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

Sri Sai Charan Velisetti
Nitesh Jha

***Abstract*— In this approach of growing the tree, every new node to be added involves a check for nearest neighbours with other nodes. Also, there are checks to determine if rewiring has to be done for optimal path. This might be computationally expensive in higher dimensions and large maps. It might also be memory inefficient to track so many nodes for the task. Hence, the RRT* Fixed Nodes algorithm adds a constraint on the tree structure, that the total number of nodes that can exist in the tree is fixed. Before this number of nodes is achieved, the RRT*FN works similar to the RRT* algorithm, but after the maximum number is reached, every new addition to the tree involves a simultaneous removal of a node from the tree. The candidate for removal of the tree are the childless nodes in the tree**

I. INTRODUCTION

A. Sampling based planning

Motion planning using graph based methods perform planning exhaustively. Sampling based methods have advantages over graph-based search methods due to their ability to solve complex planning tasks quickly.

B. Randomly-exploring trees

Rapidly-exploring random trees samples a node randomly in space and connects it to the nearest node in the tree, given that the connection is feasible (considering obstacles and maximum step length). If this is not feasible, another state is sampled to expand the tree. This expansion of the tree is performed until the goal node is reached, and the path is found.

C. RRT*

As all randomly sampled nodes are connected to the nearest neighbours only, there might be cases where the path generated might not be optimal, as lower cost could have been achieved with the selected nodes. Thus, the RRT* algorithm optimizes the RRT in that the connections are made by checking the overall cost relative to the start location. If a node with a lower cost is found than the nearest neighbour, it is replaced with the nearest node for establishing a connection. Also, after a connection is made, neighbours are checked if reconnecting ('rewiring') with a new sampled node reduces the cost, and if so, the connections are rewired.

I would like to Express my Thanks to
Dr.Reza Monfaredi Faculty of University of Maryland College Park.

II. METHODOLOGY

A. Path Planning Method : A Novel RRT*-Based Algorithm for Motion Planning in Dynamic Environments

1) RRT:* The initial planning problem is solved for a global solution using the RRT* algorithm, which performs local optimization for the optimal solution. Due to the constant rewiring of the structure, this will be slower to reach the optimal solution than RRT.

2) Imposing Constrain of Fixed Nodes: When a new node is sampled and the number of nodes in the tree is greater than a fixed number, the addition of node will require a removal of a childless node. The candidate for removal will be evaluated as follows: if no node in the tree has parent has the node being evaluated, then the node is childless. Then, a childless node is randomly sampled and removed from the tree. There is a possibility that the goal node itself is childless, hence removing it would result in impaired solution. Hence, in such cases, the goal node will explicitly be excluded from random sampling.

3) Dynamic Obstacle Tolerance: The robot navigating the obstacle space will receive the map at regular intervals. Once the initial motion plan is generated, further pings of the updated map with displaced obstacles will just involve a check of whether the optimal path is obstacle-free. If so, the motion is continued as normal. If not, the tree is broken down and reconnected/regrown to retrieve as much information as possible from the initial planning stage. Thus, the planning algorithm will be robust to changes in position of obstacles. However, the map information must be updated such that it leaves enough time for re-planning motion in case a collision might take place, and so we factor in the obstacle velocity and average re-planning time into the frequency of map update.

III. IMPLEMENTATION

A. pygame

1) Static Environment: The RRT* planner class is defined by the start position of search, end position, list of obstacles in space, and area between which it will randomly sample points. The first step is to sample a random point in the defined area. This involves two conditions which determines the sampled point: the sampled points are sampled randomly, but there are some instances where the goal point itself is sampled. The random sampled points account for search around obstacles/search for shorter paths, whereas the goal point sampling accounts for straight traversal. Then, the

nearest neighboring node to the sample is found, and is used as basis to grow the tree. The tree growth takes place in the direction of this nearest search, but with a determined step size. If this new position after growth is not in obstacle, all nodes which are in the neighborhood of this new node are selected. Then, a search for the most optimal parent node takes place. This involves calculating the distance of the neighboring node such that there is no obstacle in traversal (in steps of determined length) from the new node to the neighboring node. If there are multiple such neighbors which can be reached without an obstacle in between, the node with the minimum distance is selected as the parent node of the new node.

After this, a check is performed to check if there are alternate paths which can yield lower costs. If this is possible, rewiring of the tree takes place to minimize costs.

