Sampling based mobile robot path planning using RRT* for dynamic obstacle avoidance

Balaji Selvakumar
UID 118545745
Robotics Engineering,
A. James Clark School of Engineering
University of Maryland,
College Park, USA
balase22@umd.edu

Saketh Narayan Banagiri
UID 118548814
Robotics Engineering ,
James Clark School of Engineering
University of Maryland,
College Park, USA
sbngr@umd.edu

***Abstract*—Sampling-based motion planning has evolved into a viable framework for addressing complex robotic motion planning problems. An RRT* algorithm for obstacle avoidance and re-planning in dynamic environments is applied in this study. The Random Tree Star (RRT*) idea is used in this method. The algorithm inherits probabilistic completeness and asymptotic optimality from RRT* to refine the existing paths continually by sampling the search-graph obtained from the grid search process. It can be used to increase the effectiveness of path planning in rapidly changing contexts by applying it to continuous cost spaces.**

Keywords—RRT, RRT*, motion planning,dynamic environment, sampling based method

I. Introduction

Path planning in dynamic environments is a demanding problem encountered in many robotic tasks and computer games [Rastgooet al. 2014; Sud et al. 2008]. Real-time path planning algorithms are used to react to the changes in the environment as well as to constantly look for a better path to the goal point. This paper presents a new path planning method which helps the robot to reach the target from the beginning point without colliding fixed and mobile obstacles existing in search space. According to LaVall [1], the path planning algorithms are mainly divided into two groups: 1) Compound Path planning Algorithms; and 2) Sampling-based path planning algorithms. Compound path planning algorithms are complete and they have a complicated implementation. An algorithm is considered complete if for any input it correctly reports whether there is a solution in a finite amount of time. If a solution exists, it must return one in finite time [1]. Sampling-based path planning algorithms. is not complete but it has a simple implementation. Many sampling-based approaches are based on random sampling, which is dense with “probability complete”.

A. <u>*RRT and RRT* - Comparison*</u>

The RRT algorithm is based on random sampling of a configuration space. The name itself gives a good idea of what it does. The sampled configurations are connected to a tree structure in which the resulting path can be found. The algorithm can be divided into three main parts: selection of a vertex for expansion, expansion and terminating condition. [3] An RRT algorithm can efficiently find a valid path for a static environment. Rapidly exploring random trees (RRTs) quickly expands a search tree in an environment. They are efficient for single-query tasks in a continuous environment, i.e. when the goal point is fixed But the performance of the RRT algorithm can be poor in environments containing narrow passages and also the path might not be optimal or smooth. Whereas RRT* gives a smoother path because of the rewiring and finding the best parent step. The down side to RRT* is that it takes much longer time compared to RRT to find the path from start to goal location .

B. <u>*Maintaining the Integrity of the Specifications*</u>

The different real time variants of RRT path planning either regrow the whole tree with guidance of the previous iteration for a limited time [Bruce and Veloso 2002] or prune infeasible branches of the tree after changing its root [Luders et al. 2010]. We tried to retain the whole tree in the environment and rewire the nodes of the tree based on the location of the tree root and changes in the dynamic obstacles , but due to time limitations we weren’t able to achieve that goal. Whenever an obstacle is detected , the algorithm starts to explore and re-plan and rewires the tree from the current location to the goal. The main objective of this paper is to predict the collision if the obstacle is moving towards / in the planned path to the goal, and replan the path to avoid collision of the robot with the obstacle.

II. PATH PLANNING

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. In each iteration,a new state is sampled randomly 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 by the Steer routine such that a single step reachability condition is fulfilled. This new single-step reachable state will be added to the tree if there is an obstacle free path segment from its nearest neighbor. Ultimately, RRT will explore the state space and provide a feasible solution.

Rapidly-exploring Random Tree Star (RRT*) is very similar to RRT. The path found using RRT* is shorter and smoother when compared to RRT. The reason for this is the additional two steps which are performed after the first few steps of RRT.

a) Parent node: Before adding Xnew to the tree, an additional step of finding the best parent is performed. The next step is to connect Xnew to the tree, in order to do this a vicinity is chosen around Xnew and checked for explored nodes. The node in this region with least cost to come is selected as the parent. The cost is the sum of all the distance from the root to the parent node.

