Project Report: Quadrotor Planning and Control

Team 11: Kaitian Chao, Jiakai Zheng, Puen Xu, Longqi Wei

I. Introduction and System Overview

Our goal for Lab 1.4 is to test and validate our control and navigation algorithms on the Crazyflie 2.0 quadrotor in a real-world lab environment. Additionally, we aim to ensure that Crazyflie 2.0 can relatively accurately reach all the provided positions while navigating through different maps and mazes. This is achieved by tuning parameters (such as resolution, margin, values in PD control matrices, and velocity of Crazyflie 2.0) and refining our programs that were proved valid in simulations (which, in our case, involves changing the controlling algorithm from PID to PD), in order to avoid/reduce the impact of factors that exist in real experimental environment, such as air resistance, noise in communication between lab computer and quadrotor, and so on.

As for robot capabilities throughout the two sessions before and after the spring break, Lab 1.4.1 focuses on the ability to stabilize in the hovering situation, follow predefined waypoints both vertically and horizontally relative to the ground, and adjust control parameters (values in PD control matrices and velocity) to achieve smooth and accurate flight in the real world. On the other hand, based on codes that were optimized in 1.4.1, Lab 1.4.2 considers the ability to autonomously generate and execute a feasible path through an environment with obstacles (using A* to plan the origin path and minimum jerk/RDP to optimize and gain the final trajectory), demonstrating effective path planning and control adjustments.

There are three main hardware components used in the whole experiment:

1. Crazyflie 2.0 quadrotor, which is a micro quadcopter used to validate our control and navigation algorithms in the real world. It can also provide additional sensor data from its onboard IMU;

2. VICON System, which is a motion tracking system connected to a separate PC and provides real-time position information of Crazyflie 2.0;

3. Lab computers, which have two different functions: (1) controlling the quadrotor to take off and land, and (2) running our main control and trajectory planning scripts to gain an optimized path for the map in .json file, processing the data collected from VICON and quadcopter sensors, and sending commands to the quadcopter.

To sum up, with the sensing function of Crazyflie 2.0 and VICON system, and the computation and control function of lab computers, our experiments were successfully done.

For the quadrotor, calculated trajectory, waypoints and specific commands (take off/land commands, velocity range, PD values and other control details) are sent from lab computers; As for lab computers, they receive real-time position data of Crazyflie from VICON system, sensor data from Crazyflie IMU sensor, which are used for post-experiment analysis that will be done in the following parts. This process is shown below in Fig. 1.

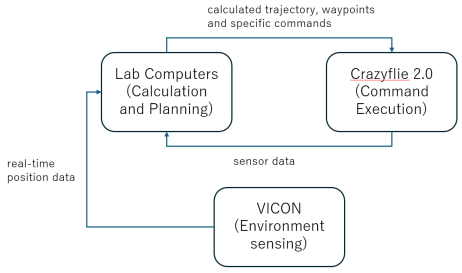

Fig. 1. Information communication inside the whole system

II. Controller

The intuition of our position controller is to calculate the error between the position $x(t)$ and the desired position $x_{des}(t)$ and the error between the velocity $\dot{x}(t)$ and the desired velocity $\dot{x}_{des}(t)$, and to perform an error correction and make the error converge to 0.

We calculated the desired acceleration and the total thrust:

$$\mathbf{a}_{\text{des}} = \ddot{\mathbf{x}}_{\text{des}} - \mathbf{K}_d(\dot{\mathbf{x}} - \dot{\mathbf{x}}_{\text{des}}) - \mathbf{K}_p(\mathbf{x} - \mathbf{x}_{\text{des}}) \quad (1)$$

The total thrust vector compensates for gravity:

$$\mathbf{F}_{\text{des}} = m(\mathbf{a}_{\text{des}} + g\mathbf{e}_z) \quad (2)$$

The actual thrust command is projected onto the drone's z-axis:

$$u_1 = \mathbf{b}_3^\top \mathbf{F}_{\text{des}}, \quad \mathbf{b}_3 = R[:, 2] \in \mathbb{R}^3 \quad (3)$$

The desired orientation matrix R_{des} is constructed by:

$$\mathbf{b}_{3,\text{des}} = \frac{\mathbf{F}_{\text{des}}}{\|\mathbf{F}_{\text{des}}\|} \tag{4}$$

$$\mathbf{b}_{2,\text{des}} = \frac{\mathbf{b}_{3,\text{des}} \times \mathbf{a}_{\psi}}{\|\mathbf{b}_{3,\text{des}} \times \mathbf{a}_{\psi}\|}, \quad \mathbf{a}_{\psi} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ 0 \end{bmatrix} \tag{5}$$

$$\mathbf{b}_{1,\text{des}} = \mathbf{b}_{2,\text{des}} \times \mathbf{b}_{3,\text{des}} \tag{6}$$

$$R_{\text{des}} = \begin{bmatrix} \mathbf{b}_{1,\text{des}} & \mathbf{b}_{2,\text{des}} & \mathbf{b}_{3,\text{des}} \end{bmatrix} \tag{7}$$

