ENAE788M Assignment 3 - Trajectory Following on the PRG Husky

Estefany Carrillo, Mohamed Khalid M, and Sharan Nayak

I. INTRODUCTION

In this project, we implement code to have the PRG Husky follow three trajectories: helix, diamond and staircase. For each of these trajectories, waypoints for positions or velocities are generated. This will be described in detail in section II. We implement an open loop controller by sending these waypoints to the Husky as control commands and perform tunning of the control gains in the inner control loop to improve trajectory tracking.

II. WAYPOINTS GENERATION

A. HELIX TRAJECTORY

For this trajectory, we are given the equations for generating x, y and z points. Since these equations are differentiable, we decided to compute the velocities in x, y and z from the given equations for a specific range of time as follows:

$$\dot{x} = \omega r \cos \omega t \tag{1}$$

$$\dot{y} = -\omega r \sin \omega t \tag{2}$$

$$\dot{z} = \omega c = \frac{h}{2\pi}, \tag{3}$$

where r is the radius of the helix, set to 1m, $2\pi c$ is a constant giving the vertical separation of the helix's loops, set to 1m, and ω is the constant angular velocity, set to 1 rad/s. For implementation, we set the time step at which we generate trajectory points to $\frac{\pi}{4}$ secs. We implemented the trajectory in two ways, generating position commands as well as velocity commands. In Fig. 1, we show a plot of this trajectory parameterized with time step = 0.1 sec. Even though we tried running experiments with this finer parameterization with time step of 0.1 secs, we did not obtain a good tracking of the trajectory, so we just set the time step to $\frac{\pi}{4}$ secs.

B. DIAMOND TRAJECTORY

For this trajectory, we implemented the control in two ways. First, we sent the waypoints forming the diamond pattern as position commands. This resulted in a noticeable delay in the quadrotor moving from one point to the next. We then tried fitting a cubic spline through the specified points using the function *interp1d* from the scipy.interpolate library available in python. Once the cubic spline fitting going through all the points was obtained, we were able to generate a trajectory time-parameterized at time step = 0.2 sec. In Fig. 2, we show a plot of this trajectory parameterized with time step = 0.2 sec.

Even though we tried running experiments by sending the points generated from the spline, the quadrotor was not able to track the trajectory in terms of the z-direction as well as sending the 5 points of the diamond pattern.

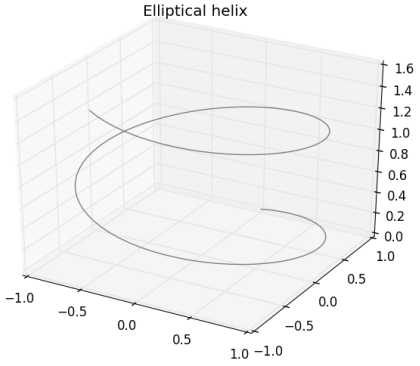

Fig. 1: Helix Trajectory

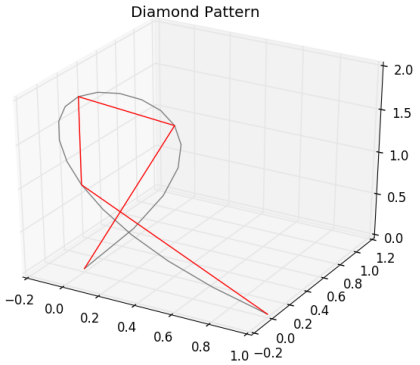

Fig. 2: Diamond trajectory (red), spline fitting for diamond trajectory (gray)

C. STAIRCASE TRAJECTORY

For this trajectory, we implemented two staircases. The fist staircase was generated based off a staircase constructed in the 2D x-z plane with initial position at $(0, 0)$ and final position at (x, z), where $x = c$, $z = c$ and $c = 3\frac{\sqrt{3.0}}{\sqrt{2.0}}$ which implies y is set to 0. Then, we rotated this staircase by 45 degrees and obtained the following set of points:

$$P = \{(0,0,0), (0,0,\frac{c}{4}), (\frac{c}{4\sqrt{2}}, \frac{c}{4\sqrt{2}}, \frac{c}{4}), (\frac{c}{4\sqrt{2}}, \frac{c}{4\sqrt{2}}, \frac{c}{2}), (\frac{c}{2\sqrt{2}}, \frac{c}{2\sqrt{2}}, \frac{c}{2}), (\frac{c}{2\sqrt{2}}, \frac{c}{2\sqrt{2}}, \frac{3c}{4}), (\frac{3c}{8\sqrt{2}}, \frac{3c}{8\sqrt{2}}, \frac{3c}{4}), (\frac{3c}{8\sqrt{2}}, \frac{3c}{8\sqrt{2}}, c), (\frac{c}{\sqrt{2}}, \frac{c}{\sqrt{2}}, c)\} \quad (4)$$