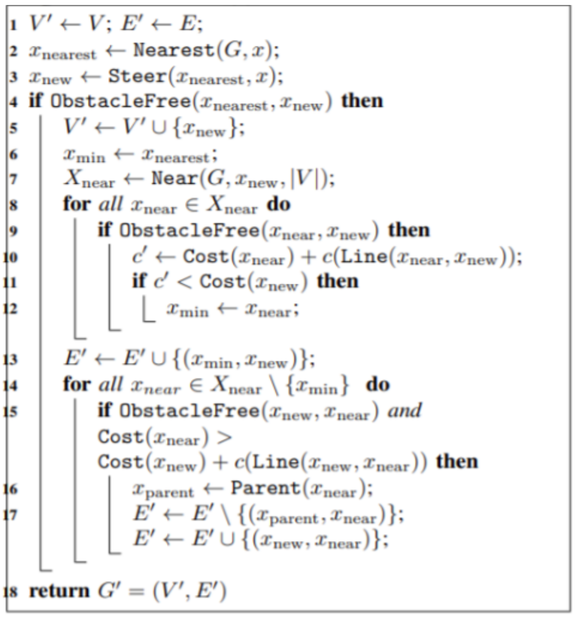
b) Rewire: Once the new node has been added to the tree, the nodes in the vicinity of the new node are checked to see if the cost to come for any of these nodes can be reduced by passing through Xnew. If so then the nodes are rewired and this makes the final path more optimal compared to RRT*.

A. The RRT Algorithm*

Given two points x, x0 ∈ Xfree, recall that Line(x, x0) : [0, s] → Xfree denotes the path defined by σ(τ) = τx + (s − τ)x 0 for all τ ∈ [0, s], where s = kx 0 − xk. Given a tree G = (V, E) and a vertex v ∈ V , let Parent be a function that maps v to the unique vertex v 0 ∈ V such that (v 0 , v) ∈ E. The RRT* algorithm differs from the RRT and the RRG algorithms only in the way that it handles the Extend procedure. The body of the RRT* algorithm is presented in Fig 1. In the description of the RRT* algorithm, the cost of the unique path from xinit to a vertex v ∈ V is denoted by Cost(v). Initially, Cost(xinit) is set to zero. Similarly to the RRT and RRG, the RRT* algorithm first extends the nearest neighbor towards the sample (Lines 2-3). However, it connects the new vertex, xnew, to the vertex that incurs the minimum accumulated cost up until xnew and lies within the set Xnear of vertices returned by the Near procedure (Lines 6-13). RRT* also extends the new vertex to the vertices in Xnear in order to “rewire” the vertices that can be accessed through xnew with smaller cost (Lines 14-17).

B. Convergence to a Feasible Solution

For all i ∈ N, let V RRT* i and E RRT* i denote the set of vertices and the set of edges of the graph maintained by the RRT* algorithm, at the end of iteration i. Let V RRT* 0 (ω) = {xinit} and E RRT* 0 (ω) = ∅ for all ω ∈ Ω. The following lemma is the equivalent of Lemma 3. Lemma 20 For all i ∈ N and all ω ∈ Ω, V RRT* i (ω) = V RRG i (ω), and E RRT* i (ω) ⊆ ERRG i (ω).



```
1  $V' \leftarrow V$; $E' \leftarrow E$;
2  $x_{\text{nearest}} \leftarrow$ Nearest($G, x$);
3  $x_{\text{new}} \leftarrow$ Steer($x_{\text{nearest}}, x$);
4  if ObstacleFree($x_{\text{nearest}}, x_{\text{new}}$) then
5      $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
6      $x_{\text{min}} \leftarrow x_{\text{nearest}}$;
7      $X_{\text{near}} \leftarrow$ Near($G, x_{\text{new}}, |V|$);
8      for all $x_{\text{near}} \in X_{\text{near}}$ do
9          if ObstacleFree($x_{\text{near}}, x_{\text{new}}$) then
10             $c' \leftarrow$ Cost($x_{\text{near}}$) $+ c$(Line($x_{\text{near}}, x_{\text{new}}$));
11             if $c' <$ Cost($x_{\text{new}}$) then
12                 $x_{\text{min}} \leftarrow x_{\text{near}}$;
13     $E' \leftarrow E' \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
14     for all $x_{\text{near}} \in X_{\text{near}} \setminus \{x_{\text{min}}\}$ do
15         if ObstacleFree($x_{\text{new}}, x_{\text{near}}$) and
           Cost($x_{\text{near}}$) $>$
           Cost($x_{\text{new}}$) $+ c$(Line($x_{\text{new}}, x_{\text{near}}$)) then
16             $x_{\text{parent}} \leftarrow$ Parent($x_{\text{near}}$);
17             $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
               $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;
18 return $G' = (V', E')$
```

fig1: pseudo-code for RRT algorithm*

*C. Shortcomings of RRT**

One of the shortcomings of RRT* is the increasing number of nodes needed to achieve optimality. From a computational perspective, this puts a burden on 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. Another variant of RRT known as Informed RRT is better suited for the problems involving multidimensional spaces.