The attitude tracking error and moment command are given by:

$$\mathbf{e}_R = \frac{1}{2}\left(R_{\text{des}}^{\top} R - R^{\top} R_{\text{des}}\right)^{\vee} \tag{8}$$

$$\boldsymbol{\tau} = I(-\mathbf{K}_R \mathbf{e}_R - \mathbf{K}_{\omega}\boldsymbol{\omega}) \tag{9}$$

The rotor speeds ω_i are obtained by solving:

$$\begin{bmatrix} u_1 \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix}}_{\text{Allocation Matrix } A} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}, \quad \omega_i = \sqrt{\frac{F_i}{k_{\text{thrust}}}} \tag{10}$$

The most recently tuned control gains are below:

$$\mathbf{K}_p = \mathbf{diag}(3, 3, 6)\,\text{s}^{-2} \tag{11}$$

$$\mathbf{K}_d = \mathbf{diag}(2.5, 2.5, 3.5)\,\text{s}^{-1} \tag{12}$$

$$\mathbf{K}_R = \mathbf{diag}(400, 400, 400)\,\text{N}\cdot\text{m/rad} \tag{13}$$

$$\mathbf{K}_{\omega} = \mathbf{diag}(80, 80, 80)\,\text{N}\cdot\text{m}\cdot\text{s/rad} \tag{14}$$

Proportional gain K_p serves to generate an acceleration command toward the target based on the error between the current position and the target position, the larger the error, the stronger the system response. The differential gain K_d is then used to suppress oscillations or overshoots due to too large K_p, which provides a damping effect based on the error between the current velocity and the target velocity, resulting in a smoother convergence of the system to the target position. In our data, its vertical (Z-axis) gain is significantly higher than the horizontal to preferentially counteract the effect of gravity on altitude control.

Our original control gains are below:

$$\mathbf{K}_p = \mathbf{diag}(20, 20, 500)\,\text{s}^{-2} \tag{15}$$

$$\mathbf{K}_d = \mathbf{diag}(20, 20, 450)\,\text{s}^{-1} \tag{16}$$

$$\mathbf{K}_R = \mathbf{diag}(23000, 23000, 13700)\,\text{N}\cdot\text{m/rad} \tag{17}$$

$$\mathbf{K}_{\omega} = \mathbf{diag}(2750, 2750, 2300)\,\text{N}\cdot\text{m}\cdot\text{s/rad} \tag{18}$$

Comparing with our previous control gains, we turned down all of the K's. The gains used in the simulation are much larger than those used in the real flight experiments, and it was only through a strategy of gradually turning them up from 0 that we were able to find gain combinations that were feasible and stable in the experimental setting.

We thought that one of the possible reasons for the adjustment is that in the simulation, we tend to assume that the thrust and rotational inertia of the motors and propellers are ideal models, whereas the real system may be overshooting too much at high gain and causing instability. In addition, in the simulation, we tend to ignore disturbing factors such as gas flow, noise, etc., so a lower K_p or K_d is often required.

In this experiment, the performance indexes of the controller are obtained and analyzed for the y-axis and z-axis directions respectively: in terms of the y-axis, the steady-state error is about 0.073m, the damping ratio is about 0.207, the rise time is about 0.42s, and the regulation time is about 9.33s, which reflects an obvious underdamping characteristic, i.e., approaching the target quickly but generating a large overshoot in a relatively short rise time, thus prolonging the overall regulation process; while the z-axis direction is about 0.1732m, the rise time is about 0.82s, and there is almost no overshoot phenomenon. The steady state error in the z-axis direction is about 0.1732m, the rise time is about 0.82s, and the regulation time is about 6.79s, which is almost no overshooting phenomenon, indicating that the damping in this direction is larger or the control gain is more conservative.