This completes the process of adding a node. At this point, the total number of nodes in the tree are checked. If this exceeds the maximum number of nodes that can exist in the tree, a simultaneous removal of a node is performed. All the leaf nodes from the tree are selected. Then, a node is randomly sampled from this set, and is removed. Only the branch from the eliminated node and its parent is broken, and the remaining branches (if any) to that parent, are retained.

This process is repeated over a maximum number of iterations, after which the search ends and the most optimal path is given. The maximum number of iterations is a parameter that requires tuning according to the search space, complexity, etc.

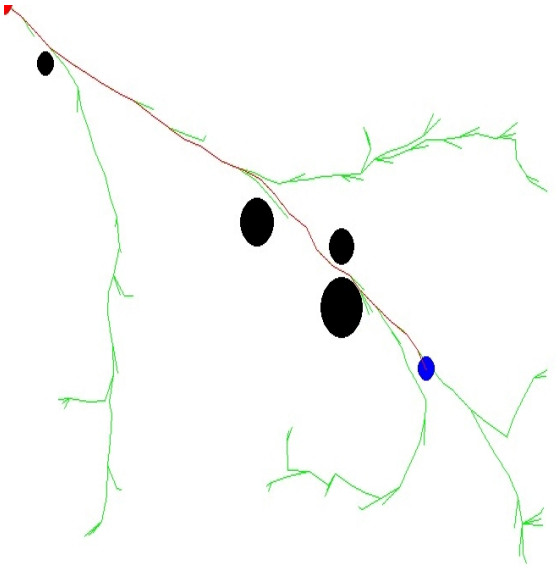

Fig 1 : Static Environment

2) Dynamic Environment: The dynamic environment is simulated in 2D as follows: an initial list of obstacles is defined for the search, and a similar process is followed as in the static case. However, in this case, we can introduce new obstacles on the 2D map using a mouse callback function. This modifies the map to put the new obstacle on it. When this happens, it is required that we check that the optimal path calculated is valid. This check involves the check of obstacles along the path, and returns True if it is still valid, and False otherwise. As the dynamic obstacles can be added according to the user, the whole process is enclosed in an infinitely running loop.

