HW 2—Sampling-based motion planning

Sapir Tubul – 305728180

Idan Lev Yehudi – 206741878

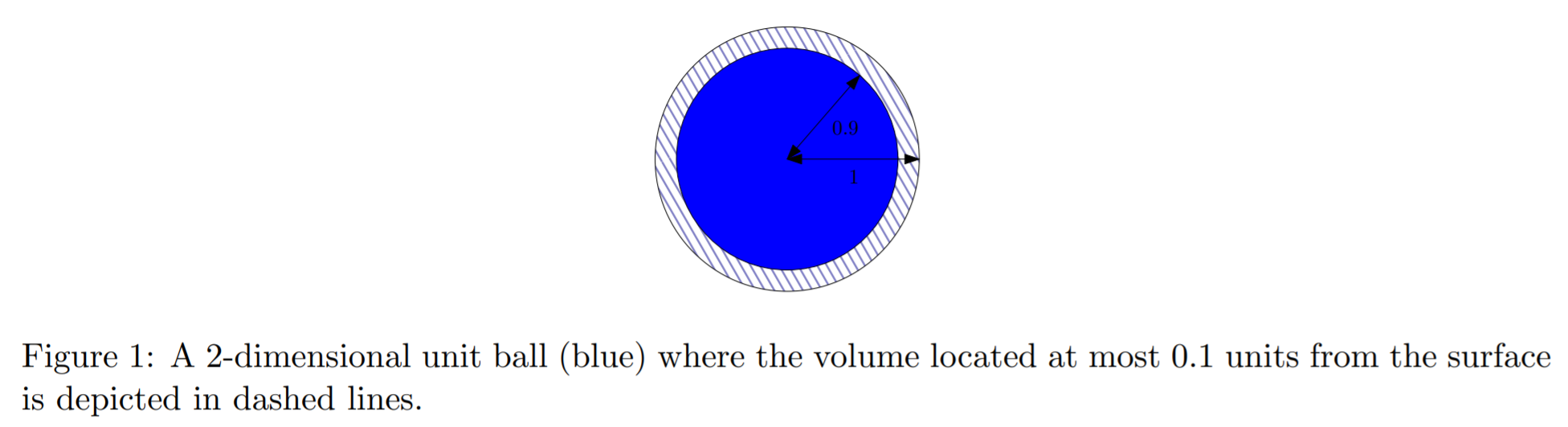
**Warmup**

1. In a 2-dimensional unit ball (namely, a disk), how much of the volume is located at most 0.1 units from the surface?

(c) Roughly 20%

The volume of ring with radius of ~0.95R, most be greater then 10% of the area.  
Volume of 60% sound to much.

The intuition is roughly 20%.



Test our intuition:

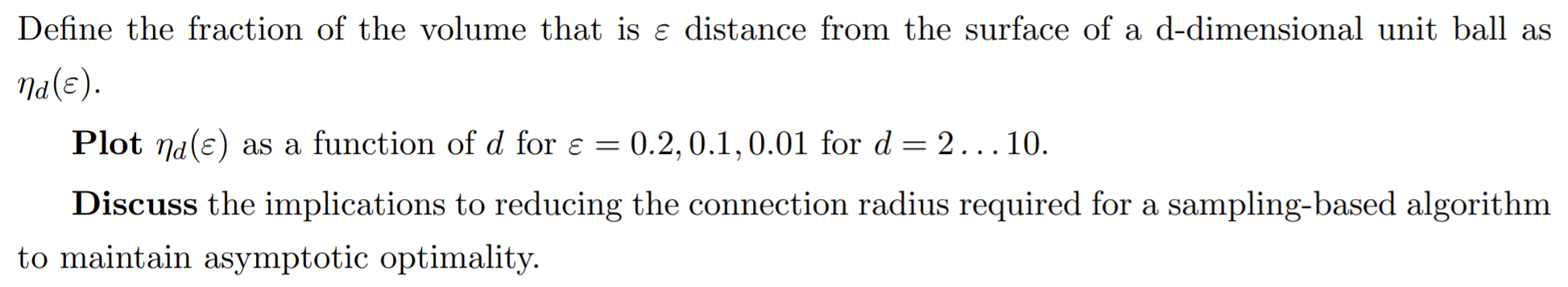
1. In a 9-dimensional unit ball, how much of the volume is located at most 0.1 units from the surface?

(d) Roughly 60%

The volume is located at most 0.1 units is proportion to ,so we assume that the relative area will increase.

