Ryan Farr – u0771896
October 13th, 2016
CS 6370 - Motion Planning

Homework 2 – Sampling Based Planning

Rapidly-Exploring Random Trees

1. See Figures 1 through 8 below.

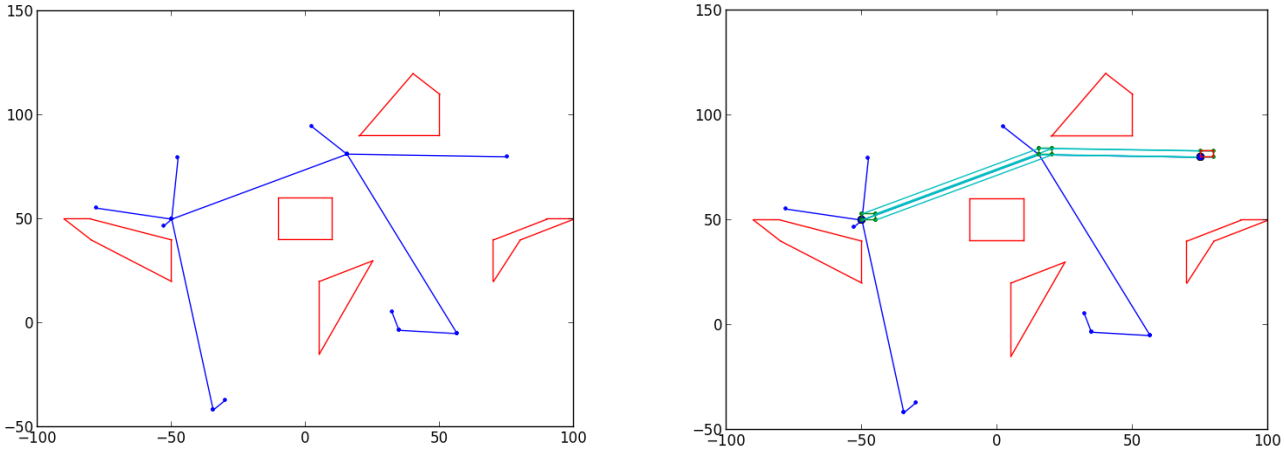

Figure 1 – RRT with 5% bias tree (left) and path (right) on env0

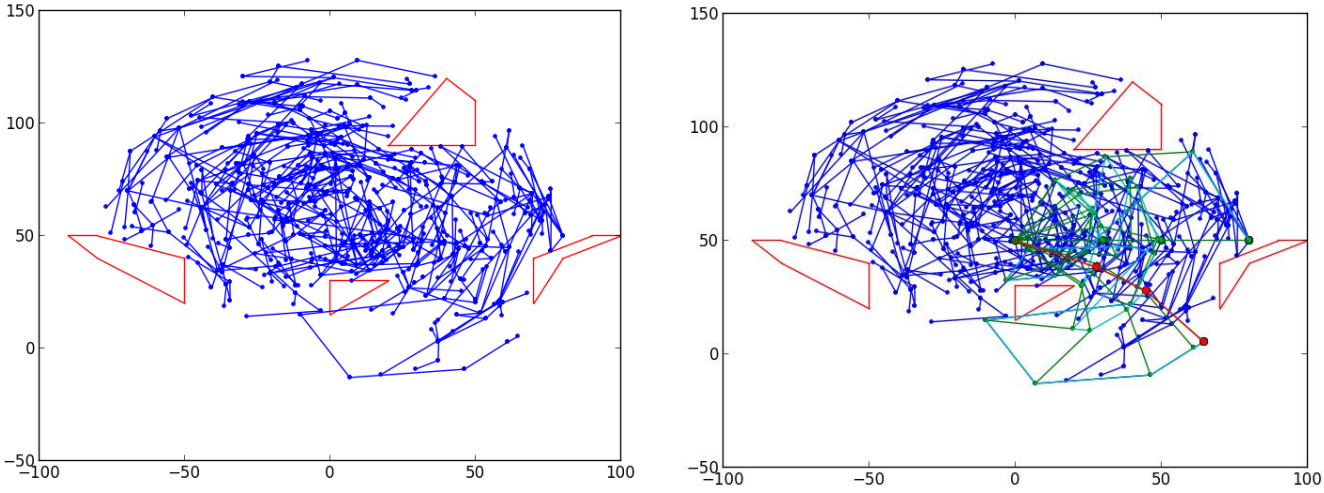

Figure 2 – RRT with 5% bias tree (left) and path (right) on env1

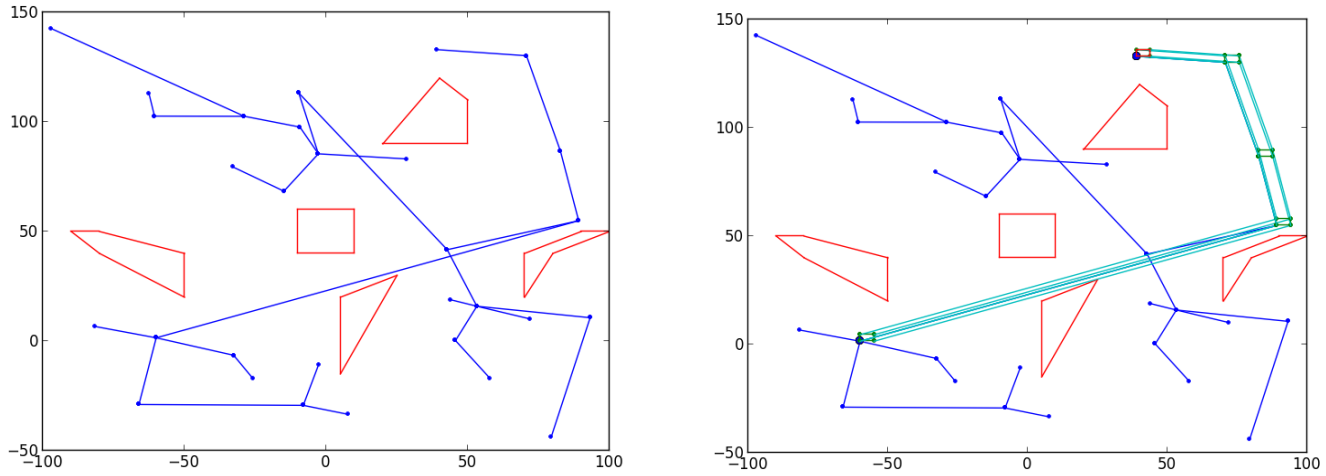

Figure 3 – RRT with 5% bias tree (left) and path (right) on env0 with start = (-60, 1.5) and goal = (39, 133)

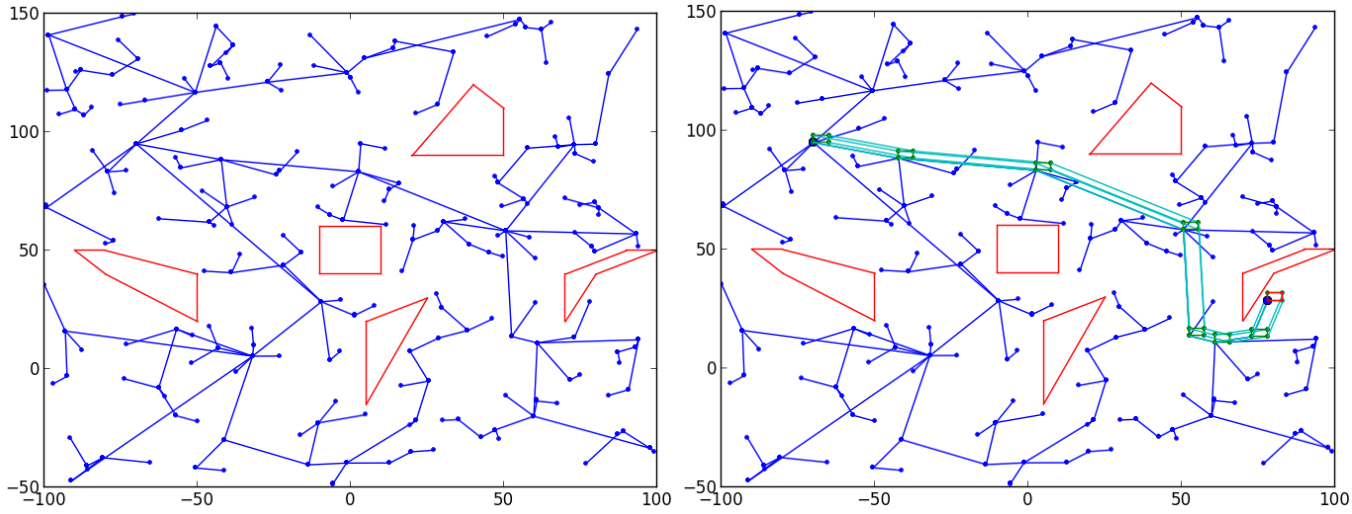

Figure 4 – RRT with 5% bias tree (left) and path (right) on env0 with start = (-70, 95) and goal = (78, 28.5)

