A Novel RRT-Based Algorithm for Motion Planning in Dynamic Environments*

Abdul Mohammed and Usnik Chawla
MAGE
University of Maryland
College Park, Maryland
amohamm6@umd.edu
usnikchawla@gmail.com

***Abstract*—This work is an extension of efficient RRT*FN algorithm to dynamic scenarios. Specifically, we retain the useful parts of the tree after a dynamic obstacle invalidates the solution path. We then employ two greedy heuristics to repair the solution instead of running the whole motion planning process from scratch. This new algorithm is RRT*FN- Dynamic (RRT*FND). To compare our method to the state-of- the-art motion planners, RRT* and RRT*FN, we created an abstract map with dynamic goal and start points to implement the various algorithms and clocked their time intervals. The results of these experiments show that RRT*FND finds the solution path in shorter time in most of the cases and verifies the efficacy of it in dynamic settings.**

***Index Terms*— Path planning, motion planning, sampling-based methods, dynamic environments, rapidly-exploring random trees.**

I. Introduction

Motion planning aims to find a sequence of discrete robot configurations from the initial to the goal state complying with constraints imposed by the environment and internal dynamics of the system. Preliminary research on the topic included cell decomposition-based methods, artificial potential fields and visibility graphs. The motion-planning problem becomes a harder challenge in higher dimensional configuration spaces, prevalent in robotics. Due to the need for explicit representation of the obstacles in the configuration space, these planning algorithms are not suitable for high dimensional spaces with high number of obstacles.

Currently, sampling-based planners are the leading family of algorithms for higher dimensional motion planning. Sampling-based methods offer significant computational savings over complete motion-planning ones, since an explicit representation of the environmental constraints (i.e., obstacles) is not necessary. Most popular sampling-based planners are Probabilistic Roadmaps (PRMs) and Rapidly Exploring Random Trees (RRTs). Both RRT and PRM are probabilistically complete. They converge to the feasible solution as the number of iterations approaches infinity. Their primary difference is the way they connect the sampled points to generate a graph.

In real-world scenarios information about the environment is often incomplete, fragmented, and only updated iteratively over the course of time, occasionally with false information. The most straightforward approach of tackling robot motion-planning problem in such cases is to treat all obstacles as static and re-run the motion planner for static environments with the updated information This approach, depending on the planning algorithm, will provide a feasible or even an optimal solution.

In this paper we implemented the RRT*FND algorithm, in this algorithm once a dynamic obstacle invalidates a part of the tree, those nodes are removed. However, the nodes on the solution path not colliding with the obstacle are kept intact. Then our algorithm tries to reconnect the tree to one of the solution path nodes disconnected from the main tree by the obstacle.

II. RRT* and RRT*FN

RRT* is an asymptotically optimal extension of the RRT. Since RRT acts as the backbone of the RRT*, it is well-suited to describe it first. In RRT, a tree is initialized at the start state. A new state is sampled randomly at each iteration and the nearest neighbor of this new sample is found. If a newly sampled state is not reachable within a single step (due to system dynamics, motion constraints and obstacles) from the nearest neighbor, a new state is generated towards the new sample. Ultimately, RRT will explore the state space and provide a feasible solution. One of the shortcomings of RRT* is the increasing number of nodes needed to achieve optimality. From a computational perspective, this puts a burden to the memory needs in high dimensional spaces requiring increased number of iterations to explore a larger search space. It also slows down motion planning due to the nearest neighbor search involving a tree with more nodes.

RRT*FN addresses these issues by limiting the number of nodes in the tree. Specifically, RRT*FN leverages childless nodes as candidates for node removal. The underlying motivation for this scheme is the simplicity of removal of such nodes.

If a childless node disappears, there is no need to deal with orphaned set of nodes in the tree. Except the node at the goal state, none of the nodes in the solution path is a childless node. RRT*FN introduces a new parameter which is the maximum number of nodes M allowed in the tree. Until the maximum number of nodes in the tree is reached, RRT*FN operates identical to RRT*. However, now when the number of nodes exceeds M, RRT*FN starts to employ. In short, after the insertion of a new node to the tree, removes one childless node randomly to keep the number of the nodes fixed.

RRT*FN-Dynamic

A straightforward approach to the path planning problem with moving and/or unknown obstacles is to update the environment model and re-run a new motion planning problem assuming that all obstacles are static. By discarding the old tree this approach loses valuable information generated during the run-time of the algorithm. Lost information includes results of obstacle collision detection routines and exploration of the configuration space by the tree. Our aim is to save important information and reuse it as much as possible. Specifically, we augment the RRT*FN with two heuristics for motion planning with moving and/or random obstacles. Pseudocode of our approach is provided in Algorithm below.

