A Comparison of RRT Connect and RRT Smart Path Planning for Higher Dimensional Robots*

Saandeep Sreerambatla

Department of Electrical Engineering, Case Western Reserve University, Cleveland, OH, USA
Email: *sxs2452[at]case.edu*

Abstract: ***Sampling based planning algorithms such as RRT Connect, RRT* and other sampling-based algorithms are extensively used for path planning of mobile robots. These algorithms are probabilistically complete and can easily be applied for higher dimensional complex problems. RRT* is an algorithm from the RRT family of algorithms which helps in faster convergence and this gives us the shortest path possible. This paper presents an analytical review of the two algorithms. Impact of different parameters on algorithm's performance is evaluated. Moreover, performance comparison of different criteria like run time and total number of nodes in the tree is performed through a simulation on a 6 DOF robot in an environment with minimal obstacles. For simplicity, obstacles are considered as circles in this experiment.***

Keywords: Path Planning, RRT Connect, RRT*, mobile robots, Comparison

1. Introduction

Path Planning for mobile robots has many applications in autonomous cars [1], Unmanned Aerial Vehicles (UAVs) [2], and industrial forklifts [3]. The goal of the path planning algorithms is to find a collision-free path from initial configuration to the goal configuration with optimal cost. Sampling Based Planning (SBP) algorithms have been used for path planning of mobile robots in recent years [4, 5]. SBP planning algorithms provide quick solutions for higher dimensional problems using random sampling in the search space [6,7]. Lavelle [8] proposed Rapidly exploring Random Tree (RRT) [8, 9], which is a well-known SBP algorithm. RRT supports dynamic environments and non-holonomic constraints for car-like robots [9] very well. Lavelle applied it successfully to problems comprising of up to twelve degrees of freedom with both holonomic and non-holonomic constraints. However, the path generated by RRT was not optimal. Karaman and Frazzoli [5] proposed a variation of RRT called RRT* with proven asymptotically optimal property. RRT* improved path quality by introducing major features of tree rewiring and best neighbor search. However, it obtained asymptotic optimality at the cost of execution time and slower path convergence rate. RRT and RRT* have numerous successful applications in robotics. Significant body of research has addressed the problem of optimal path planning using RRT* in recent years. These methodologies have gained tremendous success in solving single-query high dimensional complex problems. This paper presents a simulation-based experimental comparison of the aforementioned algorithms in an environment cluttered with obstacles. The effect of different parameters on the performance of these approaches is discussed. Further, limitations and future directions for improvement are also suggested. The next section describes the methodology of RRT and RRT* path planning approaches. Experimental results along with comparative analysis are presented in Section 3. Section 4 discusses future recommendations, followed by the conclusion in the last section.

2. Environment Overview

Environment Creation: Created a 6-DOF revolute robot with 4m length of links. The robot has provided a start configuration and goal configuration to reach. The environment also has multiple obstacles which the robot has to avoid in reaching the final goal configuration. The below figure explains the initial position of the robot in green and the final configuration in red, and the obstacles are also plotted in the graph.

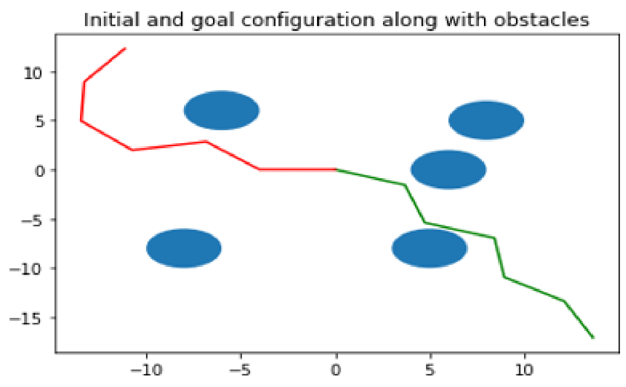

Figure 1: Initial and goal configuration along with obstacles

3. Algorithm Overview

Problem Statement: RRT Connect and RRT* operate in a configuration space with all set of possible transformations for a robot to operate. Let the given state space be denoted by a set Z R nN n, where n represents the dimension of the given search space. The area of the search space which is occupied by obstacles is represented by Z obs Z and region free from obstacles is represented by Z free Z Zobs = / . goal Z free z represents goal and init Z free z represents starting point. init z and goal z are provided to planner as inputs. The problem is to find a collision free path between initial init z and goal z states in Z free, in least possible time t R , with minimum path cost.

