Path Planning for a non-holonomic mobile robot in an obstacle space using RRT-A* algorithm

Joseph Pranadeer Reddy Katakam
University of Maryland
College Park, USA
jkatak73@umd.edu

Bharadwaj Chukkala
University of Maryland
College Park, USA
bchukkal@umd.edu

***Abstract*— The RRT path planning algorithm is quite popular for its speed and implementation. The only downside to RRT is the optimality in the planning process, as the nodes are selected at random, different planning costs are obtained due to the metric function. Therefore an improvised heuristic RRT-A* algorithm is proposed for a mobile robot motion planning with non-holonomic constraints. This algorithm makes the performance optimal by introducing the cost of the A Start algorithm into the RRT algorithm. A mobile robot such as a turtle bot with a non-holonomic constraints has been used to implement this algorithm where simulation results have shown that the Manhattan heuristic information function based RRT-A* planning algorithm is better than the other improved RRT algorithms in the optimization path and computational cost.**

***Index Terms*—Sampling-based Motion Planning, Path Planning Algorithms, A Star Algorithm, RRT Star Algorithm, Robotic Operating System (ROS), Turtle Bot, non-holonomic constraints**

I. Introduction

The traditional problem with motion planners is that they are quite slow. Unless in a scenario where the robot's movements are being planned in advance, this is a concerning issue where a robot is desired to react in real-time in accordance to the environment. The motion planning algorithms have improved significantly in the past decade and are a trend in today's world.

The functionality of High Flexibility, Efficient Movements, Better Power Consumption and Better Use of Workspace has lead to an increasing demand in the field of motion planning. In order to comprehend the concept, the jargon has been elucidated prior to going into the details of the efficient planning algorithm to be used and tested out.

A. Path Planning

Planning consists of finding a sequence of actions that transform some initial state into some desired goal state. In path planning, the states are agent locations and transitions between states represent actions the agent can take, each of which has an associated cost. Planning is performed to determine a path from the start node to the goal node and is a crucial application in the field of robotics. A path is optimal if the sum of its transition costs (edge costs) is small across all possible paths leading from the initial position (start state) to the goal position (goal state). Planning algorithm quality depends on two major factors, optimality, and completion. A planning algorithm is complete if it will always find a path in finite time when one exists and will let us know in finite time if no path exists. In a similar way, a planning algorithm is optimal when it always finds an optimal path.

There exist several approaches for path planning and they are categorized in two types: Deterministic/Heuristic based methods and Randomized/ Probabilistic methods. When the dimensionality of the planning problem is low, for example when the agent has only a few degrees of freedom, deterministic algorithms are predominantly used because they provide bounds on the quality of the solution path returned. A robotic path planning technique consists of representing the environment (or configuration space) of the robot as a graph G = (S, E), where S is the set of possible robot locations and E is a set of edges that represent transitions between these locations. The cost of each edge is the cost of transitioning between the two endpoint locations.

B. Optimality

A path-planning algorithm is considered optimal if it generates the shortest path, thereby improving energy efficiency. Hence, it is an important parameter for real-world solutions. Computation time is the time required to compute the solution is very important for practical applications.

C. Holonomic and non-holonomic

Depending on the type of constraints, there are several types of problems. The holonomic constraint is a constraint on the dynamical system that is integrable to eliminate one of the variables. A constraint that is not integrable or cannot be expressed is said to be a nonholonomic constraint. For a sphere rolling on a rough plane, the no-slip constraint turns out to be nonholonomic.

Motion planning is made complicated by the non-holonomic differential constraints. If these constraints are not handled, the non-holonomic wheeled vehicle will be unable to track the planned path. In the figure, the non-holonomic differential constraints of a simple car robot are.

