MEAM 6200 Lab 1.4 Report

Group 6: Chen Hsin Chiang, Erica Santos, Songhao Huang, Shrehari Prsad Gopalakrishnan

I. Introduction and System Overview

A. Objective

The in-person labs are used to demonstrate how algorithms and implementations developed for the simulation perform on a real robot, specifically, a miniature quadrotor operating in a motion-captured lab environment. The first lab focused on tuning the controller's XYZ gain as there were many discrepancies between the simulation and the actual quadrotor, while the second lab involved planning and executing a path through a known maze with different sets of start and end locations. The in-person lab provided hands-on experience in operating an actual drone and enhanced students' ability to translate theoretical concepts into practical applications on autonomous systems such as the Crazyflies quadrotor.

B. Hardware

The hardware setup for the lab utilized Crazyflies, Vicon, and the lab computer. The Crazyflies quadrotor served as the main device to be controlled and traversed through the test environment; the Vicon system was responsible for sensing and tracking the quadrotor's movement within the lab environment; and the lab computers handled the computation tasks and output commands to be used by the quadrotor to stay on the planned trajectory.

C. Information

There are a couple of pieces of information that are being sent to the robot from the lab computer. Firstly, the quadrotor receives commands from the Python scripts running on the lab computers, these commands include the control signals which are computed based on trajectory planning, and the quadrotor's navigation paths. Similarly, there is also crucial data to be collected by the lab computer. The lab computers run the Python scripts written by the team that controls the quadrotor, these include the se3_control.py, world_traj.py, and graph_search.py. During the lab session, the lab computer also captures and records the data from the Vicon system to generate the resultant plots and .bag files. In summary, the information sent to the quadrotor includes commands for control and navigation while the lab computers compute the command signals to be sent to the quadrotor based on Vicon data.

II. Controller

The position controller computes the acceleration required to reach the desired position and velocity. By comparing the current position and velocity with the desired ones, the controller generates appropriate acceleration commands to steer the quadrotor toward the desired position and velocity. The PD controller we used for experiments is a geometric nonlinear controller and is constructed as described in Project 1-1 Section 3.4.

Our final control gains used for experiments are:

$$K_{p\_x} = 3.0, K_{p\_y} = 3.0, K_{p\_z} = 20,$$
$$K_{d\_x} = 2.6, K_{d\_y} = 2.6, K_{d\_z} = 6,$$
$$K_{r\_x} = 100, K_{r\_y} = 100, K_{r\_z} = 20,$$
$$K_{\omega\_x} = 11, K_{\omega\_y} = 11, K_{\omega\_z} = 5$$

Where K_p scales the error in position, with units of m, and K_d scales the error in velocity, with units of m/s^2. K_r and K_ω have units of rad and rad/s.

The role of K_p primarily affects the system's response time and steady-state error. However, large K_p will bring large overshoot, thus, K_d is used to dampen the system and decrease overshoot. The response time will increase if K_d is too large, so the team also needs to find a trade-off between these two parameter sets.

Adjustments were required when the team transitioned from simulation to real-world testing. This was necessary as there were discrepancies between the simulation and the actual environment. These discrepancies include dynamics, noise, delays, and simplified models. In summary, simulation gains were halved and velocity was limited to $1m/s$.

A. Controller Performance and Analysis

Step response graphs were created from experimental data from the first lab session. After analyzing both graphs in Fig 1, we calculated steady state error (e_{steady}), maximum peak (M_p), damping ratio (ζ), rise time (T_{rise}) and settling time (T_{settle}) in Z and Y. We assumed symmetry between X and Y and claim that the characteristics are therefore the same for both X and Y. Our controller achieved a damped system responses, as both Z and X/Y have damping ratios <1. We can see that neither are critically damped, and therefore still room for improvement in our gains.

Our derivations for Z characteristics are as follows:

$$e_{steady} = 1 - z_{steady} = 1 - 0.94 = 0.06m$$

$$M_p = \frac{z_{peak} - z_{steady}}{z_{steady}} = \frac{1.15 - 0.94}{0.94} = 0.223$$

$$\zeta = \sqrt{\frac{\ln^2 M_p}{\ln^2 M_p + \pi^2}} = \sqrt{\frac{\ln^2 0.223}{\ln^2 0.223 + \pi^2}} = 0.431$$

$$T_{rise} = t_{90\%\text{final}} - t_{10\%\text{final}} = 1.75s - 1.40s = 0.35s$$
$$T_{settle} = 4.5s$$

