EE581 Optimal Control Project Report

Group Members:

Ratnesh Yadav (PSU ID: 962052529)

Dhruva Uplap (PSU ID: 941955375)

Title: Adaptive Quadcopter Trajectory Planning via Sampling-Based Algorithms

1. Motivation:

In the rapidly evolving realm of automation, quadcopters emerge as a pinnacle of technological advancement, demonstrating exceptional agility and self-sufficiency. Initially popularized for recreational purposes, these versatile aerial machines have now transcended into pivotal roles across various sectors. Their applications span from intricate aerial monitoring and infrastructure evaluation to the rapid transportation of goods. At the heart of these expanding functionalities is the crucial process of motion planning—the algorithmic backbone that dictates the self-directed flight paths of quadcopters. This process is integral to developing routes that not only fulfill operational objectives but also prioritize safety, efficiency, and adaptability to dynamic and unpredictable environments.

This project underscores an academic initiative aimed at crafting and simulating pathfinding technologies tailored for quadcopters within controlled environments. It emphasizes the adoption of sampling-based algorithms to develop practical and optimized flight trajectories. These algorithms are designed to be responsive to both static obstacles, enhancing the quadcopters' operational efficiency and safety. By integrating academic research with practical drone applications, this simulation endeavors to push the boundaries of current drone capabilities through comprehensive algorithmic evaluation and testing.

Our methodology adopts a structured approach that begins with a precise definition of the problem followed by the creation of a specialized simulation environment. This environment serves as the testing ground for iterative assessments and enhancements of various motion planning algorithms. By employing established computational strategies, such as Dijkstra's and A*, the project aims to refine the trajectory planning capabilities of quadcopters. These efforts highlight how theoretical models can be transformed into effective, real-world applications that improve the navigational abilities of drones, particularly in complex environments laden with obstacles.

Moreover, the project's focus on sampling-based algorithms shows the efficiency in handling the stochastic nature of real-world scenarios where environmental unpredictability is common. These algorithms offer a robust framework for dynamically adjusting flight paths in real time, a critical feature for operations requiring high levels of precision and adaptability.

2. Objective:

The primary objective of this project is to design and simulate an adaptive trajectory planning system for quadcopters using sampling-based algorithms. The focus is to enhance the quadcopter's ability to navigate efficiently and safely through static environments. This will involve the creation of a simulated environment where various motion planning algorithms can be implemented and rigorously tested. The project aims to demonstrate the practicality of these algorithms in real-time navigation and obstacle avoidance, providing a robust solution that can be scaled and adapted to different operational needs. Through this simulation, the project seeks to bridge the gap between theoretical knowledge gained in class and its application in solving real-world problems that involve complex dynamic systems.

An integral part of the project is the development of a dynamic model of the quadcopter, which includes accurately modeling its physical behavior and integrating this model within a virtual testing environment. This environment will mimic real-world conditions, including potential obstacles and environmental variables that the quadcopter might encounter during flight. The use of MATLAB will be essential for creating this simulation framework, allowing for detailed analysis and adjustment of the algorithms. By customizing the motion planning algorithms to fit the specifics of the modeled quadcopter, the project will evaluate the performance of these algorithms across various simulated scenarios. This comprehensive approach ensures not only the validation of the concept but also the refinement of the algorithms to optimize pathfinding efficiency and safety.

To further this objective, the project will also involve extensive simulation trials to assess the adaptability of the trajectory planning algorithms under different environmental scenarios. Ultimately, the aim is to produce a set of validated, efficient, and adaptable algorithms that could be implemented in real-world quadcopter applications.

3. Methodology:

3.1. Vehicle Rotor Model

The first step involves developing a comprehensive vehicle model that accurately represents the dynamics and physical properties of a quadcopter. This model is crucial for simulating real-world behaviors and interactions within a virtual environment. By leveraging the principles of physics that govern aerial dynamics, such as lift, drag, and thrust, alongside the quadcopter's mass distribution and motor characteristics, we establish a foundation for realistic simulations. The model is constructed in MATLAB, which facilitates complex computations and dynamic visualizations essential for real-time analysis and debugging. This high-fidelity representation allows for the precise evaluation of different trajectory planning algorithms under a variety of simulated conditions, ensuring that our simulations closely mirror potential real-world applications.

The key parameters assumed for the rigid body dynamics of the quadrotor are as follows:

(a) mass: m = 0.030 kg;

(b) the distance from the center of mass to the axis of a motor: L = 0.046 m; and

(c) the components of the inertia dynamics:

$$[IC] = \begin{matrix} 1.43 * 10^{-5} & 0 & 0 \\ 0 & 1.43 * 10^{-5} & 0 \\ 0 & 0 & 1.43 * 10^{-5} \end{matrix}$$

Each rotor has an angular speed ω_i and produces a vertical force F_i according to

$$F_i = k_F \omega_i^2.$$

Experimentation with a fixed rotor at steady-state shows that kF ≈ 6.11 × 10−8 N/rpm2. The rotors also produce a moment according to

$$M_i = k_M \omega_i^2.$$

This motor gain, k_m, is found to be about 20 s^{-1} by matching the performance of the simulation to the real system. The desired angular velocities, ω^{des}, are limited to a minimum and maximum value determined through experimentation.