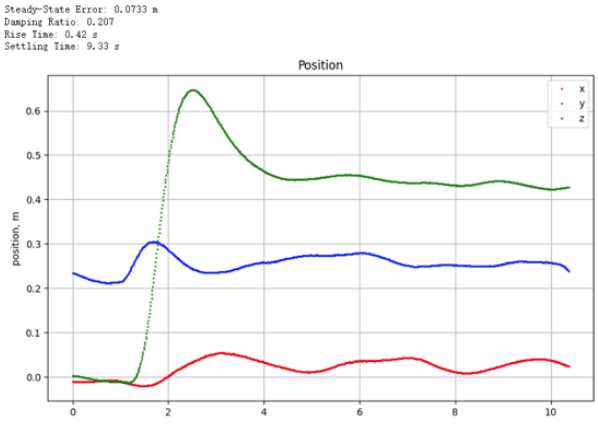

Fig. 2. Y Position versus Time

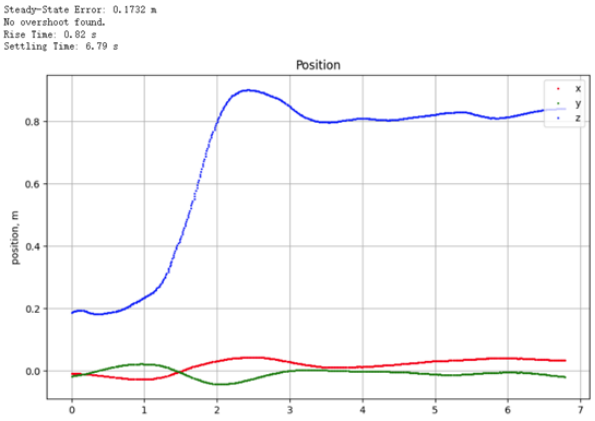

Fig. 3. Z Position versus Time

III. Trajectory Generator

To generate waypoints, we used the A* algorithm to compute the optimal path on the maze. The Manhattan distance was used as the cost metric, while the Euclidean distance was used as the heuristic. After obtaining the initial path, we applied the Ramer-Douglas-Peucker (RDP) algorithm for waypoint simplification, setting the tolerance parameter $\epsilon = 0.1$.

To allocate time between waypoints, we calculated the speed for each segment based on the distance between consecutive waypoints. The drone moves faster when the distance is large (indicating a straight path) and slower when the distance is small (indicating an incoming turn).

For trajectory generation, we formulated a QP to solve for a third-order polynomial. This approach minimizes acceleration and ensures smooth transitions between waypoints.

$$x^*(t) = \arg\min_{x(t)} \int_0^T (\ddot{x})^2 \, dt$$

The objective function is obtained by calculating the quadratic cost for each time segment

$$\begin{bmatrix} c_{i,2} & c_{i,3} \end{bmatrix} \begin{bmatrix} 4t_i & 6t_i^2 \\ 6t_i^2 & 12t_i^3 \end{bmatrix} \begin{bmatrix} c_{i,2} \\ c_{i,3} \end{bmatrix}$$

Assuming there is M segments, the position constraints are

$$p_1(0) = x_0, \quad p_M(t_M) = x_M$$
$$p_i(t_i) = p_{i+1}(0) = x_i, \ \forall i = 1, 2, \cdots, M-1$$

The velocity boundary constraints are

$$\dot{p}_1(0) = 0, \quad \dot{p}_M(t_M) = 0$$

The velocity continuity constraints are

$$\dot{p}_i(t_i) = \dot{p}_{i+1}(0), \ \forall i = 1, 2, \cdots, M-1$$

After formulating the objective function and constraints, we obtain the optimal solution by solving the KKT system.

In Fig. 4, it depicts a plot over time of position, velocity, acceleration, and jerk from the Map 2 run. We could see that the trajectory is continuous in position and velocity.

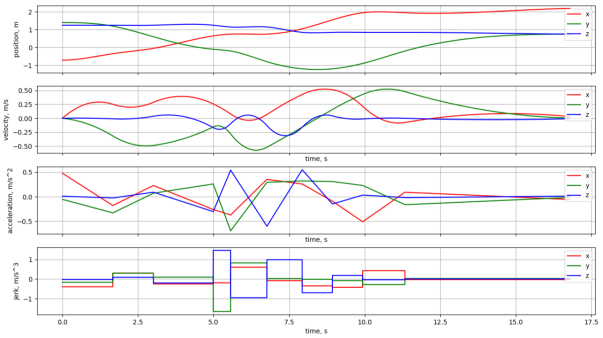

Fig. 4. Map 2: Position and Derivatives versus Time

IV. Maze Flight Experiments

We have successfully completed the navigation tasks through all three maps using our planner, trajectory generator, and controller. This section presents our experimental results.

Fig. 5 through Fig. 7 illustrate the results for each of the three maze runs. In each 3D plot, we show the environment, the waypoints, the planned trajectory, and the actual flight path taken by the quadrotor.