The second staircase was generated by solving the following nonlinear equations for each point not in the diagonal corresponding to $P1$, $P3$, $P5$, $P7$. Points in the diagonal were easily computed by increasing the initial point $P0 = (0,0,0)$ by $3/4$ to produce the next point in the diagonal until reaching the final point $P8 = (3,3,3)$. To compute the coordinates of $P1$ (not a point on the diagonal), the following equations were used:

$$((x - P_{0x})^2 + (y - P_{0y})^2 + (y - P_{0y})^2) = d^2 \quad (5)$$

$$((x - \frac{P_{2x}}{2})^2 + (y - \frac{P_{2y}}{2})^2 + (y - \frac{P_{2z}}{2})^2) = h^2 \quad (6)$$

$$((x - P_{2x})^2 + (y - P_{2y})^2 + (y - P_{2z})^2) = d^2, \quad (7)$$

where d is the length between P_0 and P_1, calculated as $d = \frac{3\sqrt{3}}{8\cos 45}$ and h is the length from the midpoint between P_0 and P_2 on the diagonal and the point not on the diagonal, P_1, calculated as $h = d \sin 45$. In Fig. 3, we show a plot of this trajectory parameterized with time step = 0.1 sec.

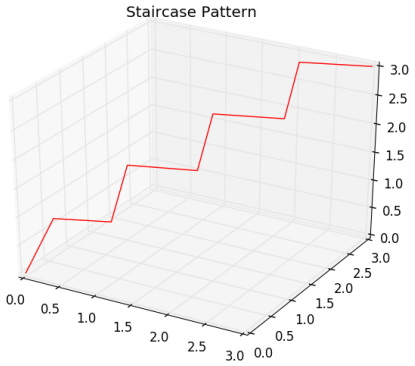

Fig. 3: Staircase Pattern

From our experiments, we obtained better tracking for the first staircase and decided to use that for demonstration and analysis.

D. TUNING OF PARAMETERS

In order to test the accuracy of motion of the quadrotor when sending position commands. Our first experiment consisted in setting linear.x to 1 in order to see how much the quadrotor translated forward. We performed this experiment a total of 15 trials and computed the average distance obtained. In Fig. 4, we show the step of obtaining the measurements of translation.

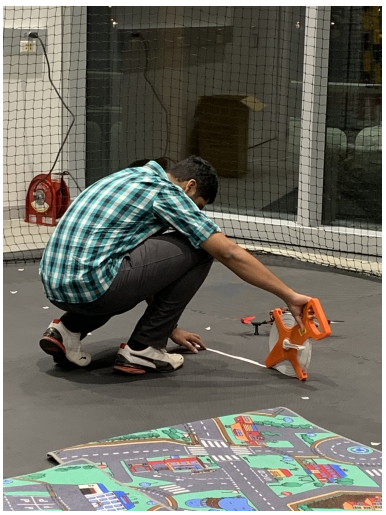

Fig. 4: Measurement step during calibration experiment.

In terms of the inner controller gains, we performed some testing by i) changing the default K_P gain set at 0.8 to 0.9 and ii) changing the default K_I gain set at 0.0 to 0.1. For the first test, we observed that the steady state error increased and for the second test there was more drifting and larger deviation from the trajectory. Given these observations, we decided to keep the default gains in place.

In terms of the outer loop controller, we modified and tuned 4 parameters. This included the delay between waypoints, the rate at which each delta position command was sent multiple times, the amount of time for which each command was sent (timeout) and the scaling for the X, Y, Z delta positions. In reference to the delay command, when the delay was decreased, a smoother trajectory was obtained, however, when the delay was reduced to about below 0.1, the quadrotor did not track the trajectory as well. The rate command caused the quadrotor to increase its speed. The timeout command caused the trajectory between waypoints to be executed over a longer period of time. Increasing the scaling caused more distance to be covered along the X, Y or Z directions. These parameters were tuned empirically to get the helix, staircase and diamond trajectories as close as possible to the ideal trajectories.