RRT Connect:

The RRT-Connect planner method is based on two ideas: the Connect heuristic that attempts to move over a longer distance, and the growth of RRTs from both qinit and qgoal. The Connect heuristic is a greedy function that can be considered as an alternative to the EXTEND function in basic RRT. Instead of attempting to extend an RRT by a single step, the Connect heuristic iterates the EXTEND step until q or an obstacle is reached. Below images show the RRT CONNECT PLANNER algorithm, which may be compared to the Basic BUILD RRT algorithm. Two trees, Ta and Tb, are maintained at all times until they become connected and a solution is found. In each iteration, one tree is extended, and an attempt is made to connect the nearest vertex of the other tree to the new vertex. Then, the roles are reversed by swapping the two trees. This causes both trees to explore Cfree, while trying to establish a connection between them. The growth of two RRTs was also proposed for kinodynamic planning; however, in each iteration, both trees were incrementally extended toward a random configuration. The current algorithm attempts to also grow the trees towards each other, which has been found to yield much better performance. The below Algorithm 1 is the basic RRT Connect algorithm. Algorithm 2 explains the initial tree and free random configuration. Algorithm 3 explains the target trying to connect to the new random configuration to which the initial tree is connected. Algorithm 4 is the planner algorithm which takes the q configuration and divides it into multiple steps so that the trees are able to connect.

Algorithm 1 RRT CONNECT:

```
Ta.initialize(qstart)
Tb.initialize(qgoal)
success = False

for i = 1 to MAXNODES do
  qrand = RANDOMCONFIG()
  [result, Ta, qtarget] = RRTEXTENDSINGLE(Ta, qrand, steplength)

  if (result == TRUE)
    [result2, Tb, qconnect] = RRTEXTENDMULTIPLE(Tb, qtarget, steplength)
    if (result2 == TRUE)
      success = True
      return (success, qconnect, Ta, Tb)
    end if
  end if
  SWAP(Ta, Tb)
end for
return (success)
```

Algorithm 2 RRT EXTEND SINGLE:

```
qtarget = []
qnear = FINDNEAREST(Ta, qrandom)
qint = LIMIT(qrandom, qnear, steplength)
result = LOCALPLANNER(qnear, qint, stepsize)

if (result == TRUE)
  Ta = ADDNODETOTREE(Ta, qint)
  qtarget = qint
end if

return (result, Ta, qtarget)
```

Algorithm 3 RRT EXTEND MULTIPLE:

```
qconnect = []
qnear = FINDNEAREST(Tb, qtarget)
qint = LIMIT(qtarget, qnear, steplength)
qlast = qnear
numsteps = CEIL(NORM(qtarget - qnear) / steplength)

for i = 1:numsteps do
  result = LOCALPLANNER(qint, qlast, stepsize)
  if (result == FALSE)
    return (result, Tb, qconnect)
  end if
  Tb = ADDNODETOTREE(Tb, qint)
  qconnect = qint
  if (i < numsteps)
    qlast = qint
    qint = LIMIT(qtarget, qint, steplength)
  end if
end for

return (result, Tb, qconnect)
```

Algorithm 4 LOCAL PLANNER:

```
deltaq = q2 - q1
numsteps = CEIL(NORM(deltaq) / steplength)
q = q1

for i = 1:numsteps do
  q = q + step
  collision = collision || CHECKCOLLISION(q, obstacles)
end for

success = !collision
return (success)
```

RRT*: RRT* inherits all the properties of RRT and works similar to RRT. However, it introduced two promising features called near neighbor search and rewiring tree operations. Near neighbor operations find the best parent node for the new node before its insertion in the tree. This process is performed within the area of a ball of radius defined by

$$K = \frac{logn(\frac{1}{d})}{n(\frac{1}{d})}$$

where d is the search space dimension and is the planning constant based on environment. Rewiring operation rebuilds the tree within this radius of area k to maintain the tree with minimal cost between tree connections as shown in Figure 2 [12]. Space exploration and improvement of path quality is shown in Figure 3. As the number of iterations increase, RRT* improves its path cost gradually due to its asymptotic quality, whereas RRT does not improves its jaggy and sub optimal path. The below algorithm explains RRT* implementation.

Algorithm 5 T= (V, E) → RRT*(zini)

```
T = (V, E) → RRT*( zinit)
T → InitializeTree();
T → InsertNode(Ø, zinit, T)
for i = 0 : n do
    zrand → Sample(i)
    znearest → Nearest(T, zrand)
    (znew, Unew) → Steer (znearest, zrand)
    if ObstacleFree(znew) then
        znear → Near(T, znew, |V|)
        zmin → ChooseParent(znear, znearest, znew)
        T → InsertNode(zmin, znew, T)
        T → Rewire(T, znear, zmin, znew)
    end if
end for
return T
```

4. Experiments and Simulated Results

In this section analysis of two algorithms RRT Connect and RRT* is presented. To simulate the experiment the experiments are conducted in MATLAB in the environment explained in the section 2, in a windows 10 PC. Experiments were carried out different number of times using RRT Connect and RRT*. The below figure explains the collision free path for RRT Connect in the environment.