The robot's state equation can be expressed as follows:

$$x^{\cdot} = u_v \cos\theta \tag{1}$$

$$y^{\cdot} = u_v \sin\theta \tag{2}$$

$$\theta^{\cdot} = u_v \tan u_\phi / L \tag{3}$$

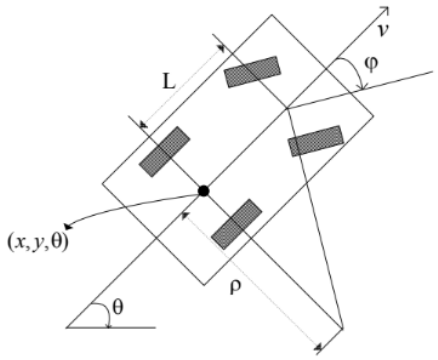

Fig. 1. non-holonomic differential constraints of a simple car robot

D. Heuristics and Manhattan heuristic Function

A heuristic function is an alternative to the search algorithm in each branch step, which decides which branch to follow based on available information. Heuristics basically estimate how close a state is to a goal. An acceptable heuristic function never overestimates the cost of reaching the goal, i.e. it estimates that the cost of reaching the goal is no higher than the lowest possible cost at the current point in the path.

- The Manhattan Distance is the distance between two points measured along axes at right angles to each other. In a plane with a point (p1) at (x1, y1) and another point (p2) at (x2, y2), the Manhattan distance between them is given by $|x1 - x2| + |y1 - y2|$.

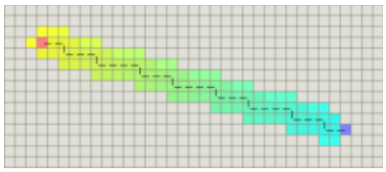

Fig. 2. A point robot using 4 actions and an admissible Manhattan heuristic for path planning.

II. Background

The trend of path planning and obstacle avoidance has been on the rise since few decades. Given the trend, numerous techniques have been proposed to tackle the issue. To address the problem of optimization, the robot takes into account the distance to the goal and the proximity with respect to the obstacle. One way to get along with this situation is to reduce the complexity of the problem by smartly choosing few samples from the free configuration space without losing completeness in the process. The popular algorithm RRT, which is prevalent in the application of motion planning, does not produce an optimal path. Due to this background, various version of RRT path planning algorithms have been proposed.

A. Literature Review

The following paper [1] Li et al has been taken as reference for the application of this project. The paper proposes the introduction of the A* cost function which makes the drawback of deficiency of cost function up for the RRT algorithm, guiding the tree to a better path. The heuristic information function can influence the performance of a planner. Simulation results of several improved heuristic algorithms verify the excellent performance of the improved RRT algorithm, and there are various performances in the heuristic RRT algorithms based on different heuristic information functions. The best result is obtained when the Manhattan distance was used as the heuristic information function.

In [2] Chang-an et al. presented simulation results of the RRT algorithm that is tested in an unknown environment and compared these results against an improved version of RRT that they have developed. The modern trend as we see is not using RRT itself as a path planning algorithm but as a sampling based tree generator which gives us n number of paths that can be explored using other search algorithms.

In [3], Kuwata et al. presented a real-time motion planning algorithm that is based on RRT approach and is applicable to autonomous vehicles operating in urban environments.

In [4], Pan and Zhang presented a motion planning algorithm based on RRT and then an extended version of the RRT algorithm is presented where RRT is combined with simultaneous localization and mapping (SLAM). The extended RRT-SLAM algorithm is tested in the MobileSim environment and the paper is concluded with promising simulation results. As discussed in [3] and [4], we see rising use of improved RRT methods in real world applications. There is extensive research that has been going on in this area.

Another RRT-SLAM approach is presented by Huang and Gupta in [5] where the motion planning problem that arises from robotic exploration is addressed while sensing, localization and mapping uncertainties were taken into account. Their algorithm is also simulated in MobileSim and the effectiveness of the RRT-SLAM algorithm is presented. With improvements come challenges and through them ground breaking technological advancements.

In [6], Seif and Oskoei presented a new path planning method that is applicable to dynamic environments. The drawback of the algorithm is that it requires a camera to be installed above the workspace hence it is only applicable to closed search spaces e.g. stores and factories. Like discussed prior, even if the solution is contained within a range of usage, the weight that this solution carries can be very effective in the bigger picture.

In [7], Moon and Chung addressed the real-time trajectory generation problem of two-wheeled differential drive mobile robots considering non-holonomic and dynamic constraints. The proposed algorithm is named dual-tree rapidly exploring random trees (DT-RRT) and the simulation results demonstrates the superior performance of DT-RRT in terms of reduced computation time and high success rate.