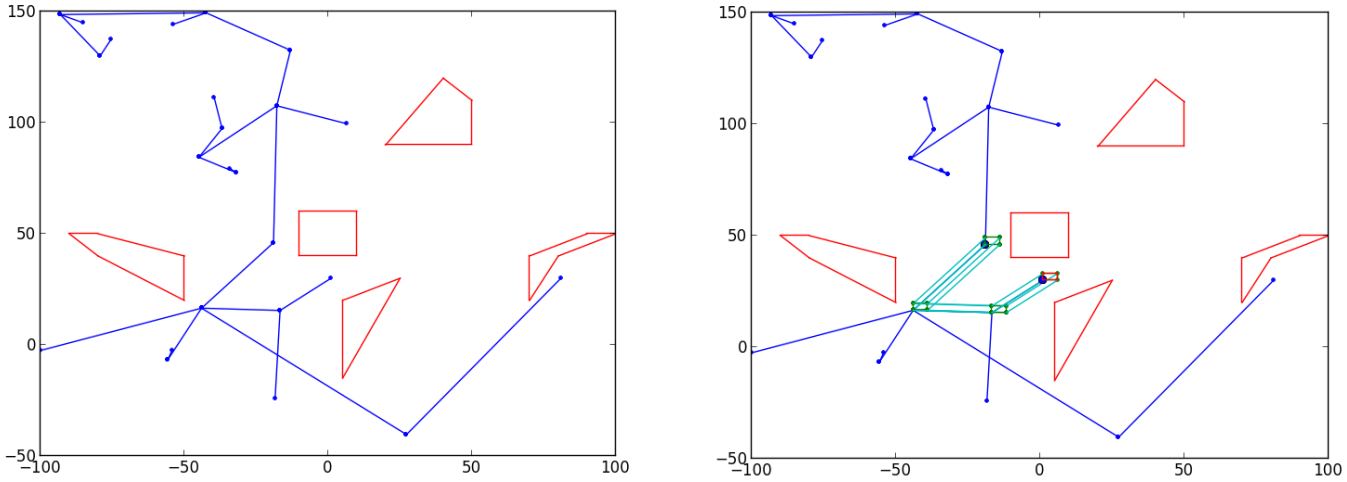

Figure 5 – RRT with 5% bias tree (left) and path (right) on env0 with start = (-19, 46) and goal = (1, 30)

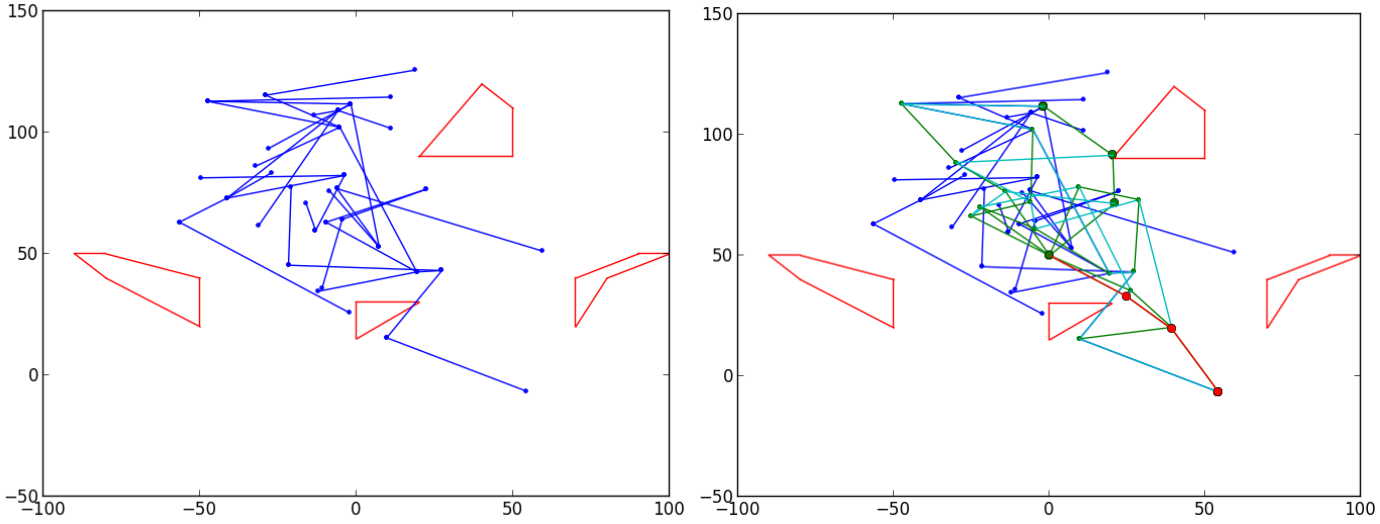

Figure 6 – RRT with 5% bias tree (left) and path (right) on env1 with start = (0.8, 0.8, 0.8) and goal = (-0.6, -0.15, -0.3)

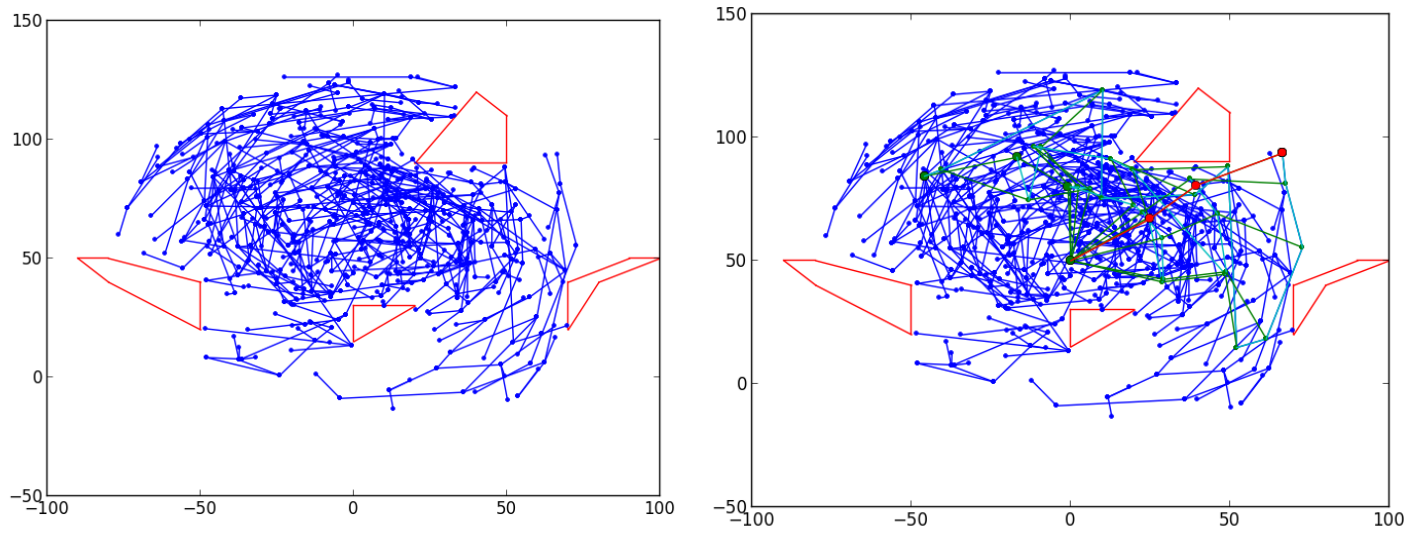

Figure 7 – RRT with 5% bias tree (left) and path (right) on env1 with start = (1.6, 0.9, 0.9) and goal = (0.6, 0.15, -0.3)

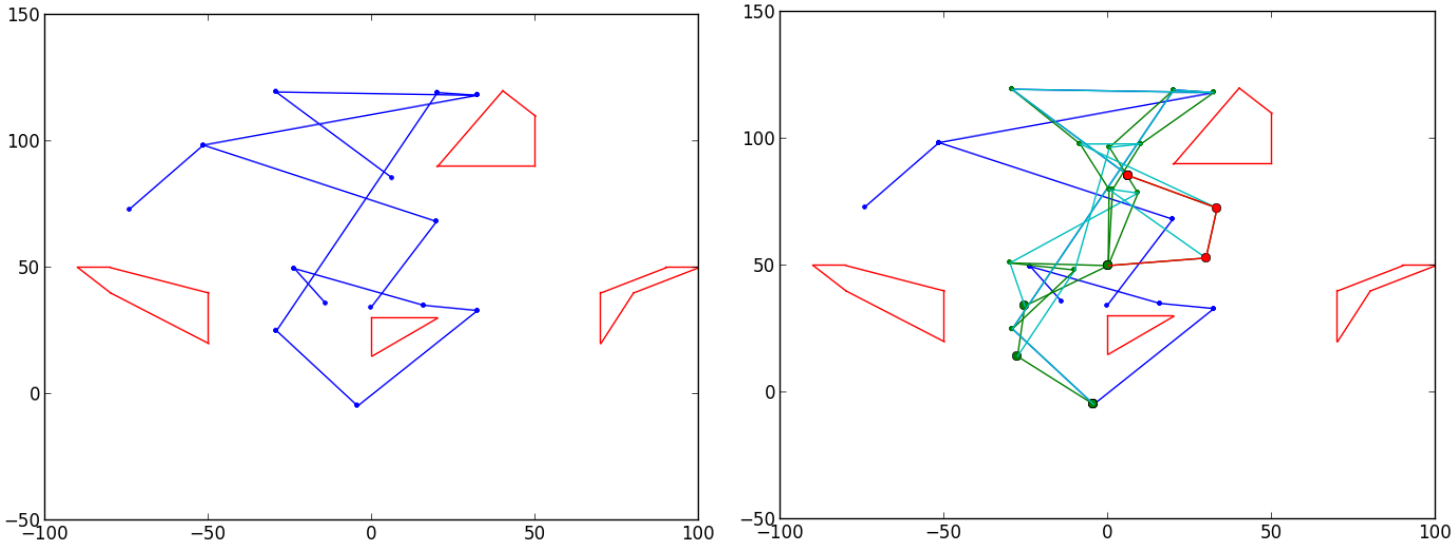

Figure 8 – RRT with 5% bias tree (left) and path (right) on env1 with start = (3.7, 0.9, 1.0) and goal = (0.1, 1.3, 1.3)

2. As can be seen in figures 9 through 12, increasing the bias tends to greatly reduce the number of nodes required to connect the start and goal. Notice that with 0% bias we weren't even able to find a path, even though there are far more nodes on the map than in any other Figure. In fact, even a small bias greatly reduces the number of nodes, as seen in Figure 9 and Figure 10. Of course, on average we may require more nodes for any given start and goal. This map has very few obstacles between the start and goal and lots of open space, which is a good situation for biasing.

