Sampling-based Motion Planning

Comparing RRT and RRT with different robot bodies*

Michael Dimitrov
Computer Science
CS 458 - Intro to Robotics
Purdue University - West Lafayette
dimitrm@purdue.edu

I. Introduction

This report summarizes results comparing the sampling-based motion planning algorithms RRT and RRT* under 3 robot bodies: Point-mass, Circular Rigid Body, Rectangular Rigid Body. Both algorithms work by randomly exploring the workspace, over a number of iterations, to build a trajectory tree, and then finding the minimum cost path from the start configuration to goal configuration. When building the tree the algorithms consider only the obstacle free-space and take into account constraints on the configuration space. We compare RRT and RRT* on 2 metrics Path Cost and Computation Time, to evaluate where each algorithm will apply best.

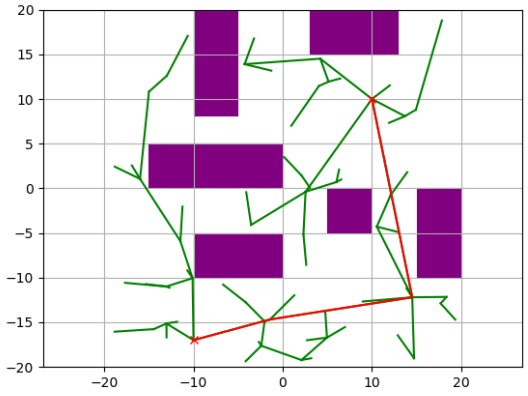
Note: When reading figures and results, note that both algorithms were always run with 100 iterations. Moving forward, if a path could not be found, it is because the tree did not explore enough to find a path to the goal configuration in those 100 iterations. This is due to collisions with the obstacles.

II. 2D Point-Mass

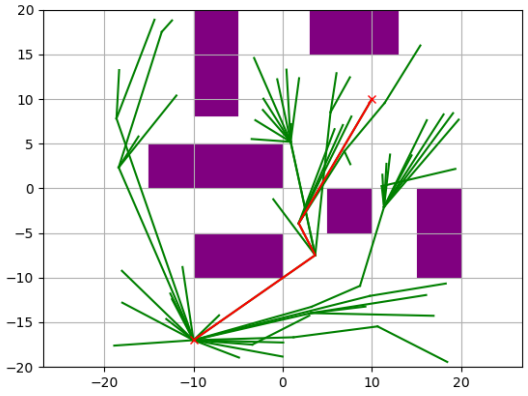
In this section I will present the results for RRT and RRT* on a 2D point-mass robot.

Fig. 1 shows the trees and final paths for each motion planner. As you can see RRT seems to be more sporadic and uniform in its exploration of the obstacle-free space, while RRT* is more discerning, focusing more on exploring areas close to where it has already explored, as evidenced by the tightly packed branches. This is largely due to the ”re-wiring” stage of RRT*. We can also glean from Fig. 1 that RRT* is better at approximating an optimal path, and in fact, given infinite iterations, would find an optimal path. RRT on the other hand is at the mercy of its more uniform exploration, and thus is not guaranteed to find the optimal path.

Table 1 supports this, by showing that the path cost for RRT* is on average 34% lower than for RRT (indicating shorter paths). However, to get this optimality guarantee, RRT* sacrifices in computation time, taking 24 times longer to complete on average. This can be attributed to the recursive update of partial path costs on all children that is required for the re-wiring step. The table also indicates that for the point mass robotic body, there was never an issue with finding a path, since the randomly sampled point mass has a lower probability to collide with the obstacles, than the circular or rectangular robot rigid bodies.



(a) RRT

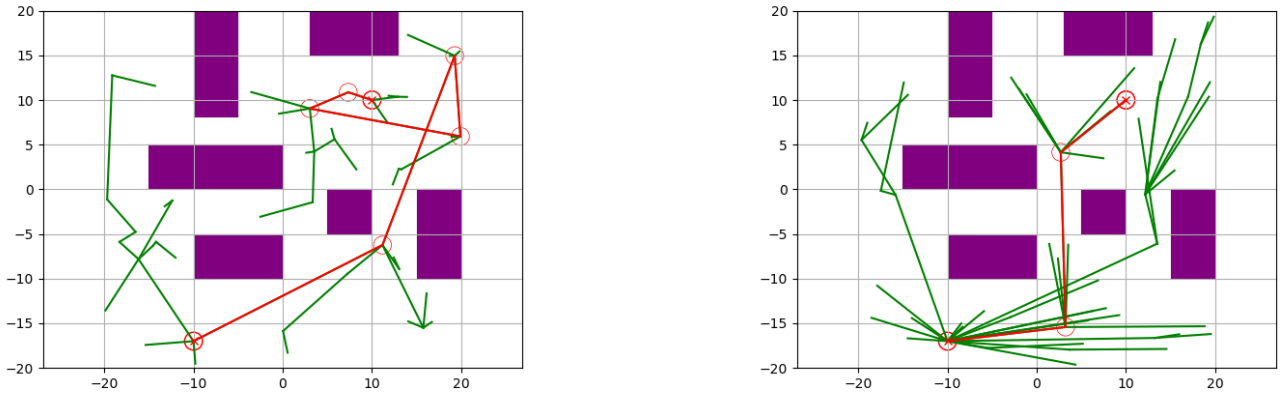


(b) RRT*

Fig. 1: Comparison of RRT and RRT* for the 2D point-mass. Green lines show the exploration tree, and the red line is the final path from the start to the goal. The start and goal configurations are each marked with an “×”.