Algorithm 5 $\tau = (V, E) \leftarrow$ RRT*FND(p_{init})

```
1: $\tau \leftarrow$ RRT*FN($p_{init}$) {Grow phase}
2: $p_{current} \leftarrow p_{init}$
3: $\sigma \leftarrow$ SolutionPath($\tau$, $p_{current}$)
4: InitMovement()
5: while $p_{current} \neq p_{goal}$ do
6:     $D \leftarrow$ UpdateObstacles()
7:     if DetectCollision($\sigma, p_{current}$) then
8:         StopMovement()
9:         $\tau \leftarrow$ SelectBranch($p_{current}$, $\tau$)
10:        $p_{separate} \leftarrow$ ValidPath($\sigma$)
11:        ReconnectFailed $\leftarrow$ true
12:        $P_{near} \leftarrow$ Near($\tau$, $p_{separate}$)
13:        for $p_{near} \in P_{near}$ do
14:            if ObstacleFree($p_{near}$, $p_{separate}$) then
15:                $\tau \leftarrow$ Reconnect($p_{near}, p_{separate}, \tau$)
16:                ReconnectFailed $\leftarrow$ false
17:                break
18:            end if
19:        end for
20:        if ReconnectFailed = true then
21:            $\tau \leftarrow$ Regrow($\tau$, $p_{separate}$, SetBias($\sigma_{separate}$))
22:        end if
23:        $\sigma \leftarrow$ SolutionPath($\tau$, $p_{current}$)
24:        ResumeMovement()
25:    end if
26:    $p_{current} \leftarrow$ NextNode($\sigma$)
27: end while
```

The code implemented in this project follows the main logic of the pseudocode given and hence retains the branch information.

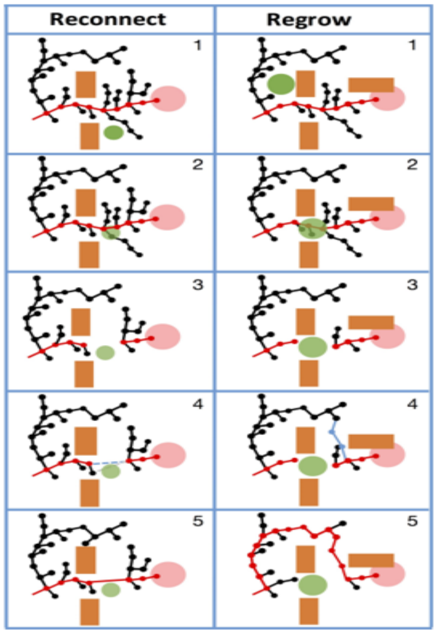

Figure shows the reconnect and regrow strategies for tree repair.

III. Experimentation

The main idea was to experiment with different dynamic obstacles in a defined map with the mobile robot moving towards to the goal position. The code uses a tolerance of 0.5mm which is used to avoid contact with the obstacle. The radius of the mobile robot is set to 20mm. The observations are recorded at the start coordinates of [-12, -12] and the goal location of [12, 12]. The borders are well defined, and the visualization is done using matplotlib and OpenCV libraries. The complex part was to write the code for branch retaining and using that information to include various algorithms.

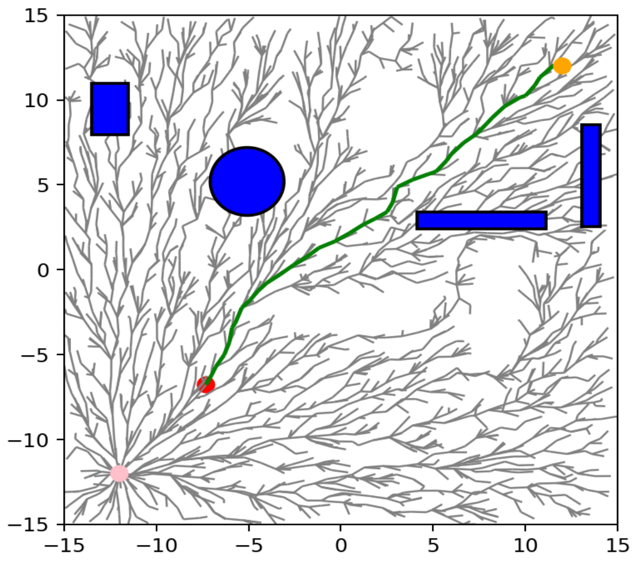

Figure shows our simulation experiment and the branching structure created. This is a still from the simulation as the obstacles are dynamic.

IV. Results

For the given test coordinates the above results are obtained and presented.

Algorithm	Time	Total cost to goal
RRT*	49.62s	50.19
RRT*-FN	40.40	50.24
RRT*-FND	27.18	43.87

From the above table it can be clearly seen that our RRT based FND algorithm is the better and much more efficient choice in contrast to other algorithms.

References

[1] T. Lozano-Perez *et al.*, "Spatial planning: A configuration space approach," *IEEE Transactions on Computers*, vol. 32, no. 2, pp. 108–120, 1983.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *The International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[3] T. Lozano-Pe´rez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[5] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," 2000.

[6] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[7] O. Adiyatov and H. A. Varol, "Rapidly-exploring random tree-based memory efficient motion planning," in *Proc. of IEEE International Conference on Mechatronics and Automation (ICMA)*, 2013, pp. 354–359.

[8] M. Svenstrup, T. Bak, and H. J. Andersen, "Trajectory planning for robots in dynamic human environments," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 4293–4298.

[9] M. Kallman and M. Mataric, "Motion planning using dynamic roadmaps," in *Proc. of IEEE International Conference on Robotics and Automation*, vol. 5, 2004, pp. 4399–4404.

[10] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," DTIC Document, Tech. Rep., 1993.

[11] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[12] J. Van Den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. of IEEE International Conference on Robotics and Automation*, 2006, pp. 2366–2371.

[13] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proc. of IEEE International Conference on Robotics and Automation*, 2006, pp. 1243–1248.