Test our intuition:

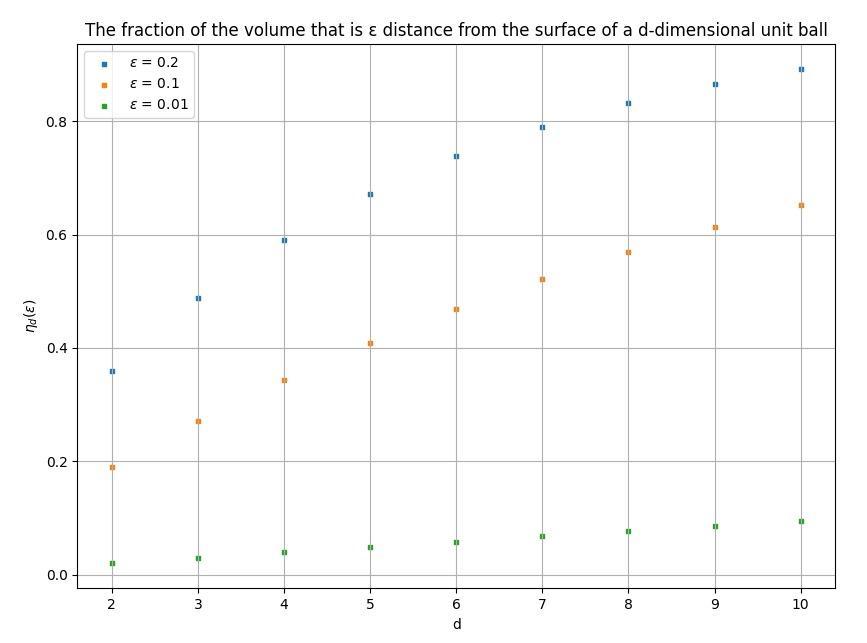
**Theoretical Part**



From Wikipedia: "The volume of a n-ball of radius R is where is the volume of the unit n-ball, the n-ball of radius 1."

*Source: Volume of an n-ball -* [*https://en.wikipedia.org*](https://en.wikipedia.org/wiki/special:search/Volume%20of%20an%20n-ball)

Therefore as we did in the earlier examples:



A slight change in causes a much larger change in volume difference in higher dimensions. The rate of change (derivative) at dimension :

We can see that for higher dimensions we get a much greater rate of change (exponential in the dimension). This means that in order to maintain a certain expected number of neighbors, the changes in radius as the tree size grows become much smaller.

Recall the PRM’s connection radius for asymptotic optimality:

This connection radius maintains expected number of neighbors.

The relative change in radius between consecutive samples:

The expression tends to 1 as tends to infinity. This means that tends to 0 as tends to infinity, and this happens much quicker for higher dimensions (since taking the n’th root of a number smaller than 1 brings it closer to 1, bringing closer to 0).

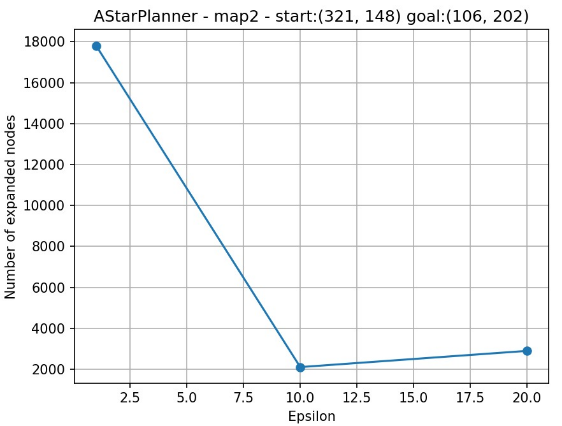
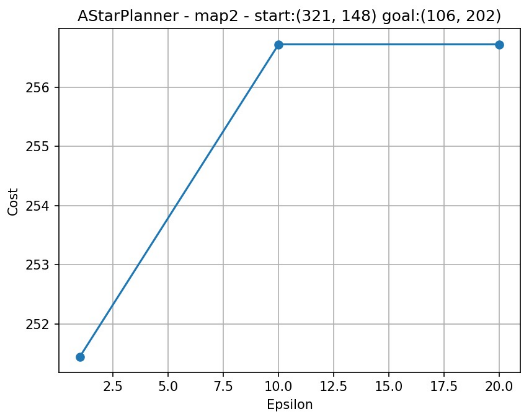
This analysis of the connection radius holds for RRT\*’s optimal connection radius, since it has the same functional form:

**Motion Planning: Search and Sampling**

1. **A\* Implementation**

We implemented the weighted version of where the heuristic is weighted by a factor of .

We tried different values of to see how the behavior changes:



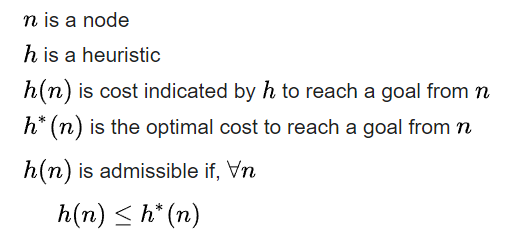
For : The final cost is ~**251.45** and the number of expanded states is **17805**.

For : The final cost is ~**256.72** and the number of expanded states is **2103**.

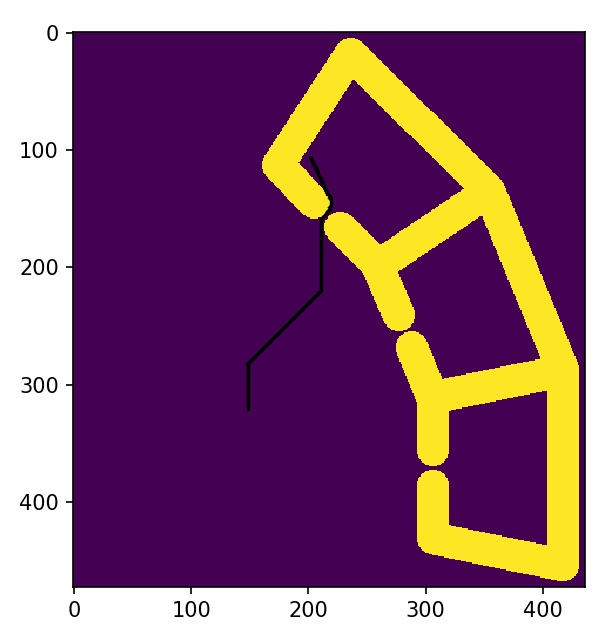
For : The final cost is ~**256.72** and the number of expanded states is **2894**.

