Project Report: Quadrotor Planning and Control

Team 7: Madeline Griffith, Anthony Nguyen, Yongxin Guo

I. Introduction and System Overview

The goal of this lab was to develop and implement algorithms for motion planning, collision avoidance, and path optimization on a Crazyflie 2.0 from Bitcraze. The crazyflie is fully equipped with a STM32 micro controller and an IMU, which allows us gather information on the current state of the drone and successfully complete the required tasks. This document details how our group manages to complete the maze as fast and smooth as possible without aggressive maneuvers.

The project can be split up into three subdivisions:
(a) Developing a robust PD controller
(b) Developing a way-point generator
(c) Developing a trajectory planner
We will specifically go into depth with (a) and (c). Because the free-space is pre-defined, (b) can be solved by implementing the A* search algorithm.

II. Controller

We implemented a geometric nonlinear controller derived in the provided *proj1_1* document. Below is a summary

Variables

F^{des}	$Desired\ Force$
$\ddot{r}^{des}$	$Desired\ acceleration$
$\dot{r}_T$	$Desired\ velocity$
r_T	$Desired\ position$
R	$RotationMatrix$
u_1	$Sum\ of\ motor\ forces$
u_2	$Sum\ of\ motor\ torques$
K_p	$Position\ gain$
K_d	$Damping\ gain$

We ultimately want to calculate inputs u_1 and u_2 to give us F^{des}

We first want to define our desired Force and acceleration:

$$F^{des} = mr^{des} + \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = u_1 R \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (1)$$

$$r^{des} = \ddot{r}_T - K_d(\dot{r} - \dot{r}_T) - K_p(r - r_T) \quad (2)$$

To obtain u_1 we project F^{des} onto the b_3 basis vector

$$u_1 = b_3^T F^{des} \quad (3)$$

because thrust can only be produced along the b_3 vector.

Next, we want to apply another PD controller to the attitude control. We align b_1^{des} to match the desired yaw angle ψ_T and define b_2^{des} so that it is perpendicular to b_1^{des}. A diagram of the vector space b can be seen in Figure 2 Now we are able to compute the desired rotation matrix R^{des}

$$R^{des} = [b_1^{des}, b_2^{des}, b_3^{des}] \quad (4)$$

The error in orientation is found by

$$e_R = \frac{1}{2}(R^{desT}R - R^T R^{des})^V \quad (5)$$

. Now it is possible to compute u_2

$$u_2 = I(-K_R e_R - K_\omega e_\omega) \quad (6)$$

The gains used in the controller can be found in Table II. You can observe that two sets of gains are provided, ones used in simulation and ones used for the actual test in lab. This is because in simulation, actuators are ideal and actuate perfectly. Therefore, the gains can be unrealistically ramped up. In actual lab testing, the gains needed to be reduced. The attitude gains are not provided because in lab an on-board IMU on the crazyflie measures the orientation is used instead of a controller.

Controller Performance

The controller performance is evaluated based on experiment results and controller settings from the first lab session in terms of overshoot, rise time, settling time, steady state error, and damping ratio. The controller gains used in the first lab session is shown in Table II. The controller evaluation is only performed on a specific section of the trajectory tested in the first lab session in order to focus on the convergence portion better.

For simplicity and purpose of capturing the essence of the controller performance, the evaluation methodology is to target down the z-axis step response from an initial position of 0.25 m to 1 m as shown in Fig 1, which is the first segment of the trajectory performed in the first lab session. The global start time is chosen to be 2.5 s by observation. The rise time is computed by taking the difference between the start time and the reach time.

