Path Planning With RRT*

J Michael Boren

***Abstract*—This project explores the implementation of RRT* path planning for miniature air vehicles (MAV), with the specific goal of integration into class projects for ECEn 674. It is shown that RRT* results in more optimized paths than RRT, at the cost of longer computation time. The effects of altering various algorithm parameters are also explored.**

I. Introduction

The Rapidly Exploring Random Tree (RRT) algorithm creates a path between two given configurations by filling the traversable space between them with a tree structure whose nodes are generated from a random process. Pseudocode for the basic algorithm is given in Figure 1. Though it does not produce optimal paths, RRT will find a path, if it exists, even in very complex or dense obstacle fields.

```
Input: Terrain map $T$, start configuration $\mathbf{p}_s$, end configuration $\mathbf{p}_e$
1. Initialize RRT graph $G = (V, E)$ as $V = \{\mathbf{p}_s\}$, $E = \emptyset$
2. while the end node $\mathbf{p}_e$ is not connected to $G$, i.e., $\mathbf{p}_e \notin V$ do
3.   $\mathbf{p} \leftarrow$ generateRandomConfiguration($T$)
4.   $\mathbf{v}^* \leftarrow$ findClosestConfiguration($\mathbf{p}, V$)
5.   $\mathbf{v}^+ \leftarrow$ planPath($\mathbf{v}^*, \mathbf{p}, D$)
6.   if existFeasiblePath($T, \mathbf{v}^*, \mathbf{v}^+$) then
7.     Update graph $G = (V, E)$ as $V \leftarrow V \cup \{\mathbf{v}^+\}, E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{v}^+)\}$
8.     Update edge costs as $C[(\mathbf{v}^*, \mathbf{v}^+)] \leftarrow$ pathLength($\mathbf{v}^*, \mathbf{v}^+$)
9.   end if
10.  if existFeasiblePath($T, \mathbf{v}^*, \mathbf{p}_e$) then
11.    Update graph $G = (V, E)$ as $V \leftarrow V \cup \{\mathbf{p}_e\}, E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{p}_e)\}$
12.    Update edge costs as $C[(\mathbf{v}^*, \mathbf{p}_e)] \leftarrow$ pathLength($\mathbf{v}^*, \mathbf{p}_e$)
13.  end if
14. end while
15. $W$ = findShortestPath($G, C$)
16. return $W$
```

Fig. 1. General RRT Algorithm

The RRT* algorithm is very similar, but with a few differences highlighted in Figure 2. Instead of simply selecting the nearest node as a parent, the chooseParent() function in RRT* compares all nodes within a fixed radius of the newly generated node and selects as parent the node with minimum path cost. Once a parent is chosen, the rewireGraph() function looks at nodes within the same radius and determines whether their path cost could be reduced by reassigning the new node as their parent. In this way, the RRT* algorithm continuously minimizes path cost. The result is convergence over time to the minimum possible path.

II. Methods

In this project, performance of the RRT* algorithm was analyzed in terms of path length and computation time. Simulations were run in Matlab with both empty terrain maps and maps with constant obstacle fields. Simulation parameters were chosen as:

```
Input: Terrain map $T$, start configuration $\mathbf{p}_s$, end configuration $\mathbf{p}_e$
1. Initialize RRT graph $G = (V, E)$ as $V = \{\mathbf{p}_s\}$, $E = \emptyset$
2. while the end node $\mathbf{p}_e$ is not connected to $G$, i.e., $\mathbf{p}_e \notin V$ do
3.   $\mathbf{p} \leftarrow$ generateRandomConfiguration($T$)
4.   $\mathbf{v}^* \leftarrow$ findClosestConfiguration($\mathbf{p}, V$)
5.   $\mathbf{v}^+ \leftarrow$ planPath($\mathbf{v}^*, \mathbf{p}, D$)
6.   if existFeasiblePath($T, \mathbf{v}^*, \mathbf{v}^+$) then
7.     chooseParent()
8.     Update graph $G = (V, E)$ as $V \leftarrow V \cup \{\mathbf{v}^+\}, E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{v}^+)\}$
9.     rewireGraph()
10.    Update edge costs as $C[(\mathbf{v}^*, \mathbf{v}^+)] \leftarrow$ pathLength($\mathbf{v}^*, \mathbf{v}^+$)
11.  end if
12.  if existFeasiblePath($T, \mathbf{v}^*, \mathbf{p}_e$) then
13.    Update graph $G = (V, E)$ as $V \leftarrow V \cup \{\mathbf{p}_e\}, E \leftarrow E \cup \{(\mathbf{v}^*, \mathbf{p}_e)\}$
14.    Update edge costs as $C[(\mathbf{v}^*, \mathbf{p}_e)] \leftarrow$ pathLength($\mathbf{v}^*, \mathbf{p}_e$)
15.  end if
16. end while
17. $W$ = findShortestPath($G, C$)
18. return $W$
```

Fig. 2. General RRT* Algorithm

1) number of iterations
2) maximum segment length between nodes
3) search radius around new nodes

Finally, a comparison is made between performance of RRT and RRT* in a constant obstacle field. Averages are taken over several simulations with both smoothed and unsmoothed versions of the RRT and RRT* algorithms.

