P2a: Tree Planning Through The Trees

Shakthibala Sivagami Balamurugan
Robotics Engineering
Worcester Polytechnic Institute
Email: sbalamurugan@wpi.edu

Aditya Patwardhan
Robotics Engineering
Worcester Polytechnic Institute
Email: apatwardhan@wpi.edu

Late Day Used: 1

***Abstract*—The aim of the project is to traverse a known 3d map from a start point to the stop point. To achieve this we develop a path planner with RRT*, a trajectory generator for the planned path with cubic spline and tune PID gains in cascaded controller where outer loop controls position and inner loop controls velocity.**

I. Map Visualization

The known maps are stored in .txt files with the axes boundary limit information of the environment and the cuboid obstacles. The obstacles are colored with rgb, The boundary and blocks are given as :

TABLE I
Simulation Parameters

Type	x	y	z	w	h	d	Color (RGB)
boundary	0	0	0	45	35	6	–
block	1.0	1.0	1.0	3.0	3.0	3.0	(0, 255, 0)

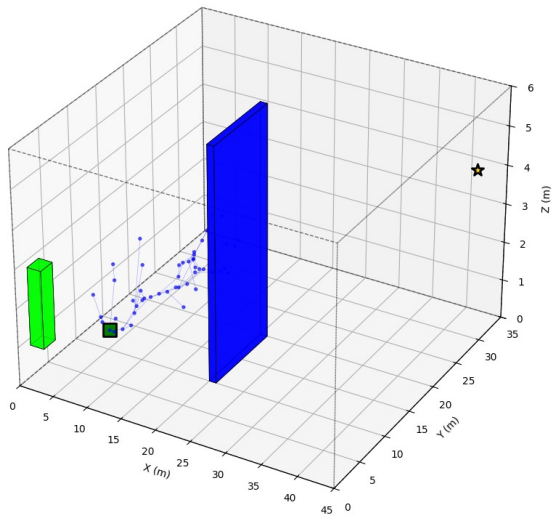

Fig. 1. Orthographic view of Map 4 - Train

II. Path Planning using RRT*

The algorithm for path planning using RRT* is as follows:

Algorithm 1: RRT* (Rapidly-exploring Random Tree*)

Input: sampling radius r, max iterations n
Output: graph $G = (V, E)$
$Rad \leftarrow r$;
$G \leftarrow (V, E)$; // Graph containing vertices and edges
for $itr \leftarrow 0$ **to** $n - 1$ **do**
  $X_{\text{new}} \leftarrow \text{RandomPosition}()$;
  if $\text{Obstacle}(X_{new})$ **then**
    ; // Try again if in an obstacle
    continue;
  end
  $X_{\text{nearest}} \leftarrow \text{Nearest}(G, X_{\text{new}})$;
  $Cost(X_{\text{new}}) \leftarrow Cost(X_{\text{nearest}}) + \text{Distance}(X_{\text{new}}, X_{\text{nearest}})$;
  $(X_{\text{best}}, X_{\text{neighbors}}) \leftarrow \text{findNeighbors}(G, X_{\text{new}}, Rad)$;
  $Link \leftarrow \text{Chain}(X_{\text{new}}, X_{\text{best}})$;
  foreach $x' \in X_{neighbors}$ **do**
    if $Cost(X_{new}) + \text{Distance}(X_{new}, x') < Cost(x')$ **then**
      $Cost(x') \leftarrow Cost(X_{\text{new}}) + \text{Distance}(X_{\text{new}}, x')$;
      $Parent(x') \leftarrow X_{\text{new}}$;
      $G \leftarrow G \cup \{(X_{\text{new}}, x')\}$; // Add edge to the graph
    end
  end
  $G \leftarrow G \cup Link$; // Add the new link to the graph
end
return G;

III. Trajectory Generation

By applying the RRT* global path planning algorithm, we can determine a shortest path from the Start to the Goal within the map environment. However, the initial path often contains sharp turns, making it dynamically infeasible for execution, as the quadrotor would tend to overshoot. To address this, the discrete waypoints obtained are transformed into a smooth spline trajectory. We employed a cubic polynomial trajectory

