Robot Motion Planning - Homework V

David Akhihiero

November 16, 2024

1 Introduction

In this assignment, I implemented the Simplified Probabilistic Roadmap (sPRM), the rapidly exploring random trees (RRT), the RRT*, and the informed RRT* algorithms for a sparse 2D environment and a dense 2D environment. All 4 algorithms plan paths in the roadmaps they generate from start to goal configurations. I then compared the preprocessing time, path-finding time, and the costs of the paths from each algorithm for the same start and goal configurations.

2 Rapidly-Exploring Random Trees * (RRT*)

I wrote an $RRTstar()$ MATLAB function that implements the RRT* algorithm shown in class. The graph was stored in a *PGraph* object, the function generated 800 nodes for both environments, the radius used for the goal region was 0.05, the radius for the $Near()$ function was 0.1, the $steer()$ function step size was 0.05 and the seed for the random number generator was 7. The function started the root of the tree from a q_{init} specified by a user. It also stored the parent and the cost of each node in the graph using the *PGraph* $setvdata()$ function and this data was used to recover the path from the start node to the node in the goal region with the smallest cost from q_{init} to q_{goal} (also user-specified). I used the breadth-first search (BFS) algorithm to propagate new node costs during the tree rewiring. The BFS algorithm starts from the node whose parent has just been changed and updates the costs of all the nodes neighboring it (excluding its parent). This propagates the cost updates down to the leaf nodes. I tested the algorithm in an obstacle-free environment with 3000 nodes for random start and goal configurations, and the graph's branches gradually converged to straight lines as shown in Figure 1. Example paths for a sparse and dense environment (with 800 nodes each) are shown in Figure 2

3 Informed Rapidly-Exploring Random Trees * (RRT*)

I also wrote an $InformedRRTstar()$ MATLAB function that implements the RRT* algorithm (Gammell et al., 2014) shown in class. I implemented an $informedSampleFree()$ function to sample configurations inside an ellipsoid after an initial solution has been found. The graph was stored in a *PGraph* object, and the function generated 800 nodes for both environments, the radius used for the goal region was 0.05, the radius for the

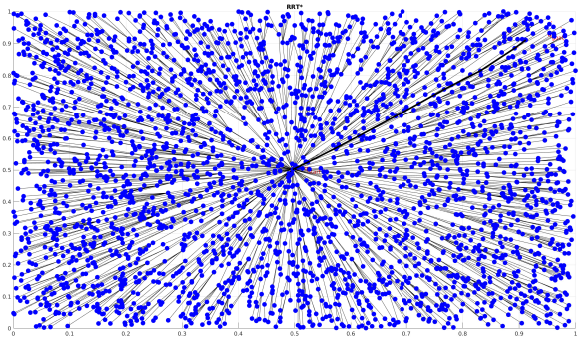

Figure 1: Paths in an RRT* graph from a start configuration (random) to a goal configuration (random) for an obstacle-free environment.

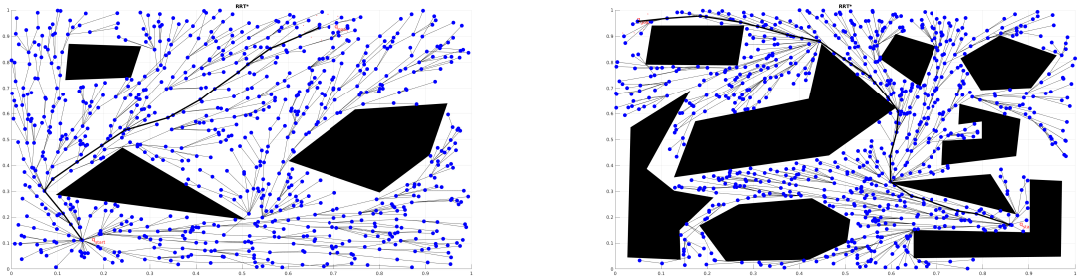

Figure 2: Paths in an RRT* graph from a start configuration (user-specified) to a goal configuration (user-specified) for the sparse and dense environments.

