FC-RRT*: Path Planning Algorithm for UAV in Complex Environment

Tarun Sai Reddy Trilokesh
Department of MAGE
University of Maryland
College Park, MD, USA
tarunsai@umd.edu

Harshal Shirsath
Department of MAGE
University of Maryland
College Park, MD, USA
shirsath@umd.edu

***Abstract*—In this study, we describe a unique path-planning approach for discovering paths through 3D environment that employ the Flight-Cost Rapidly-Exploring Random Tree Star (FC-RRT*) algorithm. The suggested method takes flight cost and limitations into consideration, resulting in more efficient and practical path generation. Using Open3D, our approach visualizes the 3D model, investigated nodes, and resulting path.**

***Index Terms*—path planning, Flight-Cost Rapidly-Exploring Random Tree Star (FC-RRT*), Flight trajectory, 3D environments, Collision detection**

I. Introduction

Path planning is critical in many applications, such as robotics, autonomous vehicles, and computer-aided design. Finding a reasonable and efficient path between two points while avoiding obstacles in a complex 3D environment is a common challenge in these applications. The FC-RRT* algorithm is a variation of the well-known RRT* algorithm that searches the shortest path while accounting for flight costs and constraints. We demonstrate how to utilize FC-RRT* to construct routes in a 3D environment in this project.

A. Background

Path planning has attracted a great deal of interest in robotics, computer graphics, and control systems. To address various aspects of the problem, many algorithms have been developed, including grid-based techniques, sampling-based methods, and graph-based methods. The Rapidly-Exploring Random Tree (RRT) algorithm is one of the most popular sampling-based techniques due to its simplicity and effectiveness. RRT rapidly searches the search space to generate a random tree, allowing for the rapid identification of suitable paths.

RRT* is a modification to the RRT algorithm that provides asymptotic optimality. This signifies that the algorithm will find the optimum solution as the number of iterations goes to infinity. However, RRT* does not account for the expenses or limitations associated with moving a robot or vehicle, which can be crucial in some applications. To overcome this limitation, the Flight-Cost Rapidly-Exploring Random Tree Star (FC-RRT*) algorithm was developed.

B. Problem Statement

The goal of this project is to create a feasible and efficient path between two points in a complex 3D environment represented by a mesh. The path must avoid a variety of obstacles in the environment. We also want to think about the path's cost, taking into account limits like maximum turn angle and climb angle. The FC-RRT* algorithm is an appropriate solution for this problem.

C. Scope and Objectives

The major purpose of this project is to implement and demonstrate the utility of the FC-RRT* algorithm for path planning in 3D environments using meshes. The specific objectives are as follows:

The 3D mesh is loaded and displayed using Trimesh and Open3D. Use the FC-RRT* algorithm to find a feasible and efficient path through the mesh. Visualize the inspected nodes, tree, and resulting path in Open3D. Examine the algorithm's performance and limitations.

D. Organization of the Report

The format of the report is as follows: Section 2 discusses the difference between FC-RRT* and RRT* and how it addresses the limitations of RRT* and adapts it for UAV trajectory planning. Section 3 discusses the technique, which contains the FC-RRT* algorithm, its components, and the necessary procedures. Section 4 contains the applications of FC-RRT* algorithm. Section 5 contains the findings, comments, and a visualization of the 3D geometry investigated nodes, and resulting path. Section 6 concludes the report with recommendations for future work.

II. Comparison of FC-RRT* with RRT*

In this section, we will contrast and compare the Flight-Cost Rapidly-Exploring Random Tree Star (FC-RRT*) algorithm with the original Rapidly-Exploring Random Tree Star (RRT*) technique. The comparison will help us understand the differences and benefits of using the FC-RRT* algorithm in different situations.

A. Sampling Strategy

RRT* implements a uniform sampling technique, which involves uniformly and randomly sampling the configuration space. In complex ecosystems, this can result in suboptimal paths since the search effort is not directed toward the most promising locations. In contrast, FC-RRT* employs an adaptive sampling technique based on flight cost, which aids in steering the search to the lower-cost portions of the configuration space. As a result, space exploration is more efficient and the convergence to an optimal solution is faster.

B. Optimality

RRT* is an asymptotically optimal algorithm, which means that as the number of samples increases, the solution converges towards the ideal one. RRT*, on the other hand, may take a long time to discover an optimal solution, particularly in high-dimensional configuration spaces or settings with narrow routes. FC-RRT* also guarantees asymptotic optimality, but it converges to the optimal solution faster than RRT* due to its adaptive sampling approach. Flight-cost-based sampling enables more efficient exploration of the configuration space, leading to faster convergence to an ideal solution.