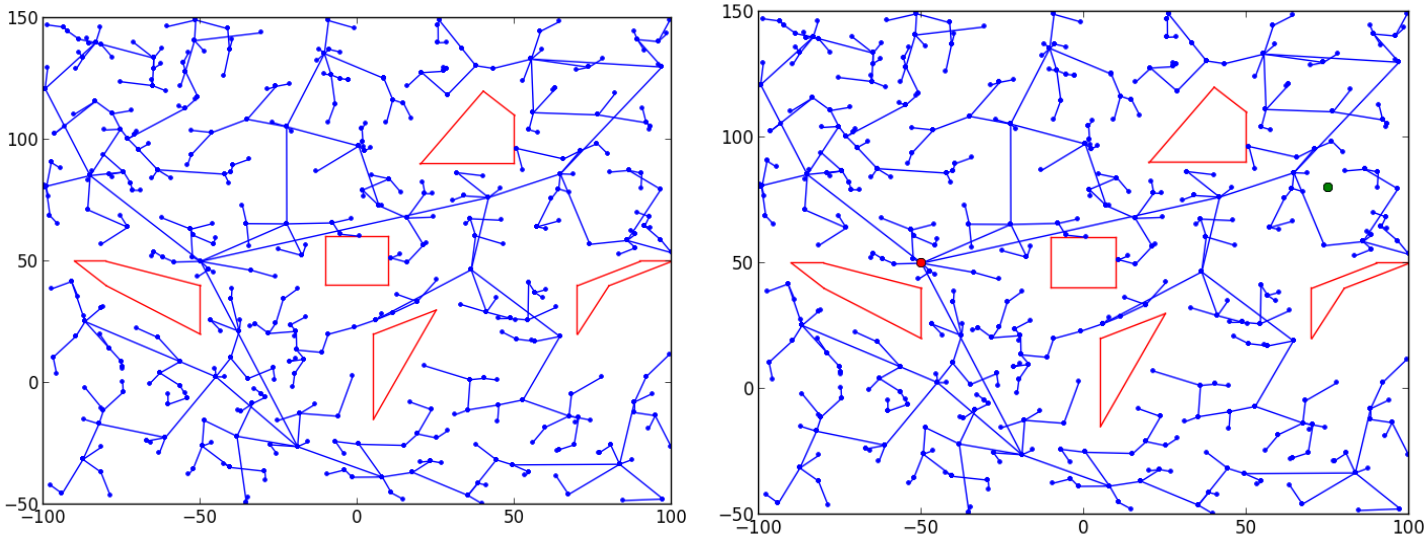

Figure 9 – RRT with 0% bias tree (left) and path (right) on env0

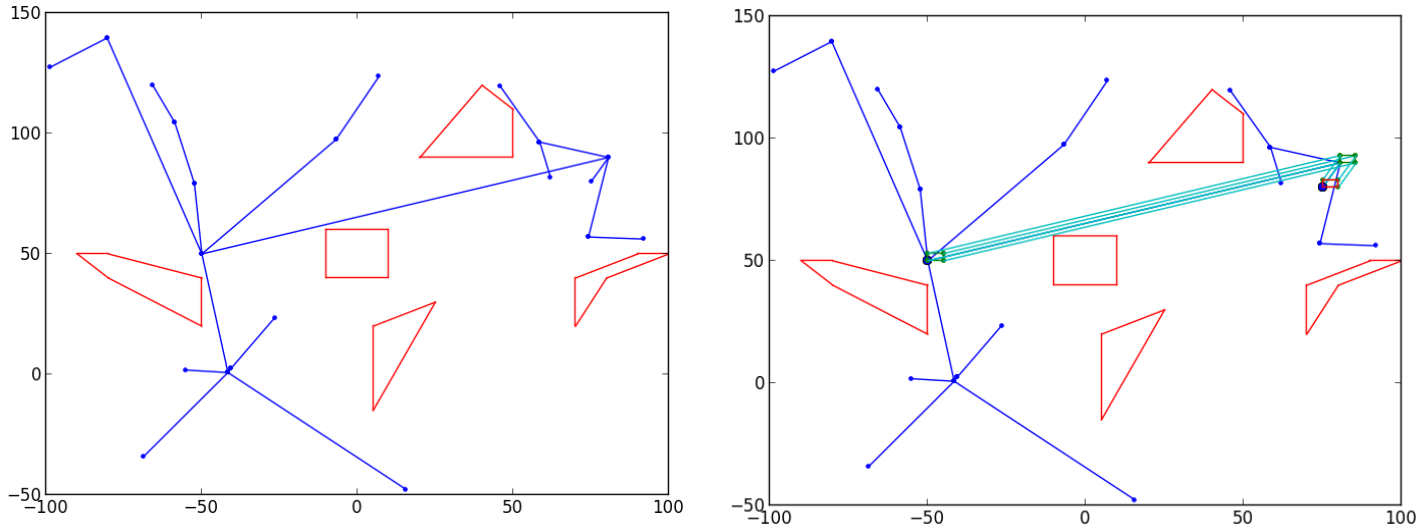

Figure 10 – RRT with 5% bias tree (left) and path (right) on env0

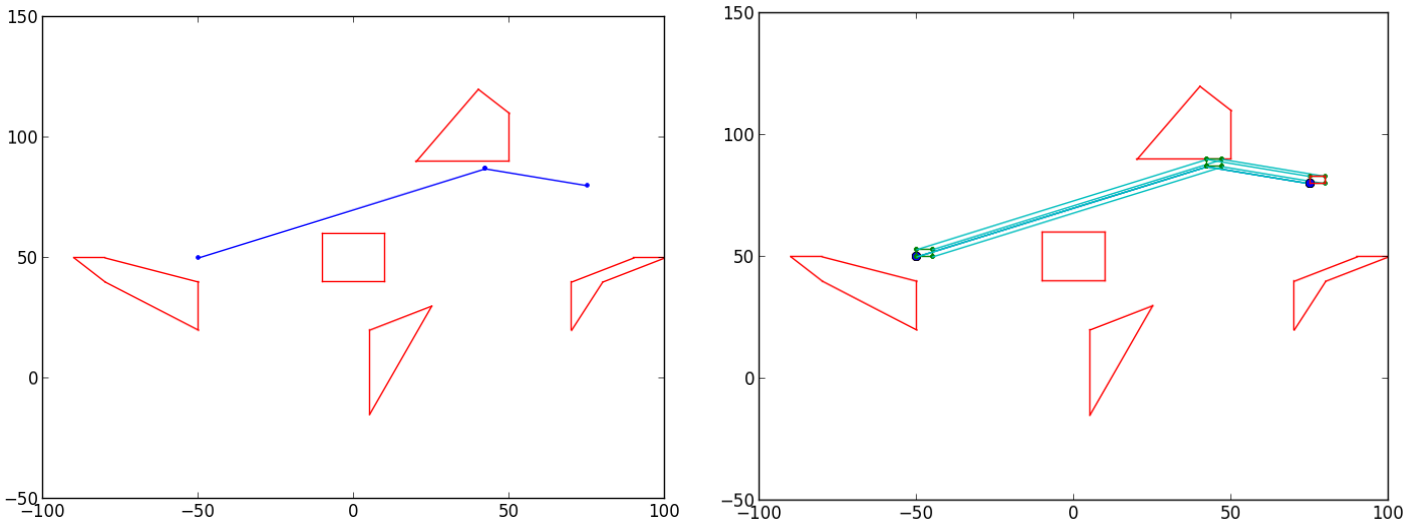

Figure 11 – RRT with 10% bias tree (left) and path (right) on env0

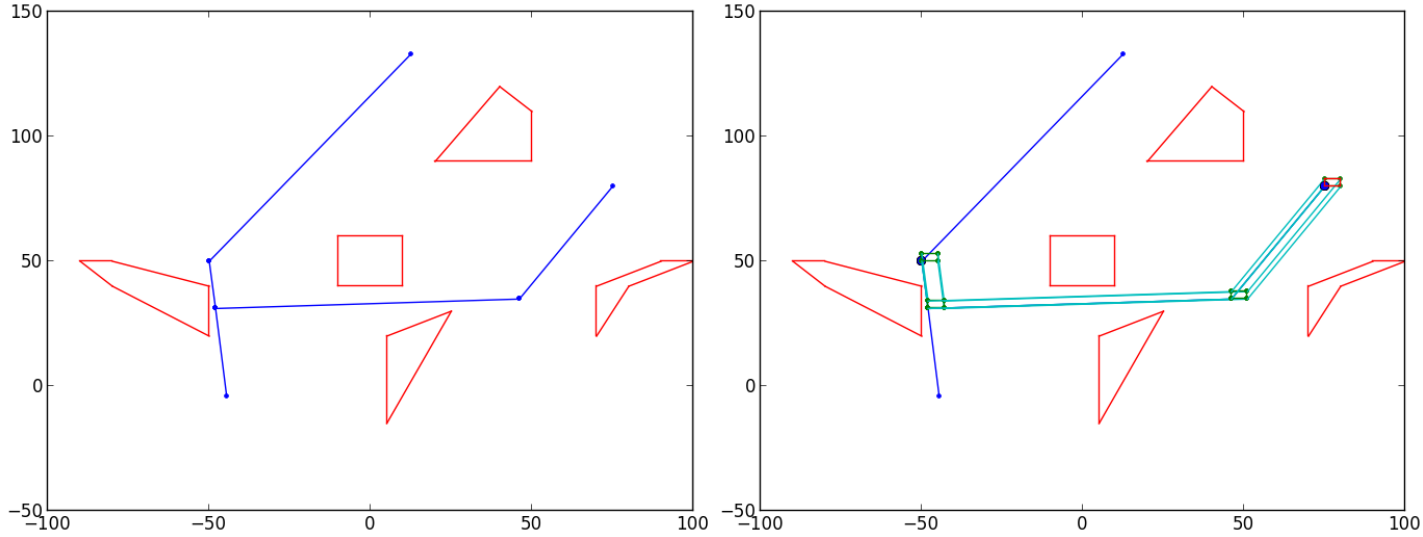

Figure 12 – RRT with 50% bias tree (left) and path (right) on env0

3. See figures 13 and 14. In this case, the number of nodes seems to be roughly similar, if not more (because of the 'connect'); however, the paths are found far more quickly. This is evident by the white space in each figure. This is particularly noticeable in figure 15 when compared to figure 2. Additionally, the path in figure 14 appears for smoother.