E. PLOTTING IN RVIZ

We used Rviz to plot real-time waypoint positions of the quadrotor. The ROS TF transform containing the position coordinates were broadcasted from our program to Rviz every 0.1 sec. Whenever our program finished sending a waypoint to the quadrotor, the position coordinates were updated for broadcasting to Rviz. Fig. 5 shows the world and quadrotor coordinate systems being displayed in Rviz.

F. IMPLEMENTATION AND RESULTS

To achieve a soft landing and as a result of multiple crashes resulting in the legs getting broken, we modified the landing

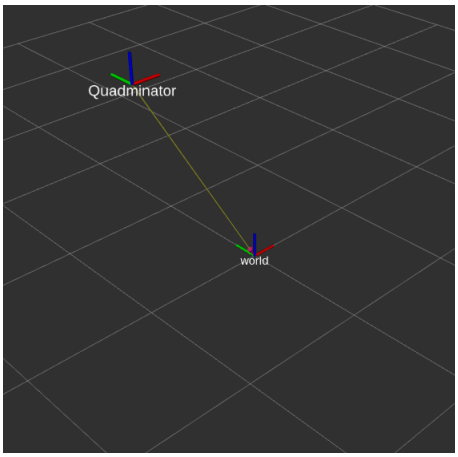

Fig. 5: World and quad coordinate systems displayed in Rviz

gear of the quadrotor by replacing the legs with cushions made of foam. We also attached a substitute board on top of the quadrotor to obtain additional weight. We did this to account for the weight of the upboard during tuning of the control commands without actually using the upboard to avoid potential damage to it. Fig. 5 shows the current hardware configuration of the quadrotor used for testing.

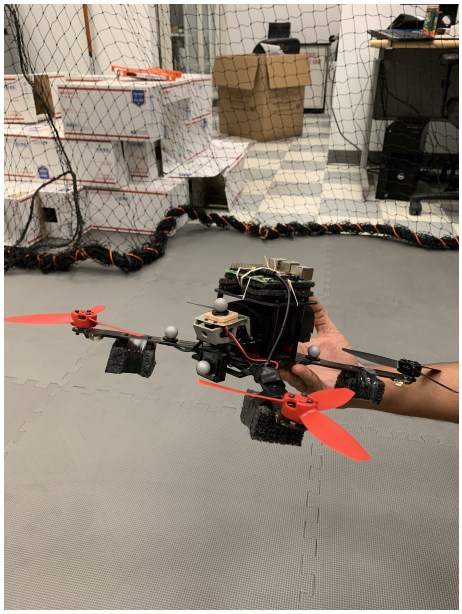

Fig. 6: Hardware modification of drone for soft landing.

The following plots shown in Fig. 7 through Fig. 12 correspond to the resulting trajectories obtained from the Vicon data and the actual waypoints sent to the quadrotor during runtime. From these plots, it can be observed that the quadrotor is better at tracking the helix trajectory than the other two given trajectories, however, there is a noticeable difference between the ideal trajectory and the actual trajectory for all given patterns. This shows the limitations of using an open loop controller scheme.

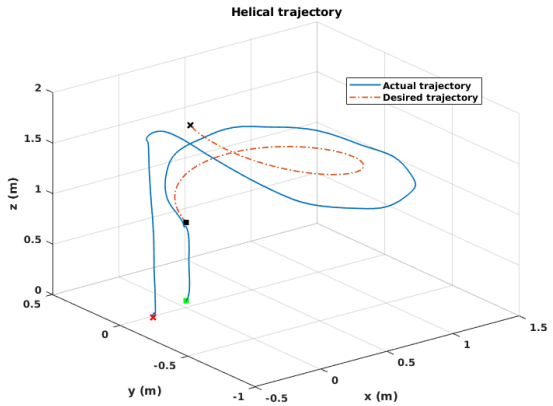

Fig. 7: 3D view of helix trajectory resulting from experiment vs ideal trajectory.

