Crazyflie Experimental Report

Charles Yoo, Frederik Zhu, Shangzhi Le, Ryan Jacobowitz

I. Introduction and System Overview

In this lab, a controller, trajectory generator and path planner for a quadroptor was programmed and tested in a physical maze. A PD geometric nonlinear controller, A* path planner, and mini-snap trajectory optimizer was used on a Crazyflie 2.0 quadrotor. This program was tested first in a simulated environment and then in two lab periods. The first lab tested the controller, and mostly tested the quadrotor's ability to fly and stabilize to a given position. The second lab was done in a maze with three different starting and ending positions to ensure that the program was robust and efficient enough to work in different configurations. In both labs, the quadrotor position and velocity were recorded using a Vicon sensor system. One of the lab computers was dedicated to the Vicon system to capture the motion of the quadrotor while the other sent the trajectory information to the robot. The quadrotor was successful in traversing the maze in lab reliably and efficiently and is capable of safely controlling itself such that it reaches a particular position.

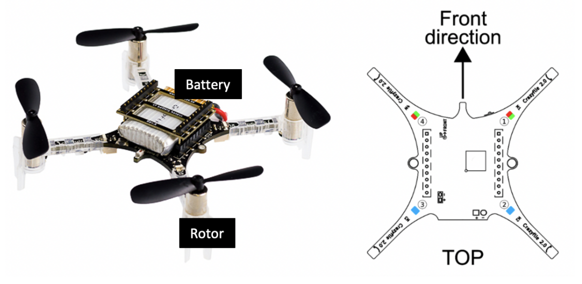

Fig. 1. Schematics of the Quadcopter

II. Controller

A geometric nonlinear controller was used for this project. This controller works based on the geometric intuition that the direction of the thrust and the b_3 axis of the quadcopter are aligned with the desired direction. The PD controller used was:

$$\ddot{r}^{des} = \ddot{r}_T - K_d(\dot{r} - \dot{r}_T) - K_p(r - r_T) \tag{1}$$

where r is the current measured state, r_T is the trajectory generated state, and K_p and K_d are controller gains. The 2 inputs for the quadcopter are the total thrust created by the motors in the inertial frame, u_1, and the moments created about the b_1, b_2, and b_3 axis of the quadcopter frame, u_2. The total thrust can be found using Newton's equation for force:

$$F^{des} = m\ddot{r}^{des} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = u_1 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2}$$

$$u_1 = b_3^T F^{des} \tag{3}$$

To find the 2nd input, we can first find the direction of the b_3^{des} by computing the direction of the force:

$$b_3^{des} = \frac{F^{des}}{\|F^{des}\|} \tag{4}$$

The b_2^{des} can be defined as perpendicular to the plane created by $b_3 des$ and the trajectory generated yaw direction:

$$b_2^{des} = \frac{b_3^{des} \times \begin{bmatrix} \cos(\psi_T) \\ \sin(\psi_T) \\ 0 \end{bmatrix}}{\|b_3^{des} \times \begin{bmatrix} \cos(\psi_T) \\ \sin(\psi_T) \\ 0 \end{bmatrix}\|} \tag{5}$$

where ψ_T is the trajectory generated yaw direction. The b_1^{des} is orthogonal to b_2^{des} and b_3^{des} and can be found by taking the cross product between the two. The desired rotation matrix for the quadcopter can be defined as:

$$R^{des} = [b_1^{des}, b_2^{des}, b_3^{des}] \tag{6}$$

The moments created about the axes of the quadcopter frame can then be found using the equation:

$$u_2 = I(-K_R e_R - K_\omega e_\omega) \tag{7}$$

where I is the inertial tensor, K_R and K_ω are controller gains, e_R is the error in orientation:

$$e_R = \frac{1}{2}(R^{des^T} R - R^T R^{des})^V \tag{8}$$

and e_ω is the error in angular velocities:

$$e_\omega = \omega - \omega^{des} \tag{9}$$

For the error in orientation, R is the measured orientation of the quadcopter and R^{des} is the desired orientation. The V operator works as follows:

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}^V = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \tag{10}$$

The final tuned gains for the controller were:

$$K_p = \begin{bmatrix} 5.81 & 0 & 0 \\ 0 & 5.81 & 0 \\ 0 & 0 & 5.81 \end{bmatrix} s^{-2} \quad (11)$$

$$K_d = \begin{bmatrix} 3.15 & 0 & 0 \\ 0 & 3.15 & 0 \\ 0 & 0 & 3.15 \end{bmatrix} s^{-1} \quad (12)$$

$$K_R = \begin{bmatrix} 152.6 & 0 & 0 \\ 0 & 152.6 & 0 \\ 0 & 0 & 152.6 \end{bmatrix} s^{-2} \quad (13)$$

$$K_w = \begin{bmatrix} 14 & 0 & 0 \\ 0 & 14 & 0 \\ 0 & 0 & 14.7 \end{bmatrix} s^{-1} \quad (14)$$

The k_p and k_d gains affect the positional movement and the k_R and k_w gains affect the angular movement of the quadcopter. The k_p and k_R gains are the proportional components and affect how quickly the quadcopter reaches the desired state. Increasing these gains will decrease the rise time but increase overshoot of the response. The k_d and k_w gains are the derivative components and act as a dampening factor to reduce overshoot caused by the proportional gains. Increasing these gains decrease both the overshoot and the settling time of the response.

There were a few differences between the simulation and hardware. Firstly, for the simulation, the torques and thrust of the quadcopter motors could be directly commanded. However, for the hardware, the torque and thrust cannot be directly commanded. Rather the motor speeds are what is directly commanded. Therefore, the desired torque and thrust computed by the controller must be converted to motor speeds. Another adjustment that had to be made to the controller between simulation and hardware was to reduce the gains by 30%. One possible reason why the gains had to be reduced is because of motor saturation. For simulation this is not an issue, but for hardware this could cause the motors to overheat beyond safe temperatures.

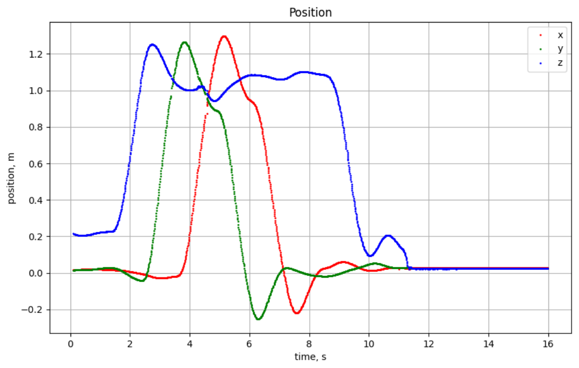

Fig. 2. Quadrotor Response from Lab 1

Figure 2, shows the response of the PD controller used in lab. Using the following equation for logarithmic decrement:

$$\delta = \frac{1}{n} \ln \frac{x(t)}{x(t+nT)} \quad (15)$$

$$\zeta = \frac{\delta}{\sqrt{4\pi^2 + \delta^2}} \quad (16)$$

The damping ratio is 0.066, the rise time is 0.75 seconds, and the settling time is 4.5 seconds. The steady state error is about 0.02 meters in the x and y directions, and 0.20 meters in the z direction. The steady state error cannot be stated as a percentage as the desired value at the end is 0. There is also a percent overshoot of about 20%.

III. Trajectory Generator

In this project, the trajectory is mainly based on 2 parts: path planning based on A* and trajectory optimization based on mini-snap and computed A* path waypoints. Waypoints were generated in a given map using an A* algorithm. Since it found a valid path in an environment with obstacles through A*, all the waypoints needed to be selected in this path.

The team tried 2 methods to choose waypoints: keeping all waypoints from the A* algorithm or using Ramer–Douglas–Peucker algorithm to do the post processing to remove some redundant points. It turned out that they both worked well. In the experiment in lab, all of the A* points were saved and it worked well. To allocate time between waypoints, for simplicity, a constant velocity was used and the time segment between waypoints was just distance divided by velocity.