In [9] Faiza Gul et al proposed a system that influences the concept of Bi-directional Rapidly Exploring Random Trees (Bi-RRTs) for path planning together with path smoothness for minimizing the processing time to search for the path

length. Produced results show that the presented approach as high robustness and use less computational effort. Extensive research has been made to solve bigger issues that were glanced above, but the solution of optimality is always an intruguing question. New answers keep coming and the old ones keep fading. [1] Li et al has provided a robust solution layering many factors of comparison and concluding which algorithm can be effective in an all round manner.

III. Algorithms

Choosing an appropriate path planning algorithm helps to ensure safe and efficient point-to-point navigation and the optimal algorithm depends on the robot geometry as well as the computing constraints, including static/holonomic and dynamic/non-holonomic constrained systems, and requires a comprehensive understanding of modern solutions. The applicability of algorithms considering the performance, cost, speed, and efficiency, popular path planning algorithms have been discussed below. Also, an improved algorithm obtained by the amalgamation of the two popular algorithms for a better output is elucidated.

A. A Star Algorithm

A* is an informed search algorithm, or a best-first search, meaning that it is formulated in terms of weighted graphs: Starting from a specific starting node of the graph, it aims to find a path to a given destination node with minimal cost (shortest distance traveled, shortest time, etc.). It does this by maintaining a tree of paths originating at the start node and extending those paths one edge at a time until its end criterion is satisfied.

At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path to the goal. Specifically, A* selects the path that minimizes f(n) = g(n) + h(n). Where n is the next node on the path, g(n) is the cost of the path from the start node to n, and h(n). is a heuristic function that estimates the cost of the cheapest path from n to the goal. A* terminates when the path it chooses to extend is a path from start to goal or if there are no paths eligible to be extended.

The heuristic function is problem-specific. If the heuristic function is admissible, meaning that it never overestimates the actual cost to get to the goal, A* is guaranteed to return a least-cost path from start to goal. It is also possible to switch the direction of the search in A*so that planning is performed from the goal state towards the start state. This is referred to as 'backward' A*.

Algorithm 1 A Star Algorithm Pseudo Code

```
A* (start, goal)
Closed set = the empty set
Open set = includes start node
G[start] = 0, H[start] = H_calc[start, goal]
F[start] = H[start]
While Open set ≠ ∅
    do CurNode ← EXTRACT-MIN- F(Open set)
    if ( CurNode  == goal ), then return BestPath
    For each Neighbor Node N of CurNode
        If ( N is in Closed set ), then Nothing
        else if ( N is in Open set ),
            calculate N's G, H, F
            If ( G[N on the Open set] > calculated G[N] )
                RELAX(N, Neighbor in Open set, w)
                N's parent=CurNode & add N to Open set
        else, then calculate N's G, H, F
                N's parent = CurNode & add N to Open
```

B. RRT Algorithm

RRT, A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. The premise of RRT is quite straight forward. Points are randomly generated and connected to the closest available node. Each time a vertex is created, a check must be made that the vertex lies outside of an obstacle. Furthermore, chaining the vertex to its closest neighbor must also avoid obstacles. The algorithm ends when a node is generated within the goal region, or a limit is hit.

The robotic path planning problem where the robot, with certain dimensions, is attempting to navigate between point A and point B while avoiding the set of all obstacles. The robot should be able to move through the open area, Configuration space, which is not necessarily discretized all the time. This creates a problem where the robot does not have any nodes to travel between in the configuration space. This problem to create nodes in a non-discretized configuration space, for robot traversal, can be solved using RRT algorithm. It does not necessarily give an optimal path but can handle problems with obstacles and differential constraints (nonholonomic and kinodynamic) and has been widely used in autonomous robotic motion planning.