D. Grid Search

First, RRT* pre-plans in a grid space and an initial node tree is obtained with the goal point as the root. Secondly, a set of new points is created by sampling the tree, and the tree is optimized under the principle of RRT*, then the final path in the initial scenario is obtained. When the environment changes, the node tree is trimmed and the cost is updated according to the changes in obstacles respectively. Then the initial optimized path and the areas where the grid path-connecting trend change are obtained. Because some branches of the tree can't form the optimal path in a changing environment, we employ the branch-and-bound technique to delete these branches to simplify the node tree. As the sampling points increase, the path converges toward the optimal solution.

The grid search process plays two roles:

1) Getting the initial node tree, so a higher-quality initial solution can be obtained quickly;

2) Getting the area of the grid where the optimal path area is located, thus the sampling space is limited and the efficiency of the sampling optimization is improved;

E. Notations

We formulate the problem of real-time path planning in dynamic environments as follows. We want to find a path, i.e. (x0, x1, ..., xgoal), from the agent to xgoal. Also, the path to xgoal should be of minimum length and avoid the obstacles at all costs. For finding a minimal length path, we use cost-to-reach values (denoted by ci) which are computed using the Euclidean length of the path from x0 to xi. Furthermore, we restrict our algorithm to be real-time. This means that we have a limited amount of time for Tree Expansion-and-Rewiring and Path Planning with the expanded tree. In real-time path planning it is important to have a real-time response whether or not we have a path to the goal. When the tree is growing and xgoal is not found, we use the cost function fi = ci + hi to plan a path from x0 to a point close to xgoal(goal tolerance). hi denotes cost-to-go values from xi to xgoal and is computed using Euclidean distance between these two nodes. Note that, when time for path planning is up, next immediate node after x0 in the planned path, x1, is committed and should be followed. In multi-query tasks, each point in Xfree(free nodes) of the environment has the potential of being xgoal while dynamic obstacles are moving around and may block some paths. In our algorithm we change x0(present state) when the agent moves to keep the agent near the tree root, and by retaining the whole tree between iterations we can return a path to any point in the environment (see Fig 1). Therefore, our main problem is, how to rewire the tree really fast and in a real-time manner to react to the changes in the environment (changes in Xobs) as well as to return a minimal length path to xgoal while xgoal can be any point in the environment.

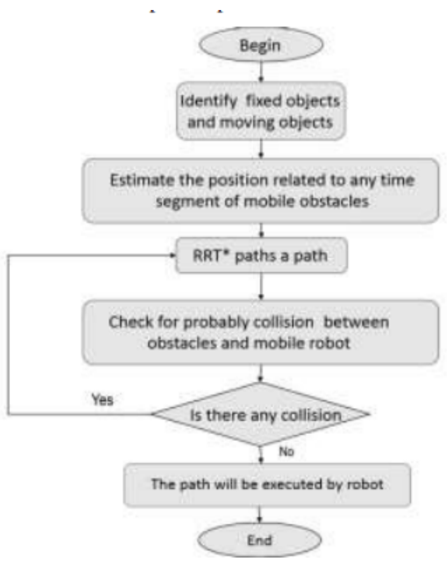

Begin
Identify fixed objects and moving objects
Estimate the position related to any time segment of mobile obstacles
RRT* paths a path
Check for probably collision between obstacles and mobile robot
Yes
Is there any collision
No
The path will be executed by robot
End

Fig 2: Flowchart of Algorithm

III. OBSTACLES AVOIDANCE

In first step, the RRT* algorithm samples a node in space then checks if it is in free space or identifies as an obstacle and in case the sample selected from space occupied by obstacle, search tree will delete it and select another sample. In this way it is possible to design a path without any obstacles. In this section RRT* algorithm was implemented in an environment with moving obstacles, which was executed in two phases: first prediction of moving obstacles and second: phase synchronization of moving obstacles and path planner system in many tiny offline parts which seems on-line and can be used in real time systems.