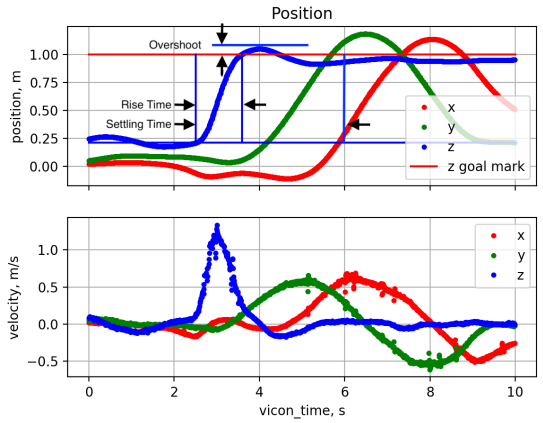

Fig. 1. Step Response for Controller Evaluation

The overshoot is the difference between the target z-axis distance and the maximum distance the drone reached after passing the target. The steady error is evaluated after 6 s where the fluctuation is mostly diminished. The damping ratio is calculated from the error dynamics of the control model,

$$\ddot{e} + \boldsymbol{K_d}\dot{e} + \boldsymbol{K_p}e = 0 \tag{7}$$

with $\boldsymbol{K_d} = \boldsymbol{2\zeta\omega_n}$ and $\boldsymbol{K_p} = \boldsymbol{\omega^2}$. One can compute the damping ratio as,

$$\zeta = \frac{\boldsymbol{K_d}}{\boldsymbol{2\sqrt{K_p}}} \tag{8}$$

All the values of the controller matrix evaluation are tabulated in Table 1. It is found that the overshoot is around 5 cm, which is quite acceptable according to the real experiment. The rise time takes 1.09 s since the distance travelled is 0.75 m upward. Particularly, the settling time takes a bit longer since the nature of the selected trajectory, which is a first piece of the entire trajectory thus the course of other two axes also affect the convergence of the z-axis a bit in real-life. More specifically, the drone still needs to travel to the next waypoint after it reached the first waypoint, and this continuity affects the drone's ability in holding the z-axis position. The damping ratio is calculated to be 0.78 and it is closed to the critical damping, and the real-life data has also shown that there is no oscillations at all for z-axis and it quickly converges to the target position after a slight overshoot of 5 cm.

	K_p	K_d
simulation	[23, 23, 23]	[7.5, 7.5, 7.5]
real test	[2, 2, 23]	[4, 4, 7.5]

TABLE I

GAINS USED IN THE POSITION CONTROLLER

Evaluation Matrix	Value
Steady State Error	0.28 m
Rise Time	1.09 s
Settling Time	3.5 s
Overshoot	0.05 m
Damping Ratio	0.78

TABLE II

CONTROLLER EVALUATION

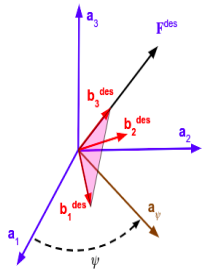

Fig. 2. Alignment of basis vectors b in the inertial frame a

III. TRAJECTORY GENERATOR

Before generating a trajectory, the waypoints calculated by the A* algorithm to connect a path between the start and goal were filtered out into a sparser set of waypoints. The second and second-to-last waypoints were excluded, to avoid sharp turns at the beginning and end; then, starting from the first point, if excluding the next point would cause a collision, that point would be added. A point was also added if it was far (>5 m) from and not collinear with the last accepted point.

Time between waypoints was scaled by distance between the points for points more than 0.5 m apart, according to Eqn. 9:

$$t = \frac{\boldsymbol{d} * \boldsymbol{a} + \boldsymbol{v^2}}{\boldsymbol{a} * \boldsymbol{v}} \tag{9}$$

where $\boldsymbol{d}$ is the euclidean distance and $\boldsymbol{a}$ and $\boldsymbol{v}$ represent the maximum acceleration and velocity, respectively. For points less than 0.5 m apart, the time between waypoints was scaled by the square root of the distance, according to Eqn. 10:

$$t = \frac{\boldsymbol{2} * \sqrt{\boldsymbol{d}}}{\boldsymbol{a}} \tag{10}$$

Maximum acceleration and velocity were set to 3 m/s^2 and 2 m/s, respectively; based on laboratory performance, these were reduced to 2 m/s^2 and 0.5 m/s to avoid movements that appeared to be too quick or aggressive.