Algorithm 2 RRT Algorithm Pseudo Code

```
RRT( )
1  Tree_init(q_init);
2  while !flag do
3      q_rand ← SamplePoint();
4      q_nearest ← Nearest_node(T, q_rand);
5      q_new, u_best ← Control(q_nearest, q_rand);
6      if collisionfree
7          tree ← tree ∪ {q_new};
8      if ‖q_new − q_goal‖ ≤ d
9          return flag=true;
10     else return flag=false;
11 function Control(q_nearest, q_rand)
12     d_min ← ‖q_nearest, q_rand‖, u_best ← ∅
13     for u ∈ U do
14         q_new ← integrate(q_nearest, u);
15         if collision(q_nearest, u, q_new)
16             next u;
17             d ← ‖q_new, q_rand‖;
18             if d < d_min then
19                 d_min, u_best ← d, u
20     return q_new, u_best;
```

C. Improved RRT Algorithm

RRT, A rapidly exploring random tree (RRT) is an algorithm designed to efficiently search nonconvex, high-dimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples drawn randomly from the search space and is inherently biased to grow towards large unsearched areas of the problem. The premise of RRT is quite straight forward. Points are randomly generated and connected to the closest available node. Each time a vertex is created, a check must be made that the vertex lies outside of an obstacle. Furthermore, chaining the vertex to its closest neighbor must also avoid obstacles. The algorithm ends when a node is generated within the goal region, or a limit is hit.

The robotic path planning problem where the robot, with certain dimensions, is attempting to navigate between point A and point B while avoiding the set of all obstacles. The robot should be able to move through the open area, Configuration space, which is not necessarily discretized all the time. This creates a problem where the robot doesn't have any nodes to travel between in the configuration space. This problem to create nodes in a non-discretized configuration space, for robot traversal, can be solved using RRT algorithm. It doesn't necessarily give an optimal path but can handle problems with obstacles and differential constraints (nonholonomic and kinodynamic) and has been widely used in autonomous robotic motion planning.

Algorithm 3 Heuristic RRT Algorithm Pseudo Code

```
Heuristic – RRT( )
1      Tree_init(q_init);
2      while !flag do
3      for i = 1 to K do
4          q_rand(i) ← SamplePoint();
5      q_nearest(i) ← Nearest_node(T, q_rand(i));
6      f(i) = f(q_nearest(i)) = g(q_nearest(i)) + h(q_nearest(i))
7      n = arg min_{i} f(i);
8      q_nearest = q_nearest(n)
9      q_new, u_best ← Control(q_nearest, q_rand);
10     if collisionfree
11     tree ← tree ∪ {q_new};
12     if ‖q_new − q_goal‖ ≤ d
13     return flag=true;
14     else return flag=false;
```

IV. Methodology

- Motion planning was merely a simple geometric path planning in its initial stages. Considering only the geometric constraints (holonomic) like obstacles in the configuration space, we perform path planning of low complexity so that the path planning configuration doesn't take into consideration the non-holonomic constraints like acceleration and velocity constraints and is easier to implement.
- When path planning is being performed on a wheeled robot, it is obvious that non holonomic constraints must be taken into consideration. In this case traditional path planning algorithms fail to solve the problem of non-holonomic constraints. RRT sampling-based method has been implemented to deal with the non-holonomic constant.
- There are many modifications layered on RRT which led to improved algorithms like goal bias RRT. These improvements give a probabilistic completeness to successfully guarantee path planning. But the catch being, the cost function which plays a key role in the path finding has to be considered for the performance of the algorithm.
- A cost function has been introduced to the RRT algorithm. This cost function is a part of the A* algorithm which tackles the problem of optimal path with less

cost. An ideal cost function can perfectly reflect on the performance of the optimal path index such as length of path, run-time of algorithm and the energy that is being consumed.

- Further to improve the RRT-A* algorithm, a metric function with the Manhattan heuristic information has been used between nodes. This effectively reduces the cost even more making this algorithm highly optimized.
- Further, this algorithm has been implemented on a mobile robot with non-holonomic constraints.

V. Simulation Work

The open-source 3D robotics simulator, Gazebo has been used to simulate the discussed algorithm. This way the challenges likely to be faced in the real world can be safely and efficiently foreseen. A mobile robot called turtle bot has been used to test the algorithm in the simulated environment.