A. Position and time Prediction of moving obstacles

To avoid collision in moving obstacles which have repetitive moving patterns like circular movement pattern, movement pattern towards straight line or sinusoidal form movement pattern and so the movements will be predictable the system uses image processing techniques so it is possible to estimate the moving speed of objects and their future position in the specific time. Moreover image processing techniques estimate present and future locations of obstacles. In this way we will be able to estimate and predict the possibility of collision between mobile robots with moving obstacles.

B. Synchronization

After recognizing the situation of fixed obstacles in the search environment and prediction of the movement of moving obstacles, a programmed system studies the feasibility of the probable contact of a mobile robot and available moving obstacles in the environment; and based on that will decide to plan another path. As all paths will locate in a specific area for optimizing the asymptotic paths and moving obstacles follow a repetitive pattern on their movement; so, in next steps, planning will be done by algorithm RRT* in the area recognized with the probability of contact and obstacles are considered as fixed one and a new path is planned.

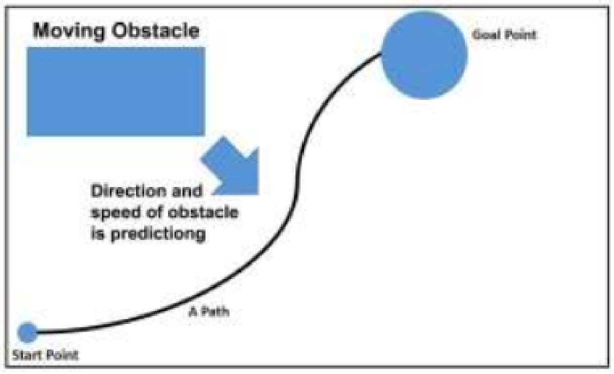


Fig. 3. It is shown an initial path named A that is planned without considering the future position of obstacle

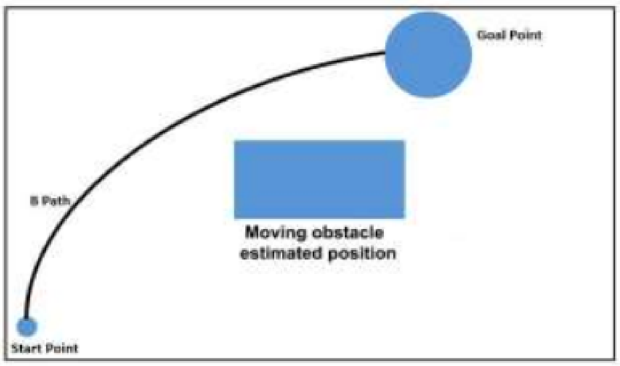


Fig. 4. It is shown path named B that is planned with considering the future position of obstacle

IV. SIMULATION

All simulations are performed in the python environment and image processing techniques are simulated by using Matlab Plot libraries.

A. Environment

For simulation, an environment of 30×30 is considered with a relatively complex map with multiple dynamic obstacles and a single static obstacle. Figure below shows the map used in simulations