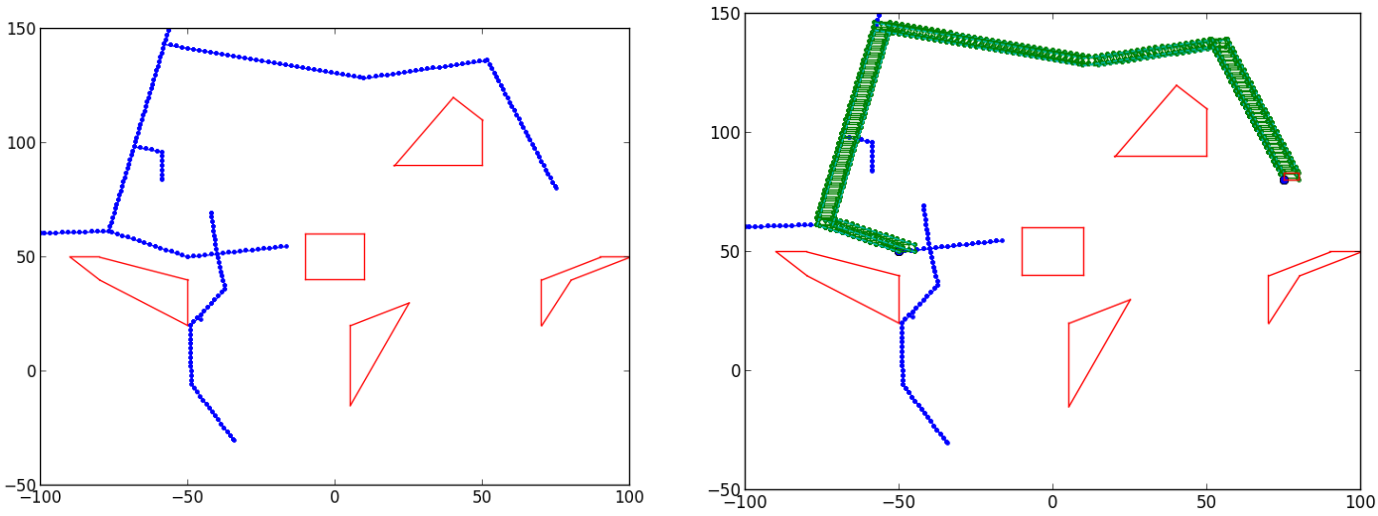

Figure 13 – RRT-Connect with 5% bias tree (left) and path (right) on env0

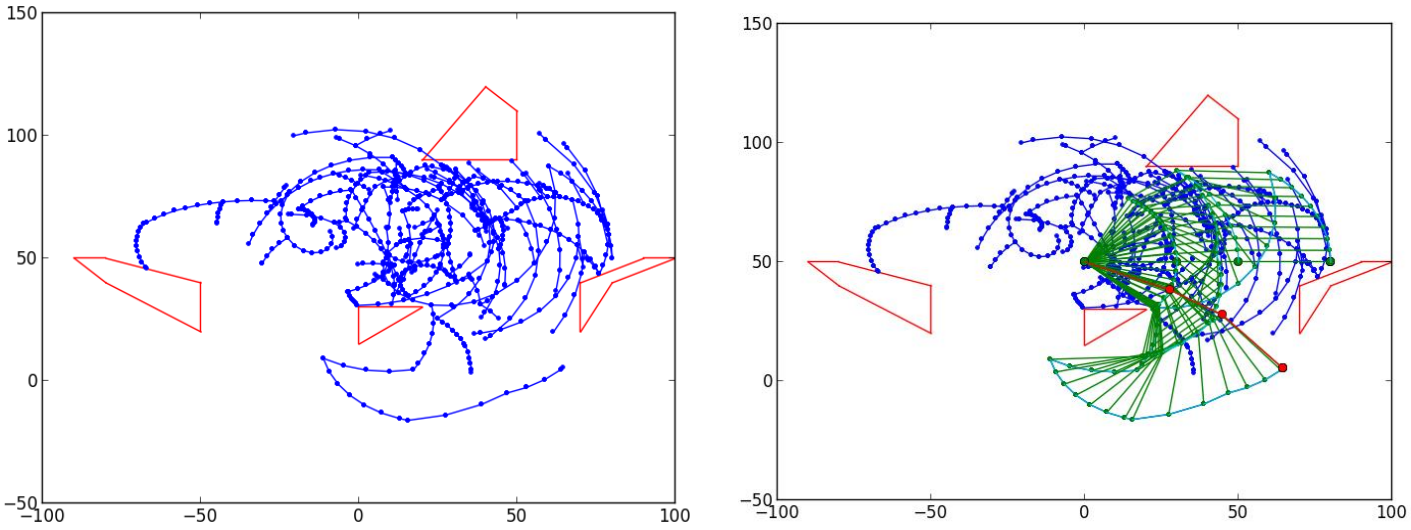

Figure 14 – RRT-Connect with 5% bias tree (left) and path (right) on env1

4. See figures 15 and 16. The main difference for environment 0 is that we see two distinct trees in figure 14. The number of nodes and whitespace is similar to regular RRT-Connect. The difference for environment 1, however, is drastic. The path is found far more quickly and thus there is much more white space. Intuitively, this appears to be because the goal is slightly hidden behind an object. One of the first branches for the goal's tree opens it up very quickly for connecting through the center whitespace. For the default scenario in environment 1, bidirectional RRT-Connect is far superior to the other methods.

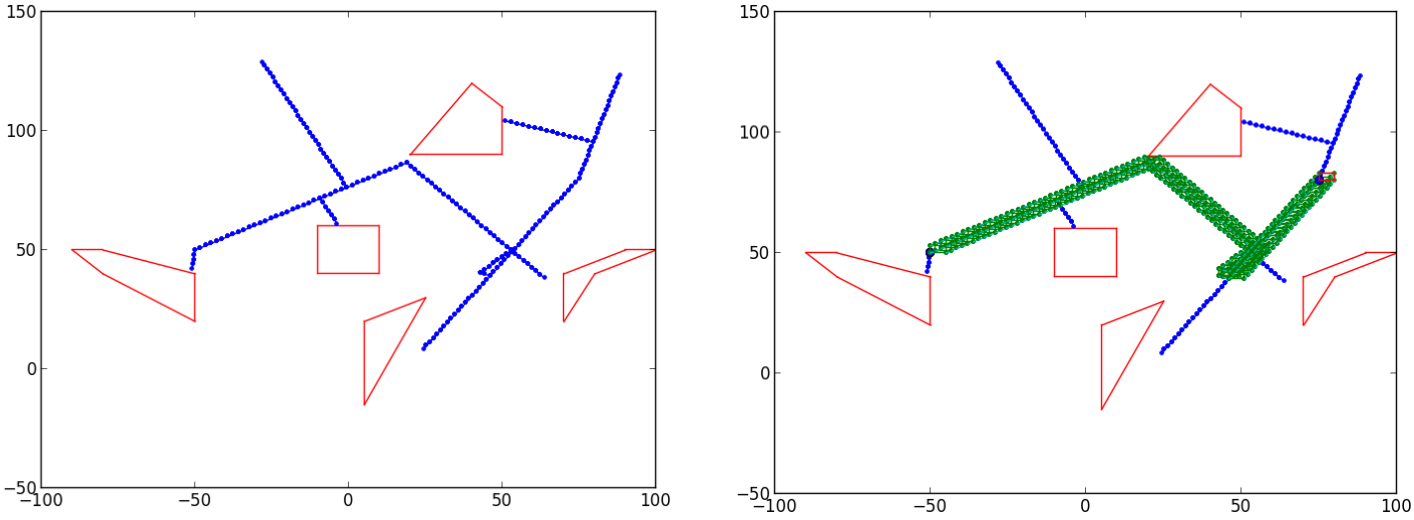

Figure 15 – Bidirectional RRT-Connect with 5% bias tree (left) and path (right) on env0