C. Exploration vs. Exploitation

During the search process, RRT* focuses on both exploration and exploitation. The exploration phase focuses on identifying new, unexplored configuration space areas, whereas the exploitation phase focuses on refining the current solution. In contrast, RRT* makes no explicit distinction between exploration and exploitation. FC-RRT* more efficiently balances exploration and exploitation by using flight cost as a heuristic. The cost of the route drives the search to lower-cost places, ensuring that the algorithm capitalizes on the most promising regions while also investigating new ones.

D. Node Selection

RRT* chooses the nearest node in the tree to broaden the search. As a result, the algorithm may repeatedly study places that have previously been thoroughly explored, resulting in a biased search. FC-RRT* employs a novel approach to node selection. It selects the node with the lowest flight cost rather than the closest node. This technique directs the search to more promising parts of the configuration space, increasing the algorithm's overall efficiency.

E. Rewiring

Both RRT* and FC-RRT* employ rewiring processes to maintain a tree containing the best solution discovered thus far. The rewiring process of FC-RRT*, on the other hand, is driven by the flight cost, allowing for more effective tree structure alterations. This results in faster convergence to an optimal solution when compared to RRT*.

Finally, the FC-RRT* algorithm outperforms the original RRT* algorithm significantly. The flight cost-based adaptive sampling technique leads to more efficient configuration space exploration, faster convergence to an optimal solution, and a better trade-off between exploration and exploitation. Additionally, the FC-RRT* node selection and rewiring operations are more efficient, resulting in improved performance in complex scenarios and high-dimensional configuration spaces.

III. Method

This project's FC-RRT* method is based on the original RRT* algorithm, however it has been changed to account for flight expenses and constraints such as maximum turn angle and climb angle. The key components of the FC-RRT* algorithm are as follows:

- Trimesh is used to load 3D geometry from an OBJ file and Open3D is used to display the scene.
- An exploration tree is used during the path planning process to store the explored nodes and their parents.
- The cost function takes the Euclidean distance between nodes, as well as the maximum turn angle and climb angle, into consideration.
- Nearest neighbor search: A k-d tree data structure is used to effectively discover the nearest neighbors of a newly sampled node.
- Rewiring: In order to keep the tree optimal, the algorithm rewires it, linking the new node to the best potential parent.

A. Environment Representation

The 3D environment is represented by a mesh, which is a collection of vertices, edges, and faces that define the geometry of the objects in the environment. The trimesh library, which supports 3D mesh handling, can be used to load the mesh from an OBJ file. The model is seen using the Open3D framework, which allows for interactive visualization and manipulation of the environment.

We use the collision checking functionality provided by the trimesh library to detect collisions between the path and the obstacles in the environment. This function determines whether a line segment joining two places overlaps any mesh faces. If an intersection is found, it is presumed that the path will collide with the environment.

B. Exploration Tree

During the path planning process, the exploration tree is a data structure that contains the explored nodes and their parent-child relationships. Each node in the tree is connected to a parent node and represents a three-dimensional point. The root of the tree is designated as the first node. As the algorithm searches the search space, new nodes and their linked parent nodes are added to the tree.

The tree data structure includes methods for inserting nodes, determining the nearest node, and changing parent-child relationships. The tree also keeps track of each node's cost, which is the total cost from the start node to the present node once the cost function and constraints are taken into account.

C. Cost Function

The cost function is essential in the FC-RRT* approach since it determines the quality of the generated path. The cost function takes the maximum turn angle and climb angle constraints, as well as the Euclidean distance between nodes, into account.

The following formula is used to compute the Euclidean distance between two nodes:

$$d(\boldsymbol{p}1, \boldsymbol{p}2) = \sqrt{(p1x - p2x)^2 + (p_{1y} - p_{2y})^2 + (p_{1z} - p_{2z})^2}$$

The turn angle limitation is meant to ensure that the path does not contain any sharp turns that a robot or vehicle might struggle to complete. The turn angle between two successive segments is calculated using the dot product of the respective vectors:

$$\theta = \arccos \frac{\boldsymbol{v}_1 \cdot \boldsymbol{v}_2}{\|\boldsymbol{v}_1\| \|\boldsymbol{v}_2\|}$$