- Initially, a custom robot configuration space has been created in a 2D map.
- The improvised algorithm is tested out and visualized on this 2D map, to observe the efficiency in the performance.
- Tweaks were made until an appropriate output is obtained in this 2D space.
- Next, the same robot configuration space is recreated in the gazebo world, taking proper care of all the positions of the objects in the environment with respect to the robot.
- The robot's spawning point is made sure to be the same like that in the 2D space, in order to replicate the same outcome.
- At last, the algorithm is tested out in the gazebo world.

VI. Test Runs

To test the optimality and robustness of the algorithm, multiple tests have been performed and kept a track of to observe the conditions in which the algorithm outperformed others.

- For regular testing, burger model of Turtle Bot was used to simulate the algorithm in the gazebo environment.
- The same model of Turtle Bot has been used to simulated a star algorithm as well to compare the results. (RRT?)
- Making the necessary changes, the same algorithm was tested on waffle and waffle pi models of the Turtle Bot as well. As the drive constraints for every model of the robot changes, a significant amount of time has been consumed to tweak many different parameters to test the algorithm.
- Dynamic environments were also used to test the performance of the improvised algorithm.

A. Performance of RRT astar in different obstacle spaces with euclidean heuristic

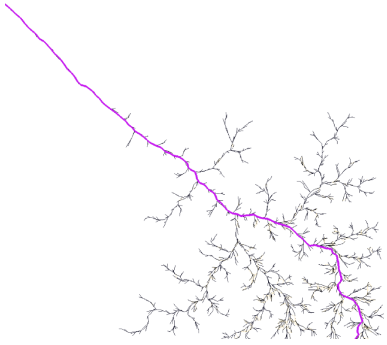

Fig. 3. RRT astar with euclidean heuristic in no obstacle space

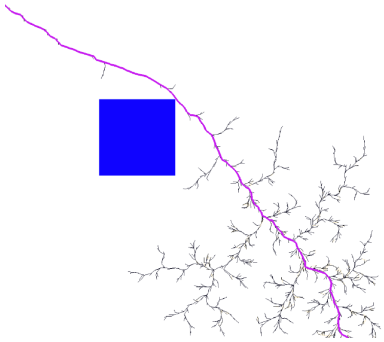

Fig. 4. RRT astar with euclidean heuristic in sparse obstacle space

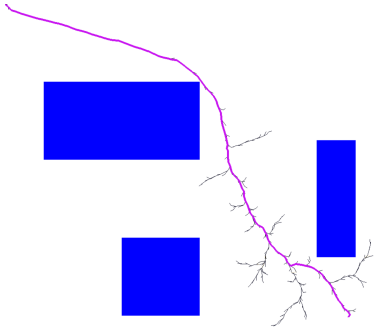

Fig. 5. RRT astar with euclidean heuristic in dense obstacle space

B. Performance of RRT astar in different obstacle spaces with Manhattan heuristic

Fig. 6. RRT astar with Manhattan heuristic in no obstacle space

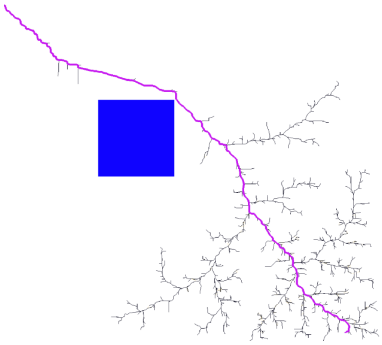

Fig. 7. RRT astar with Manhattan heuristic in sparse obstacle space

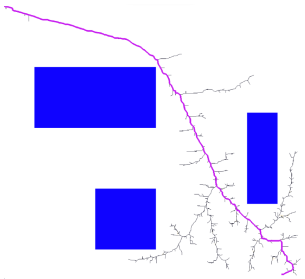

Fig. 8. RRT astar with Manhattan heuristic in dense obstacle space

C. Performance of RRT in different obstacle spaces with euclidean heuristic

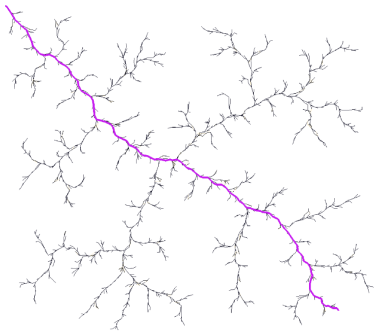

Fig. 9. RRT with Euclidean heuristic in no obstacle space

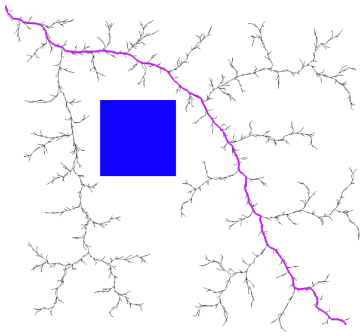

Fig. 10. RRT with Euclidean heuristic in sparse obstacle space

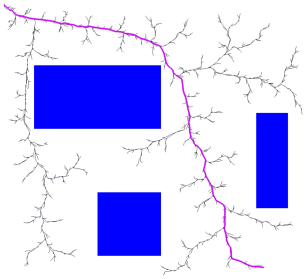

Fig. 11. RRT with Euclidean heuristic in dense obstacle space

