goal)
      # start and goal id will be 'start' and 'goal' instead of some integer
      self.graph.add_nodes_from(['start', 'goal'])

      start_pairs = []
      goal_pairs = []
      _,idx=self.kdtree.query(start,20)
      for i in idx:
          if self.check_collision(start,self.samples[i]):
              continue
          start_pairs.append(('start',i,self.dis(start, self.samples[i])))

      _,idx=self.kdtree.query(goal,20)
      for i in idx:
          if self.check_collision(goal,self.samples[i]):
              continue
          goal_pairs.append(('goal',i,self.dis(goal, self.samples[i])))
```

For PRM, the Advantages of the four sampling methods compared to each other.

- **Uniform Sampling**
  - Uniform item sampling covers the entire grid map for sampling.
  - Straightforward and easy implementation.
- **Random Sampling**
  - Easy to implement.
  - Can sample the points in the area that might be unsampled by any standard approach.
  - Can be implemented on various dimensions easily.
- **Gaussian Sampling**
  - Requires fewer samples to reach the goal.
  - Useful for sampling around the obstacles.
- **Bridge Sampling**

- Performs better sampling for the narrow paths between the obstacles.
- More efficient number of samples is generated (less unnecessary sampling.)

For PRM, the Disadvantages of the four sampling methods compared to each other.

- **Uniform Sampling**
  - Super expensive in computing, as the number of dimensions or grid size, increases the computation for the graph increases too.
  - Samples unnecessary areas in the grid map.
- **Random Sampling**
  - Random sampling might not give enough samples in the region that is desired in some cases.
  - It might get caught up in clustering in some cases, also it cannot provide a bias towards the goal as well.
- **Gaussian Sampling**
  - Requires to have accurate variance and tuning, otherwise, clustering of points could be observed.
  - In some cases it might cause oversampling.
- **Bridge Sampling**
  - Gets stuck in the corners of the obstacles where gaussian sampling performs better.
  - Difficult to tune, if not tuned properly might not find the path even if it exists.

Results

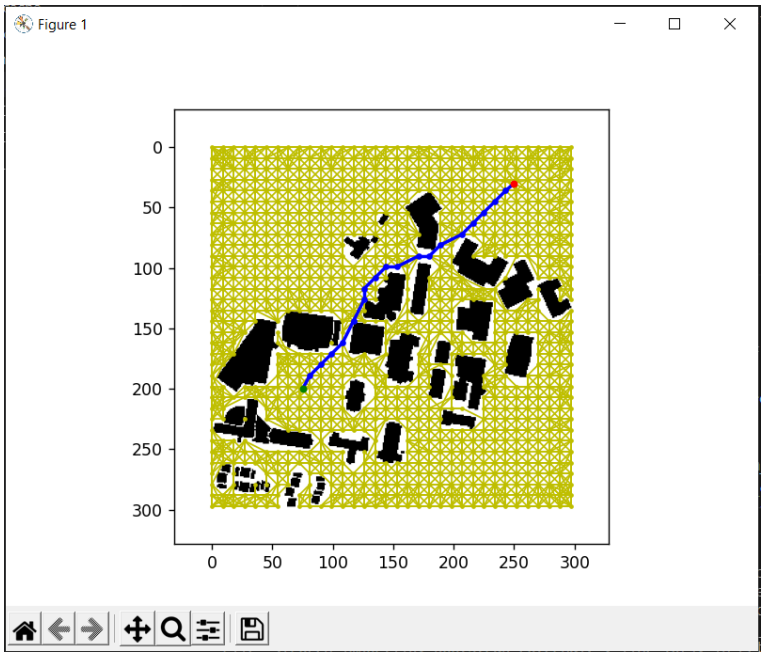

Figure 1: Image of the path after Uniform Sampling.

Explanation: This is uniform sampling where the sampling grid is uniformly spread over the entire grid. Then the start point connects the nodes in its nearest neighbours of 20 (tuning parameter) and so does the goal and then the path is matched by using in-built Dijkstra.

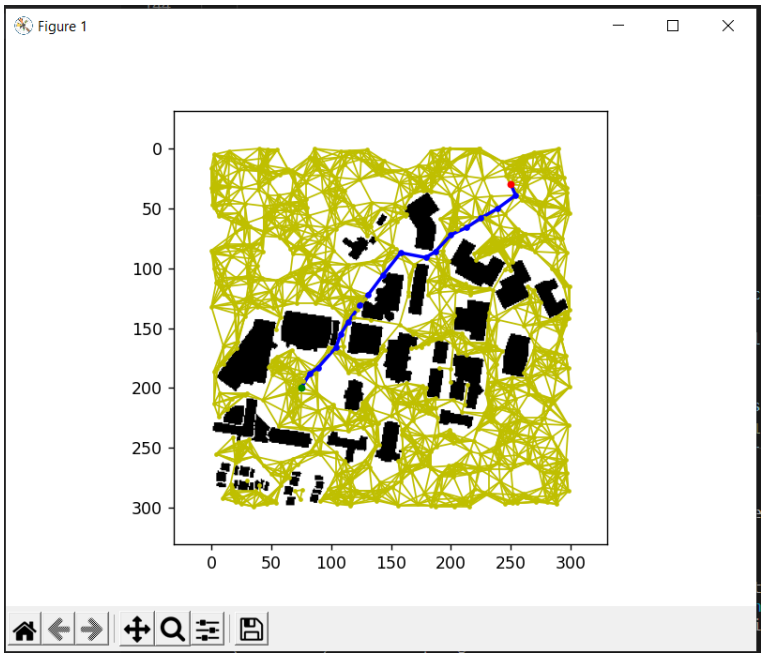

Figure 2: Image of the path after Random Sampling.