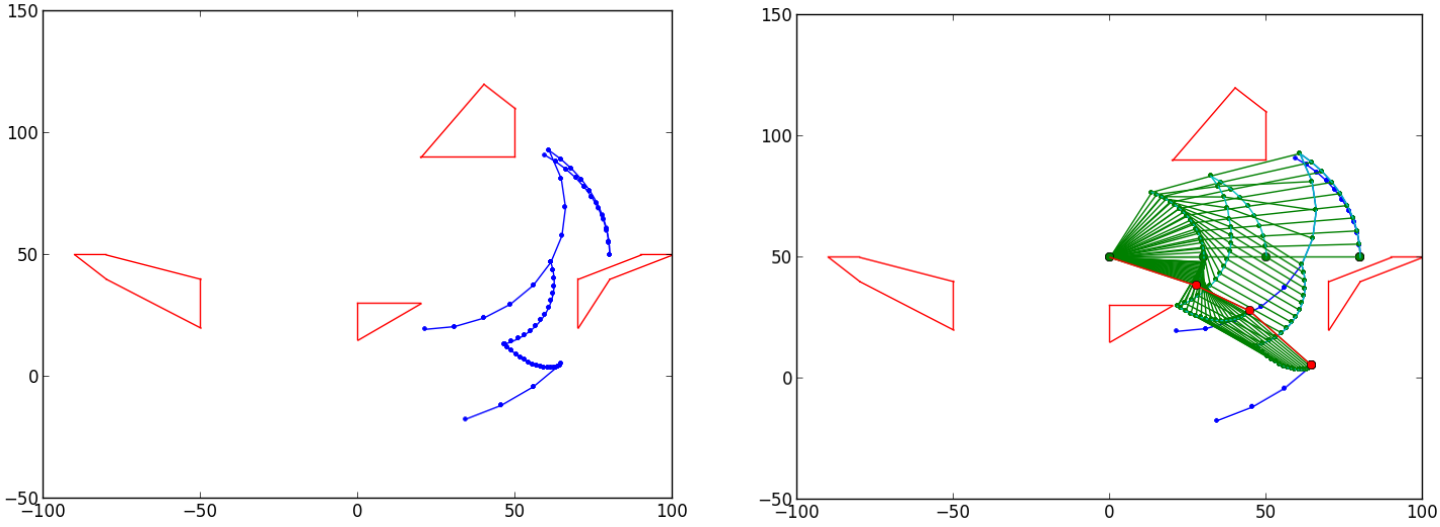

Figure 16 – Bidirectional RRT-Connect with 5% bias tree (left) and path (right) on env1

Probabilistic Roadmaps

1. As can be seen, the actual plans look somewhat similar to those found using vanilla RRT. The obvious difference is that each map completely fills its space and is re-usable. Note, however, that if there is a query that returns no path, the only thing you can do is re-build the map with more samples and/or a higher 'k' (number of nearest neighbors) value. Therefore, you do want the maps to be fairly dense.

   As can be seen in figures 21 and 22, as the number of samples increases, the amount of whitespace between nodes decreases. The graphs 'fill in'. This means that the area covered by the graph remains roughly the same (nearly all of the space is connected), but white space is filled and connected more thoroughly. Similarly, in figures 23 and 24 we see that as we increase the number of nearest neighbors we go from having many distinct graphs, to having one well-connected graph. As we increase the number of nearest neighbors, we increase the connectivity of the graph.

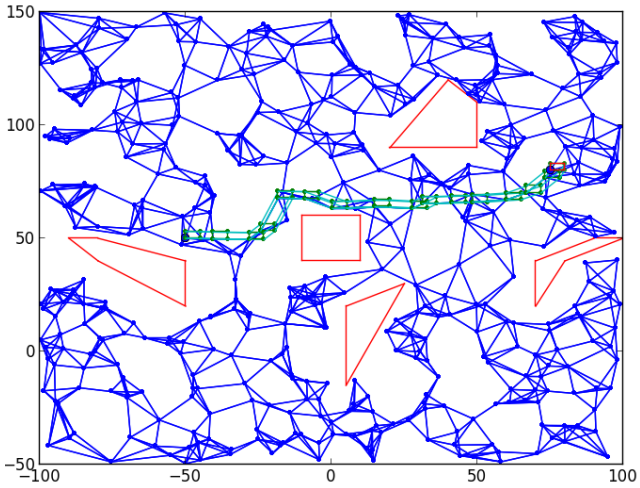

Figure 17 – PRM for env0 using 500 samples and 5 nearest neighbors

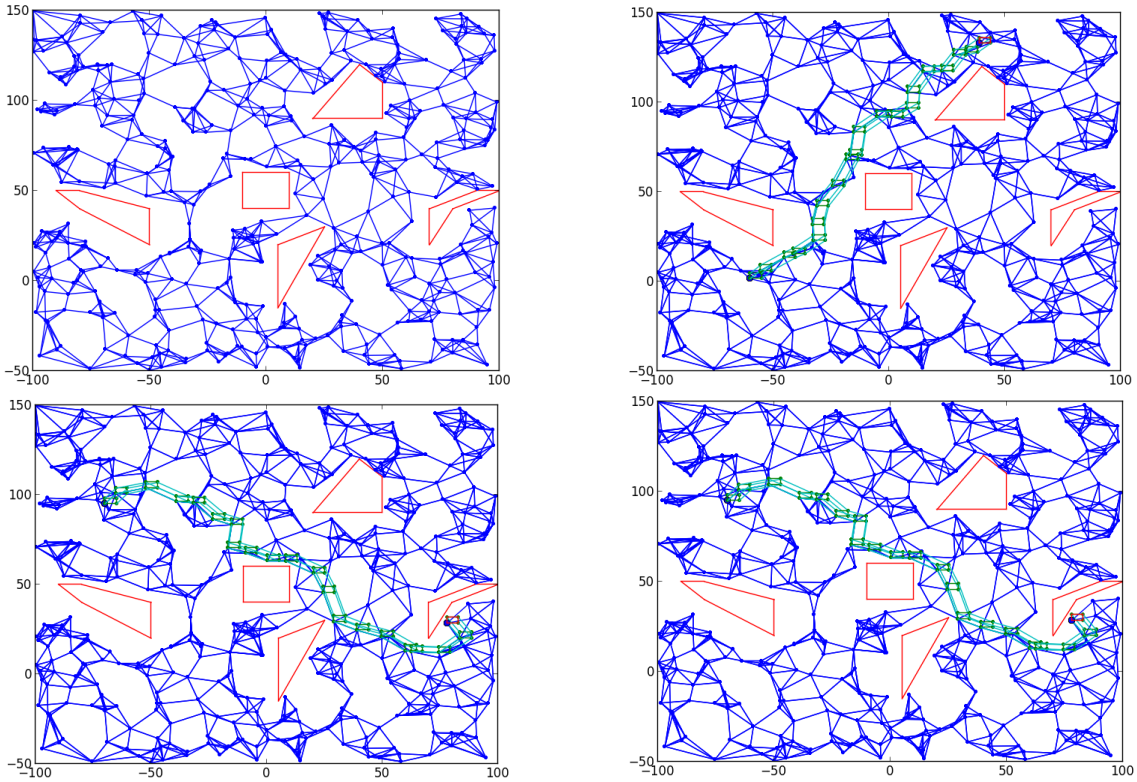

Figure 18 – Queries from map on Figure 17 using points from figures 1, 3, 4 and 5 respectively (left to right, top to bottom)