TABLE I: 2D Point-Mass

	RRT	**RRT***
Avg. Path Cost	56.03	37.32
Avg. Computation Time (s)	0.112	2.81
Success Rate	30/30	30/30



(a) RRT

(b) RRT*

Fig. 2: Comparison of RRT and RRT* for the Circular Rigid Body. Green lines show the exploration tree, and the red line is the final path from the start to the goal. The start and goal configurations are each marked with an "×".

III. Circular Rigid Body

This section will give the results for RRT and RRT* on a circular rigid body.

Similar to the point-mass, it seems clear from Fig. 2 that RRT* is better at approaching an optimal path (even with as low as 100 iterations). This is again supported by the results in Table 2, which show a 20% lower average path cost for RRT*. In addition, the average computation time of RRT* was 18 times higher than RRT, mirroring the results of the point-mass.

One clear difference between the point-mass and the circular rigid body, is that the completion time for the latter was more than 2 times higher, likely due to the more complicated nature of checking for collisions of the circle. This collision check also led to higher failure rates in the circular rigid body, since it was harder to find a path through the obstacles that satisfied the unit radius circle constraint, than it was for the infinitesimally small point-mass.

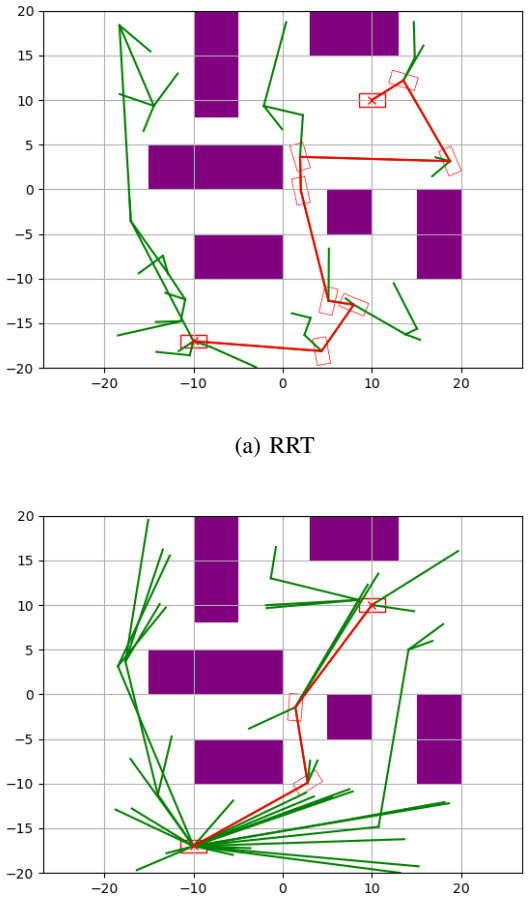
TABLE II: Circular Rigid Body

	RRT	**RRT***
Avg. Path Cost	51.37	41.07
Avg. Computation Time (s)	0.375	7.42
Success Rate	21/30	27/30

IV. Rectangular Rigid Body

This section will present results for RRT and RRT* on a rectangular rigid body.

Much like the point mass and circular rigid body, the rectangular one highlights how RRT* is better at approaching an optimal trajectory, as seen in Fig. 3.



(a) RRT

(b) RRT*

Fig. 3: Comparison of RRT and RRT* for the Rectangular Rigid Body. Green lines show the exploration tree, and the red line is the final path from the start to the goal. The start and goal configurations are each marked with an "×".

While the average path cost is very similar to the circular rigid body, the computation time is again more than double (and more than 5 times higher than the point-mass). This large jump in computation is caused by having to use the Separation of Axes Theorem to compute collisions, which is costly in the number of operations (must compute projections of all points onto the 4 unique normal axes). The performance hit would be even worse for more complex rigid bodies and obstacles.

We also see a dip in the success rate from the unit circle rigid body to the rectangle, but this is likely because the rectangle has a slightly larger volume, and thus is more likely to collide with an obstacle.

Finally, notice the systematic difference between the success rate of RRT and RRT* across the different robot bodies (with RRT* doing slightly better). This could be because RRT* tended to densify the tree near lower-cost corridors, which may have helped it find a valid path more often within the limited iteration budget for these specific runs.

TABLE III: Rectangular Rigid Body

	RRT	**RRT***
Avg. Path Cost	52.82	41.89
Avg. Computation Time (s)	0.61	15.72
Success Rate	18/30	21/30

V. Conclusion

Overall, RRT* is better at approximating an optimal trajectory, while RRT is faster. These methods can be combined so that a robotic system can react quickly (yet imprecisely) to stimuli with RRT, and use RRT* for more precise tasks. This is very similar to how humans act when we need to move our hands to pickup objects, where we move quickly to get closer to the object, and as we get closer, we become slower, more careful, and more precise. The experiments also showed that as the robot body becomes more complex (point $\rightarrow$ circle $\rightarrow$ rectangle), both algorithms generally require more computation and experience occasional failures at a fixed iteration budget, highlighting the impact of configuration-space dimension and collision geometry on sampling-based planning.