Because we used an admissible heuristic, we expect that vanilla () will give the optimal solution with the minimum cost. Increasing epsilon and thus believing more in heuristics will decrease the number of expanded states but increase the cost (this is because the heuristic would be no longer admissible).   
In our graphs, the cost increases slightly between to , and remains unchanged between to . The number of expanded nodes decreases between to , but then increases between to .

Admissible heuristic formulation:

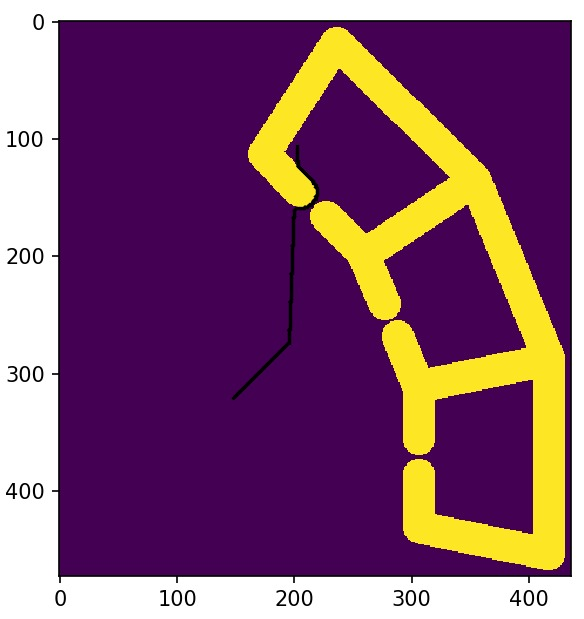


Path returned from A\* planner with on map2:



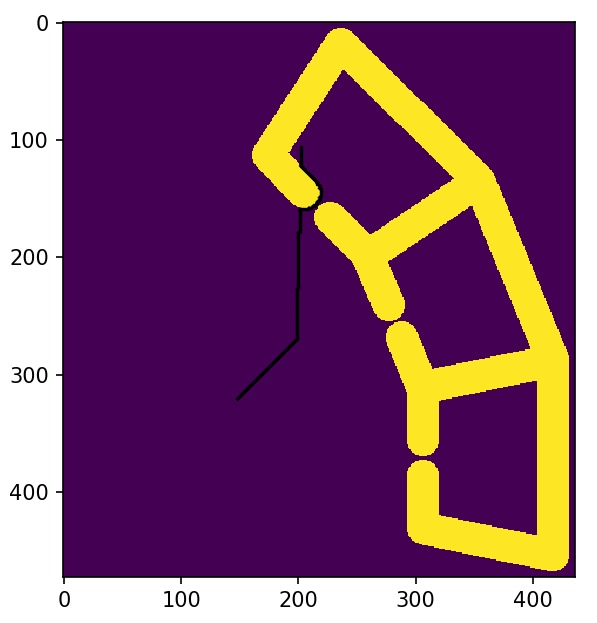
The starting point is (321,148) and the goal point is (106,202). This is a path with the minimum cost.

Path returned from A\* planner with on map2:



For , returned path after running on map2. The starting point is (321,148) and the goal point is (106,202). The planner attempts to go directly to the goal point (because the heuristic, the Euclidean distance to the goal, gets more weight) and after he cannot proceed further because of an obstacle, it went around it.

Path returned from A\* planner with on map2:



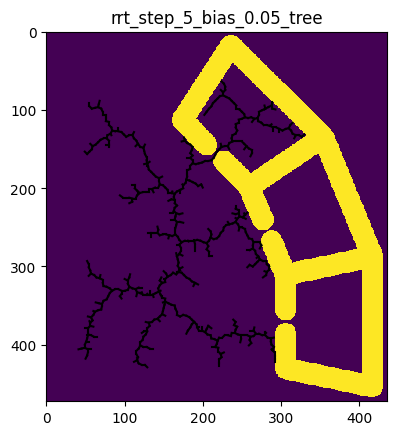
For , returned path after running on map2.The starting point is (321,148) and the goal point is (106,202). The path selection is similar to running the algorithm with .