$Near()$ function was 0.1, the $steer()$ function step size was 0.05 and the seed for the random number generator was 7. The function started the tree's root from a q_{init} specified by a user. It also stored the parent and the cost of each node in the graph using the $PGraph\ setvdata()$ function and this data was used to recover the path from the start node to the node in the goal region with the smallest cost from q_{init} to q_{goal} (also user-specified). I used the breadth-first search (BFS) algorithm to propagate new node costs during the tree rewiring. The BFS algorithm starts from the node whose parent has just been changed and updates the costs of all the nodes neighboring it (excluding its parent). This propagates the cost updates down to the leaf nodes. I tested the algorithm in an obstacle-free environment with 1500 nodes for random start and goal configurations, and the samples were inside ellipsoidal areas as shown in Figure 3. Example paths for both environments are shown in Figure 4

4 Comparison between Algorithms

To compare these algorithms, I used each algorithm to plan a path from a start configuration ($[0.05, 0.03]$) to a goal configuration ($[0.9, 0.9]$) on a sparse environment and a dense environment. I measured the time each algorithm took for pre-processing and path-finding and the length of the path returned. To make the comparison fair, I used the same random generator seed for the algorithms (7) and gave each RRT algorithm 2 seconds to improve its initial solution with the same limit on the number of nodes each

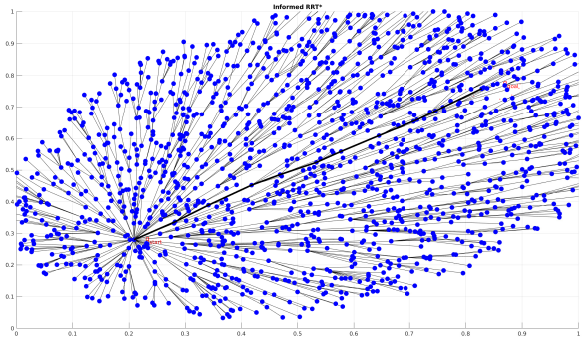

Figure 3: Paths in an informed RRT* graph from a start configuration (random) to a goal configuration (random) for an obstacle-free environment. Samples are in an ellipsoidal region that progressively becomes smaller.

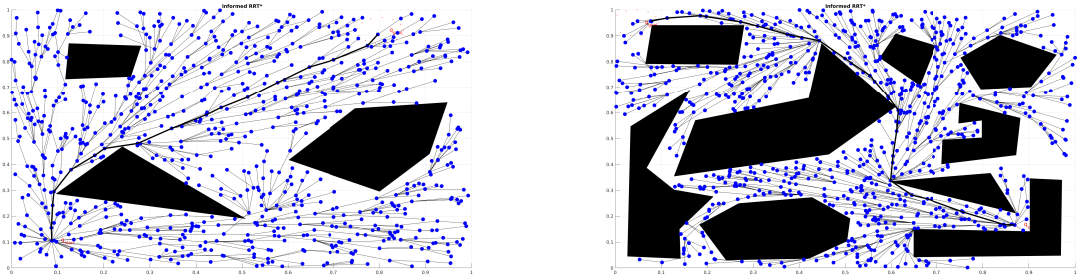

Figure 4: Paths in an informed RRT* graph from a start configuration (user-specified) to a goal configuration (user-specified) for the sparse and dense environments.

could use (800). The number of nodes in the sPRM, RRT, RRT*, and informed RRT* was 800. The results for the sparse and dense environments are shown in Tables 1 and 2 and the paths with these algorithms are shown in Figures 5 and 6.

Table 1: Path Length and Computation Time in Sparse Environment

Algorithm	Path Length	Pre-processing Time (s)	Path Finding Time (s)
sPRM	1.294723	20.816526	10.523598
RRT	1.704615	0	0.174399
RRT*	1.316388	0	3.192782
Informed RRT*	1.316388	0	0.598460

From the results, the sPRM algorithm computes the shortest path but takes the longest time (more than 30 seconds), RRT* and the informed RRT* compute similar paths that are close to that of the sPRM but the informed RRT* computed the path faster. Since the sPRM, the RRT*, and the informed RRT* algorithms are asymptotically optimal, they will eventually converge to the same solution but the results show that the informed RRT* algorithm will converge much faster. The RRT algorithm takes the shortest time to compute but it gave the worst solution even with the same number of nodes. RRT is real-time like the RRT* and the informed RRT* but it is not asymptotically optimal.