The trajectory generator part was created using a 7th order polynomial since it is based on mini-snap which allowed the robot to follow a smoother path. There were several constraints this generator needed to follow and to solve for every coefficient in closed form:

1) All the segments start and end points need to be waypoints in A* path.

$$\begin{bmatrix} t^7 \\ t^6 \\ t^5 \\ t^4 \\ t^3 \\ t^2 \\ t^1 \\ 1 \end{bmatrix}^T \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = coord$$

2) The start and goal point derivative constraints need to be set. Here the velocity, acceleration and jerk are all set to 0.

$$\begin{bmatrix} 7t^6 \\ 6t^5 \\ 5t^4 \\ 4t^3 \\ 3t^2 \\ 2t \\ 1 \\ 0 \end{bmatrix}^T \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = 0 \ , \quad \begin{bmatrix} 42t^5 \\ 30t^4 \\ 20t^3 \\ 12t^2 \\ 6t \\ 2 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = 0 \ ,$$

$$\begin{bmatrix} 210t^4 \\ 120t^3 \\ 60t^2 \\ 24t \\ 6 \\ 0 \\ 0 \\ 0 \end{bmatrix}^T \begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = 0$$

3) All the derivatives in the end point of previous segment and the start point of latter segment had to be the same to ensure the smoothness and continuity of the trajectory. For each point except start and end point, the big A matrix is shown below:

$$\begin{bmatrix} 7t^6 & 42t^5 & 210t^4 & 840t^3 & 2520t^2 & 5040t \\ 6t^5 & 30t^4 & 120t^3 & 360t^2 & 720t & 720 \\ 5t^4 & 20t^3 & 60t^2 & 120t & 120 & 0 \\ 4t^3 & 12t^2 & 24t & 24 & 0 & 0 \\ 3t^2 & 6t & 6 & 0 & 0 & 0 \\ 2t & 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -720 \\ 0 & 0 & 0 & 0 & -120 & 0 \\ 0 & 0 & 0 & -24 & 0 & 0 \\ 0 & 0 & -6 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$$

If there are n segments, then there would be a total of 8n coefficients. Based on all the constraints above, 1 unique group of coefficients can be solved for and hence generate the trajectory. The smoothness and feasibility are guaranteed since it has continuity constraints. The following figure shows this smoothness demonstrating the position, velocity, acceleration, jerk and snap vs time.

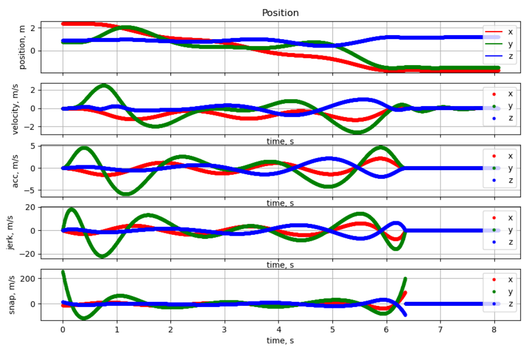

Fig. 3. Derivatives vs Time

IV. Maze Flight Experiments

During the lab period, the quadcopter flew through a maze three times with three different starting and ending points. The quadcopter was able to reach all three end points from each of the start points. For the first and third maps a margin of 0.3 was used and for the second map a margin of 0.3 was used. This was because the second map would not solve properly with the higher margin. This was possibly because the higher margin meant no path was possible by the quadcopter. All three graphs produced similar results, and the position and velocity graphs for the first map configuration can be seen in Figure 7.

The difference between the x, y, and z positions of the expected trajectory versus the actual trajectory is very small as can be seen in Figure 8. There is an offset, however, in the z position, where the trajectory follows closely to the expected z, but with an offset throughout the path. This resulted in a hard (although still successful) landing, since the quadcopter was higher in the z direction than expected in the final position. To decrease these errors, the controller and trajectory parameters would need to be further optimized.