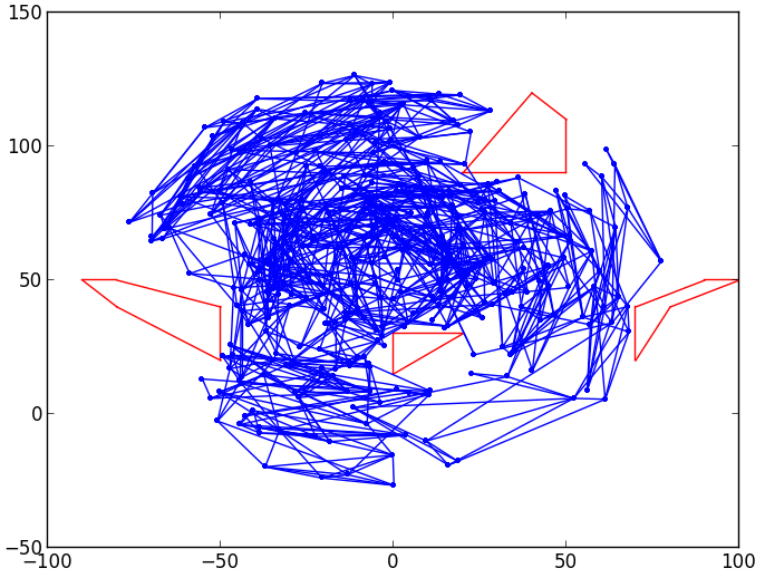

Figure 19 – PRM for env1 using 500 samples and 5 nearest neighbors

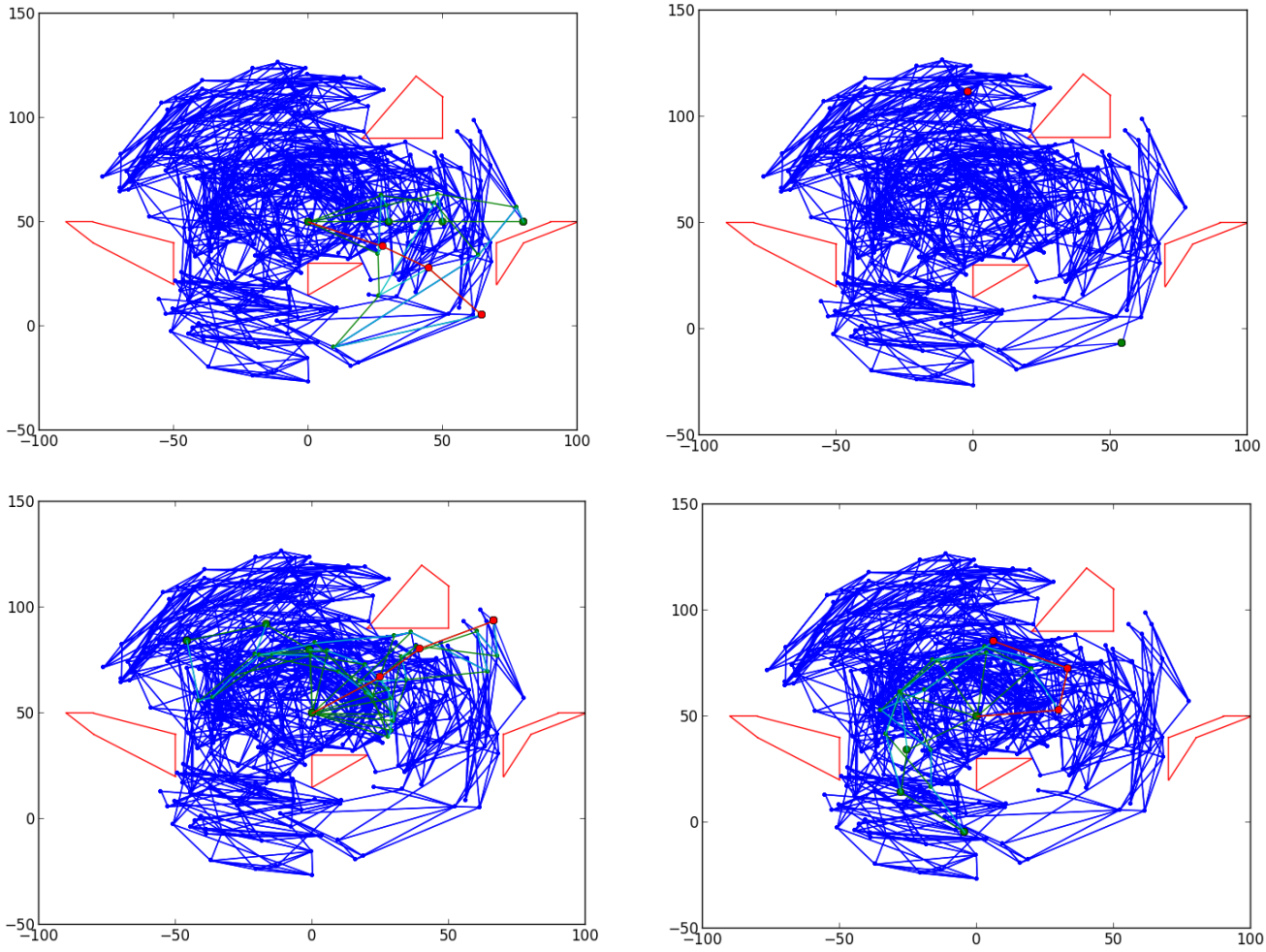

Figure 20 – Queries from map on Figure 19 using points from figures 2, 6, 7 and 8 respectively (left to right, top to bottom). Note that the top right could not find a path

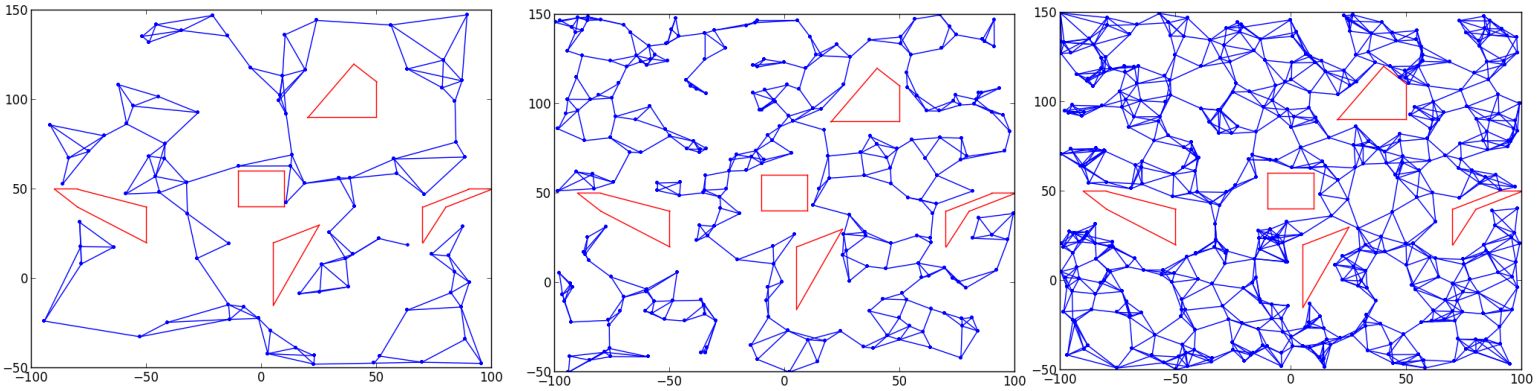

Figure 21 – PRM for env0 using 5 nearest neighbors and 100, 300 and 500 samples, respectively

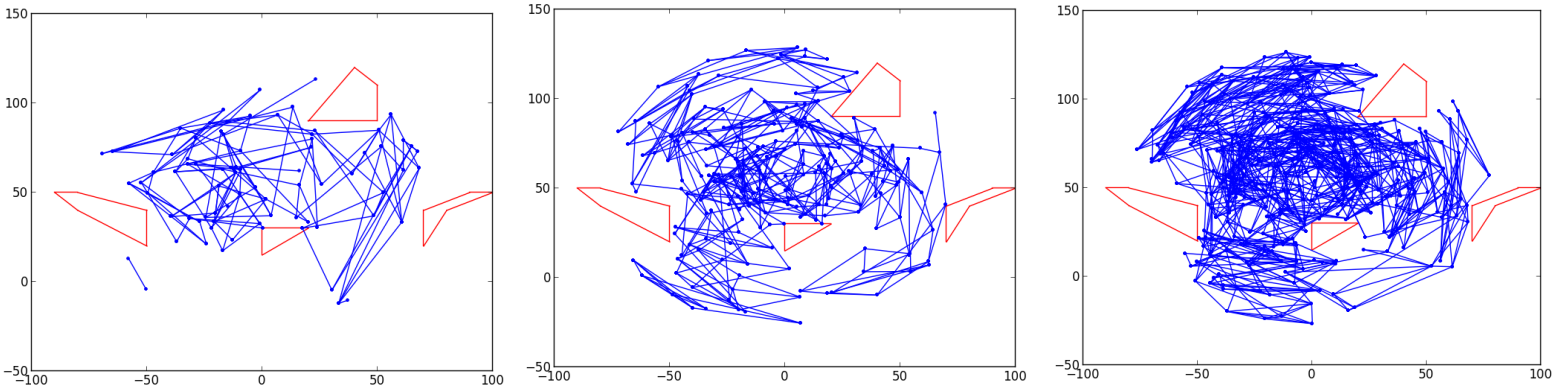

Figure 22 – PRM for env1 using 5 nearest neighbors and 100, 300 and 500 samples, respectively

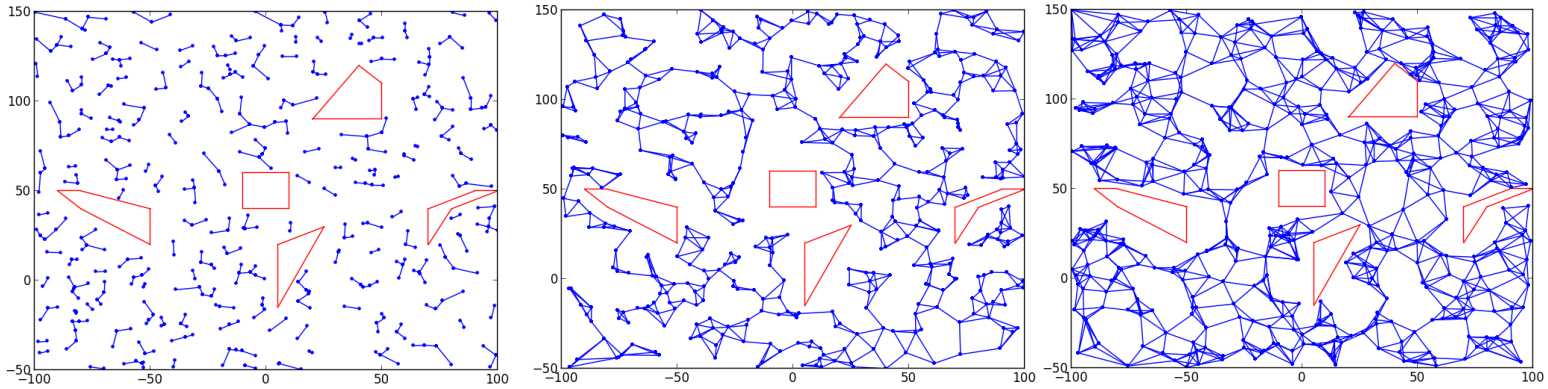

Figure 23 – PRM for env0 using 500 samples and 1, 3 and 5 nearest neighbors, respectively