And the corresponding characteristics in Y:

$$e_{steady} = 1 - y_{steady} = 0.5 - 0.44 = 0.06m$$

$$M_p = \frac{y_{peak} - y_{steady}}{y_{steady}} = \frac{0.586 - 0.44}{0.44} = 0.332$$

$$\zeta = \sqrt{\frac{\ln^2 M_p}{\ln^2 M_p + \pi^2}} = \sqrt{\frac{\ln^2 0.332}{\ln^2 0.332 + \pi^2}} = 0.331$$

$$T_{rise} = t_{90\%\text{final}} - t_{10\%\text{final}} = 1.95s - 1.45s = 0.5s$$
$$T_{settle} = 4s$$

III. Trajectory Generator

A. Waypoint Generation

Given a map, start, and goal positions, the team employed the A* algorithm to find the shortest collision-free path, utilizing the Euclidean distance as the cost function and the Chebyshev distance as the heuristic function. Additionally, the team implemented a tie-breaker in the heuristic function to expedite path searching. Post-processing of the path was necessary due to its initial density. The Ramer-Douglas-Peucker (RDP) algorithm was utilized for recursive pruning. Subsequently, the team assessed waypoint distances; if they were too large, they may cause trajectory deviation from the A* path and result in potential collision. In such cases, the algorithm would reintroduce intermediate waypoints. Tunable parameters in this stage include map resolution (set at 0.1m for xyz for finer solutions due to the small map scale), margin (0.2m), RDP threshold distance (0.1m), and the maximum distance threshold between waypoints (set at 5 index in the A* path array).

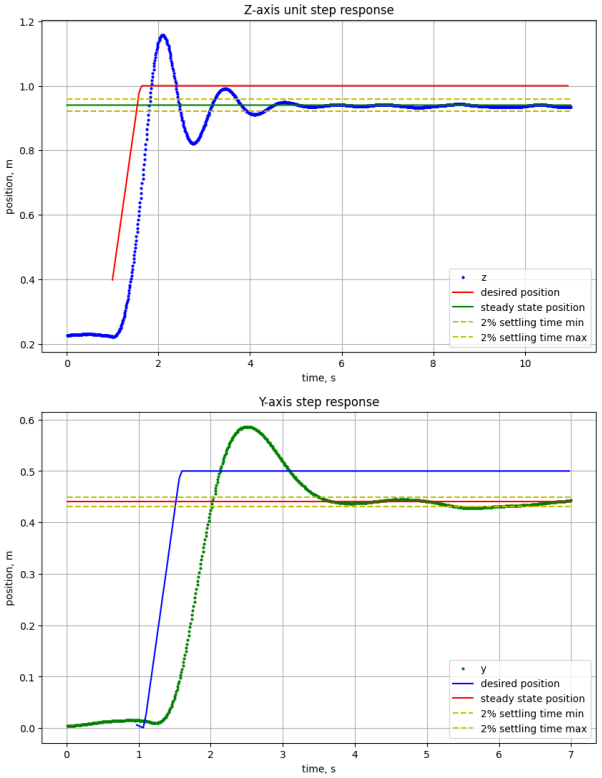

Fig. 1. Z and Y-Step Response for Controller Performance and Characterization

B. Time Allocation

Trapezoidal velocity profile time allocation was implemented between waypoints. Assuming the UAV begins and ends its motion at rest, the movements are divided into three stages. During the first stage, the UAV accelerated to maximum velocity using maximum acceleration. It maintained velocity during the second stage and finally decelerated with maximum acceleration to come to rest at the goal. The total distance of the path was determined. Then, the time duration for each of the three stages was computed. Finally, the timestamps were assigned to each waypoint.

C. Trajectory Generation Overview

A minimum jerk 5-order polynomial trajectory was generated. The optimization problem for one of three dimensions was formulated as shown in Fig. 2. Note: to make the trajectory less conservative and twisted, a simple square safe corridor centred at each waypoint was created and the trajectory would only go through this region, the corridor size (0.05m) would need to be smaller than the margin.

The generated trajectory was quite smooth as shown in Fig.3: the position was followed with low overshoot and error, and the velocity and acceleration have no

$$\begin{aligned} &\text{Minimize} \quad p^T Q p \\ &\text{subject to} \quad Ax \leq b \end{aligned}$$

where

$$Q = \begin{bmatrix} Q_0 & & & \\ & Q_1 & & \\ & & \ddots & \\ & & & Q_m \end{bmatrix}$$

$$Q_i(t) = \int_{T_{i-1}}^{T_i} qq^T$$

$$q = \begin{bmatrix} 0 & 0 & 0 & 6 & 24t & 60t^2 \end{bmatrix}$$

$$p = \begin{bmatrix} p_0^T & p_1^T & \dots & p_m^T \end{bmatrix}^T$$

$$p_i = \begin{bmatrix} p_{i0} & p_{i1} & \dots & p_{i5} \end{bmatrix}^T$$

m is number of segment. The inequality constraint includes both equality constraints and inequality constraints, p_i, v_i, a_i stand for position, velocity, acceleration at i^{th} segment. T_i is the i^{th} duration, pt_i is the i^{th} waypoint position