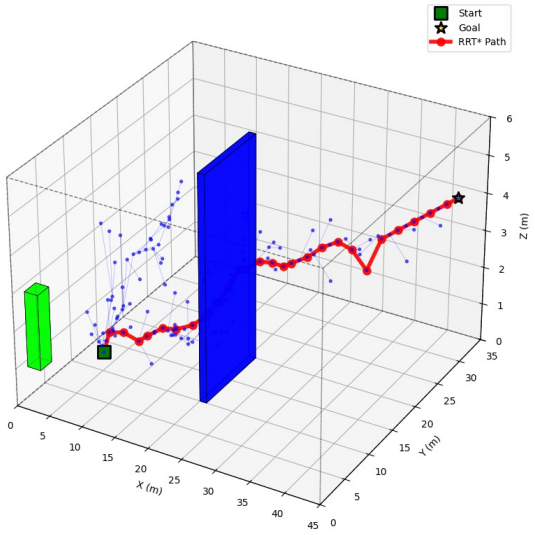

Fig. 2. Orthographic view of RRT* planner

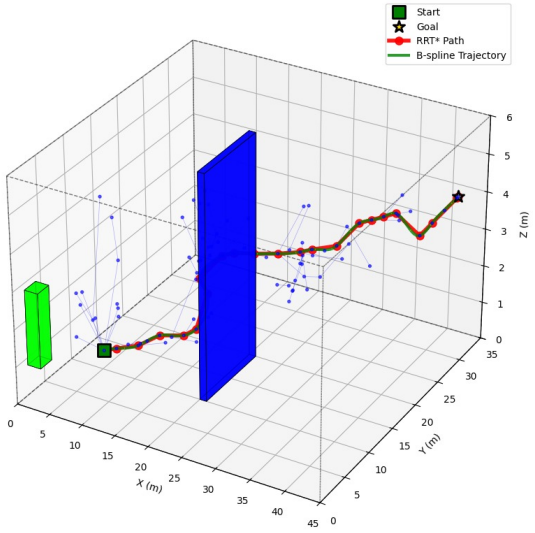

Fig. 3. Cubic Spline trajectory generation

to achieve this, resulting in smoother and more natural motion, which reduces mechanical stress on the robot or vehicle while enhancing efficiency and safety. Through experimentation, we identified that an average velocity of 2 m/s provided the most reliable performance for following the generated trajectories.

A. Collision Handling

We are handling collision detection by calculating if the current position of the quadrotor is in free space and not in an obstacle or outside the boundary. If a collision is detected, the quadrotor halts its motion and the simulation stops.

IV. Cascaded Controller Design

The controller is designed in a cascaded manner in which there is an outer position control loop and then there is an inner velocity control loop. Both of these are PID controllers. The

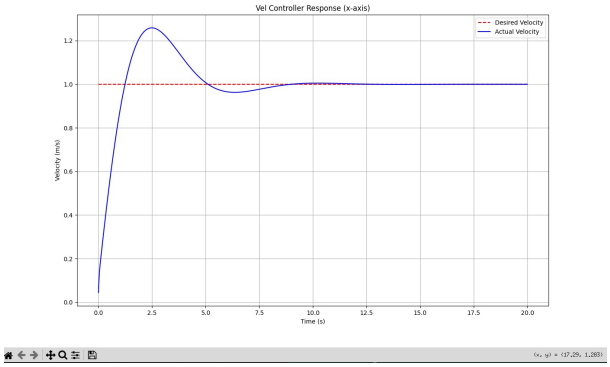

Fig. 4. Tuned response of Controller

controllers are tuned in a step-by-step manner. First, the velocity controller is tuned since it is the inner control loop. The PID tuning values are: (1.3,0.2,0.2) for x-direction; (1.3,0.2,0.2) for y-direction and (20,0.1,0.1) for z-direction. When the velocity controller is satisfactorily tuned, the position control loop is tuned. The PID values are: (1,0,0) for the xdirection; (1,0,0) for the y-direction and (0.4,0.06,0.1) for the z-direction.