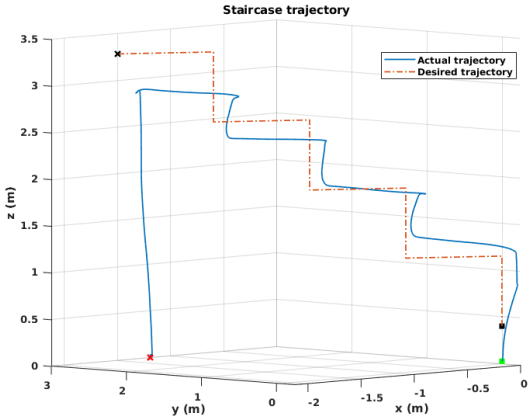

Fig. 8: 3D view of staircase trajectory resulting from experiment vs ideal trajectory.

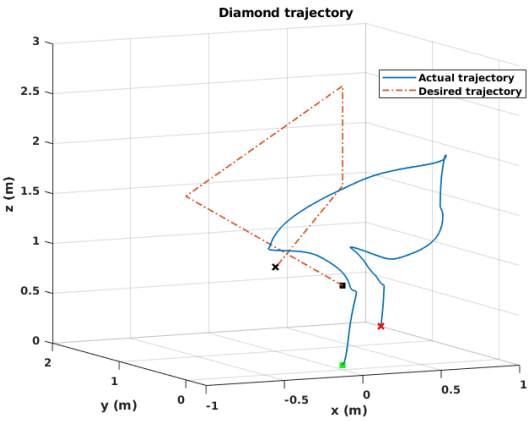

Fig. 9: 3D view of diamond trajectory resulting from experiment vs ideal trajectory.

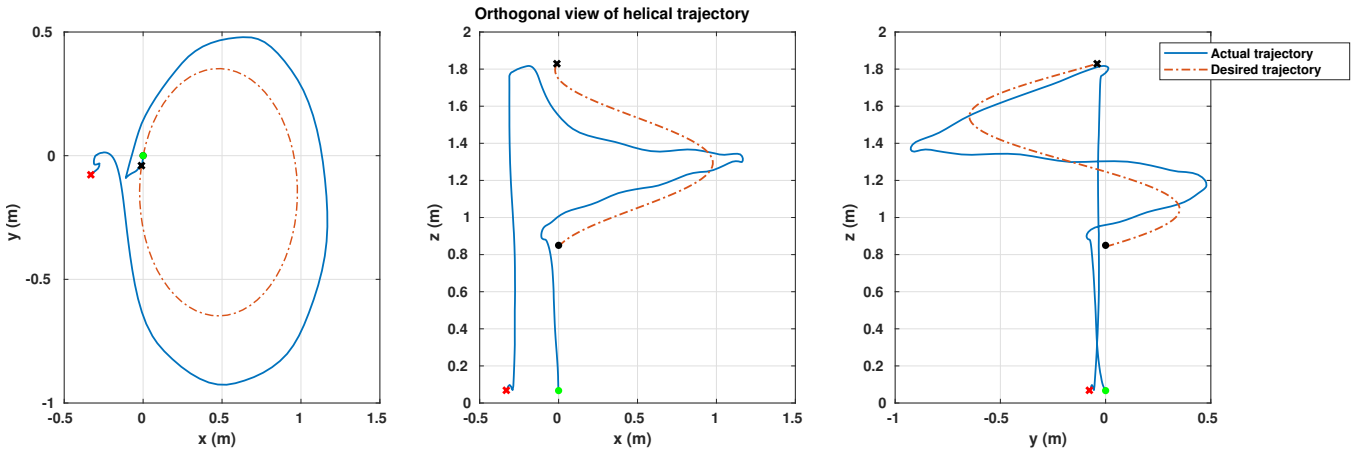

Fig. 10: 2D view of helix trajectory resulting from experiment vs ideal trajectory in the y-x, z-x and z-y planes.