The trajectory between waypoints was calculated as a minimum jerk (5th order) polynomial, with the quadrotor constrained to stop at each waypoint with zero velocity and acceleration. This could be represented by the matrix equation in Eqn. 11.

$$\begin{bmatrix} t_1^5 & t_1^4 & t_1^3 & t_1^2 & t_1 & 1 \\ t_2^5 & t_2^4 & t_2^3 & t_2^2 & t_2 & 1 \\ 5t_1^4 & 4t_1^3 & 3t_1^2 & 2t_1 & 1 & 0 \\ 5t_2^4 & 4t_2^3 & 3t_2^2 & 2t_2 & 1 & 0 \\ 20t_1^3 & 12t_1^2 & 6t_1 & 2 & 0 & 0 \\ 20t_2^3 & 12t_2^2 & 6t_2 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} p(t_1) \\ p(t_2) \\ \dot{p}(t_1) \\ \dot{p}(t_2) \\ \ddot{p}(t_1) \\ \ddot{p}(t_2) \end{bmatrix} \tag{11}$$

The distances and times were normalized to calculate the polynomial, so that each in each segment the quadrotor travelled [0,1] m in [0,1] s. Thus, the matrix equation in Eqn. 11 could be simplified based on the 6 boundary constraints to the matrix equation shown Eqn. 12, which was then solved for the ploynomial coefficients.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5 & 4 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20 & 12 & 6 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{12}$$

During the update function, the polynomial would then be converted back to dimensional units based on the length and time for the current segment.

Using this planned trajectory, the quadrotor was able to complete the 3 maps in the laboratory without any collisions. The position and velocity of the quadrotor for one of these laboratory maps is shown in Fig. 3. While this trajectory is feasible, as it generally avoids high velocities (except a few points near the beginning) and does not overshoot the desired positions, it is not as smooth as desired, as can be seen in the time range [10,15] in Fig. 3. This tended to happen near obstacles in the laboratory, or where the waypoints were more dense, as these portions of the trajectory involved more starts and stops, shorter path lengths, and shorter time durations.

A potential solution to further smooth this trajectory could be to increase the amount of time between waypoints that had a shorter distance between them. Another solution, which may also improve the speed, would be to use a trajectory that does not stop at each individual waypoint but rather connects a smooth, continuous trajectory between them. This could be achieved by using continuity constraints on the velocity and acceleration at each waypoint, rather than constraining velocity and acceleration to zero at each waypoint.

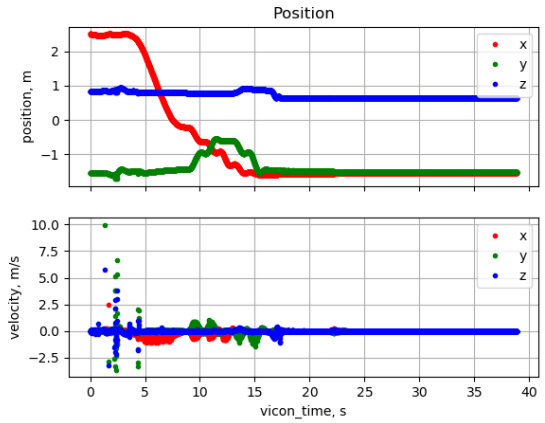

Fig. 3. Position versus time and velocity versus time for the quadrotor travelling the map 3 path in the laboratory.

IV. Maze Flight Experiments

Using the planner, trajectory generator, and controller described above, 3 maps were flown in the laboratory. The obstacles, waypoints, desired trajectory, and flight for each map can be seen in Figs. 4, 5, and 6.

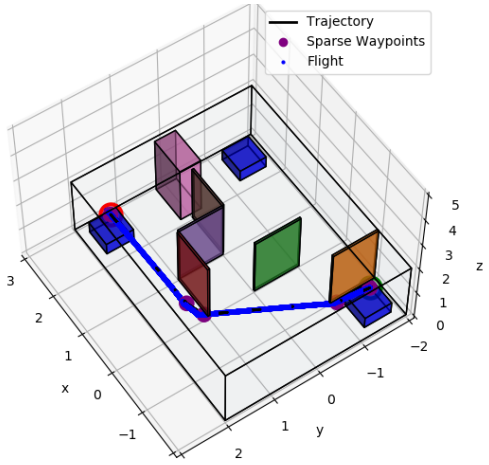

Fig. 4. Waypoints, trajectory, and flight around laboratory obstacles for map 1.