3.2. Controller Dynamics

The second major component of our methodology is the development of controller dynamics, which are essential for executing the computed trajectories efficiently and accurately. The controller interprets the trajectory planning algorithms' output, translating them into actionable commands that adjust the quadcopter's flight path in real time. This involves the design of both a high-level path planner and a low-level stabilizing controller, which work in tandem to ensure optimal performance. The high-level controller focuses on trajectory planning, using sampling-based algorithms to generate feasible paths that avoid obstacles and meet predefined objectives. The low-level controller, meanwhile, stabilizes the flight based on feedback from the quadcopter's onboard sensors, maintaining the desired attitude and position against disturbances.

Our controller dynamics also incorporate robust fault-tolerance mechanisms to handle potential failures or unexpected environmental changes. By designing the controllers to be adaptive, they can recalibrate and compensate for sensor inaccuracies or motor failures, thereby enhancing the reliability of the quadcopter in critical applications. This adaptability is achieved through advanced algorithms such as Kalman filters, which continuously learn from the quadcopter's performance and adjust improve control accuracy and system resilience.

```
% Proportional and derivative gains for position and attitude control
kp = [7, 7, 58];
kd = [4.3, 4.3, 18];
kpt = [2500, 2500, 20];
kdt = [300, 300, 7.55];

% Desired acceleration
acc_des = qd{qn}.acc_des - diag(kd)*(qd{qn}.vel - qd{qn}.vel_des) - diag(kp)*(qd{qn}.pos - qd{qn}.pos_des);

% Desired angles (roll, pitch) calculated to achieve desired acceleration
phi_des   = (sin(qd{qn}.yaw_des)*acc_des(1) - cos(qd{qn}.yaw_des)*acc_des(2)) / g;
theta_des = (cos(qd{qn}.yaw_des)*acc_des(1) + sin(qd{qn}.yaw_des)*acc_des(2)) / g;
psi_des   = qd{qn}.yaw_des;

% Control input u for thrust and torques
u = zeros(4,1);
u(1) = acc_des(3)*m + m*g;  % Total thrust

% Angular error and angular velocity error
ang_error = angdiff([phi_des; theta_des; psi_des], qd{qn}.euler);
ang_vel_error = qd{qn}.omega - [0; 0; qd{qn}.yawdot_des];

% Torque calculation
u(2:4) = I * (-diag(kpt) * ang_error - diag(kdt) * ang_vel_error);
```

Figure 1: Adaptive Control Computation

Additionally, the controllers are optimized for computational efficiency to ensure they can operate in real time on the hardware typically available in commercial quadcopters. This involves algorithmic optimizations and hardware-in-the-loop simulations to balance processing power and response time. Such optimizations are crucial for deploying these advanced control systems in practical applications, where delays or computational lags can compromise the mission's success.

3.3. Path Planning Algorithms:

In the domain of autonomous navigation, effective path-planning algorithms are critical for ensuring that a quadcopter can navigate through complex environments. This project utilizes Dijkstra's algorithm and the A* algorithm, which are fundamental for finding the shortest path between two points in a graph.

Dijkstra's algorithm, renowned for its efficiency and accuracy in finding the shortest path, operates on the principle of relentlessly exploring all possible paths until the optimal route is determined. This exhaustive approach ensures that no potential paths are overlooked, making it highly reliable for applications where precision is paramount. However, Dijkstra's method can be computationally intensive, as it does not prioritize which paths to explore based on proximity to the goal.

```
% dijkstra algorithm
while (~all(unvisited(:)==inf))
    % find the min dist pos from unvisited node
    dist_unvisited = dist.*unvisited;
    [~, id] = min(dist_unvisited(:));
    % mark visited
    unvisited(id) = inf;
    num_expanded = num_expanded + 1;
    % check 6-connected neighbors
    [i,j,k] = ind2sub([map.nx,map.ny,map.nz],id);
    for d = 1:size(dijk,1)
        nijk = bsxfun(@plus, int32([i,j,k]), int32(dijk(d,:)));
        % if neighbor (in the map) and (unvisited) and (unoccupied)
        if all(nijk > 0) & all(int32([map.nx, map.ny, map.nz]) >= int32(nijk)) ...
            & unvisited(nijk(1),nijk(2),nijk(3)) == 1 ...
            & map.occ_map(nijk(1),nijk(2),nijk(3)) ~= 1
            alt = dist(id) + sqrt(sum(dijk(d,:).^2));
            % Got rid of function call for optimization
            nid = (nijk(3)-1)*map.nx*map.ny + (nijk(2)-1)*map.nx + nijk(1);
            if alt < dist(nid)
                dist(nid) = alt;
                prev(nijk(1), nijk(2), nijk(3),:) = [i,j,k];
            end
        end
    end
    if id == (goal(3)-1)*map.nx*map.ny + (goal(2)-1)*map.nx + goal(1);
        break
    end
end
```