Explanation: This is random sampling where the sampling grid is randomly sampled where there is no collision and the nodes are connected checking the collision as explained previously in the pseudo-code. Then the start point connects the nodes in its nearest neighbours of 20 (tuning parameter) and so does the goal and then the path is matched by using in-built Dijkstra.

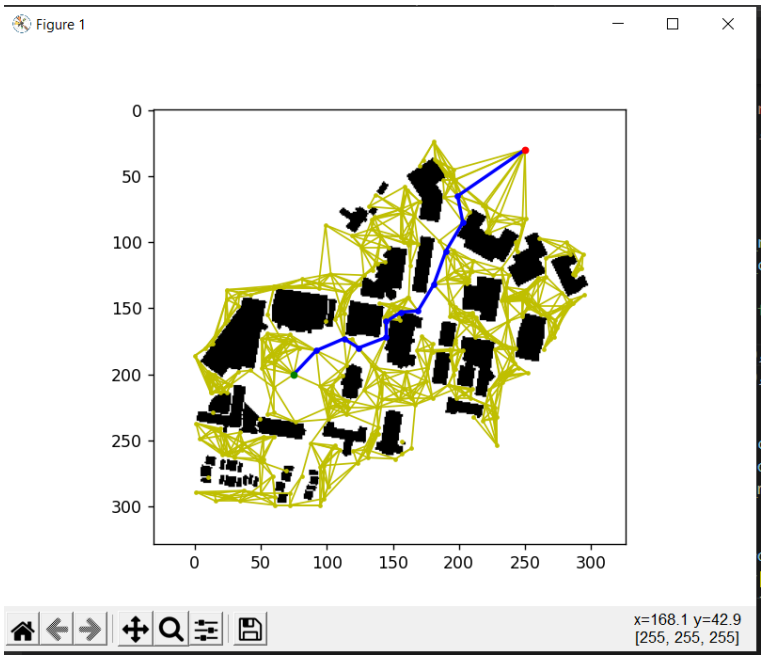

Figure 3: Image of the path after Gaussian Sampling.

Explanation: This is gaussian sampling where we take a random point and then using that point as the centre for gaussian we randomly pick another point in that gaussian distribution. If both of the points are in free space or collision space we reject them and pick only those pairs where one is in obstacle and the other in free space, we finally add the point in the free space to the samples list. Then the nodes are connected to each other as explained in the learning phase above. Then the start point connects the nodes in its nearest neighbours of 20 (tuning parameter) and so does the goal and then the path is matched by using in-built Dijkstra. One can Observe that the number of samples are reduced as compared to uniform and random sampling.

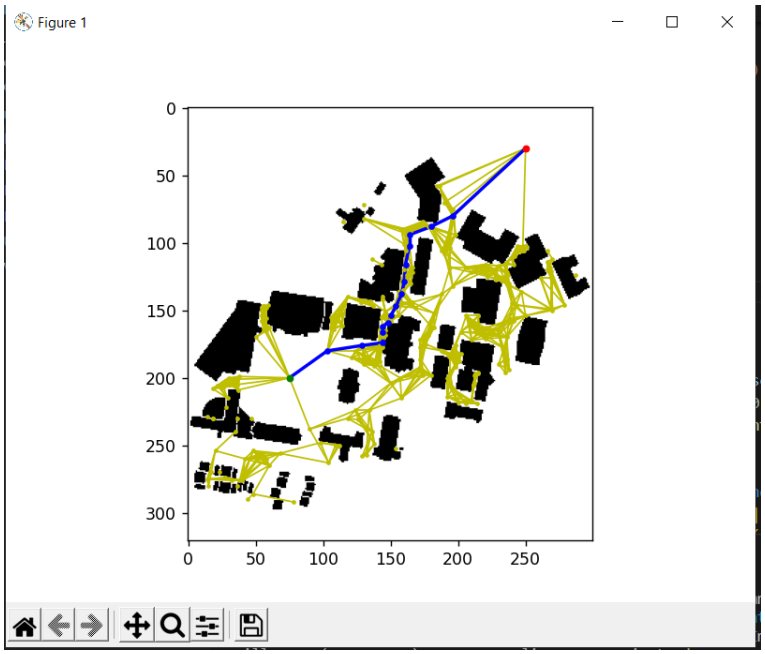

Figure 4: Image of the path after Bridge Sampling.

Explanation: In this method of bridge sampling, we pick up random points in the obstacle or collision space, and then we pick up another the point that is gaussian centric to the previous point. Then we take the midpoint of these points and add it to the sample only if it is in the free space Here we tuned the variance of the points picked such that we could encounter the narrow tunnels of the path between the obstacles. Later for path find and start and end goal connection similar approach is used.

Note The algorithm for the bridge sampling does not always provide the optimal path, as it is defined with random samples.

References

1. https://github.com/dawnjeanh/motionplanning
2. https://webspace.science.uu.nl/ gerae101/pdf/compare.pdf
3. Class lecture slides