TABLE II
PID Controller Parameters

Parameter	K_p	K_i	K_d
position_x	1	0.0	0
position_y	1	0.0	0
position_z	1.4	0.0	0.1
velocity_x	1.3	0.2	0.2
velocity_y	1.3	0.2	0.2
velocity_z	20	0.1	0.1

V. Assumptions

- The boundary of `Map3` is modified $\{x_{\min}, y_{\min}, z_{\min}\}$ from $\{0.0,\ 0.0,\ 0.0\}$ to $\{-0.1,\ 0.0,\ 0.0\}$.
- It is assumed that Quadrotor is a point object for Collision Handling.

VI. Results and Analysis

The RRT* planner provides about 29 points for path planning and B-spline Trajectory gives 785 waypoints for Quadcoptor to navigate, With Drone velocity of 3m/s and Simulation time of 25 sec, The success rate to reach goal is 98.2% in map4, 98.1% in map1, 98.1% in map2 .

VII. Conclusion

We compute collision-free paths using RRT*, then apply B-spline parameterization for waypoint generation and smoothing of the quadrotor trajectory. Closed-loop performance after tuning 18 PID gains achieves 98.1% success on Map-1 , 98.2% on Map-4, 98.1% on Map-2.

VIII. Supplementary Material

The videos of the complementary filter along with its comparison against ground truth is given: bit.ly/482grWh