For the dense environment, the results show similar trends as in the sparse environment

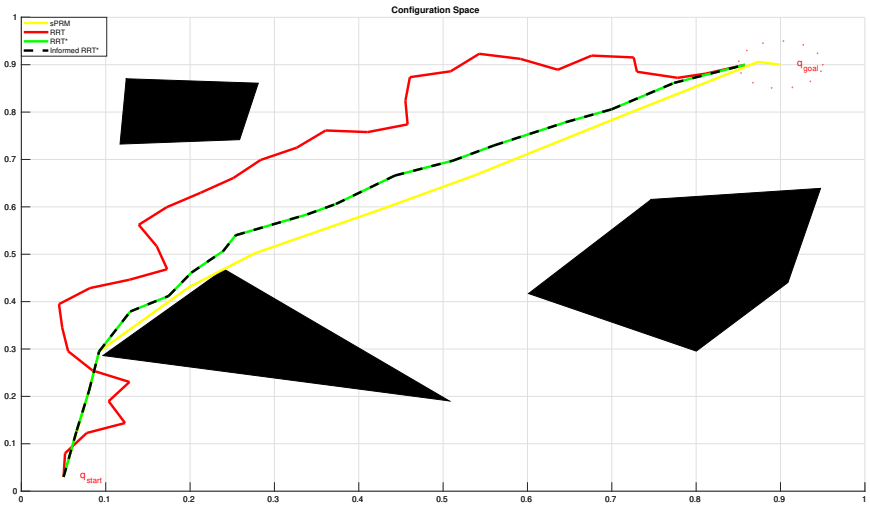

Figure 5: Paths in the configuration space from a start node to a goal for a sparse environment for all algorithms

Table 2: Path Length and Computation Time in Dense Environment

Algorithm	Path Length	Pre-processing Time (s)	Path Finding Time (s)
sPRM	1.319868	46.564160	5.919067
RRT	1.917256	0	0.228738
RRT*	1.365159	0	2.006252
Informed RRT*	1.320216	0	1.210103

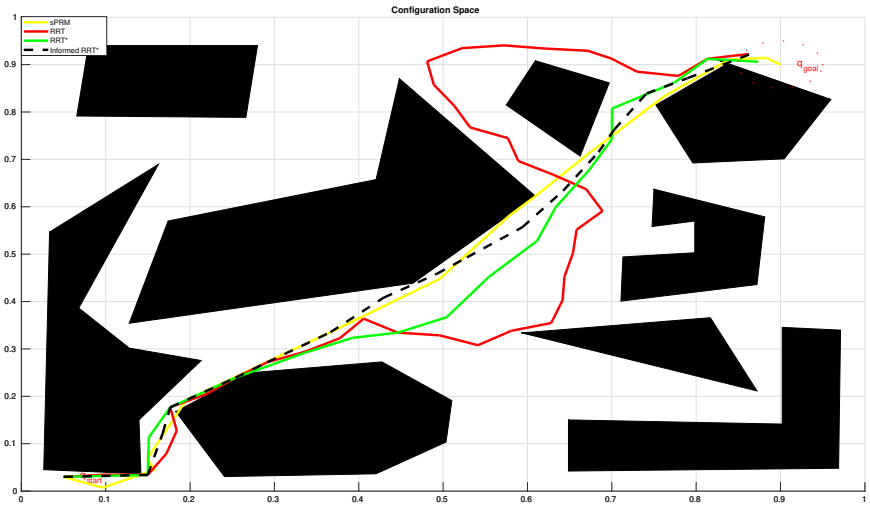

Figure 6: Paths in the configuration space from a start node to a goal for a dense environment for all algorithms

but this time, the RRT* was not able to converge to the same path as the informed RRT* algorithm within 2 seconds so the path is slightly worse. The sPRM still has

the best solution but the computation time is much longer now and the difference in solution between the sPRM and the informed RRT* is negligible. The path from the RRT algorithm is the longest but it was computed in the shortest time.

References

Gammell, J. D., Srinivasa, S. S., and Barfoot, T. D. (2014). Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ international conference on intelligent robots and systems*, pages 2997–3004. IEEE.