D. Performance of RRT in different obstacle spaces with Manhattan heuristic

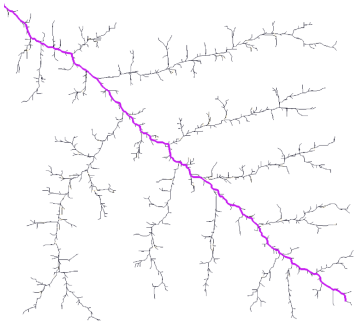

Fig. 12. RRT with Manhattan heuristic in no obstacle space

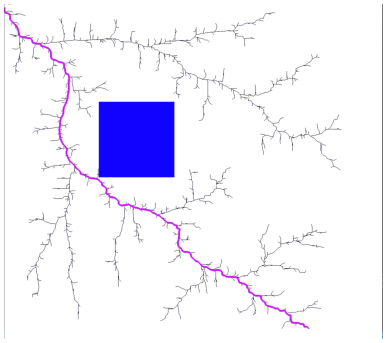

Fig. 13. RRT with Manhattan heuristic in sparse obstacle space

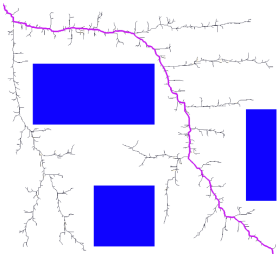

Fig. 14. RRT with Manhattan heuristic in dense obstacle space

VII. Discussion

In the process of simulating these results, there have been many trials and errors that the algorithm went through to check the credibility and optimally. RRT sampling based method, even though is traditional and is being used less when compared to the improved versions of itself. This attempt to of our implementation is a direct example of an improvement. When we consider the fact that RRT is a generated space with a start and goal node and many paths connecting the both, it can be clearly understood that a variety of algorithms can be used to find the path between the start and the goal. Thus improved versions are created which each have a specific functionality. There is a lot of scope to this niche in path planning, where based on the requirements, one can make necessary modifications to get consistent results. For example, consider the factors cost, optimality, computation power, time etc, an improved algorithm may vary based on which of these factors is given priority.

RRT, although a very robust and adjusting algorithm being used for efficient path planning and traversal has a major drawback. It works poorly in convex environments and the path that is generated by RRT looks very chaotic and usually it will take longer distance from initial point to the goal point.This can be partially eliminated with some improvements but not completely. One of the many advantages of RRT is to generate open-loop trajectories for nonlinear systems with state constraints. An RRT can also be considered as a Monte-Carlo method to bias search into the largest Voronoi regions of a graph in a configuration space.

The Challenges we faced when implementing the algorithm is to figure out the individual sub functions in the algorithm. This is because there are times when the problem lies in the sub-function but not in the whole algorithm. And sometimes, it is hard to tell whether the sub-functions are behaving as they should behave. The only method we can find out is to slowly debug the individual sub-functions, which consume most of our time.

VIII. Future work

This work done will be extended to apply on real world scenarios.