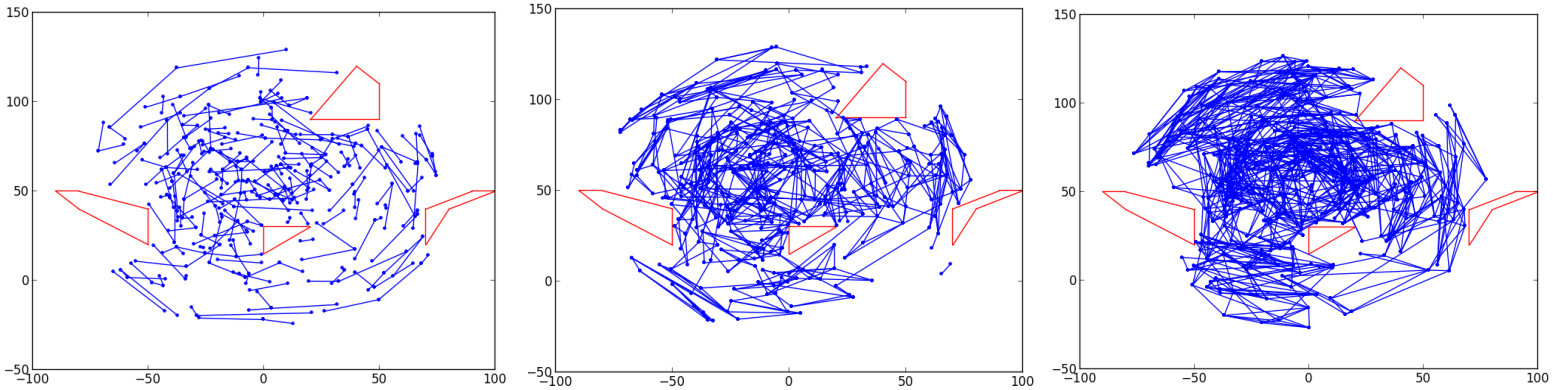

Figure 24 – PRM for env1 using 500 samples and 1, 3 and 5 nearest neighbors, respectively

2. As can be seen in figures 25 and 26, the resulting roadmaps looks very different. As is particularly prominent in figure 25, all nodes tend to be around edges. With Gaussian, the nodes conglomerate at edges or configurations that nearly lead to collision. While this works well for environment 1, it makes traversing environment 0 very difficult. Despite having significantly more nodes and a significantly higher nearest neighbors value than anything seen in question 1, the graph isn't even fully connected. In fact, as we increase the sample size we see a tendency toward n distinct graphs where n is the number of obstacles. This is because there will be many neighbors near the same edge as the node, and it's unlikely to try to connect with a node away from its edge as a result. While this isn't a big deal for environment 1, whose results look somewhat similar to problem 1's, this makes it much more difficult to create a path for environment 0.

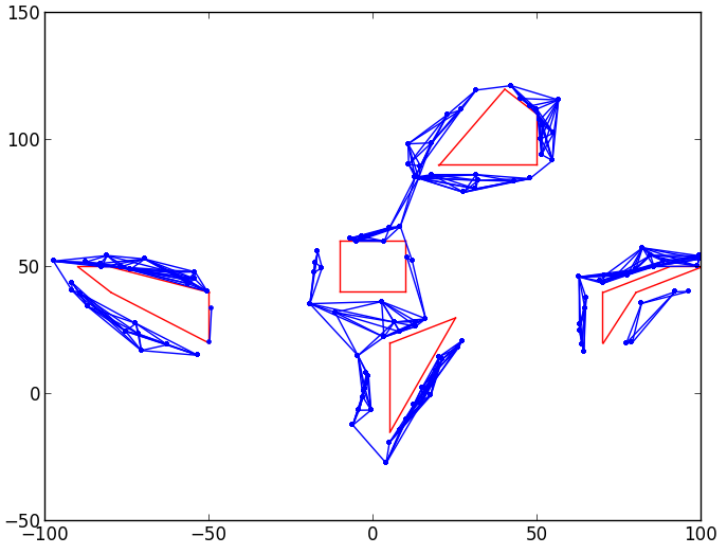

Figure 25 – Gaussian PRM for env0 using 3000 samples and 10 nearest neighbors

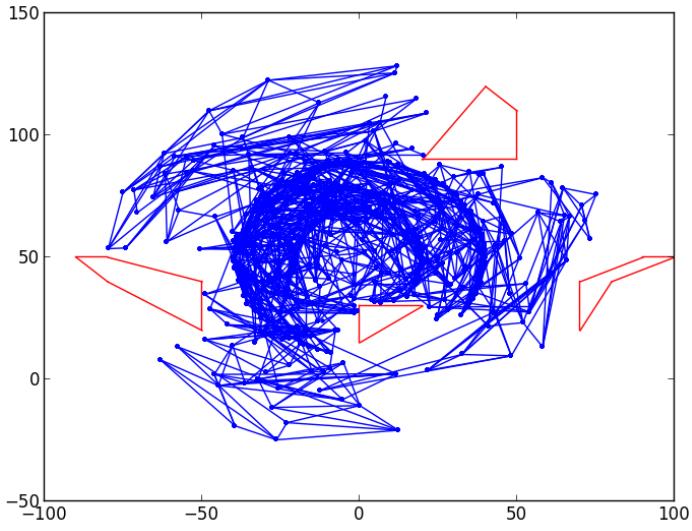

Figure 26 – Gaussian PRM for env1 using 1500 samples and 5 nearest neighbors

3. As can be seen in figures 27 through 30, using the RRT local planner actually appears to have done a worse job when compared to the PRM using a basic planner. This is largely due to the fact that we're only using 10 samples for the RRT, and thus are not making many connections that are very simple for the basic planner. We only use 10 samples for RRT-Connect because, as can be seen in tables 1 and 2, it requires somewhat ridiculous amounts of time to use it as a planner. Additionally, increasing the number of samples or nearest neighbors drastically increases the time required. At the same time, when we increase the number of samples using a basic planner, we only make more independent graphs that don't connect. This doesn't help at all for planning (Figure 29). While increasing the nearest neighbors does connect more distinct graphs, it doesn't do nearly as well as the basic planner (Figure 30). In the end, the RRT-Connect planner reduces performance of the PRM, particularly when considering time.

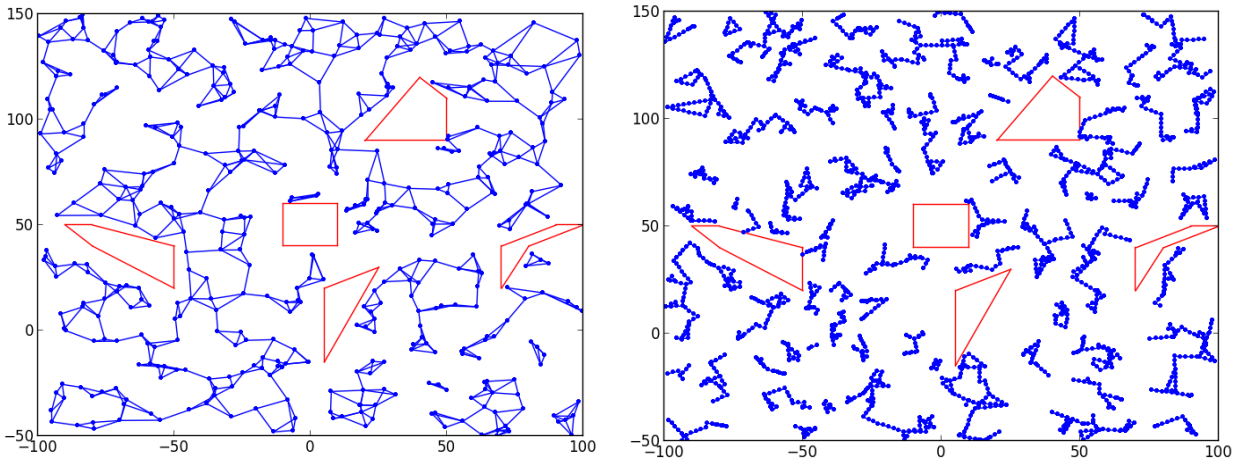

Figure 27 – PRM in env0 using basic local planner (left) and RRT-Connect local planner (right) for 500 samples and 3 nearest neighbors. RRT uses 10 samples and 10% bias

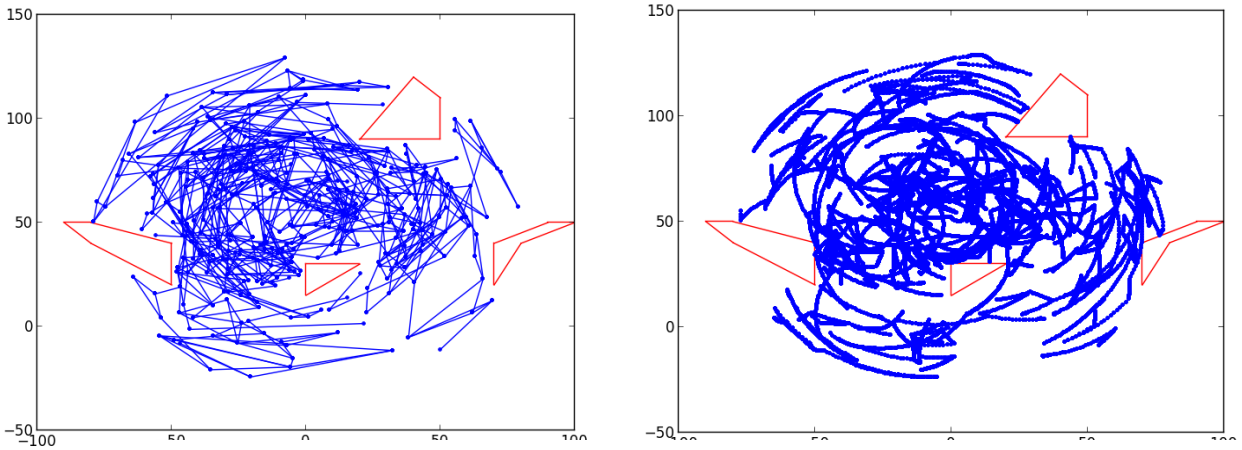

Figure 28 – PRM in env1 using basic local planner (left) and RRT-Connect local planner (right) for 500 samples and 3 nearest neighbors. RRT uses 10 samples and 10% bias