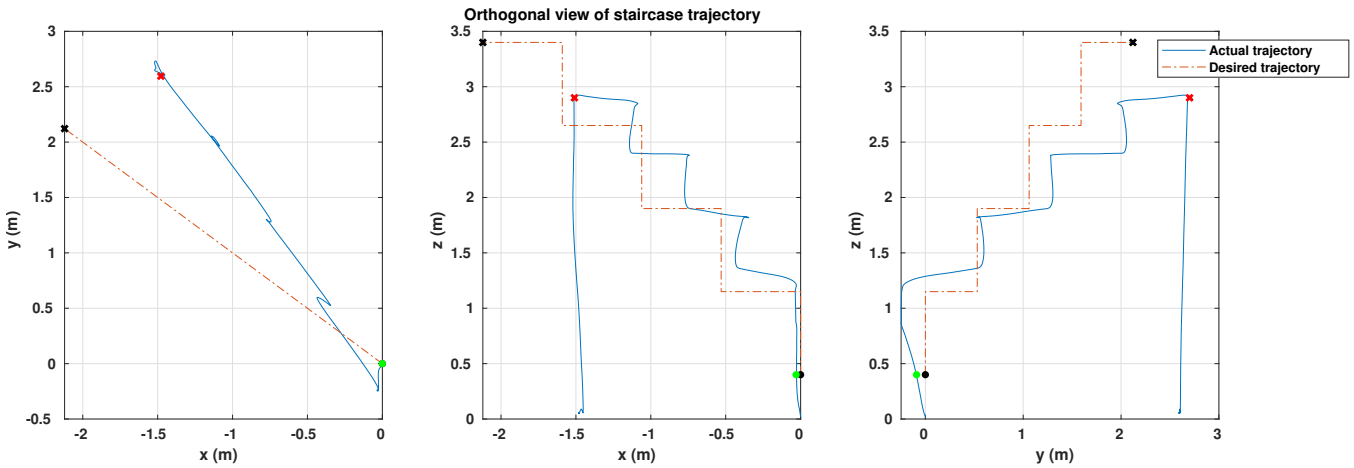

Fig. 11: 2D view of staircase trajectory resulting from experiment vs ideal trajectory in the y-x, z-x and z-y planes.

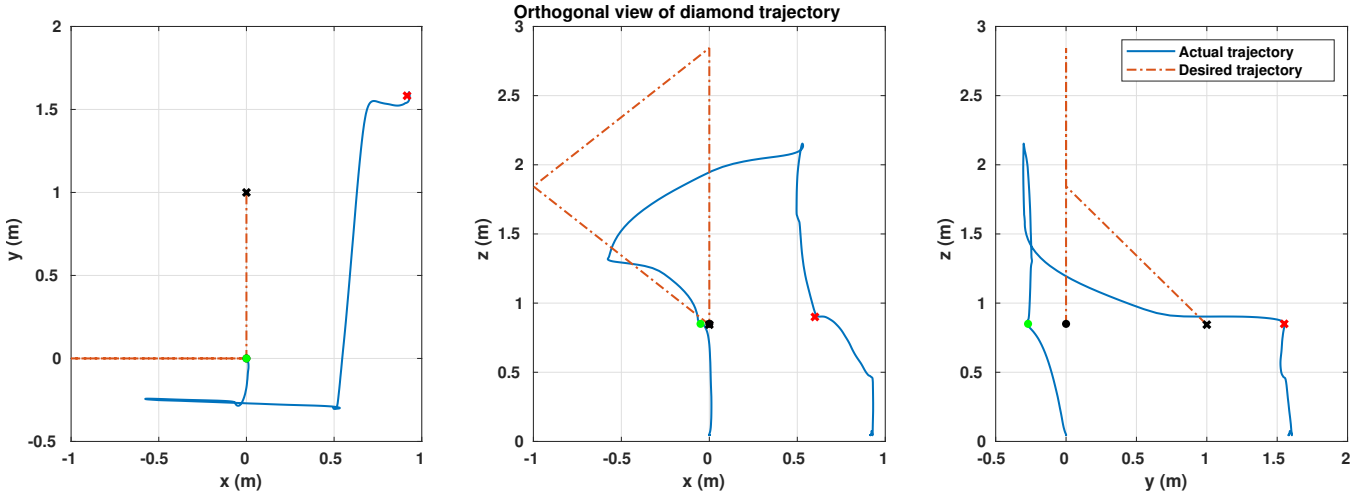

Fig. 12: 2D view of diamond trajectory resulting from experiment vs ideal trajectory in the y-x, z-x and z-y planes.