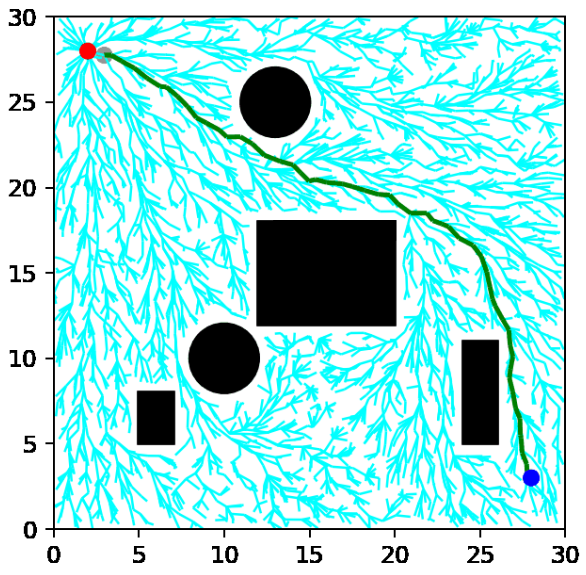


fig 5: Tree exploration and path at the beginning of the program

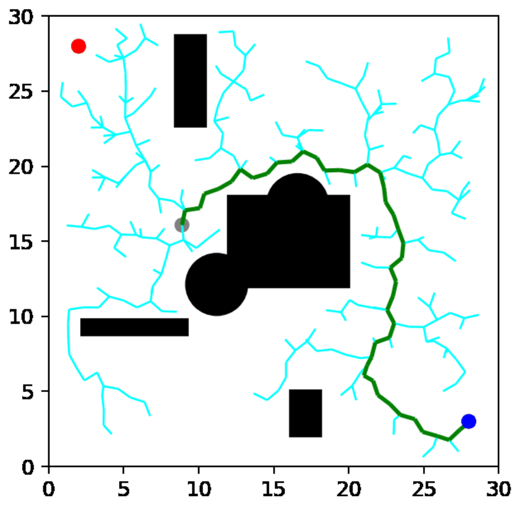


fig 6: Tree generation and rewiring after the environment changes

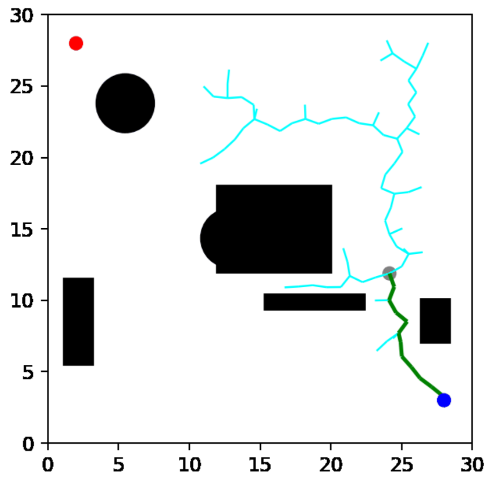


fig7: exploration tree while robot is near the goal

V. RESULTS AND DISCUSSION

The effectiveness and superiority of the proposed algorithm were verified in the simulated dynamic environments with changing obstacles. Simulation results show that RRT* has better performance in providing the shortest path and avoiding obstacles in both static and dynamic environments. In addition, it has the ability to quickly find a high-quality initial path and constrains the sampling space to improve the efficiency of sampling, in scenarios that involve changing obstacles and dynamic environments. However, it can be inferred that the process is computationally very expensive. It would be suggested to use combinational algorithms such as a combination of D* lite and RRT* algorithms to decrease the computational power.

References

[1] 1-S.Lavalle. Planning Algorithms. Cambridge University Press, 2006.

[2] M. Zucker, J. Kuffner, and M. Branicky. Multiple RRTs for rapid replanning in dynamic environments. In IEEE Conference on Robotics and Automation, 2007

[3] LUDERS, B. D., KARAMAN, S., FRAZZOLI, E., AND HOW, J. P. 2010. Bounds on tracking error using closed-loop rapidly exploring random trees. In American Control Conference, IEEE.

[4] OTTE, M., AND FRAZZOLI, E. 2015. RRTX: Real-time motion planning/replanning for environments with unpredictable obstacles. In Algorithmic Foundations of Robotics XI. Springer

[5] RASTGOO, M. N., NAKISA, B., NASRUDIN, M. F., NAZRI, A., AND ZAKREE, M. 2014. A critical evaluation of literature on robot path planning in dynamic environment. Journal of Theoretical & Applied Information Technology

[6] SUD, A., ANDERSEN, E., CURTIS, S., LIN, M., AND MANOCHA, D. 2008. Real-time path planning for virtual agents in dynamic environments. In ACM SIGGRAPH 2008 Classes, ACM.

[7]] 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.

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

[9] 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.

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

[11] N. E. Du Toit and J. W. Burdick, “Robot motion planning in dynamic, uncertain environments,” IEEE Transactions on Robotics, vol. 28, no. 1, pp. 101–115, 2012.

[12] Seif, Roodabeh. (2015). Mobile Robot Path Planning by RRT* in Dynamic Environments.

[13] Adiyatov, Olzhas; Varol, Huseyin Atakan (2017). *[IEEE 2017 IEEE International Conference on Mechatronics and Automation (ICMA) - Takamatsu, Japan (2017.8.6-2017.8.9)] 2017 IEEE International Conference on Mechatronics and Automation (ICMA) - A novel RRT*-based algorithm for motion planning in Dynamic environments. , (), 1416–1421.* doi:10.1109/ICMA.2017.8016024

[14] J. Qi, H. Yang and H. Sun, "MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment," in *IEEE Transactions on Industrial Electronics*, vol. 68, no. 8, pp. 7244-7251, Aug. 2021, doi: 10.1109/TIE.2020.2998740.

[15] DL-RRT* algorithm for least dose path Re-planning in dynamic radioactive environments