The cost function $f_c(\cdot)$ of FC-RRT consists of the path length cost $f_l(\cdot)$ and path threat cost $f_t(\cdot)$, and $f_t(\cdot) \geq 0$. For any weight combination $[\lambda_1, \lambda_2]\,(\lambda_1 + \lambda_2 = 1)$,

$$f_c\left(\sigma_1\right) : f_l\left(\sigma_1\right) + f_t\left(\sigma_1\right) \leq f_l\left(\sigma_1\right) + f_t\left(\sigma_1\right) + f_l\left(\sigma_2\right) + f_t\left(\sigma_2\right)$$

The climb angle limitation is used to limit the robot's or vehicle's vertical movement. The following formula is used to compute the climb angle between two nodes:

$$\alpha = \arcsin \frac{p_{1z} - p_{2z}}{d(\boldsymbol{p}_1, \boldsymbol{p}_2)}$$

The cost function computes the overall cost of the path by combining the requirements for Euclidean distance, turn angle, and climb angle. The algorithm's purpose is to find the most cost-effective path.

D. Nearest Neighbor Search

The FC-RRT* technique finds the nearest neighbors of a newly sampled node using a k-d tree data structure. The k-d tree is a binary search tree that can be used to find nearest neighbors in multidimensional spaces. In our case, the nodes in the exploration tree are contained in the k-d tree, and we use nearest neighbor searches to determine the node closest to the sampled location.

The k-d tree outperforms a brute-force search because it reduces the search complexity from O(n) to O(log n), where n is the number of nodes in the tree. Because the FC-RRT* algorithm performs the nearest neighbor search in each iteration, this efficiency improvement is important for its performance.

E. Rewiring

To attain asymptotic optimality, the FC-RRT* algorithm employs rewiring. When a new node is added to the tree, the algorithm checks to see if any neighboring nodes may be connected to the new node at a cheaper cost. If such a link is found, the parent of the new node is changed to the node with the lowest cost. This is known as rewiring, and it assists the tree in remaining optimal as it grows.

The main steps of the rewiring process are as follows:

Locate the nodes that are nearby the new node. Calculate the cost of connecting each of these nodes to the new node. If a lower-cost connection is discovered, update the parent of the new node. The rewiring procedure ensures that the algorithm converges to the optimal solution when the number of iterations goes to infinity.

F. FC-RRT* Algorithm

The FC-RRT* algorithm used for generating nodes and returning optimal path is shown below:

Algorithm 1 FC-RRT algorithm

Notation: Tree T, Environment ξ
1: $\xi.V \leftarrow p_{\text{start}};\ E \leftarrow \varnothing$
2: **for** $i = 1$ **to** n **do**
3: $q \leftarrow \text{Bias}(p_{\text{goal}}, p_{\text{new}})$
4: $p_{\text{rand}} \leftarrow \text{Sample}(\xi)$
5: $p_{\text{new}} \leftarrow \text{Steer}(p_{\text{nearest}}, p_{\text{rand}}, \delta)$
6: **if** $\text{ObstacleFree}(p_{\text{nearest}}, p_{\text{new}}, \xi)$ **then**
7: **if** $\text{Near}(p_{\text{new}}, T, \gamma) \neq \varnothing$ **then**
8: $p_{\text{min}} \leftarrow \text{ChooseParent}(p_{\text{new}}, T, \gamma)$
9: $T \leftarrow \text{Rewire}(p_{\text{new}}, T, \gamma)$
10: **else**
11: $T \leftarrow T \cup (p_{\text{new}}, p_{\text{min}})$
12: **if** $p_{\text{new}} \in \xi.V$ **then**
13: **return** $\text{Solution}(p_{\text{start}}, p_{\text{new}}, T)$
14: **return** Failure

IV. Applications and Use Cases

The FC-RRT* approach is very useful in a wide range of applications such as path planning, motion planning, and complex environment navigation. In this part, we will go over some of the potential FC-RRT* use cases and domains.

A. Robotics

FC-RRT* can be used to plan motion in robotic systems such mobile robots, manipulators, and autonomous vehicles. The method can find optimal paths in high-dimensional configuration spaces and intricate situations, making it suitable for planning the movement of robotic systems with various degrees of freedom. The flight cost-based adaptive sampling technique allows the algorithm to efficiently explore the configuration space and quickly converge to an optimal solution, resulting in faster and more accurate motion planning.