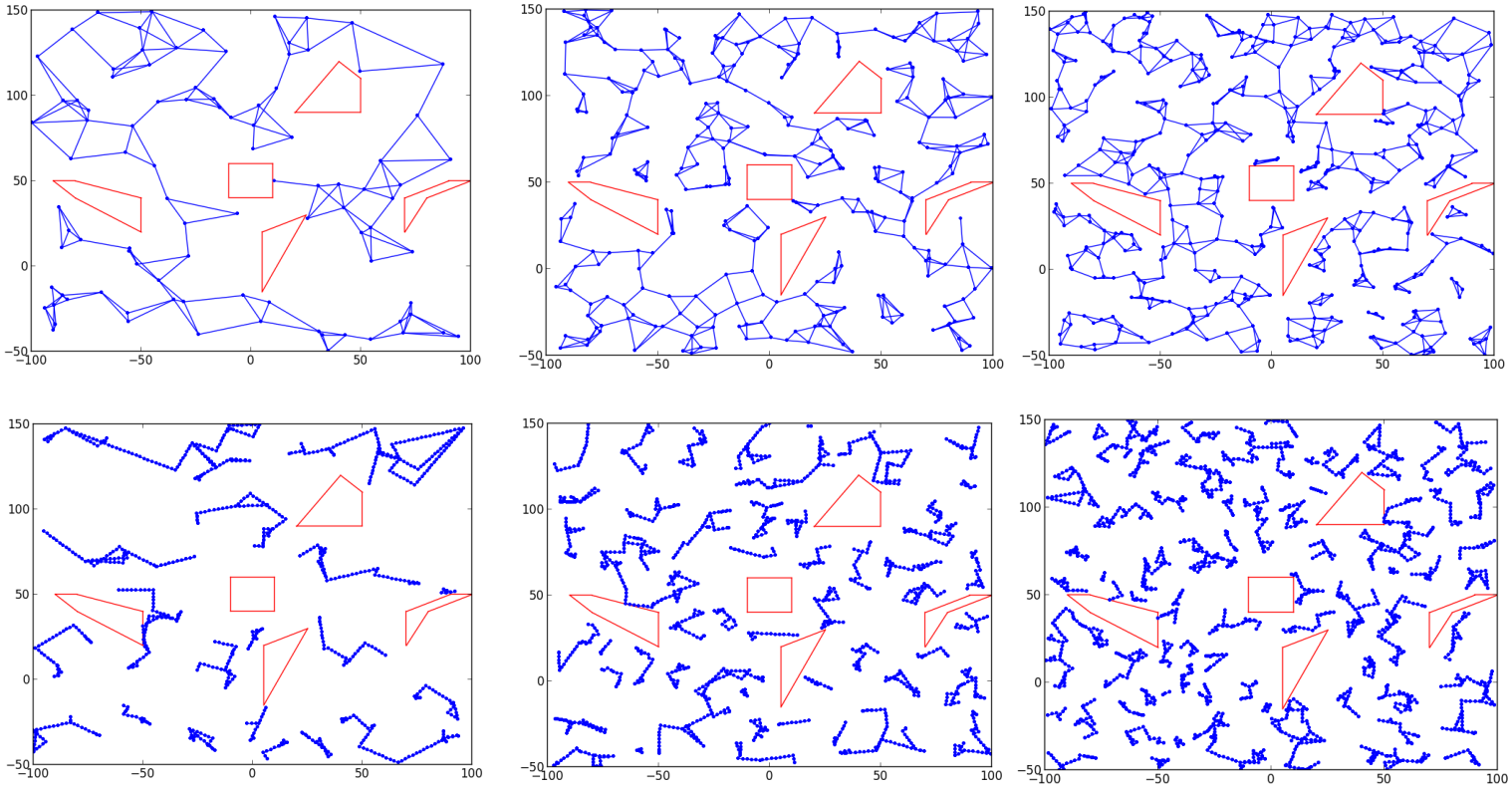

Figure 29 – PRM using basic planner (top) and RRT-Connect planner (bottom) with 100, 300 and 500 samples respectively. RRT uses 10 samples

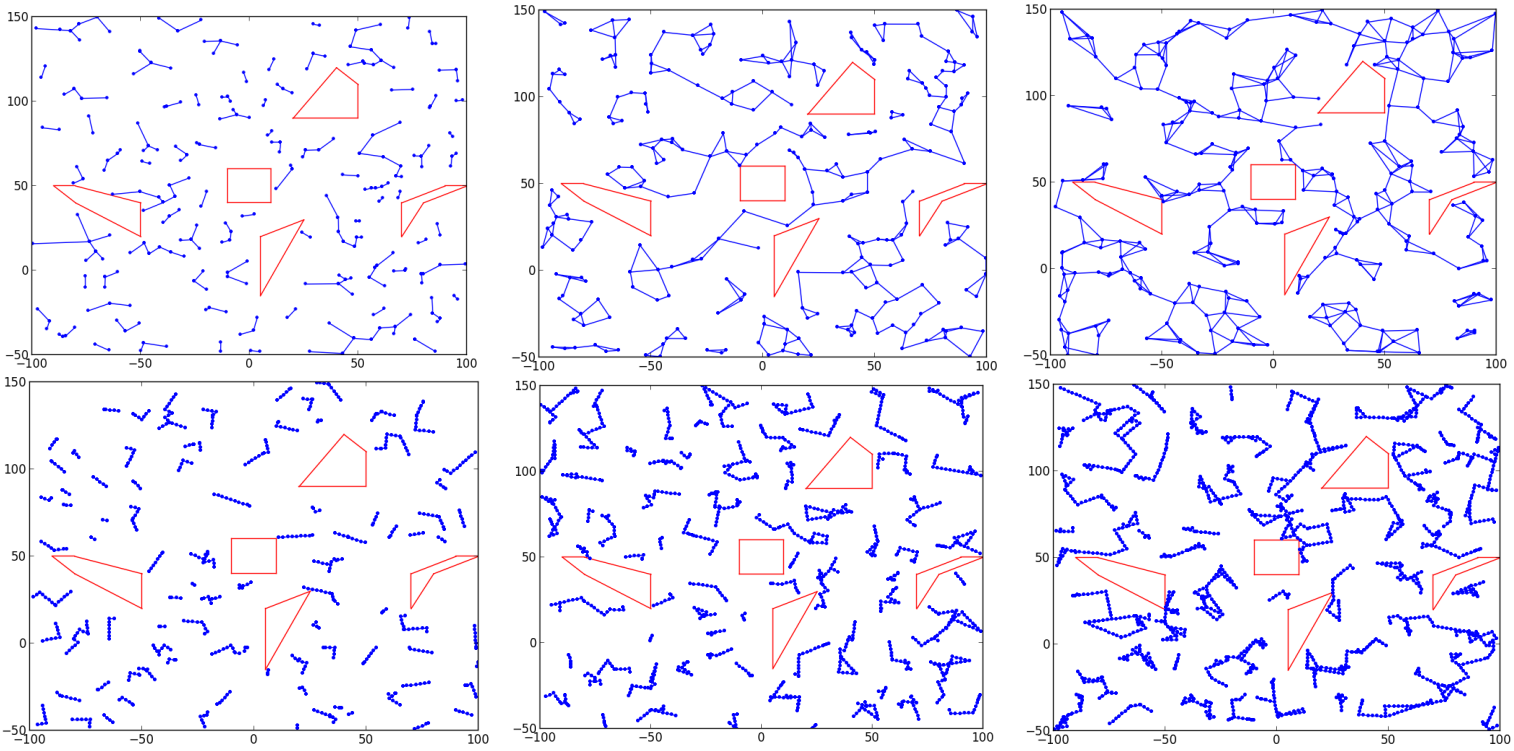

Figure 30 – PRM using basic planner (top) and RRT-Connect planner (bottom) with 1, 3 and 5 as nearest neighbor respectively. RRT uses 10 samples

Samples	Time Basic (s)	Time RRT (s)
100	1.5	29.5
300	3.8	83.5
500	5.5	142.9

Table 1 – Time required to make each graph in Figure 29

Nearest Neighbors	Time Basic (s)	Time RRT (s)
1	2.0	28.3
2	4.3	54.4
3	3.6	85.4

Table 2 – Time required to make each graph in Figure 30

Self Analysis

1. I think the hardest part was creating bi-directional RRT-Connect. I spent a fair amount of time fixing bugs that ended up dealing with passing references rather than values. Although doing bi-direction RRT-Connect isn't particularly difficult, those bugs made it the most time consuming part.
2. Implementing the basic PRM. I managed to get that done very quickly and without much trouble by following a similar outline to the one given for RRT.
3. I think building the PRM helped me the most (surprisingly), because it made me realize that I had misunderstood a basic part of building them. That portion which I had misunderstood was when to connect nodes.
4. Actually, I thought all of the assignment was helpful, none of it felt particularly tedious.
5. Overall, the homework took a very long time. Certainly more time than I was expecting. If there's any way to reduce that, it would likely be appreciated in the future. Perhaps getting rid of implementing Gaussian PRM, which was fairly simple and is likely understood by most simply through lecture.