**Homework 3 – Question 1**  
Adi Album 316563956 and Tomer Epshtein 313200750

Exercise 3.1 (A):

The aspect we chose to improve is the sampling method.

*prm\_basic.py*: The basic sampling method in *prm\_basic* is to randomly select landmarks inside a scene’s bounding-box and verify each landmark is valid, do so until *num\_landamrk* landmarks have been chosen. These are the algorithm’s landmarks.

The main disadvantage of this method occurs when there are narrow passages. The probability of sampling points inside narrow passages decreases as the passage becomes narrower.

We implemented a ‘Gaussian sampling’ method based on *“The Gaussian Sampling Strategy for Probablistic Roadmap Planners”* (Boor, Overmas, van der Strappen), as presented in class.

*prm\_gaussian\_sampling.py:* The sampling method:  
We randomly sample a point, if it is valid we ignore it and sample another point.  
So we now have a non-valid point. We sample another point from a normal distribution centered at the current invalid point. If the new sampled point is valid – We add it to our set of landmarks.

This sampling method promotes points that are near obstacles, therefor the probability of sampling points near narrow passages increases.

The intuition behind this sampling method is that it has a higher probability of sampling points near obstacles. In many cases, these coincide with difficult passageways and often the shortest paths are those that occur near obstacles.

Exercise 3.1 (B):

|  |  |  |
| --- | --- | --- |
| *our\_easy\_robot\_maze.json* | *our\_medium\_robot\_maze.json* | *our\_difficult\_robot\_maze.json* |
| A picture containing text, clock  Description automatically generated |  |  |

We designed 3 scenes for our rod robot, with difficulty levels increasing:

Exercise 3.1 (C):

**Our empirical definitions:**

* Success rate: Percentage of runs with a successful path from start position to target position, of 14 attempts
* Running time: The average amount of time (in secs) it took the program to run on 14 attempts
* Number of samples needed to reach a solution when one exists: Number of landmarks needed for 3 consecutive runs with a successful path.

We calculate these properties for everyone of the following mazes:

|  |  |  |
| --- | --- | --- |
| *our\_easy\_robot\_maze.json* | *our\_medium\_robot\_maze.json* | *our\_difficult\_robot\_maze.json* |
| A picture containing text, clock  Description automatically generated | A picture containing rectangle  Description automatically generated | Chart  Description automatically generated with medium confidence |

We will first analyze our solver’s performance on the easy maze *our\_easy\_robot\_maze.json*  
For each of the following *num\_landmark*, run 14 times and calculate the average of the following parameters:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | 10 | 20 | 50 | 100 |
| ***prm\_basic.py*** | **Success rate** | 57.13% | 78.57% | 92.86% | 100% |
| **Running time** | 0.41s | 1.18s | 2.14s | 3.60s |
| ***prm\_gaussian\_sampling.py*** | **Success rate** | 85.71% | 92.86% | 100% | 100% |
| **Running time** | 0.44s | 1.20s | 1.96s | 3.82s |

|  |  |
| --- | --- |
|  | **Number of samples needed to reach a solution when one exists** |
| ***prm\_basic.py*** | 12 |
| ***prm\_gaussian\_sampling.py*** | 5 |

Next we’ll analyze our solver’s performance on the medium maze *our\_medium\_robot\_maze.json*  
For each of the following *num\_landmark*, run 14 times and calculate the average of the following parameters:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | 10 | 20 | 50 | 100 |
| ***prm\_basic.py*** | **Success rate** | 0% | 14.3% | 35.71% | 57.14% |
| **Running time** | 0.34s | 0.83s | 1.81s | 3.03s |
| ***prm\_gaussian\_sampling.py*** | **Success rate** | 14.3% | 35.71% | 78.6% | 92.6% |
| **Running time** | 0.39s | 0.81s | 1.85s | 3.29s |

|  |  |
| --- | --- |
|  | **Number of samples needed to reach a solution when one exists** |
| ***prm\_basic.py*** | 90 |
| ***prm\_gaussian\_sampling.py*** | 30 |

Lastly we’ll analyze our solver’s performance on the difficult maze *our\_difficult\_robot\_maze.json*  
For each of the following *num\_landmark*, run 14 times and calculate the average of the following parameters:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | 50 | 100 | 250 | 500 | 1000 |
| ***prm\_basic.py*** | **Success rate** | 0% | 0% | 7.14% | 14.3% | 21.43% |
| **Running time** | 1.53s | 2.82s | 5.80s | 10.84s | 21.20s |
| ***prm\_gaussian\_sampling.py*** | **Success rate** | 7.14% | 14.3% | 35.72% | 57.14% | 71.43% |
| **Running time** | 1.61s | 3.01s | 6.88s | 12.83s | 23.1s |

|  |  |
| --- | --- |
|  | **Number of samples needed to reach a solution when one exists** |
| ***prm\_basic.py*** | 6100 |
| ***prm\_gaussian\_sampling.py*** | 2350 |

Summary:

* Our empirical results correlate well with our proposed hypothesis: The success rate of *prm\_gaussian\_sampling* is significantly better than *prm\_basic*. This difference becomes greater as the scene becomes more difficult and complex.
* Our empirical results show a minor increase in runtime for *prm\_gaussian\_sampling* over *prm\_basic*. This seems to be consistent without direct relation to the scene’s complexity.
* Our empirical results also show a major decrease in (‘minimum’) number of samples needed to reach a solution when one exits. This too becomes more dramatic as the scene becomes more complex.