- One of the scenarios where efficiency of the motion planning plays crucial role is in Urban search and rescue operations. In this application, one robot searches the map to find the victims of urban disaster, while the 'follower' robot goes to the coordinates of these victims to rescue. Currently, in order to simulate the scenario, the ArUco markers are placed in the map which are considered as victims. The coordinates of these markers are sent to the follower robot. Using the discussed motion planning algorithm, the follower robot will be observed to check the optimality in reaching the marker.
- As serial manipulators are heavily used in most of the robotic fields, this motion planning algorithm can be

extended to see the performance of a robotic arm to reach from start position to goal position.

IX. Conclusion

In this paper, we have analyzed the RRT sampling-based algorithm, A* graph traversal algorithm and implemented a combination of both. This implementation is an improved version which is named as RRT-A*. This algorithm has been tested to work for robots operating in sparse and dense configuration spaces. It has been duly observed after careful comparison that, euclidean heuristic function will give us an optimal path by exploring lesser number of nodes in a dense environment, unlike the Manhattan heuristic function. But, the Manhattan heuristic gives us the fastest result and is equally optimal. Either way, both the heuristics are admissible in many scenarios. The algorithm has been successfully simulated on Turtle Bot 3 in a real-world environment with obstacles. From the results, it is concluded that RRT-A* algorithm increases the path optimality with decreased cost, thus having broad application prospects.

References

[1] Jiadong Li, Shirong Liu, Botao Zhang, Xiaodan Zhao, "RRT-A* Motion Planning Algorithm for Non-holonomic Mobile Robot" SICE Annual Conference, September 9-12, 2014.

[2] L. Chang-an, C. Jin-gang, L. Guo-dong, and L. Chun-yang, "Mobile Robot Path Planning Based on an Improved Rapidly-exploring Random Tree in Unknown Environment," IEEE International Conference on Automation and Logistics, pp. 2375-2379, September 2008.

[3] Y. Kuwata, G. Fiore, and E. Frazzoli, "Real-time Motion Planning with Applications to Autonomous Urban Driving," IEEE Transactions on Control Systems Technology, vol. 17, no. 5, September 2009.

[4] H. Pan and J. Zhang, "Extending RRT for Robot Motion Planning with SLAM," Trans Tech Publications, Switzerland. Applied Mechanics and Materials vol. 151 (2012) pp. 493-497.

[5] Y. Huang and K. Gupta, "RRT-SLAM for Motion Planning with Motion and Map Uncertainty for Robot Exploration," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 1077-1082, September 2008.

[6] R. Seif and M. Oskoei, "Mobile Robot Path Planning by RRT* in Dynamic Environments," I.J. Intelligent Systems and Applications, pp. 24-30, May 2015.

[7] C. Moon and W. Chung, "Kinodynamic Planner Dual-Tree RRT (DT-RRT) for Two-wheeled Mobile Robots using the Rapidly Exploring Random Tree," IEEE Transactions On Industrial Electronics, vol. 62, no. 2, pp. 1080-1090, February 2015.

[8] Faiza Gul and Wan Rahiman, "An Integrated approach for Path Planning for Mobile Robot Using Bi-RRT", IOP Conference Series: Materials Science and Engineering, 697 (2019).

[9] Zhao Wang1, Xianbo Xiang*1,2, "Improved Astar algorithm for path planning of marine robot", Proceedings of the 37th Chinese Control Conference, July 25-27, 2018.

[10] Jose Bernal, Paola Ard on, Eric Paire, "Two-dimensional Offline Path Planning for a Turtlebot",University of Girona

[11] Canberk Suat Gurel, Rajendramayavan Rajendran Sathyam, Akash Guha, "ROS-based Path Planning for Turtlebot Robot using Rapidly Exploring Random Trees (RRT*)", University of Maryland

[12] MOHAMED ELBANHAWI (Student Member, IEEE) AND MILAN SIMIC, "Sampling-Based Robot Motion Planning: A Review", IEEE. Translations, 2169-3536, 2014.

[13] Jiaonan Zhang1, Tengyu Zhang1, Yi Niu2, Yuxuan Guo1, Jiahao Xia1, Yang Qiu1, Zhengyi Yuan1, Yingze Yang1, "Simulation and Implementation of Robot Obstacle Avoidance Algorithm on ROS", Journal of Physics: Conference Series, 2203 (2022) 012010