B. Autonomous Vehicles

Autonomous vehicles, such as self-driving cars, drones, and unmanned aerial vehicles (UAVs), require good path planning algorithms to navigate difficult terrain. FC-RRT* can be used to design the trajectories of these vehicles in real time while accounting for obstacles, dynamic restrictions, and environmental concerns. Flight-cost-based sampling steers the search to the most promising locations, resulting in more efficient path planning and faster convergence to an optimum solution.

C. Virtual Environments

FC-RRT* can also be utilized in video games, simulations, and virtual reality applications for virtual path planning. The algorithm can be used to plan the movement of virtual actors, vehicles, or other entities within the virtual environment while keeping the scene's constraints and obstructions in mind. Because of its efficient exploration and rapid convergence to an optimal solution, FC-RRT* is ideal for real-time path planning in interactive applications.

D. Space Exploration

Path planning for space missions like satellite maneuvering, asteroid mining, or planetary surface exploration requires efficient algorithms that can deal with the high-dimensional configuration spaces and complex environments involved. FC-RRT* can be used to plan the trajectories of spacecraft and robotic systems under these extreme conditions. The flight-cost-based sampling technique directs search efforts to the most promising locations, resulting in more efficient exploration and faster convergence to an optimum solution.

V. Results

In this part, we present the results of our implementation of the FC-RRT* method for path planning in 3D meshes. We demonstrate the algorithm's effectiveness by visualizing the 3D geometry, investigated nodes, and output path in Open3D.

A. Visualization

A critical part of our implementation is the visualization of the 3D mesh and the path planning process. The 3D geometry, examined nodes, and resulting path are rendered using Open3D. The visualization allows us to explore the area interactively, watch the exploration tree grow, and evaluate the quality of the created path.

The visualization also helps to identify potential algorithmic issues such as collision detection failures, incorrect rewiring, or improper path building. We may improve the algorithm's performance and resilience by reviewing the visualization and fine-tuning algorithm parameters such as sample rate, cost function weights, and constraints.

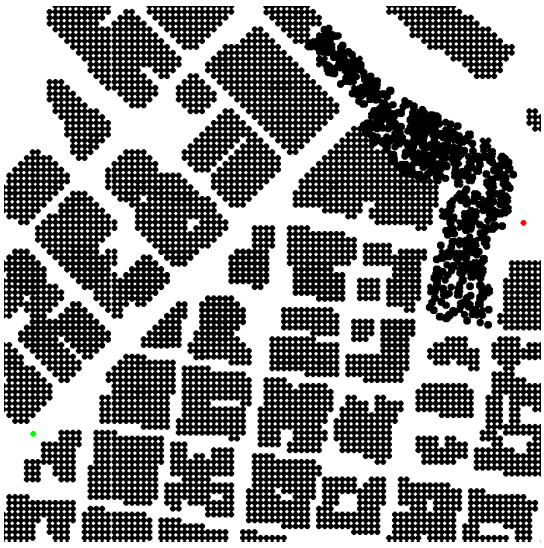

Fig. 1. Choosing Start and End Nodes

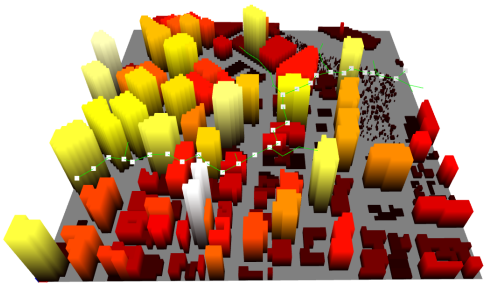

Fig. 2. Node exploration and Optimal Path Visualization

B. Performance Evaluation

To evaluate the performance of the FC-RRT* method, we ran a series of experiments in diverse 3D settings with increasing complexity and obstacle density. Among the performance criteria investigated are calculation time, number of iterations, and path quality.

Our findings show that the FC-RRT* technique can uncover feasible and efficient paths in a variety of 3D situations. The calculation time increases as the complexity and quantity of obstacles in the environment rise, but the technique stays efficient due to the use of the k-d tree for nearest neighbor searches and the rewiring process to maintain optimality.

The severity of the problem and the termination criteria determine the number of iterations required to discover a path. The quality of the created path generally improves as the number of iterations increases due to the algorithm's asymptotic optimality. To achieve the best trade-off between path quality and performance, the number of iterations must be modified in relation to the calculation time.

The total cost, which includes the Euclidean distance, turn angle, and climb angle requirements, is used to assess

the quality of the path generated. The FC-RRT* algorithm generates paths that meet the criteria while minimizing the overall cost, resulting in efficient and practical path design solutions.