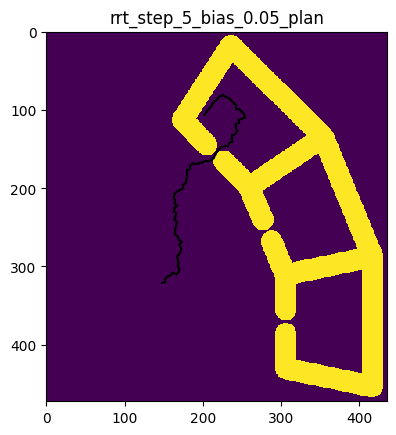
1. **RRT and RRT∗ Implementation**

RRT:

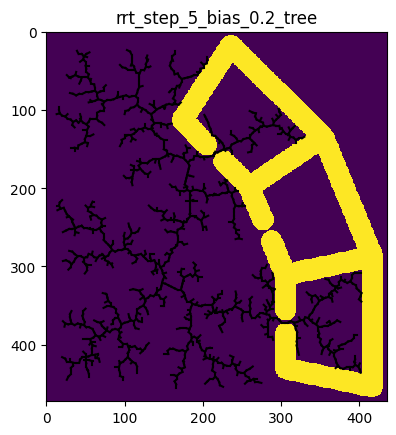
We tested RRT with step sizes of 5, 50 and infinity (for each sampled state, attempts to extend all the way from the closest state in the current tree). For each of the step sizes, we ran the algorithm with a bias of 5% and 20% chance of picking the goal state.

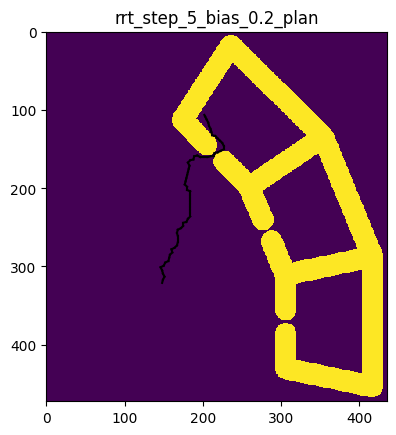
Example for step size 5, bias 5%:



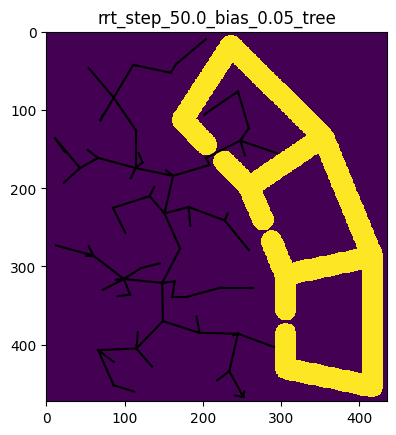


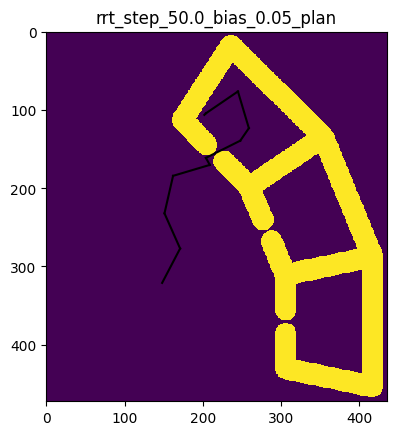
Example for step size 5, bias 20%:



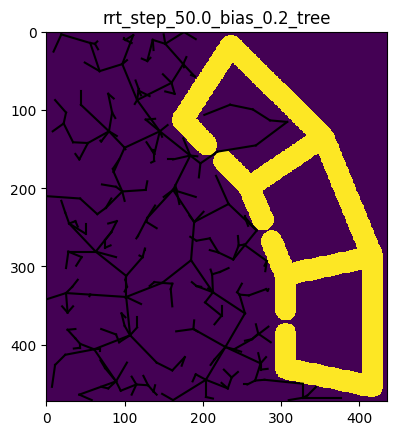


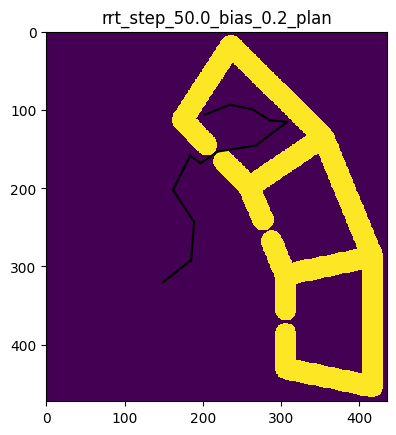
Example for step size 50, bias 5%:





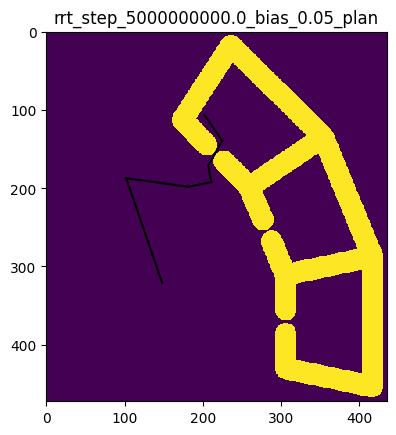
Example for step size 50, bias 20%:





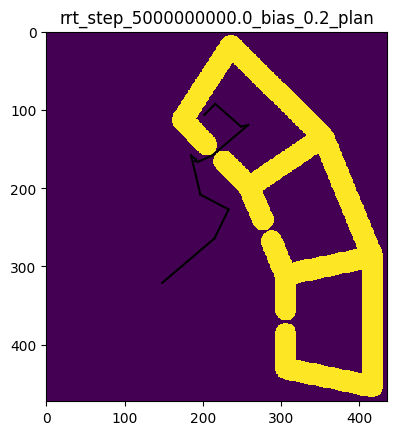
Example for step size inf, bias 5%:





Example for step size inf, bias 20%:





The following tables summarizes the mean results across 10 runs for each configuration:

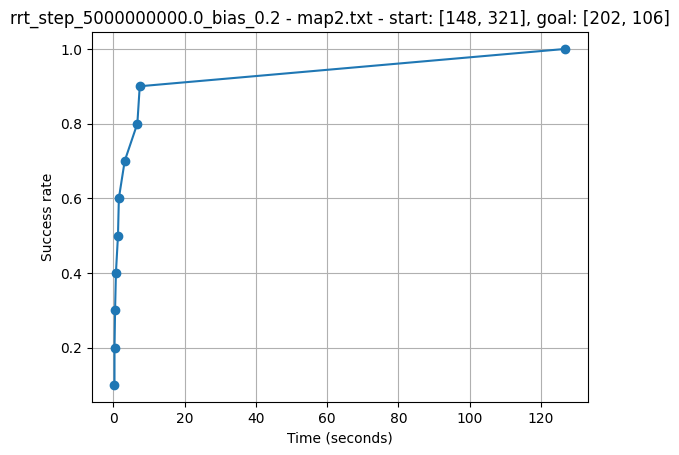
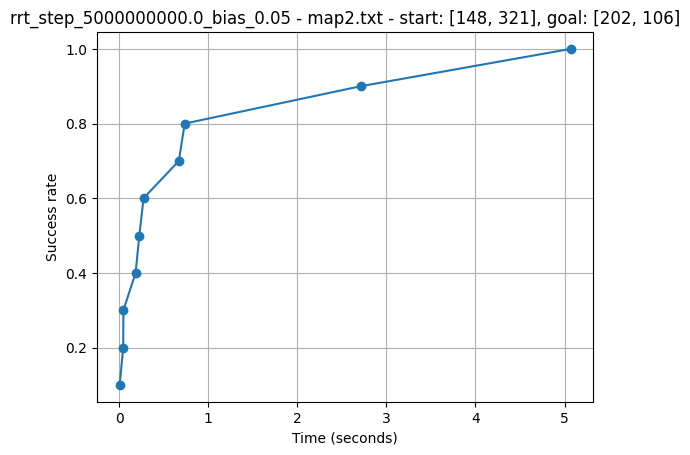
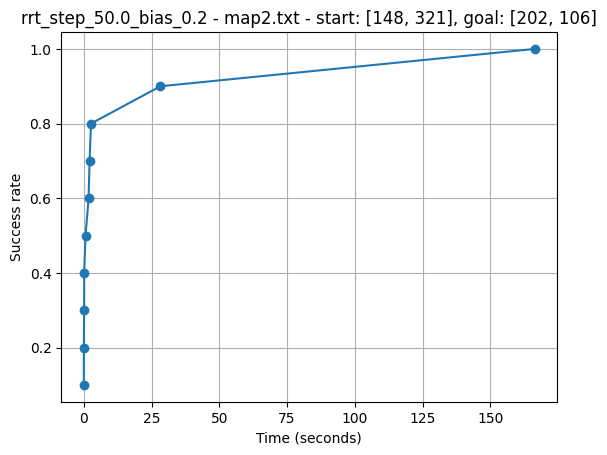
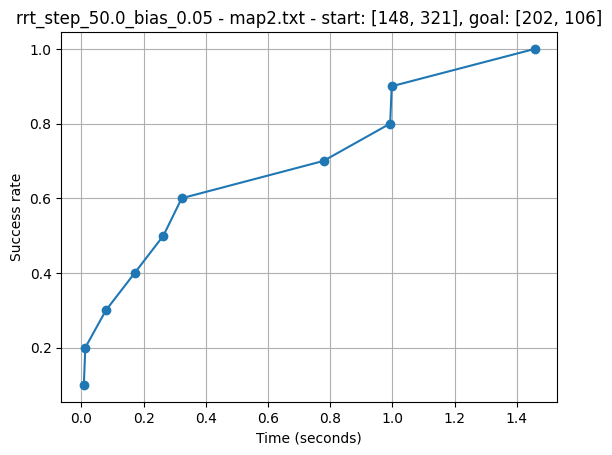
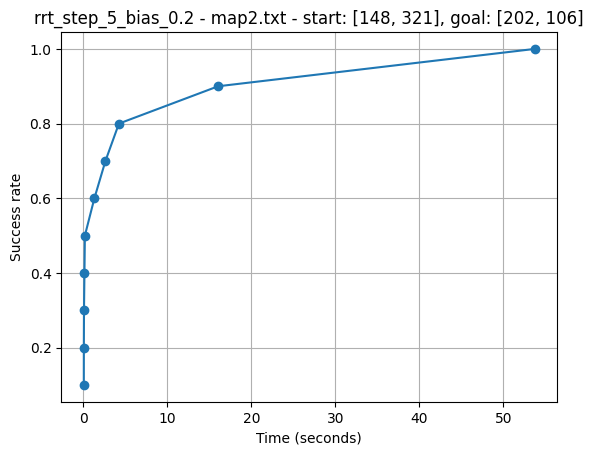
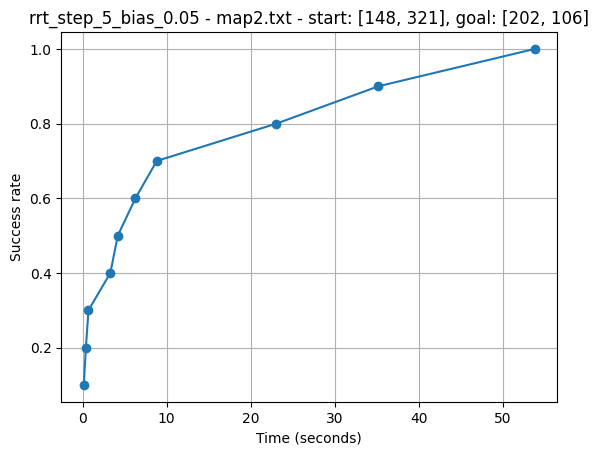
| Mean Run Time (secs) | Step Size 5 | Step Size 50 | Step Size Inf |
| --- | --- | --- | --- |
| Bias 5% | 13.58 | 0.51 | 0.99 |
| Bias 20% | 7.83 | 20.21 | 14.87 |