Figure 2: Dijikstra Algorithm

On the other hand, the A* algorithm introduces a heuristic into the pathfinding process, significantly enhancing efficiency by estimating the cost from the current node to the goal. This heuristic guides the search, enabling A* to focus on more promising paths and often reach the goal faster than Dijkstra's algorithm. By combining the reliability of Dijkstra's method with the efficiency of the A* heuristic, which often employs the Euclidean distance in our implementation, our project benefits from a balanced approach that is both accurate and computationally feasible. This dual-algorithm strategy ensures that the path-planning component of our system is robust enough to handle various operational scenarios encountered by the quadcopter.

```
% A* algorithm
fscore = inf(size(map.occ_map)); % init as inf distance
fscore(start(1), start(2), start(3)) = euclidean_heuristic(start, goal); % set start dist as 0
openset = inf(size(map.occ_map));
openset(start(1), start(2), start(3)) = 1;

while (~all(openset(:)==inf))
    % find the min point with lowest fscore in the openset
    openfscore = fscore.*openset;
    [~, id] = min(openfscore(:));

    if id == (goal(3)-1)*map.nx*map.ny + (goal(2)-1)*map.nx + goal(1);
        % if found the goal position, stop searching
        break
    end
    % remove current node from the open set
    openset(id) = inf;
    unvisited(id) = inf;
    num_expanded = num_expanded + 1;
    % check 6-connected neighbors
    [i,j,k] = ind2sub([map.nx,map.ny,map.nz],id);
    for d = 1:size(dijk,1)
        nijk = bsxfun(@plus, int32([i,j,k]), int32(dijk(d,:)));
        % if neighbor (in the map) and (unvisited) and (unoccupied)
        if all(nijk > 0) & all(int32([map.nx, map.ny, map.nz]) >= int32(nijk)) ...
            & unvisited(nijk(1),nijk(2),nijk(3)) == 1 ...
            & map.occ_map(nijk(1),nijk(2),nijk(3)) ~= 1
            % gscore is the distance from start to the neighbor via current
            % position
            nid = (nijk(3)-1)*map.nx*map.ny + (nijk(2)-1)*map.nx + nijk(1);
            gscore = dist(id) + sqrt(sum(dijk(d,:).^2));
            if openset(nid) == inf
                openset(nid) = 1;
            elseif gscore >= dist(nid)
                % then it is a longer path
                continue
            end

            % then this is the best path so far
            prev(nijk(1), nijk(2), nijk(3),:) = [i,j,k];
            dist(nid) = gscore;
            fscore(nid) = dist(nid) + euclidean_heuristic(nijk, goal);
        end
    end
end

end
if dist(goal(1), goal(2), goal(3)) == inf
```

Figure 3: A Algorithm*

Implementing these algorithms involves setting up a virtual grid in our simulation environment where each cell represents a potential node on the path. The quadcopter evaluates its movement from one node to another based on the cost associated with that move, which could be influenced by factors such as obstacles, restricted zones, or the energy required to make the move. The paths are constructed by backtracking from the goal to the start once the destination is reached, ensuring the entire route is the most efficient as per the algorithm's calculation. Our simulation tests these algorithms under different conditions to validate and fine-tune their performance, ensuring that the quadcopter can reliably follow the computed paths in real-world situations.

3.4. Simulations, Testing, and Results:

The final subsection focuses on rigorous simulation and testing to validate and refine both the vehicle model and the controller dynamics. This process begins with the creation of a virtual testing environment that simulates a wide range of operational scenarios. Each scenario is designed to challenge the quadcopter's capabilities and test the effectiveness of the trajectory planning and control algorithms under different conditions. Through iterative testing, we identify strengths and weaknesses in our designs and make necessary adjustments. The following results show the movements of a quadrotor in a 3D environment.

Keypoints to remember:

Greenpath: Desired Trajectory

RedPath: Achieved Trajectory

1. **Zigzag Motion in the horizontal plane**:

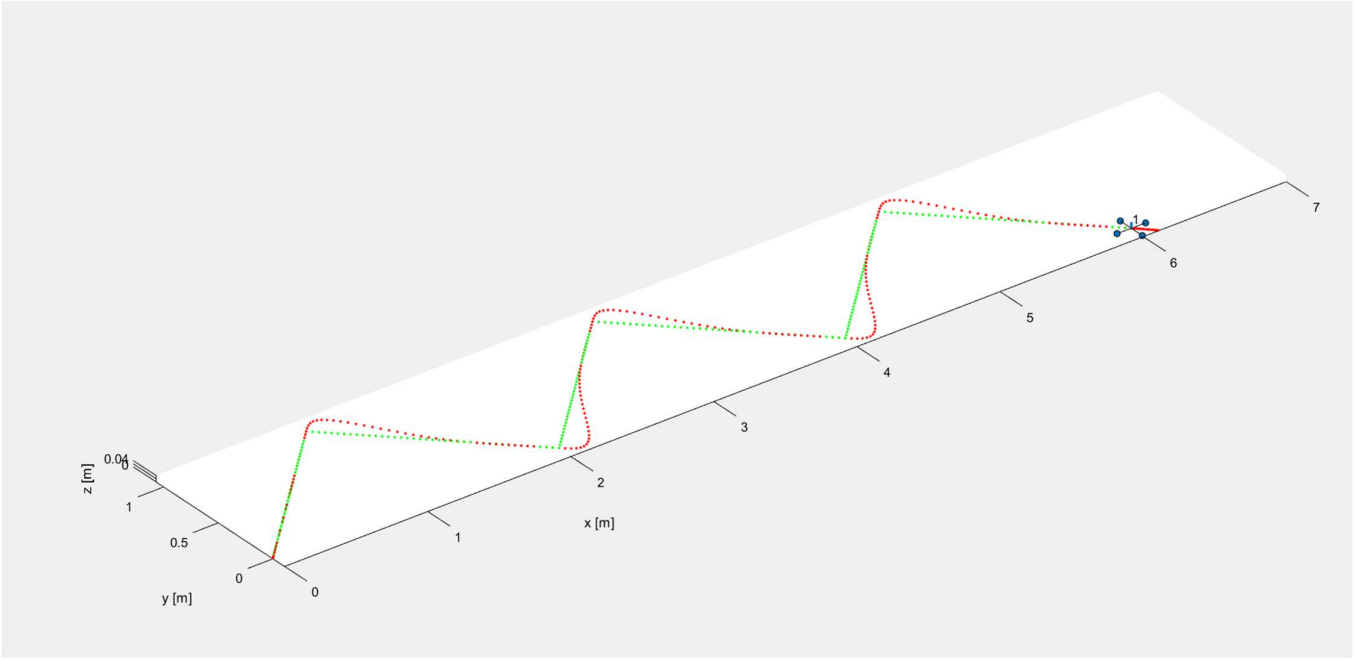

Figure 4: Motion in Horizontal Plane

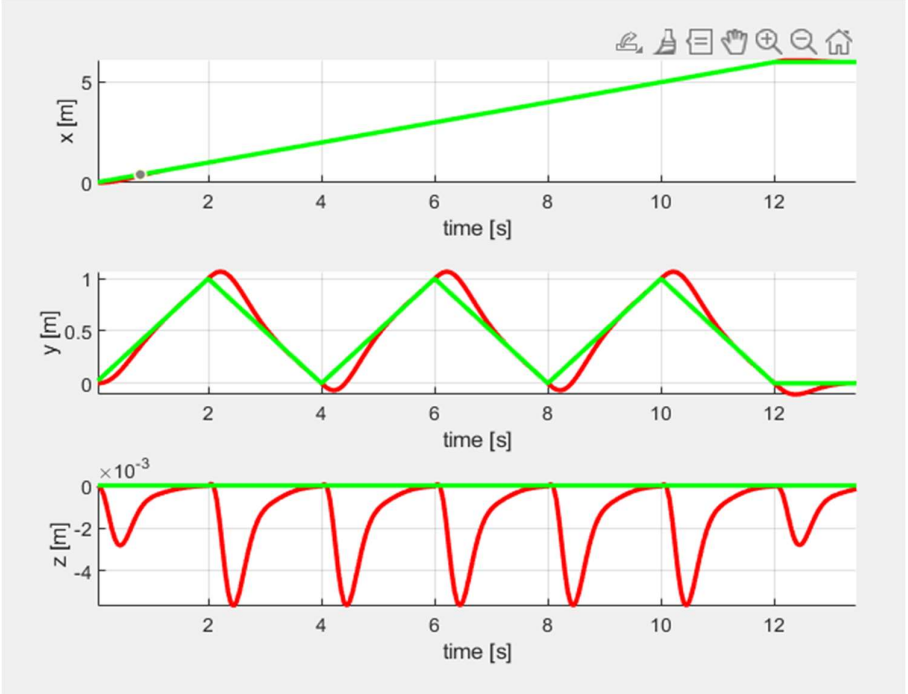

Figure 5: Positional Tracking

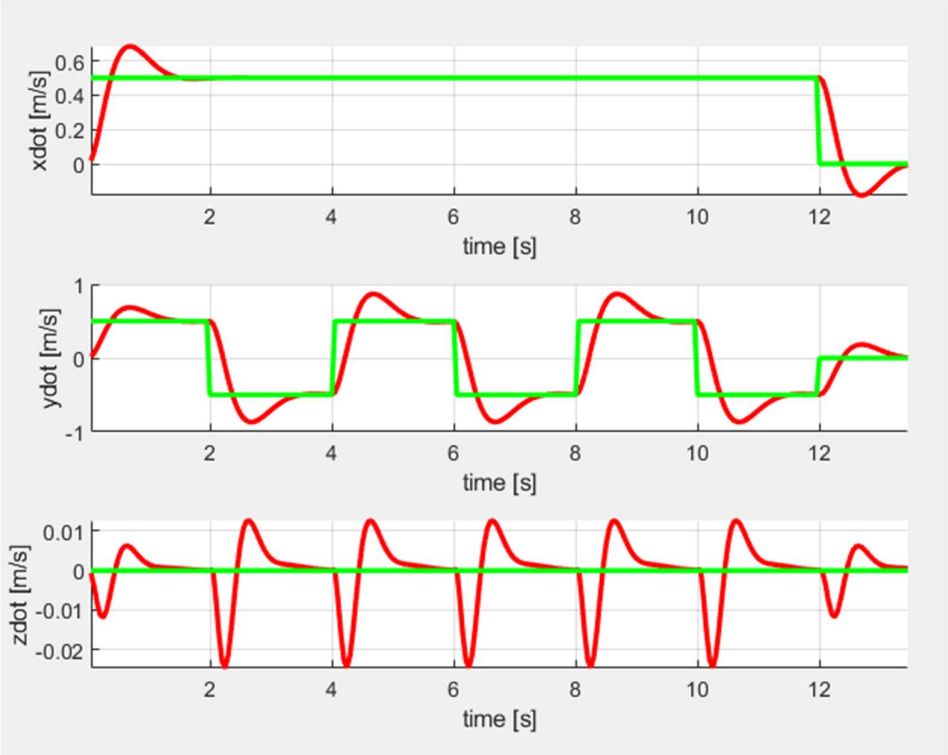

Figure 6: Velocity Tracking

2. **Square Motion in the Vertical plane**:

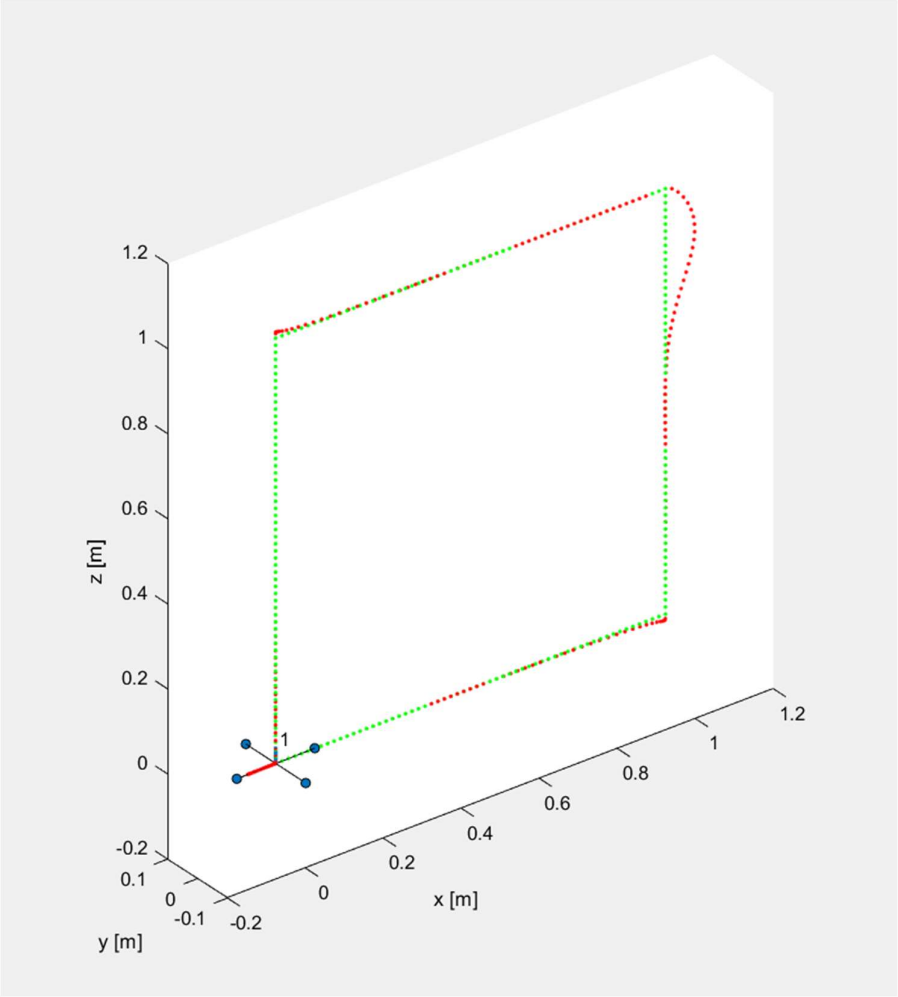

Figure 7: Motion in Vertical Plane

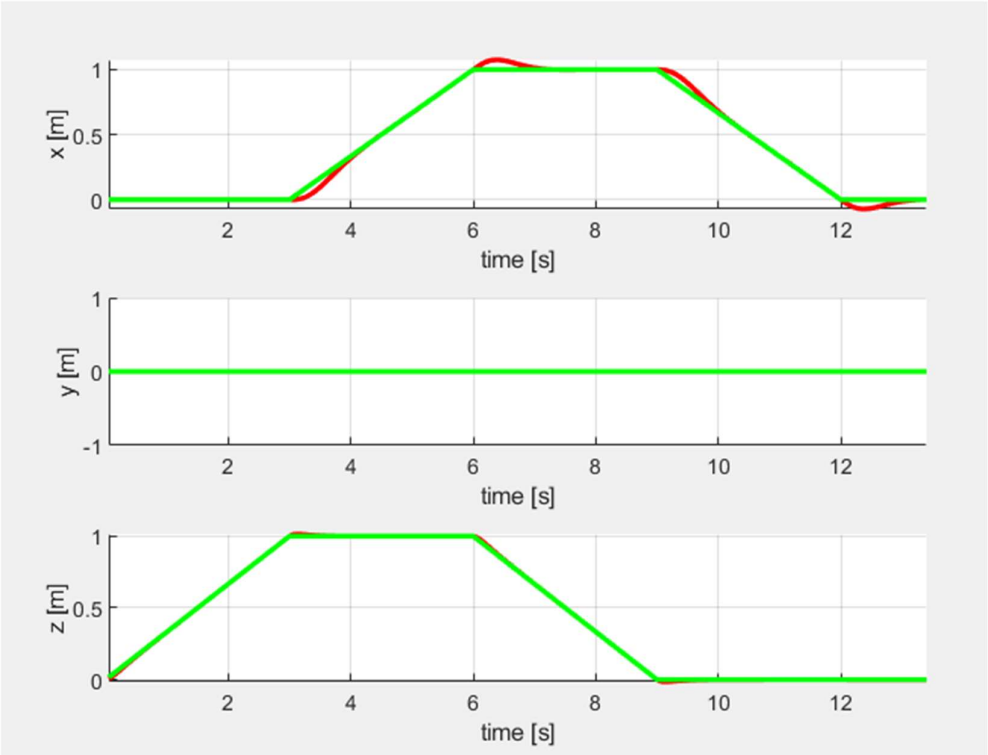

Figure 8: Positional Tracking

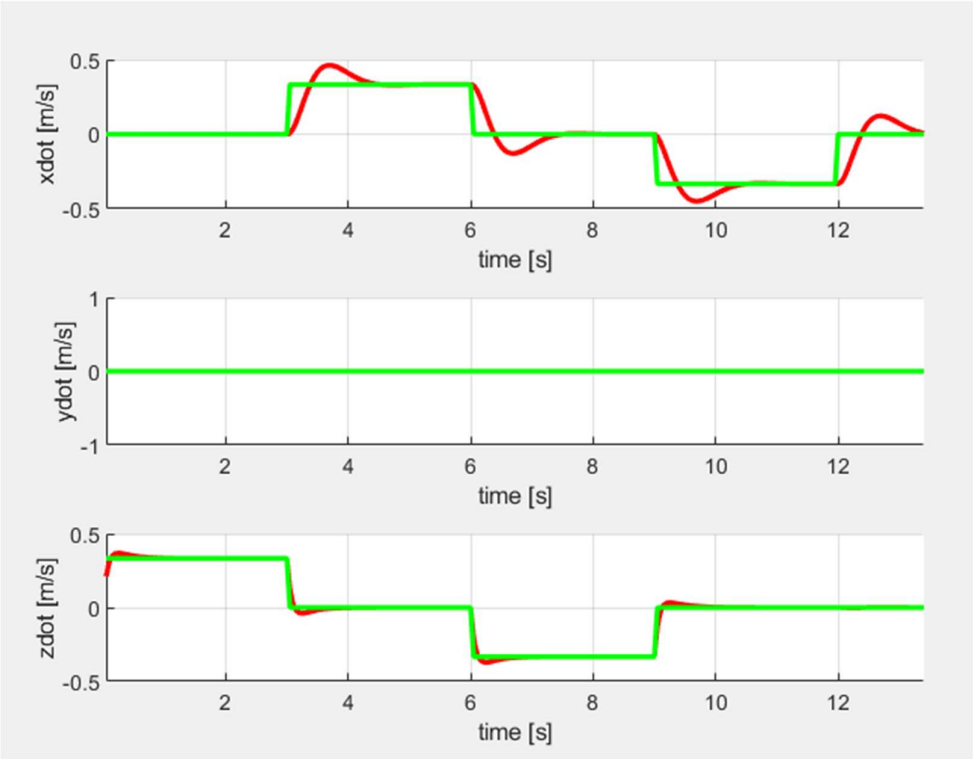

Figure 9: Velocity Tracking

The testing phase also includes stress tests that push the quadcopter to its operational limits, such as navigating through densely cluttered environments or recovering from sudden system failures. These tests are critical for ensuring that the quadcopter can handle real-world challenges safely and effectively. Moreover, the feedback gathered during these simulations is used to further refine the algorithms and controllers, enhancing their robustness and reliability.

We tried running the quadrotor through a customized map and track it's movement:

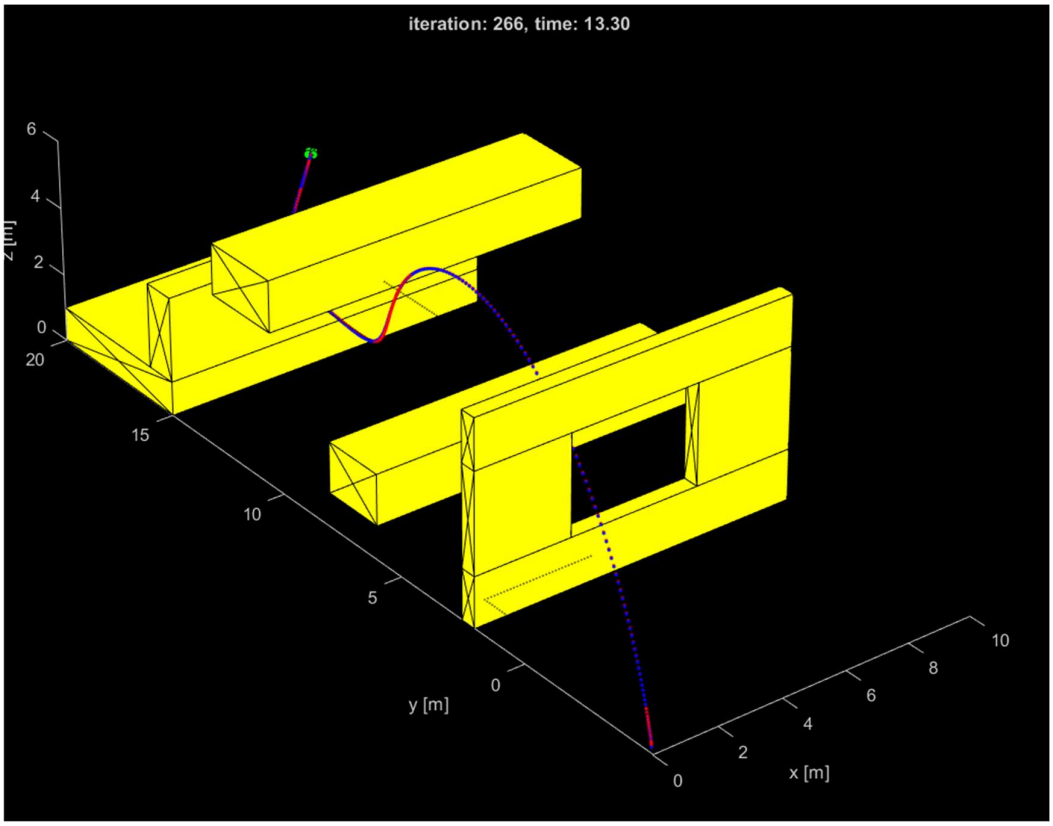

Figure 10: 3D Plane: Path Movement

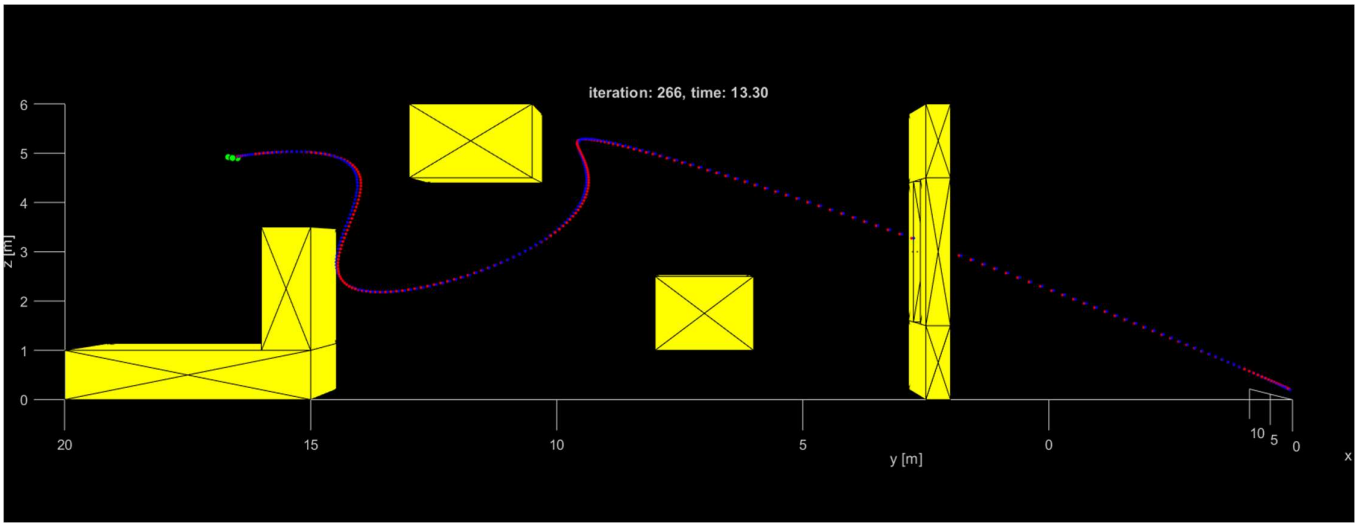

Figure 11: ZY Plane

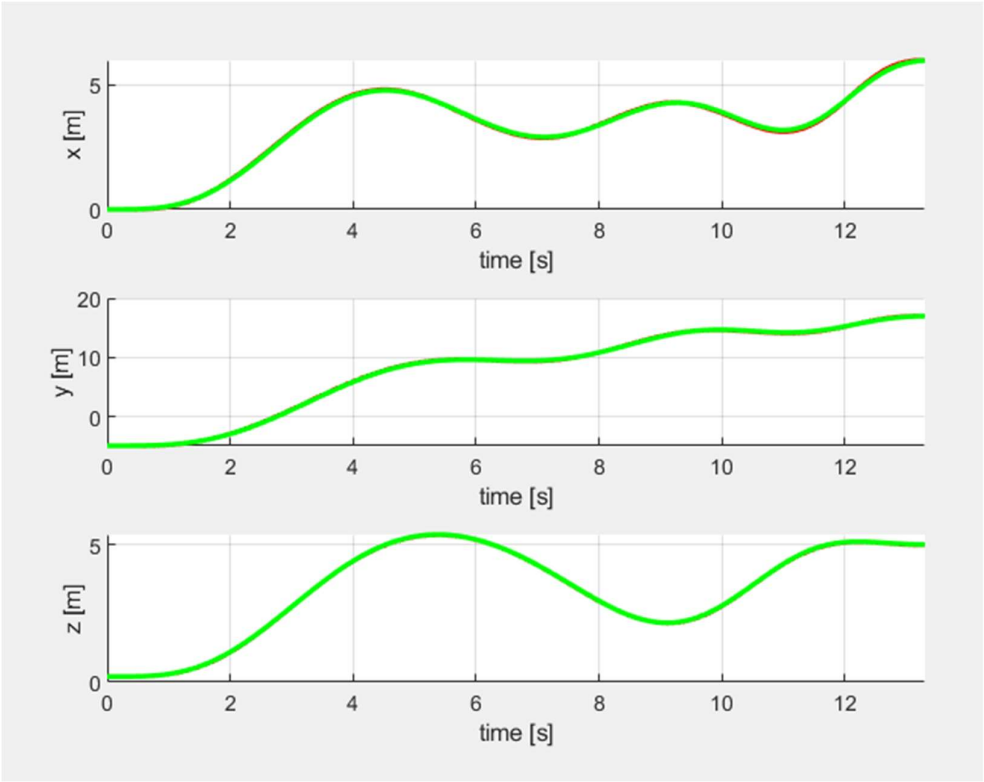

Figure 12: Position Tracking

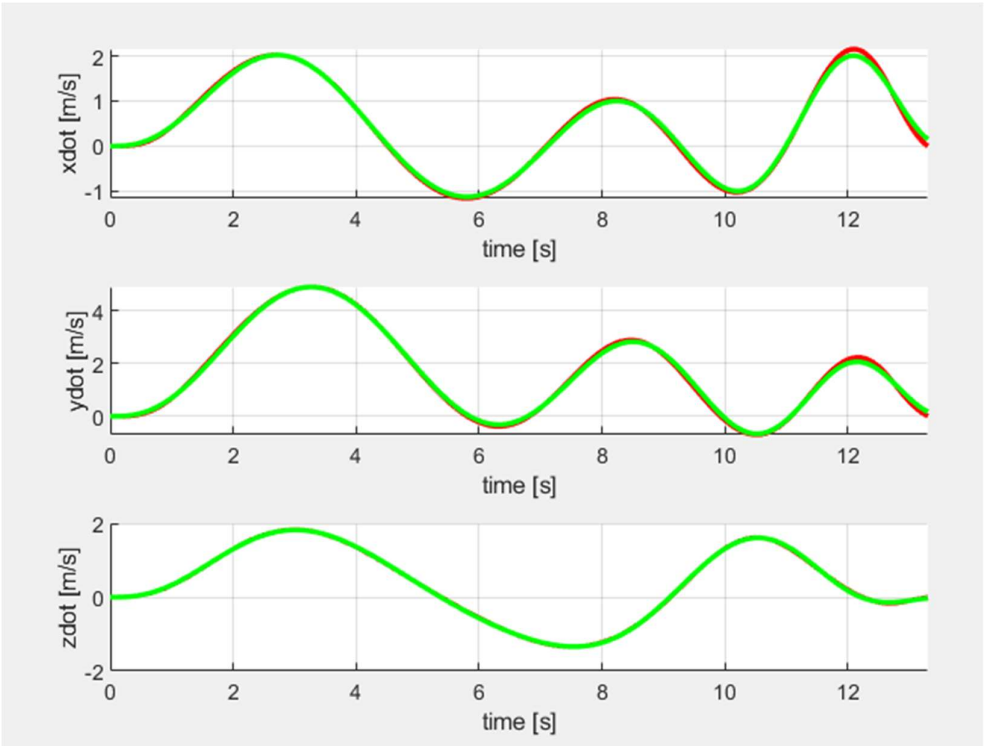

Figure 13: Velocity Tracking

4. Conclusion

Through meticulous design, simulation, and testing of both the vehicle model and controller dynamics, we have validated the effectiveness of using Dijkstra's and A* algorithms for real-time path planning. The integration of these algorithms into a comprehensive simulation environment has proven to enhance the quadcopter's navigational capabilities, enabling it to efficiently and safely maneuver through complex scenarios. The results from simulation trials underscore the robustness of our path planning system, highlighting its potential to adapt to unforeseen changes and maintain optimal performance under various operational conditions.

References

[1] S. Bouabdallah, “Design and control of quadrotors with application to autonomous flying,” Ph.D. dissertation, Ecole Polytechnique Federale de Lausanne, 2007.

[2] G. Hoffmann, S. Waslander, and C. Tomlin, “Quadrotor helicopter trajectory tracking control,” in AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii, Apr. 2008.

[3] T. Lee, M. Leok, and N. McClamroch, “Geometric tracking control of a quadrotor uav on SE(3),” in Proc. of the IEEE Conf. on Decision and Control, 2010.

[4] D. Mellinger and V. Kumar, “Minimum snap trajectory generation and control for quadrotors,” in Proc. of the IEEE Int. Conf. on Robotics and Automation, Shanghai, China, May 2011.

[5] D. E. Kirk, Optimal Control Theory. Dover Publications, 2004.