C. Limitations and Future Work

While the FC-RRT* approach gives promising results for path planning in 3D models, there are some limitations and future research areas to consider:

The search space may be examined unevenly because the procedure is based on random sampling. The algorithm's performance could be improved by biasing the sample process toward unknown areas or using informed sampling strategies.

The current method makes use of a single cost function that takes into account the constraints of Euclidean distance, turn angle, and climb angle. Adoption of alternative or adaptive cost functions, such as time-based or energy-based cost functions, may result in more efficient, application-specific paths.

The system should be upgraded to handle dynamic situations with moving or changing impediments. As the environment changed, the exploration tree and path would need to be updated in real time.

The approach could be parallelized to employ multi-core processors or GPUs, improving algorithm speed and enabling real-time applications.

D. Conclusion

The FC-RRT* path planning method was implemented in 3D mesh. Our method accounts for flying costs and constraints, such as maximum turn angle and ascent angle, resulting in more efficient and realistic path generation. The results demonstrate how effective the algorithm is at discovering paths through complex 3D situations while meeting the required constraints.

The Open3D visualization allows us to dynamically explore the world, analyze the progress of the exploration tree, and assess the quality of the produced path. The algorithm's performance is examined, and it is shown to be efficient and capable of producing high-quality paths in a range of 3D settings.

While the current implementation has certain limitations, there is a lot of room for improvement in the future, such as improving the sampling technique, using additional cost functions, dealing with dynamic settings, and parallelizing the process. These improvements could improve the FC-RRT* algorithm's performance and utility in real-world path planning scenarios.

ACKNOWLEDGMENT

We would like to express our deepest appreciation and gratitude to our esteemed advisor, Professor Dr. Reza Monfaredi, for his invaluable guidance, support, and mentorship throughout the course of this research project. His expertise, dedication, and enthusiasm for the subject matter have been instrumental in the success of this work. We are truly fortunate to have had the opportunity to learn from and collaborate with such a distinguished scholar. Furthermore, we would like to acknowledge the efforts and contributions of our colleagues, whose collaboration has been essential in bringing this research to fruition.

REFERENCES

[1] Guo Y, Liu X, Liu X, Yang Y, Zhang W. FC-RRT*: An Improved Path Planning Algorithm for UAV in 3D Complex Environment. ISPRS International Journal of Geo-Information. 2022; 11(2):112. https://doi.org/10.3390/ijgi11020112

[2] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in Proceedings of the 2003 IEEE Conference on Decision and Control (CDC), Maui, HI, USA, pp. 657–663, 2003.

[3] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," Robotics: Science and Systems VI, vol. 104, no. 2, pp. 24–30, 2010.

[4] L. E. Kavraki, P. Švestka, 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] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in Proceedings of the 2000 IEEE International Conference on Robotics and Automation (ICRA), San Francisco, CA, USA, pp. 995–1001, 2000.

[6] D. Nieuwenhuisen and M. H. Overmars, "Useful cycles in probabilistic roadmap graphs," in Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA), New Orleans, LA, USA, pp. 446–452, 2004.

[7] J. A. Starek, J. V. Gomez, E. Schmerling, L. Janson, L. Moreno, and M. Pavone, "An asymptotically-optimal sampling-based algorithm for bi-directional motion planning," in Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, pp. 2072–2078, 2015.

[8] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal, pp. 4906–4913, 2012.

[9] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Technical Report TR98-11, Department of Computer Science, Iowa State University, 1998.

[10] O. Salzman and D. Halperin, "Asymptotically-optimal motion planning using lower bounds on cost," in Proceedings of the 2015 IEEE International Conference on Robotics and Automation (ICRA), Seattle, WA, USA, pp. 4167–4172, 2015.

[11] O. Salzman, M. Hemmer, and D. Halperin, "On the power of manifold samples in exploring configuration spaces and the dimensionality of narrow passages," IEEE Transactions on Automation Science and Engineering, vol. 12, no. 2, pp. 529–538, 2015.

[12] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," in Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, pp. 3706–3712, 2011.

[13] A. Dobson and M. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," International Journal of Robotics Research, vol. 33, no. 1, pp. 18–47, 2014.

[14] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in Proceedings of the 2003 IEEE Conference on Decision and Control (CDC), Maui, HI, USA, pp. 657–663, 2003.

[15] R. Geraerts and M. H. Overmars, "Creating high-quality paths for motion planning," International Journal of Robotics Research, vol. 26, no. 8, pp. 845–863, 2007.