| Mean Plan Cost (pixels) | Step Size 5 | Step Size 50 | Step Size Inf |
| --- | --- | --- | --- |
| Bias 5% | 306.07 | 332.68 | 423.83 |
| Bias 20% | 288.09 | 339.92 | 434.24 |

From the above data, it seems like for this problem a step size of 50 and a bias of 5% give the best running time, while not losing much performance compared to the other configurations. As a general intuition: a step size of 5 might be too small and cause the algorithm not to progress fast enough, while an infinite step size might cause it too many extensions to fail because of obstacles. The bias of 20% seems to help advance faster to the goal in a small step size, but in big step sizes cause too many failed extensions.

In this particular map, there is a large space to traverse from the start position until reaching the obstacles, so this might explain why larger step sizes are favorable in this setting.

We’d note that it seems as if 10 runs per configuration were too little to be statistically significant. For some of the configurations, there was a single run that got stuck for a long while until terminating, causing the mean run time to increase significantly, while in others this didn’t happen. This can be seen in the following success rate vs. algorithm run time graphs.



RRT\*:

Based on the conclusions from the previous part, we ran the RRT\* with a bias to sample the goal state of 5%.

We ran the algorithm again with step sizes of 5, 50. For the amount of nearest neighbors to attempt the rewiring operation, we tested logarithmic and constant numbers. For the logarithmic, we chose . For constant numbers, we chose the following options: 3, 10, 30, 70.

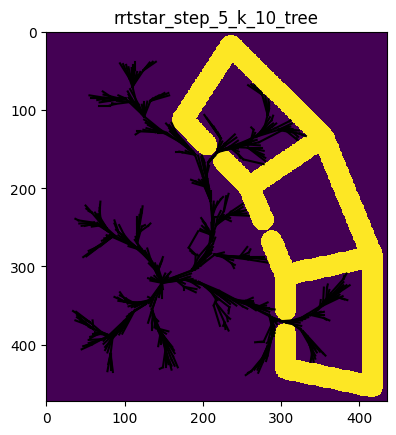
We will show one example tree and plan found for each step size and nearest neighbors choice.

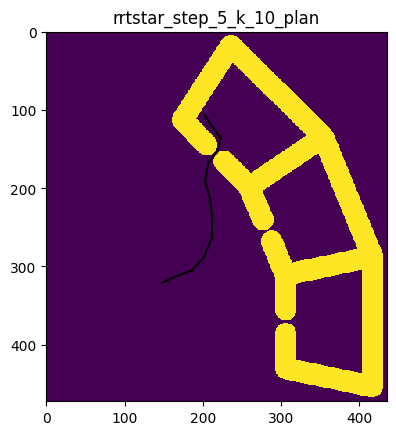
Example for step size 5, number of neighbors 3:



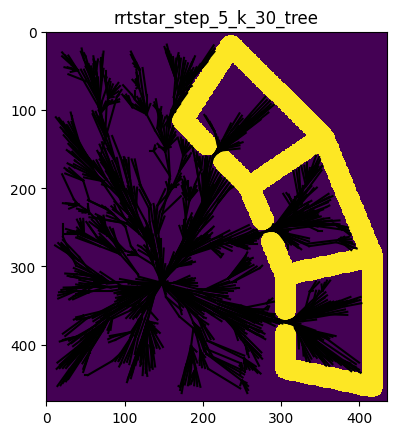


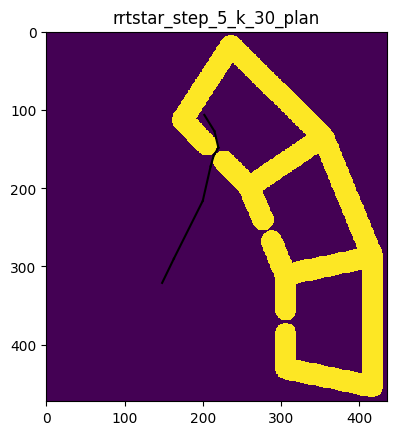
Example for step size 5, number of neighbors 10:



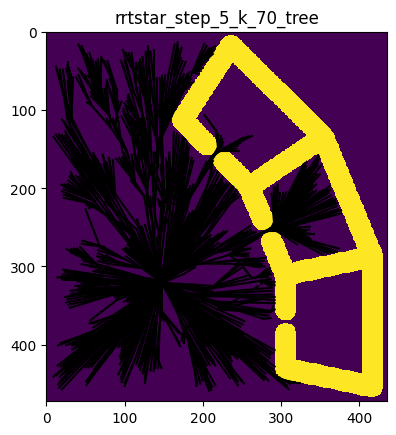


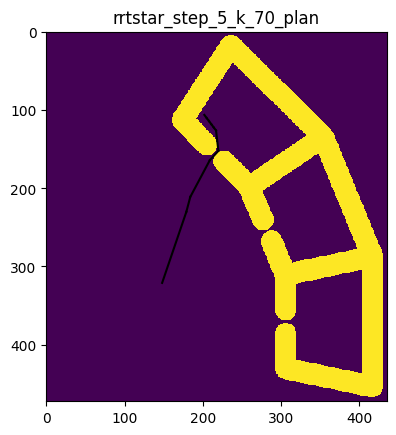
Example for step size 5, number of neighbors 30:



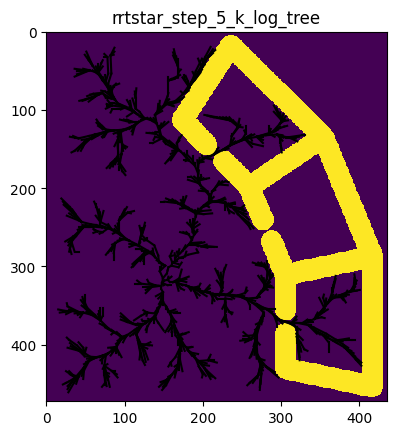


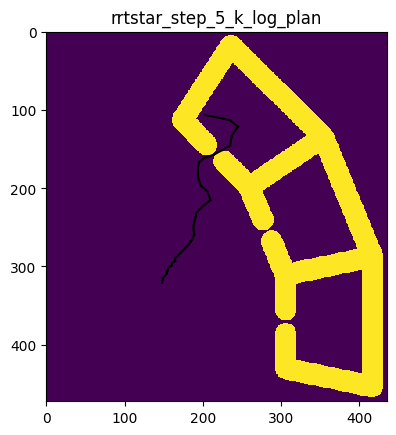
Example for step size 5, number of neighbors 70:



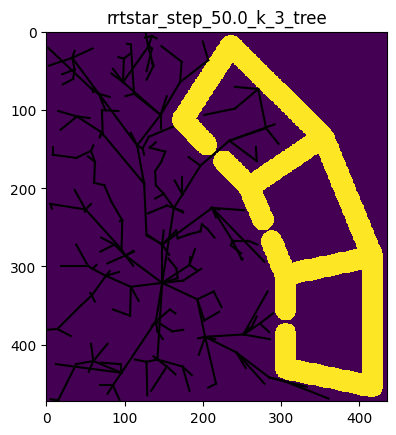


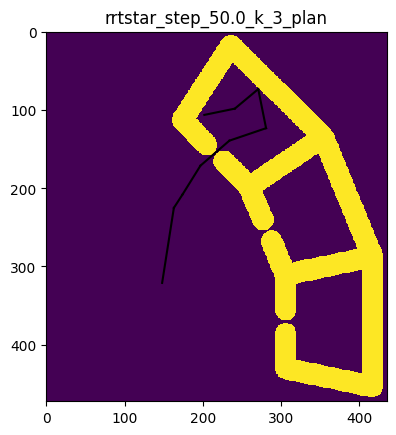
Example for step size 5, number of neighbors log:



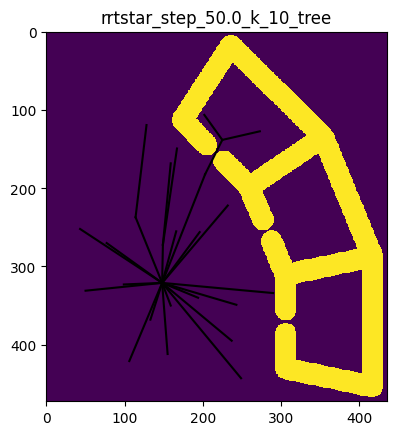


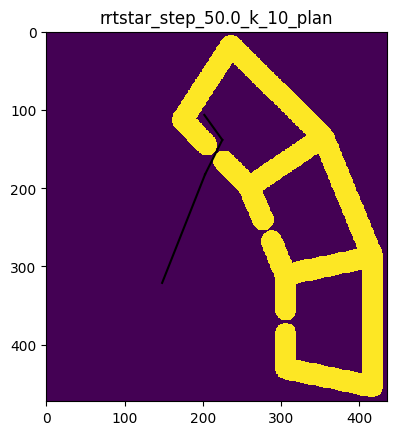
Example for step size 50, number of neighbors 3:



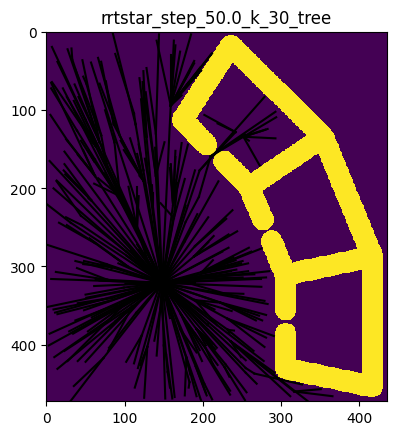


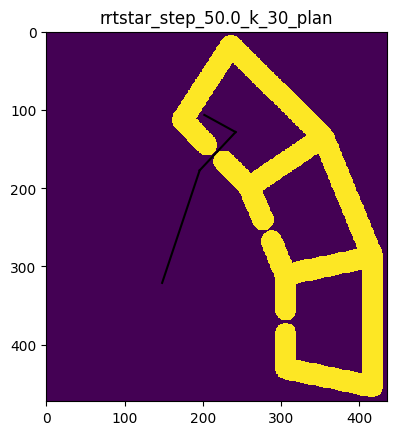
Example for step size 50, number of neighbors 10:



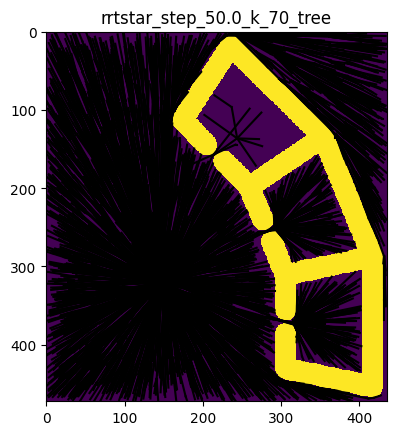


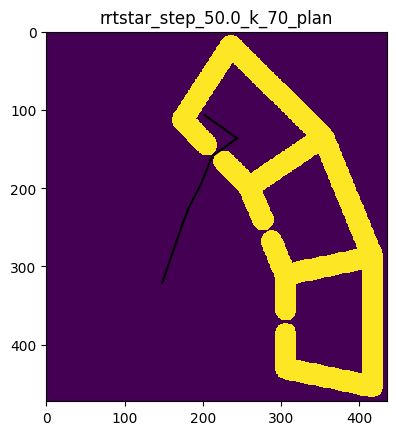
Example for step size 50, number of neighbors 30:



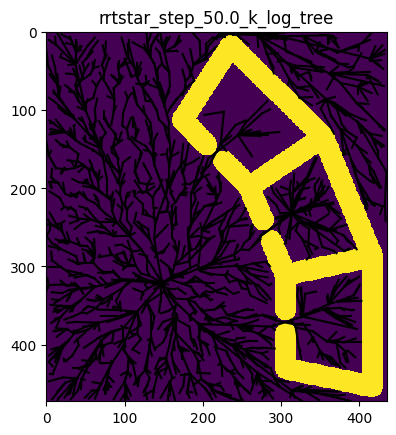


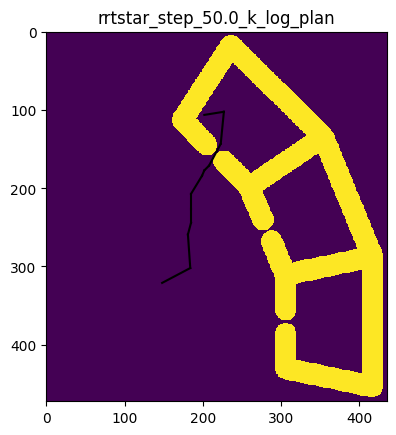
Example for step size 50, number of neighbors 70:





Example for step size 50, number of neighbors log:





| Mean Run Time (secs) | Step Size 5 | Step Size 50 |
| --- | --- | --- |
| num neighbors 3 | 8.55 | 6.94 |
| num neighbors 10 | 12.41 | 7.08 |
| num neighbors 30 | 14.44 | 6.94 |
| num neighbors 70 | 20.58 | 28.45 |
| num neighbors log | 15.02 | 2.32 |

| Mean Plan Cost (pixels) | Step Size 5 | Step Size 50 |
| --- | --- | --- |
| num neighbors 3 | 288.91 | 309.68 |
| num neighbors 10 | 252.36 | 255.40 |
| num neighbors 30 | 238.66 | 255.43 |
| num neighbors 70 | 236.19 | 256.06 |
| num neighbors log | 273.22 | 287.15 |

First of all, we can see that the plans generated by the RRT\* are generally better (lower cost) than those of the RRT. For 3 neighbors considered, the plan costs are similar (slightly lower) than those of the RRT. The more neighbors considered, the greater this effect, until the improvement in cost reaches a plateau (while the time spent continues to increase).

As for the log number of neighbors, it seems like it helped save a lot of time when the tree had fewer vertices (with step size of 50), but in a graph with a greater number of nodes, it led to a long runtime without improving the plan’s costs.

For each setting of step size and number of neighbors function, we plotted the success rate of the algorithm vs. the running time, and the path cost vs. the running time. Once more, the results seem very noisy because each configuration was only tested 10 times.