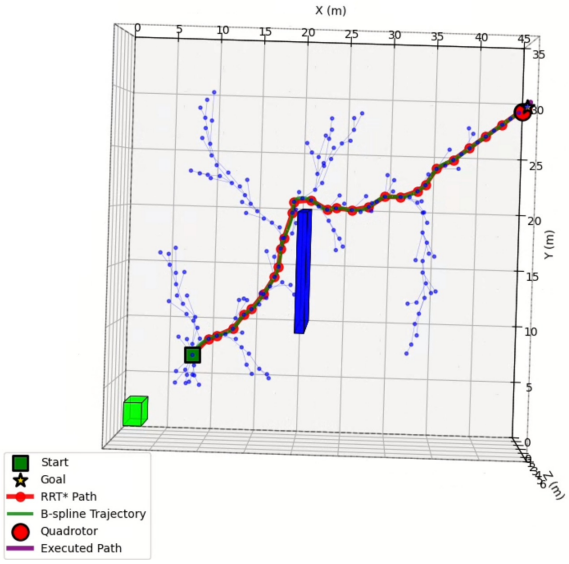

Fig. 5. Executed Trajectory map4

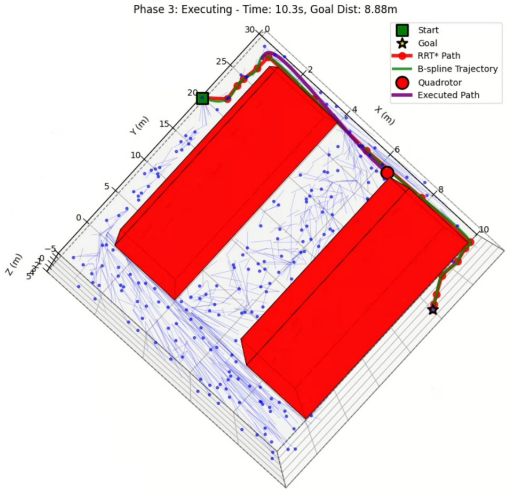

Fig. 7. Executed Trajectory map2

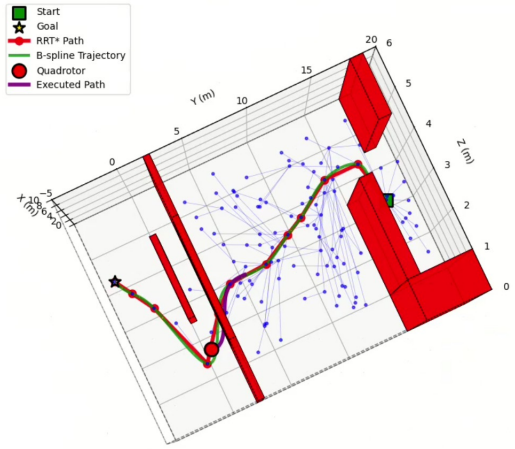

Fig. 6. Executed Trajectory map1

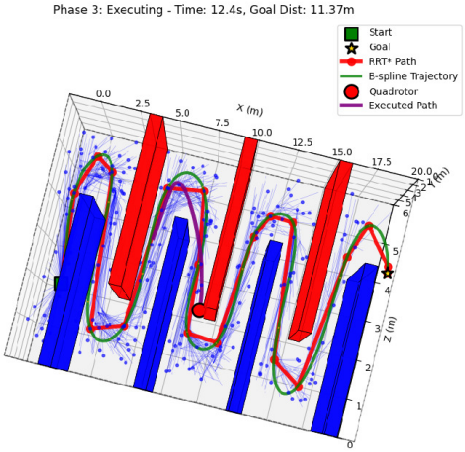

Fig. 8. Executed Trajectory map3

REFERENCES

[1] PX4 Autopilot, "AttitudeControl.hpp," GitHub, 1c1f8da7d9cc416aaa53d76254fe08c2e9fa65e6, [Online]. .

[2] R. Baker, "D-D Splines," UCLA Department of Mathematics, [Online].

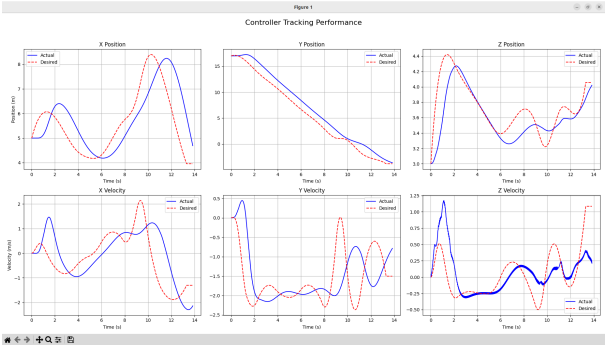

Fig. 9. Position and Velocity plots for Map1

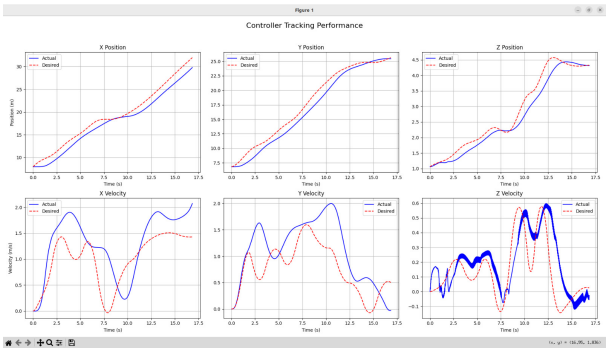

Fig. 10. Position and Velocity plots for Map2

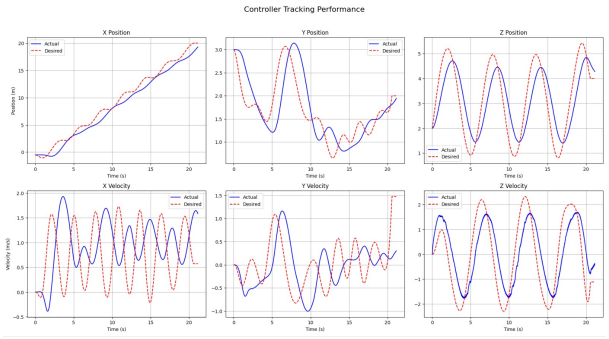

Fig. 11. Position and Velocity plots for Map3

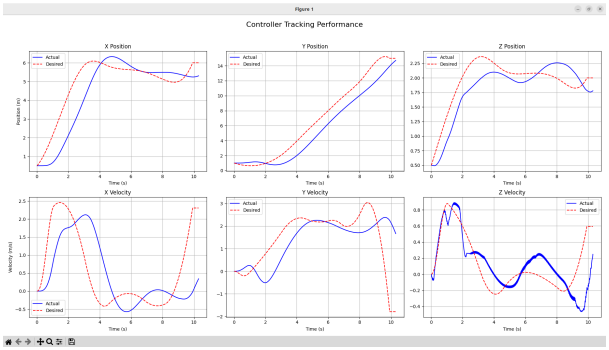

Fig. 12. Position and Velocity plots for Map4