Open Loop Trajectory Following on PRG Husky

Vishnu Sashank Dorbala
UID 116907569
University of Maryland, College Park

Surabhi Verma
UID 116949460
University of Maryland, College Park

I. Problem Statement

Given the dimensions of a helix, diamond and staircase, we try to make a PRG Husky follow the corresponding trajectory without any external feedback. The firmware in the Husky controls flight by processing feedback from different sensors present on-board. For desirable trajectory following characteristics, tuning the internal PID parameters becomes essential. In order to evaluate the flight performance, we use a VICON motion tracking system to track the trajectory of the drone, which is then compared with the desired path. We then plot the VICON outcomes with the desired trajectory and do a side by side comparison.

II. Trajectories Followed

A. Helix

Trajectory specifications given:

$$r = 1m \tag{1}$$
$$h = 1m, \tag{2}$$

where, r and h are the radius and pitch of the helix. A helix can be written in terms of parametric equations:

$$x(t) = rcos(t) \tag{3}$$
$$y(t) = rsin(t) \tag{4}$$
$$z(t) = ct, \tag{5}$$

where, $t \in [0, 2\pi)$ and c is a constant giving the vertical separation between the helix's loops such that $2\pi c = h$. Once we have the parametric equations, we divide the trajectory for one turn into $n = 300$ divisions, followed by line interpolation between the way points. However, the drone does not accept commands in terms of position and orientation but in terms of linear and angular velocities required in order to reach a desired pose. We obtain linear velocities as follows,

$$v_x(t_n) = [x(t_{n+1}) - x(t_n)]/dt \tag{6}$$
$$v_y(t_n) = [y(t_{n+1}) - y(t_n)]/dt \tag{7}$$
$$v_z(t_n) = [z(t_{n+1}) - z(t_n)]/dt \tag{8}$$

Here, v_x, v_y, v_z are the velocity components in x, y, and z direction at any instant t_n, defined as t at the n^{th} division. Depending upon how fast we want the drone to complete the trajectory, we set dt which is the time taken for the drone to traverse from one waypoint to the next waypoint.

B. Diamond

Diamond trajectory defined by its corners in space are given as follows:

$$A = [0, 0, 0]m \tag{9}$$
$$B = [0, 1, 1]m \tag{10}$$
$$C = [0, 0, 2]m \tag{11}$$
$$D = [0, 0, 1]m \tag{12}$$
$$E = [1, 0, 0]m \tag{13}$$

To reach these goal points in the order as described above, we note that the drone should be able to ascend and descend laterally along with throttling downwards. For lateral movement in say, the $y-z$ plane, we give the drone a velocity component in both, the y and z directions. According to the coordinates, we have to move 45 degrees laterally. So, ideally we should give equivalent velocities in the both, the y and z directions for dt time in order for it to cover the required distance. But since it is an open loop system, we do not give the same velocities in the practical scenario and tune it such that it follows the desired trajectory (estimated visually).

C. Staircase

The staircase trajectory is required to start from the origin, $[0, 0, 0]$, and end at $[3, 3, 3]m$, with each step having its rise and run equal to $0.707m$. To follow such a pathway, the drone has to throttle upwards and move laterally by 45 degrees in the $x-y$ plane. We use a similar strategy described in the previous section for lateral descent in $y-z$ plane. In total, we perform 5 staircase maneuvers in order to reach our goal point.

III. Implementation

In order to implement each of these trajectories on the Husky, we make us of RoS. We send a 6 DoF velocity vector to the Bebop's 'cmd_vel' RoS topic to control it. In our case, as we do not require any angular velocities to obtain the desired trajectories, we only send linear velocities.

A. PID Tuning

We also tune the internal PID controller parameters for making the trajectory on the Husky more stable. The Attitude Gain values are set as [9.570, 2.158, 7.800] for K_p, K_d, and K_i respectively. The Position Gain values for non-carpet and carpeted surfaces are set to [1.1, 0.0, 1.4], and [3.5, 0.4, 2.5], respectively. The Speed Gain values are set to [0.5632, 0.1606, 0.0].

From tuning the PID values, we practically observed that increasing K_p too much gave the drone a very 'jerky' motion. However, increasing this value in general gave better results.

Reducing the K_d values only slightly improved performance. However, increasing them significantly worked in our favour, and stabilized the drone's 'strafing' motion.

B. Plots and Analysis

Figures 1-9 present the VICON outcomes of our drone experiments along with the desired 'Ground Truth' trajectory. In many cases, our outcome differs from the desired one. This we believe, can be attributed to two major things.