Again, the position and velocity versus time for the actual flight for one maze trial (map 3) can be seen in Figure 3.

The flight generally followed the trajectory closely, but there was some tracking error, as can be seen more easily in Figure 7, which shows a top view of the map 3 trajectory. This error was roughly $\pm$ 0.1 m or less, although this could be calculated more precisely if necessary, for example by using simpler paths. Error data could be used to inform the choice of margin used in path planning, for instance by increasing the margin

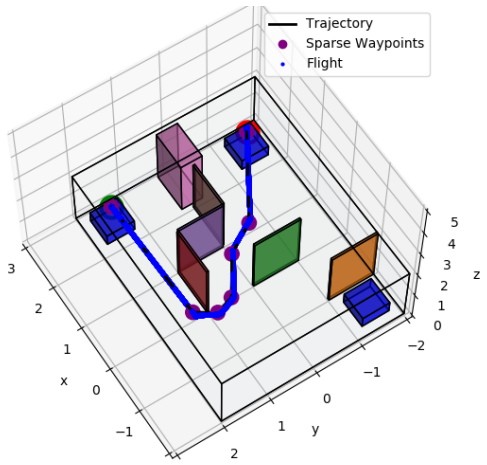

Fig. 5. Waypoints, trajectory, and flight around laboratory obstacles for map 2.

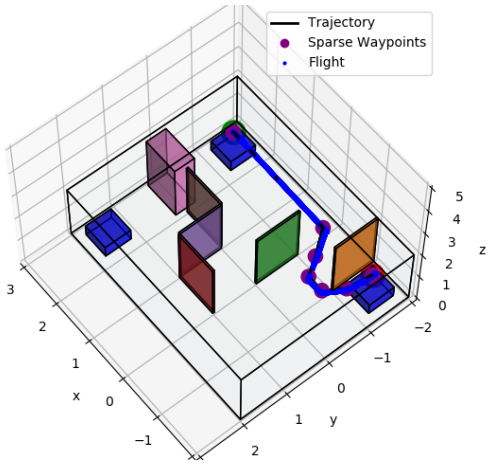

Fig. 6. Waypoints, trajectory, and flight around laboratory obstacles for map 3.

if there are larger errors associated with the current trajectory.

In our lab experiment, our trajectories could have been more aggressive. The trajectories we used were developed with the goal of safely finishing the course, which led to some slow velocities and inefficient paths (e.g., straight line paths rather than rounded corners). The trajectory generator we used stopped at each waypoint with zero velocity and zero acceleration, which added a significant amount of time to the flight. A faster but still safe trajectory could have used continuity constraints on acceleration and/or velocity at each waypoint, rather than constraining each to zero. Or, a faster and more reliable trajectory could have been achieved by increasing the order of the polynomial, as these trajectories were constructed using a minimum jerk spline, rather than a minimum snap spline.

If we had one more session in lab, I would want to try

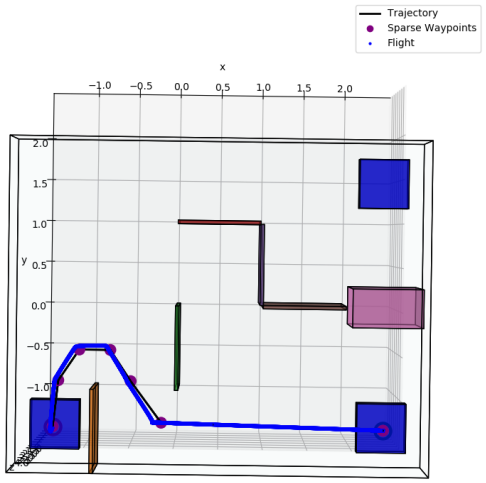

Fig. 7. Top view of waypoints, trajectory, and flight around laboratory obstacles for map 3.

to further refine these trajectories. I would want to try to implement a more aggressive trajectory and complete more challenging maps (e.g., passing through a window opening).