The path chosen was not the most aggressive path, as for the second map specifically, there was a quicker path the robot could have taken. It is possible that a lower margin or resolution could have been chosen that would lead to a more optimal path. Lowering the resolution did make the trajectory planner take longer to run, which was not ideal, and lowering the margin too much caused the robot to collide with the walls of the maze. There needs to be a balance between the aggressiveness of the trajectory planner and the stability and safety of the robot on its course. Too low of a margin will cause collisions. The trajectory planner was optimized such that these collisions didn't occur, so it would be difficult to make the trajectory planner more aggressive without running into these issues.

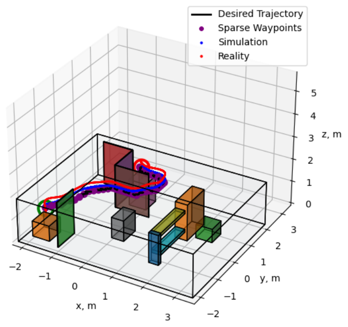

Fig. 4. Map 1 Trajectory

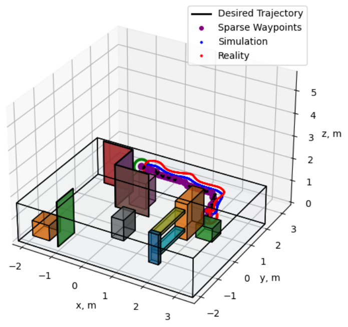

Fig. 5. Map 2 Trajectory

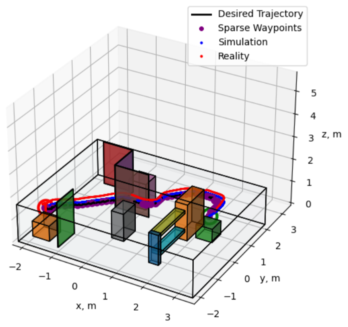

Fig. 6. Map 3 Trajectory

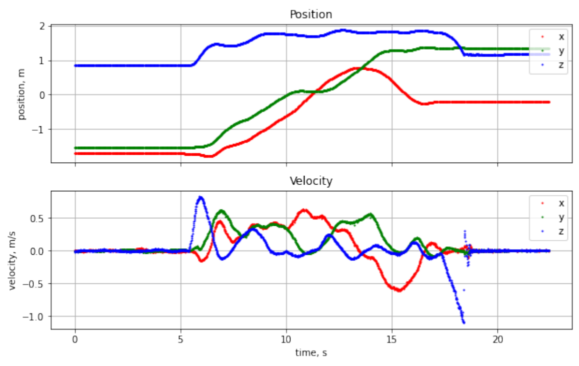

Fig. 7. Map 1 Position and Velocity vs Time

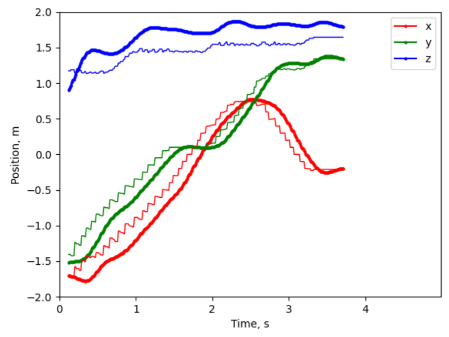

Fig. 8. Map 1 Position Error

In order to increase the speed of the robot, the robot could have taken a more aggressive path, as described earlier, or the speed of the robot could be increased. Increasing the speed, however, would increase the chance of instability and might cause the robot to overshoot waypoints. Lowering the speed would increase the reliability and improving the control parameters would make the robot move more smoothly.

Given another lab session, it would be interesting to try more complicated mazes. It could be interesting to land and then take off again and see if the robot behaves as expected in this case. It would be interesting to further test the margin, resolution, speed, and controller parameters to fine tune them to produce a smoother curve, and to remove some waypoints to create a smoother trajectory.

V. Conclusion

The quadcopter was successful in reaching the end points from each starting point in the maze and was able to closely follow the trajectory planned in simulation. The quadcopter was able to do so with small errors in the position and velocity. Further optimization could be done to ensure any errors in the trajectories were decreased, and decreasing the number of waypoints could result in a smoother path for the robot.