Firstly, we operate our drone using an 'open loop' control system, where there is no real-time feedback to correct position. Having a feedback gives the system more information which can be used to improve the trajectory of the drone.

Secondly, the internal PID of the Bebop was not efficiently tuned. This was not so much because of a lack of understanding of the effects of the controller, but more because of the time taken for implementing different configurations. Each time a new set of PID values were used, the drone needed to be restarted, which caused a lot of delay in experimentation.

Overall, we can observe that our drone does indeed replicate the basic structure of the required outcomes (atleast in terms of direction), but does not fully emulate the desired results. This is still a satisfactory result in our view however, given that the task was to make the drone follow trajectories.

IV. Acknowledgments

We would like to thank the instructors, Nitin. J. Sanket and Chahat Deep Singh for their advice and constant guidance throughout the course of this project.

V. Conclusion

In conclusion, we have configured our drone to follow 3 different trajectories in accordance with the problem statement. The outcomes showed us the importance of having external feedback in the system.

VI. References

- Configuring the PRG Husky Shell - Link
- Basic RoS Tutorials - Link
- Understanding how drones work - Link

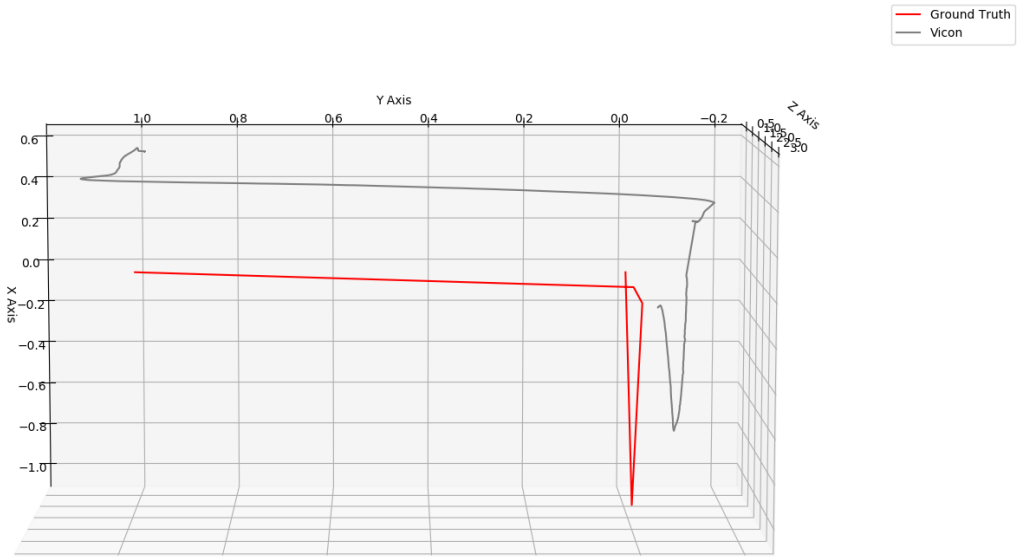

Fig. 1: Diamond Trajectory: XY View

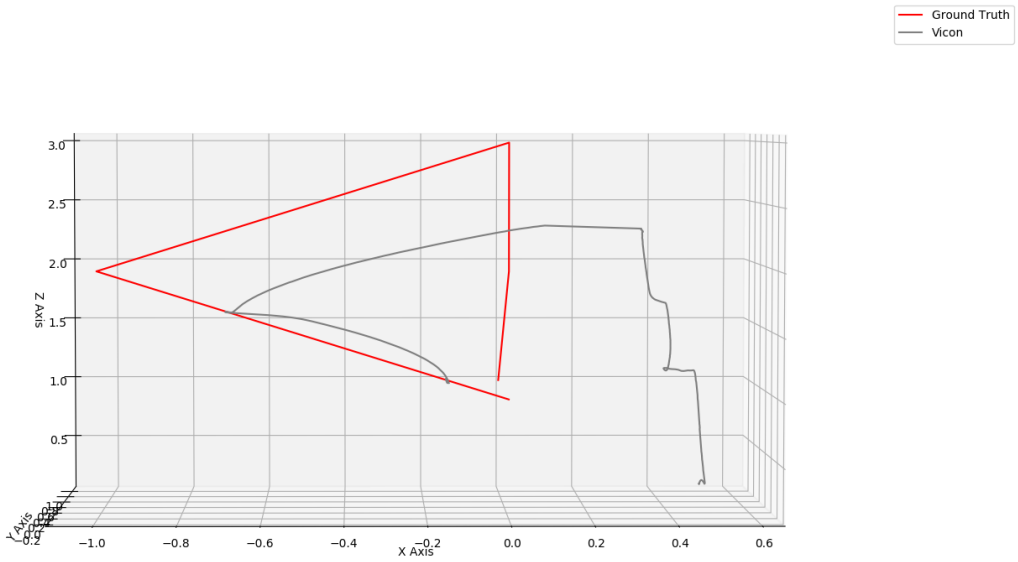

Fig. 2: Diamond Trajectory: XZ View