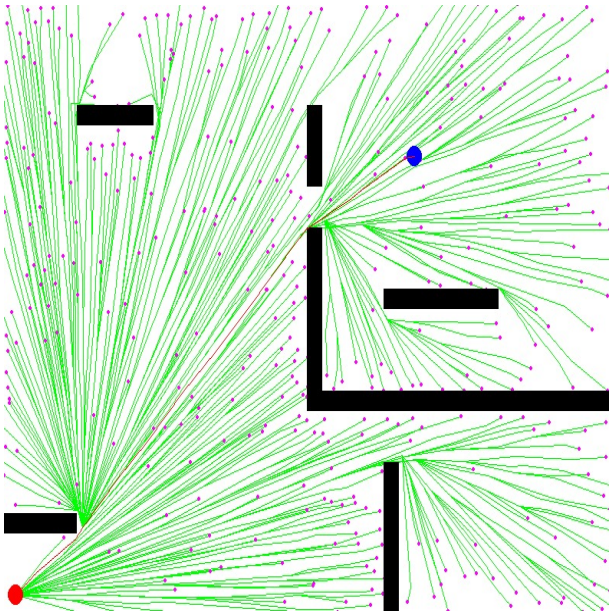

Fig 2 : Dynamic Environment

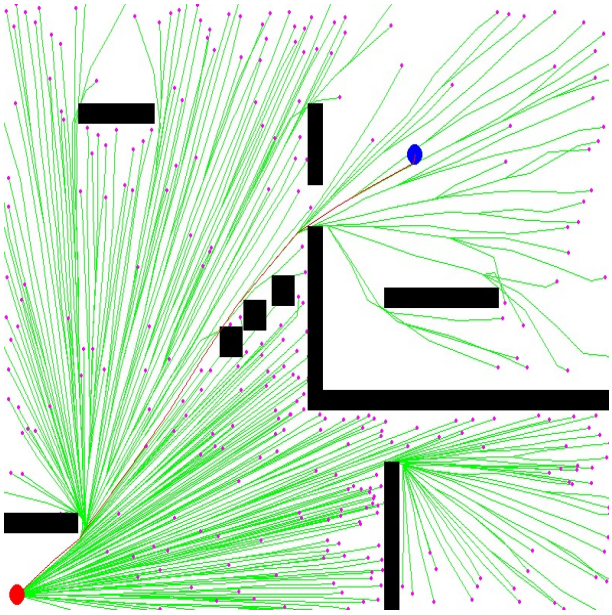

Fig 3 : Dynamic Environment with Obstacles

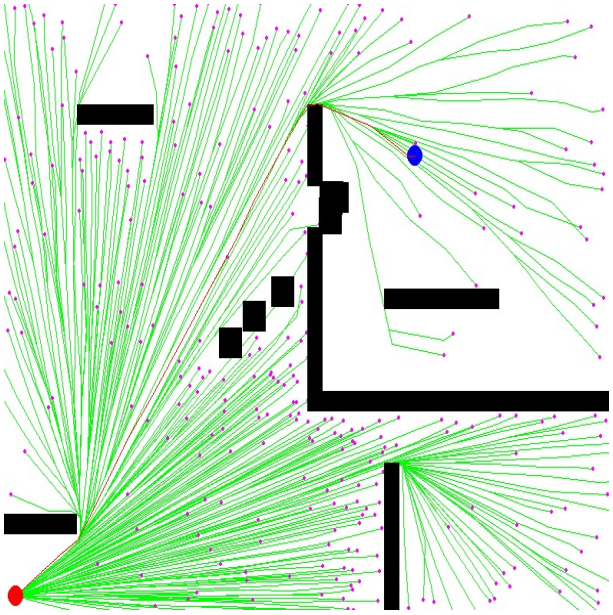

Fig 3 : Dynamic Environment with Obstacles

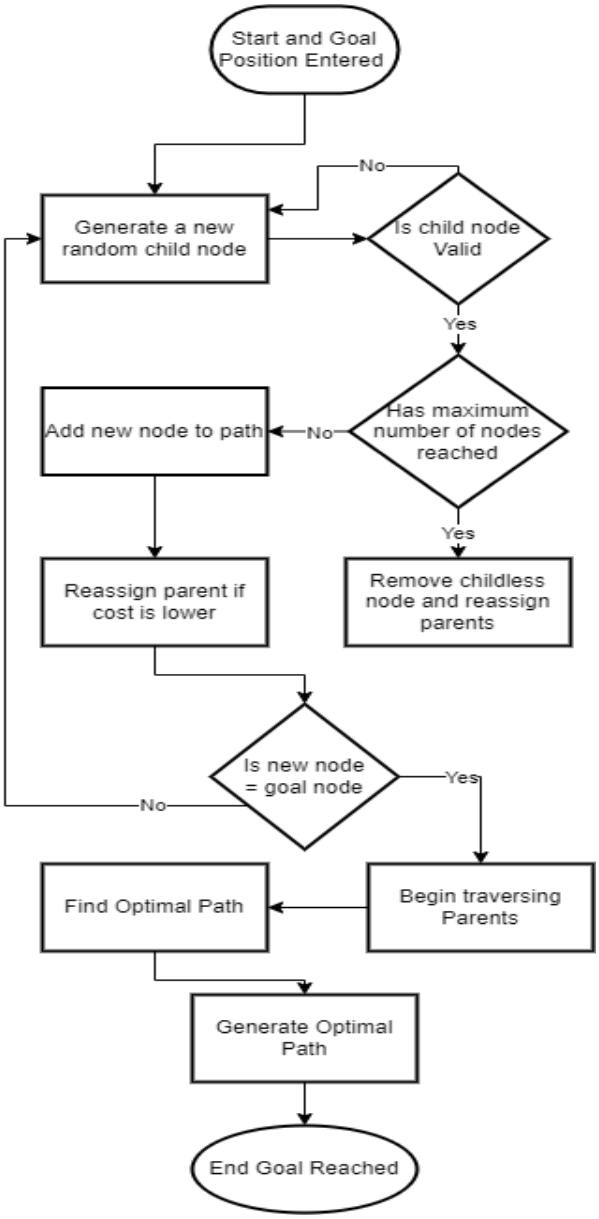

Fig 1 : Static Environment Process Flow

B. Gazebo

1) Static Environment: The gazebo implementation involves the 2-dimensional search execution. Then, a turtlebot is spawned at the start location. To control the turtlebot, an open loop controller is implemented.
This works as follows:

1 The optimal path nodes are selected.
2 From the start node, locations are taken from subsequent nodes along the optimal path as intermediate goal nodes.
3 First, we give a rotational velocity to the turtlebot to align itself such that the intermediate goal node only requires a linear translation along the heading direction. For this, small angular velocities are given until the heading angle is reached approximately.
4 Then, small linear velocity is given till the distance from the start location to the intermediate location is convered. For this step, we subscribe to the /odom topic for reference of robot position with respect to its initial location.
5 On performing the stepped-rotation and translation, we reach near the goal position. As the motion commands are not feedback controlled, there is some deviation in the reached goal position and the actual goal position. This can be improved by using a PID controller which could also compensate for extra rotations and translations.

