**Homework 3 – Question 2**  
Adi Album 316563956 and Tomer Epshtein 313200750

Exercise 3.2 (A):

We implemented a variant of the RRT algorithm presented in class.

*num\_landmark* times we:  
Randomly sample a point and find the nearest point sampled so far .  
Next, given , we try take a step in the direction of starting from with size .  
I.e., .  
We will do this with in decreasing order . We’ll stop decreasing whenever is contained in the free space and so is the line connecting and .

Every 500 landmarks are added, we try and see if a path exists to our target point.

The main difference between our implementation and the RRT pseudocode presented in class is the method of decreasing s:

Given that with the path exists, that’s standard RRT. Given that with the path doesn’t exist, That means that in between our current point and the current we have an obstacle (either on the path in between or is inside one)  
As seen in class, in motion path planning tasks, with minimizing distance the procedure’s goal, we have the motivation of selecting landmarks that are as close as possible to obstacles.

The motivation is presented in the following figure:

A picture containing diagram

Description automatically generated

To see this method is effective, we viewed the histogram of different s running on the given maze *rod\_maze\_difficult.json*:

Text

Description automatically generated with low confidence

The results:

We have that for of the sampled points was chosen to be . This implies that most of the points we selected via the RRT method such that they are (relatively) close to obstacles.

An advantage of our RRT over the basic PRM implemented in the previous question, is that RRT will stop its run (approximately) when it finds a valid path from source to target. This improves RRT’s runtime on simple scenes.

When comparing our PRM implementation and our RRT implementation, on our easy and medium scenes that we created the differences in the metrics we defined in the previous section are negligible. The differences show on the more difficult scenes:

A picture containing shape

Description automatically generated

As we showed in the previous section, our PRM implementation took an average of 2,350 landmarks to solve this scene (3 consecutive times). Our RRT implementation is significantly faster with an average of ~300 landmarks in attempting to solve the scene (3 consecutive times).

RRTs performance is significantly better than PRM’s performance as we move towards even more complex scenes. For example:

Text

Description automatically generated with medium confidence

With our PRM taking >76,000 landmarks to solve this scene, our RRT solves this scene using ~13,000 landmarks.