start and end pva equality constraints

$$p_0(0) = p_{start}, v_0(0) = v_{start}, a_0(0) = a_{start}$$

$$p_m(T_m) = p_{end}, v_m(T_m) = v_{end}, a_m(T_m) = a_{end}$$

continuous pva constraints at waypoints

$$p_0(T_0) = p_1(0), v_0(T_0) = v_1(0), a_0(T_0) = a_1(0)$$

$$p_1(T_0) = p_2(0), v_1(T_0) = v_2(0), a_1(T_0) = a_2(0)$$

$$\dots$$

$$p_{m-1}(T_{m-1}) = p_m(0), v_{m-1}(T_{m-1}) = v_m(0), a_{m-1}(T_{m-1}) = a_m(0)$$

waypoints corridor constraints

$$pt_1 - corridor \leq p_0(T_0) \leq pt_1 + corridor$$

$$\dots$$

$$pt_m - corridor \leq p_{m-1}(T_{m-1}) \leq pt_m + corridor$$

Fig. 2. Optimization problem formulation

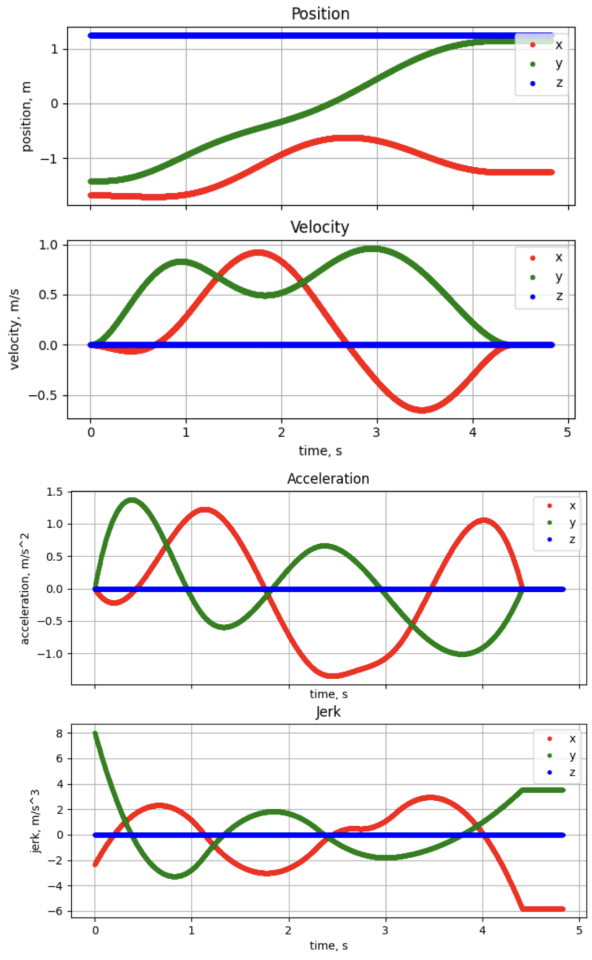

Fig. 3. Position and higher-order derivatives in Maze 1 run

spike, which would indicate a sharp turn or change in direction.

For feasibility, the velocity and acceleration were limited at 1.0 m/s and 1.0 m/s^2, respectively. While velocity remained feasible throughout the test session, acceleration sometimes exceeded the limit. However, with no specific constraint on maximum acceleration for the experiments, the team proceeded without adjustment. Two solutions emerged: 1) Lengthening time duration in segments with infeasible points ensured dynamical feasibility. This optimization with adjusted time allocation was repeated until the trajectory was fully feasible. 2) Incorporating an additional optimization term into the QP for optimal time allocation in trajectory. These avenues were potential future considerations.

IV. Maze Flight Experiments

Each maze's planned trajectories, simulated drone paths and actual trajectories are shown below in Fig 4, 5, 6. The team also selected Maze 3's position and velocity over time to be compared to the actual hardware trial in Fig 7.