III. Results

Simulations with increasing numbers of iterations in an obstacle free environment confirmed that the RRT* algorithm will converge to the minimum path with time. Results of these tests are shown in Figure 3. Note that the path length converges slowly toward its minimum (calculated to be 1132). However, the computation cost increases dramatically with greater numbers of iterations. Similar results were obtained when the algorithm was applied to a constant obstacle field. Example results of path planning through a fixed obstacle grid are shown in Figure 4. Here can also be seen the characteristic star patterns in the random tree that give this algorithm its name.

Results of simulations varying the maximum segment length between nodes are shown in Table I. Considering a segment length of 100 as standard, we see that halving the segment length gives a negligible reduction of path cost while increasing computation time by a factor of 1.79. Doubling the segment length, on the other hand, results in a significant path length reduction with comparative computation time increase (1.80 times). It is noteworthy that simulations with the standard parameters (seg length: 100, search radius: 100) given 5000 iterations and 1284.17 seconds could only achieve a path length of 3585.97 meters.

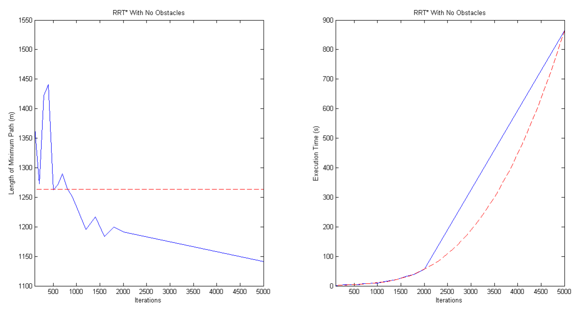

Fig. 3. RRT* with no obstacles. blue = data points, red = average/interpolation

Seg Length (m)	Iterations	Path (m)	Time (s)
50	3000	3727.37	509.02
100	**3000**	**3771.43**	**284.31**
200	3000	3116.29	514.14

TABLE I
VARYING SEGMENT LENGTH, ALL OTHER PARAMETERS CONSTANT.

Increasing the search radius around new nodes has a similar effect to that of increasing the segment length. Table II shows that, again considering a radius of 100 to be standard, doubling the search radius gives a significant reduction in path length while increasing computation time by a factor of 1.87. As noted previously with segment length, even using significantly more time and iterations, simulations using the standard parameters cannot match these results in terms of path length. It was observed during these simulations that it is inadvisable to extend the search radius far beyond the segment length. As the algorithm stands, this can result in the final path cutting corners and colliding with obstacles. Additional collision checking would be necessary to allow for this.

Radius (m)	Iterations	Path (m)	Time (s)
100	**3000**	**3771.43**	**284.31**
200	3000	3244.99	532.26

TABLE II
VARYING SEARCH RADIUS

A performance comparison between RRT and RRT* in a fixed obstacle field is found in Table III. Rather than performing a fixed number of iterations, these simulations were simply run until a path was found between start and end configurations. Note that with standard parameters, there is no discernible difference between the two algorithms. However, smoothing over each path gives a remarkable advantage to RRT*. Extending the segment length and search radius with smoothing gives slightly worse results in terms of path length than smoothing alone, but yields significant savings in calculation time.

IV. Conclusion

The RRT* algorithm will produce near optimal paths, but achieving those paths requires time. Parameters such as segment length and search radius can be extended to make each iteration of the algorithm more efficient. Optimizing these parameters can give RRT* a significant advantage over RRT even when the algorithms are terminated when the first viable solution is found. Continuous path minimization makes path smoothing functions much more effective on a tree produced by RRT* as compared to RRT.

	RRT* Path (m)	RRT* Time (s)	RRT Path (m)	RRT Time (s)
Standard	3982.3	38.14	3933.0	38.24
Smoothed	3268.9	49.59	3564.8	41.83
Extended	3327.9	26.10	3659.9	27.32

TABLE III
RRT vs RRT*

Further work may be done to find limiting values for number of iterations, segment length and search radius, i.e. at what point increasing these values no longer appreciatively contributes to the results obtained. Other methods of optimizing the algorithm might also be considered, such as biased sampling–forcing generated nodes closer to a known solution so that it converges to the optimum more quickly.

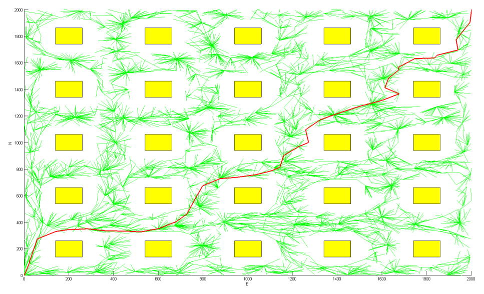

Fig. 4. RRT* path planning with fixed obstacle grid (5000 iterations)