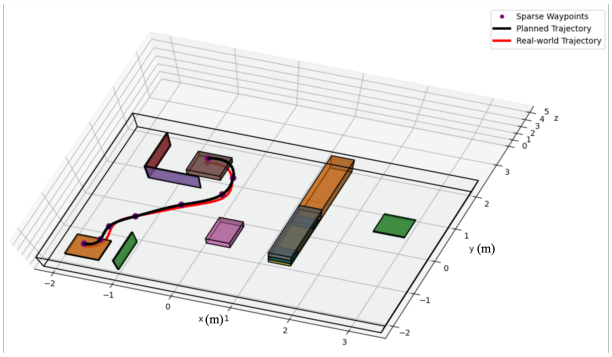

Fig. 5. Map 1: Waypoints, Planned and Real Trajectory

It is noteworthy to point out that, for Map 3, we reduced the margin to 0.25 so that the drone can reach the goal by crossing the window instead of traversing around it. Furthermore, to provide a more detailed analysis of the system's performance, we present the position and velocity profiles over time for this maze run in Fig. 8.

To evaluate the tracking errors during the drone flight, we analyzed both the 3D plot and the position profile to compare the planned trajectory with the actual flight path. For Map 1 and Map 3, the tracking errors were relatively small, with maximum errors of 0.1 meters

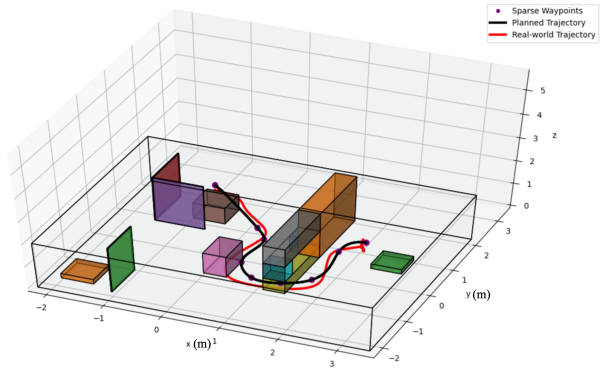

Fig. 6. Map 2: Waypoints, Planned and Real Trajectory

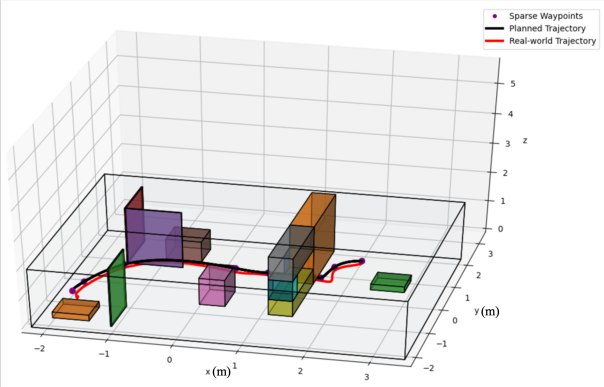

Fig. 7. Map 3: Waypoints, Planned and Real Trajectory

and 0.3 meters, respectively, occurring during turns. In contrast, the tracking error for Map 2 was larger due to a challenging U-turn, where the maximum error of 0.6 meters occurs. However, when we reduced the margin to 0.25 meters for Map 2 as well, allowing the drone to pass through a window instead of navigating around it, the errors decreased. This suggests that most tracking errors arise during turning maneuvers, possibly due to small position control gains. With this information, we can develop trajectories that minimize sharp turns to improve the tracking performance of the quadrotor.

We can create more aggressive trajectories by increasing the speed. Given that we have a function that calculates the speed for each segment based on the distance between consecutive waypoints, to make the trajectories more aggressive, we could increase the speed further during straight-line motions. The increase in speed could lead to larger tracking error but the error could be reduced by tuning the controller gains.

If we had one more session in the lab to try something new, we would like to test our algorithms on a custom, more complex map. An idea we had but couldn't implement due to time constraints was to combine all three maps into a single map by starting from the origin

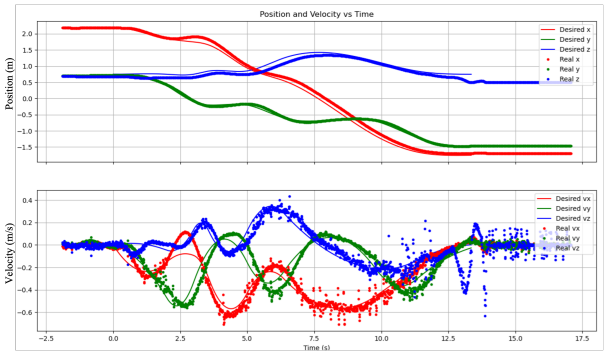

Fig. 8. Map 3: Position and Velocity versus Time

of Map 1, passing through the origin of Map 2, and reaching the goal of Map 3. This combined map would provide a more challenging environment, allowing us to further assess the robustness and effectiveness of our planner, trajectory generator, and controller.