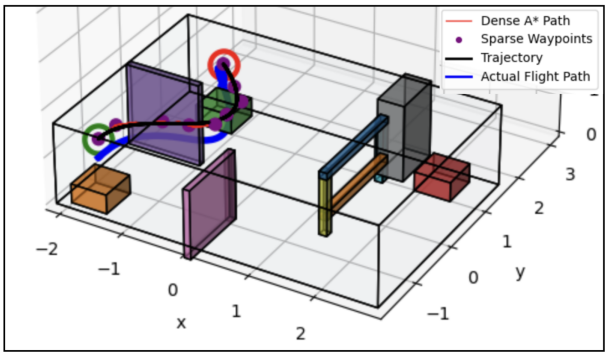

Fig. 4. Maze 1 3D plot including waypoints, trajectory, and the actual flight path

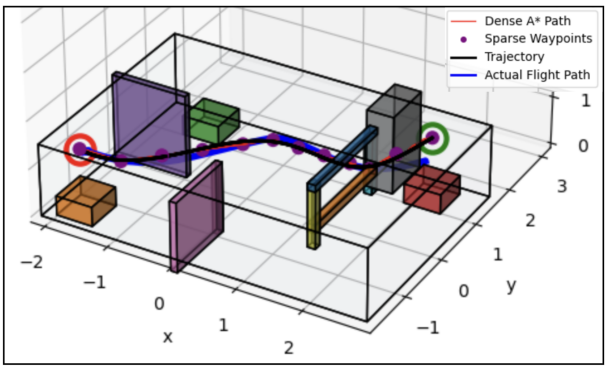

Fig. 5. Maze 2 3D plot including waypoints, trajectory, and the actual flight path

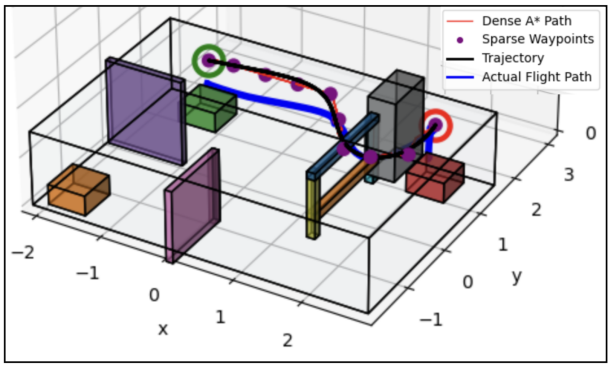

Fig. 6. Maze 3 3D plot including waypoints, trajectory, and the actual flight path

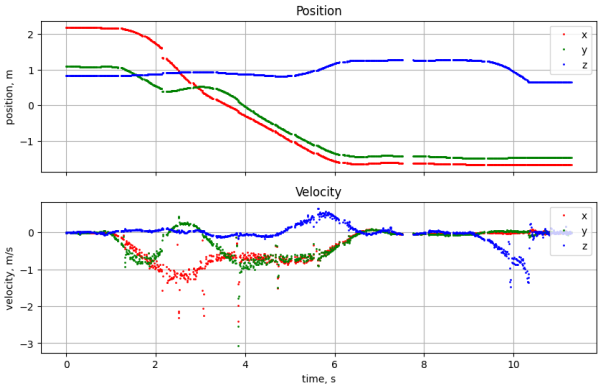

Fig. 7. Position and velocity data over time for the Maze 3 trial.

A. Discussion

The desired and actual x, y, and z trajectories are plotted. Note that the start times were non-zero as it was cropped at an approximated start time since the .bag files captured hovering data before the quadrotor's operation. Using the .bag files, the team extracted both the flat_output and the odometer data and calculated the error using the normal distance. The norm distance was used since it reflected the objective of the lab. The average 3D distance error is 0.1130 m while the xyz component-wise error is given in Table I. Given this error, the team can modify and optimize the margin to allow such deviation around obstacles when the quadrotor is following the trajectory.

x	y	z
0.103 m	0.025 m	0.019 m

TABLE I

Average Norm Error in x,y, and z positions.

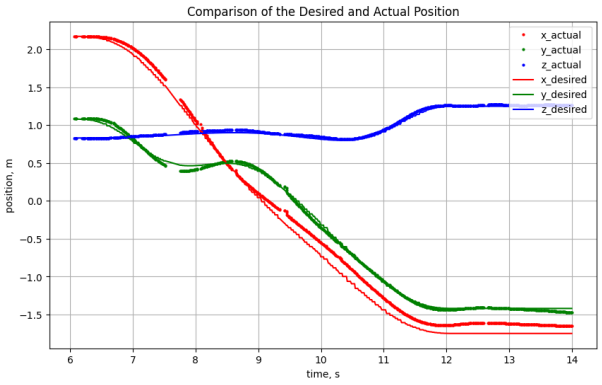

Fig. 8. The desired and actual position data from Maze 3

B. Improvements

The quadrotor system can be improved for speed and aggressiveness while maintaining its safety by generating smoother paths and adjusting the velocities adaptively based on the trajectory. Similarly, extensive testing can refine parameters to optimize trajectories and collision avoidance.

C. Future Work

Given an extra session, the team would like to try to maximize the velocity of the quadrotor and find the drone's limit to complete a much faster run.