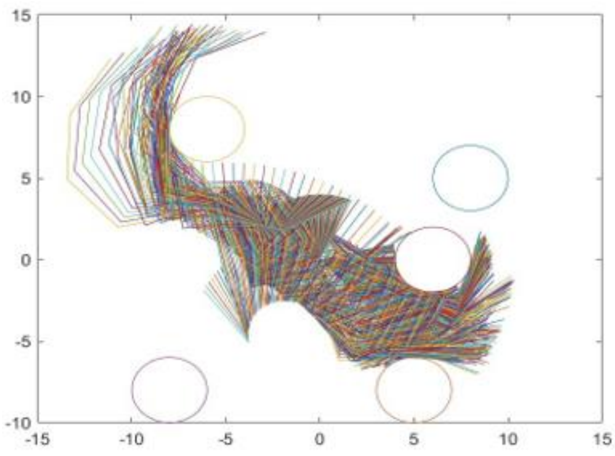

Figure 2: RRT-Connect collision free path

The below figure explains the collision free path from RRT* algorithm.

It is observed that RRT connect converges to the path faster than RRT*. RRT* needs more nodes and more time to create the final collision free path. It is also observed that RRT* though takes more time to converge it will always give the shortest path in the given environment.

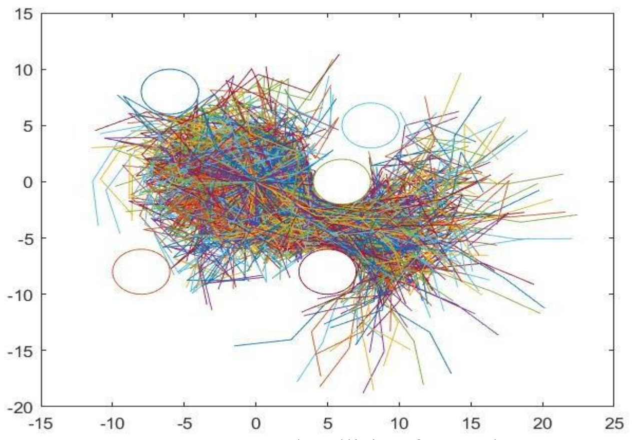

Figure 3: RRT* collision free path

5. Conclusion

We can conclude that both RRT Connect and RRT* work decently in a higher dimensional environment. If we need a faster convergence, we would need to use RRT Connect as it converges faster, and if we have enough processing and time and if we are interested in shortest path, we would go for RRT*

Acknowledgement

I would like to express my sincere gratitude to Prof. M. Cenk Cavusoglu for his guidance and patience.

References

[1] S. Thrun, W. Burgard, and D. Fox, “Probabilistic Robotics (Intelligent Robot and Autonomous Agents)”, MIT Press, 2005.
[2] X. Lan, and S. Di Cairano, "Continuous Curvature Path Planning for Autonomous Vehicle Maneuvers Using RRT*", presented at the European Control Conference (ECC), 2015.
[3] D. Alejo, J. A. Cobano, G. Heredia, J. R. Martínez De Dios, and A. Ollero, "Efficient Trajectory Planning for WSN Data Collection with Multiple UAVs", in Cooperative Robots and Sensor Networks, vol. 604, ed: Springer International Publishing, 2015, pp. 53-75.
[4] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT*", presented at the IEEE International Conference on Robotics and Automation (ICRA), 2011.
[5] S. M. Lavalle, Planning Algorithms, Cambridge University Press, 2006.
[6] S. Karaman, and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning", The International Journal of Robotics Research, vol. 30, pp. 846-894, 2011.
[7] M. Elbanhawi, and M. Simic, "Sampling-Based Robot Motion Planning: A Review Survey", IEEE Access, vol. 2, pp. 56-77, 2014.
[8] S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime Motion Planning using the RRT", presented at the IEEE International Conference on Robotics and Automation (ICRA), 2011.
[9] S. M. Lavalle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning", 1998.

[10] J. J. Kuffner, and S. M. Lavalle, "RRT-Connect: An Efficient Approach to Single-Query Path Planning", in Proceedings of IEEE International Conference on Robotics and Automation, 2000, pp. 1-7.

[11] J. Nasir et al., "RRT*-SMART: A Rapid Convergence Implementation of RRT*", International Journal of Advanced Robotic Systems, vol. 10, pp. 1-12, 2013.

[12] Y. K. Hwang, "Gross Motion Planning-A Survey", ACM Computing Surveys, vol. 24, pp. 219-291, 1992.

[13] J. W. Loeve, "Finding Time-Optimal Trajectories for the Resonating Arm using the RRT* Algorithm", Master of Science, Faculty Mechanical, Maritime and Materials Engineering, Delft University of Technology, Delft, 2012