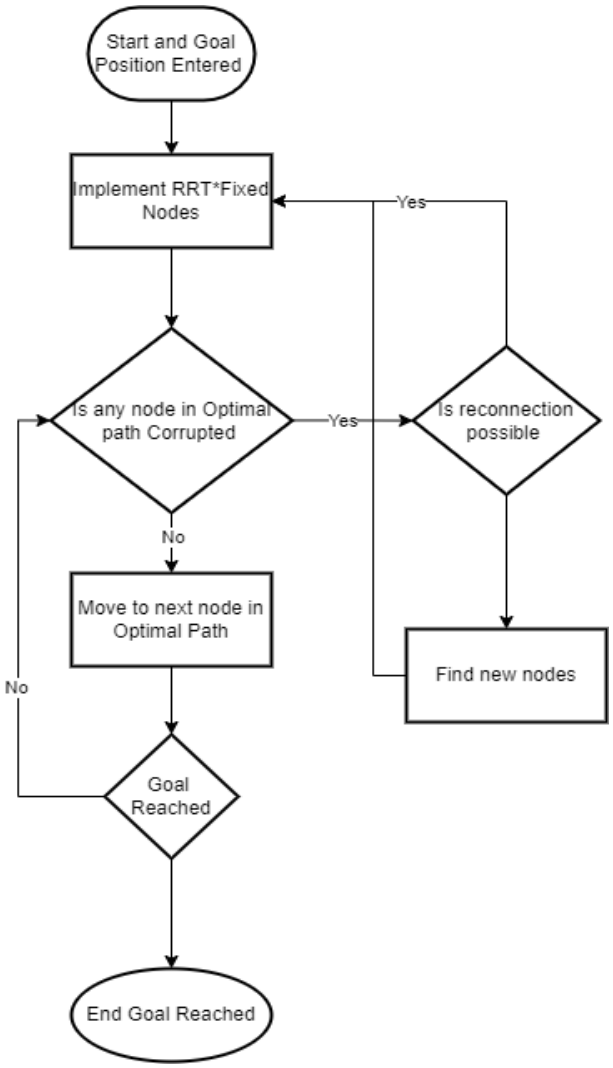

Fig 1 : Static Environment Process Flow

IV. Challenges

The major challenges with were faced during the implementation of a 3D simulation as the search algorithm. While generating obstacles while the program was being executed was possible in 2D, the corresponding spawning of obstacles in 3D at the location fed by the user was a major challenge faced. Although the robot motion in Gazebo accounts for this dynamically added obstacle by changing its path, we do not spawn the obstacle in the Gazebo world. Although we tried using service calls to spawn, and using shell script execution at runtime, this was not resolved.

Also, the selection for maximum number of nodes was done by trial and error as some values failed to reach the goal. This parameter requires tuning when the maps are changed, and also when the number of obstacles added are large, because of which a large number of intermediate nodes are required to reach the goal node.

V. CONCLUSIONS

This implementation of RRT* focuses on the space and time complexity allowing for near real time implementation even in higher-dimensional spaces. This is important in cases when the time-to-response is constrained to avoid accidents that may occur due to search delay.

ACKNOWLEDGMENT

Authors would like to thanks , Dr. Reza Monfaredi of University of Maryland College Park for their constant encouragement towards the realization of this work.

VI. REFERENCES

- Adiyatov and H. A. Varol, ”A novel RRT-based algorithm for motion planning in Dynamic environments,” 2017 IEEE International Conference on Mechatronics and Automation (ICMA), 2017, pp. 1416-1421, doi: 10.1109/ICMA.2017.8016024.
- Chengren Yuan, Guifeng Liu, Wenqun Zhang, Xinglong Pan, An efficient RRT cache method in dynamic environments for path planning, Robotics and Autonomous Systems, Volume 131, 2020, 103595, ISSN 0921-8890,
- Xu, J., Park, KS. A real-time path planning algorithm for cable-driven parallel robots in dynamic environment based on artificial potential guided RRT. Microsyst Technol 26, 3533–3546 (2020). https://doi.org/10.1007/s00542-020-04948-w
- S. Liu, H. Liu, K. Dong, X. Zhu and B. Liang, ”A Path Optimization Algorithm for Motion Planning with the Moving Target,” 2018 IEEE International Conference on Mechatronics and Automation (ICMA), 2018, pp. 2126-2131, doi: 10.1109/ICMA.2018.8484607.