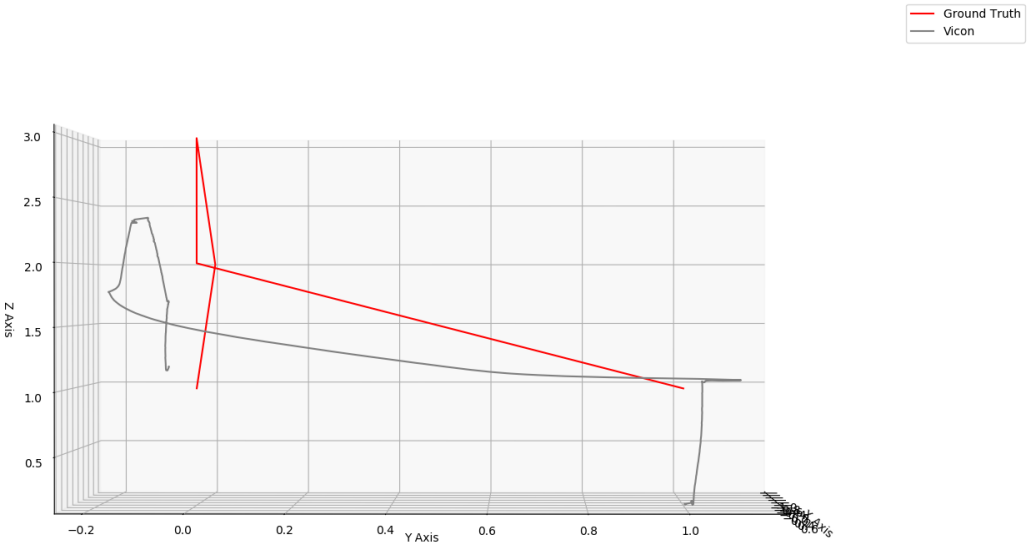

Fig. 3: Diamond Trajectory: YZ View

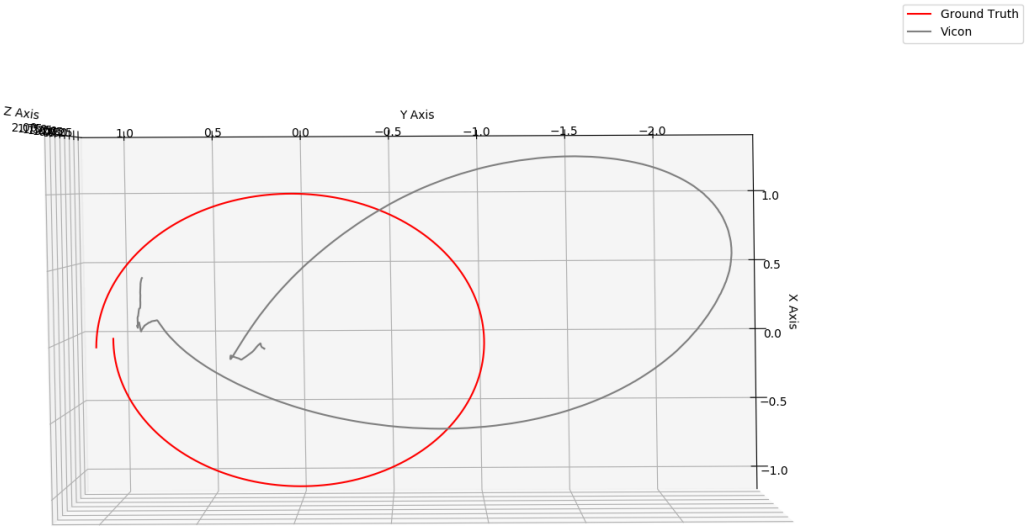

Fig. 4: Helix Trajectory: XY View

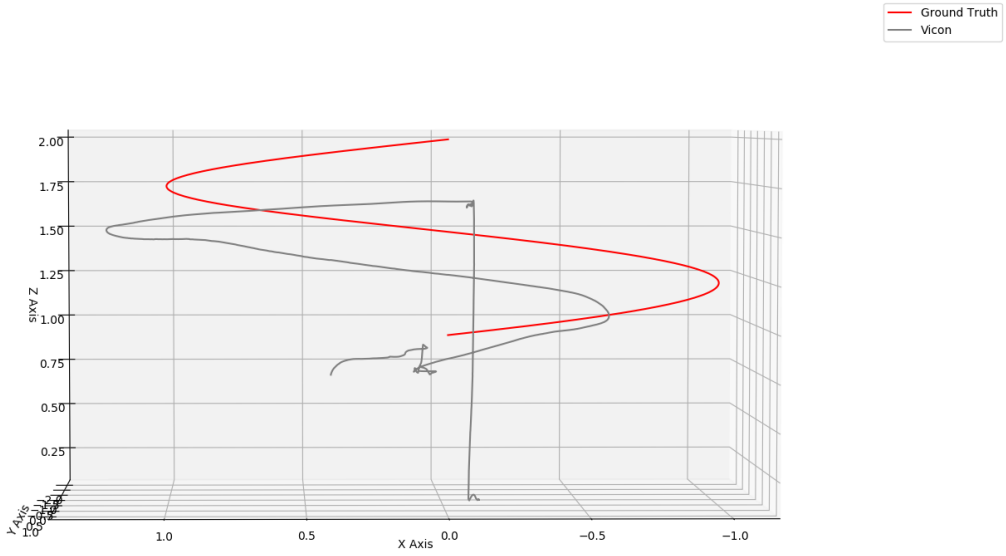

Fig. 5: Helix Trajectory: XZ View

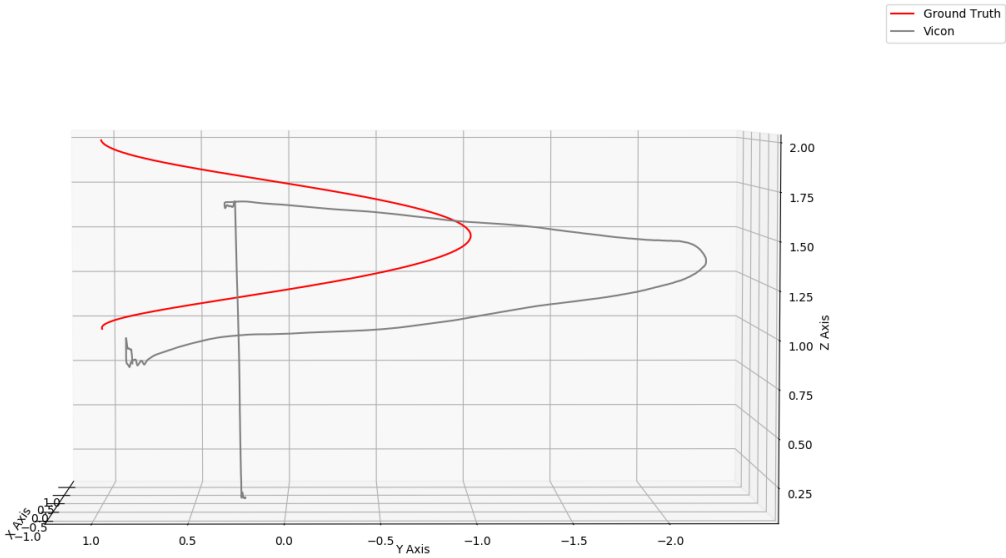

Fig. 6: Helix Trajectory: YZ View

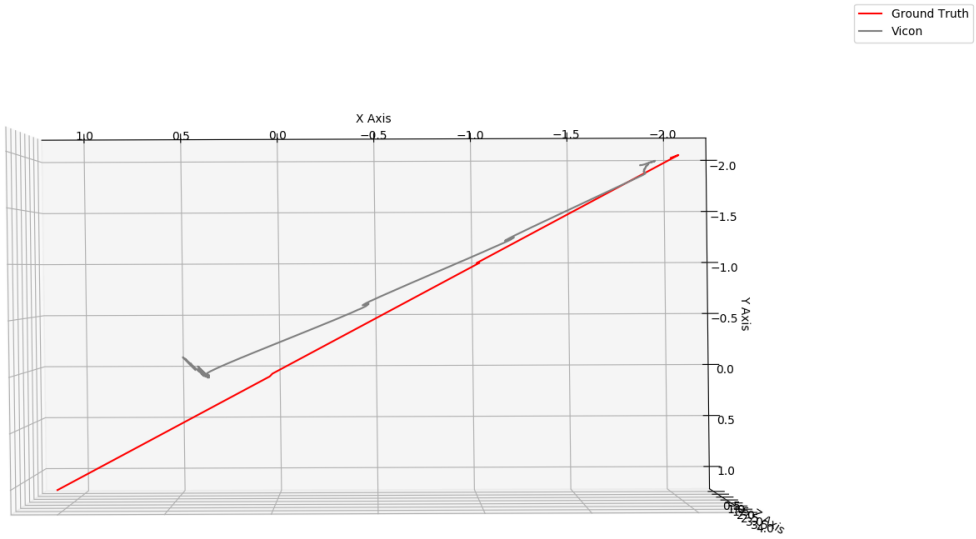

Fig. 7: Staircase Trajectory: XY View

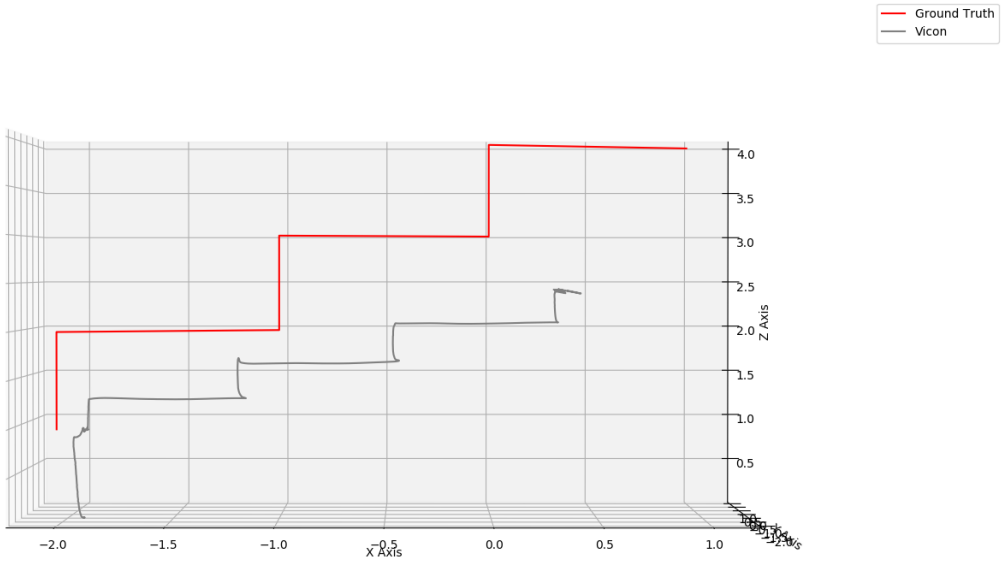

Fig. 8: Staircase Trajectory: XZ View

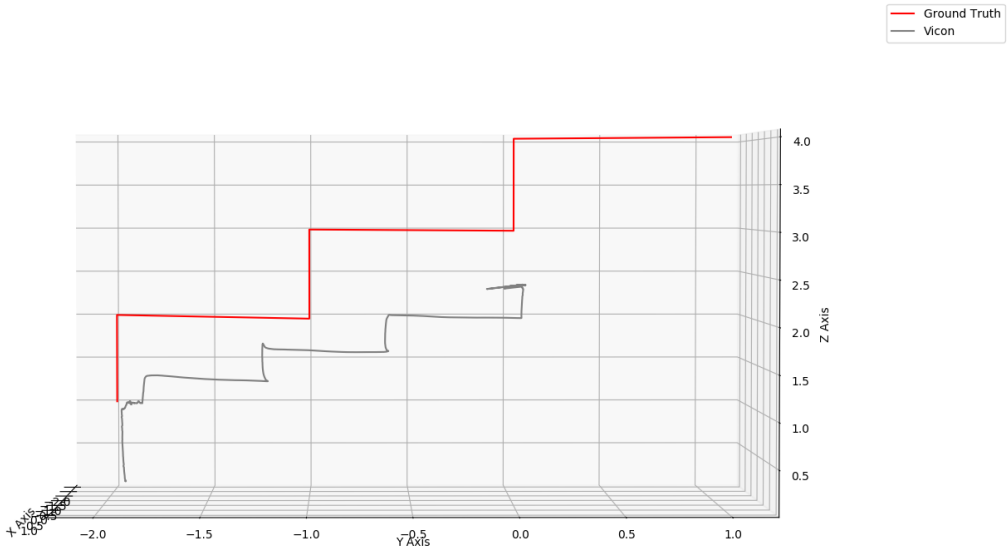

Fig. 9: Staircase Trajectory: YZ View