Trajectory Following on the PRG Husky

Mrinalgouda Patil
Alfred Gessow Center of Excellence
University of Maryland
College Park, Maryland 20742
Email: mpcsdspa@gmail.com

Curtis Merrill
Alfred Gessow Center of Excellence
University of Maryland
College Park, Maryland 20742
Email: curtism@umd.edu

Ravi Lumba
Alfred Gessow Center of Excellence
University of Maryland
College Park, Maryland 20742
Email: rlumba@umd.edu

***Abstract*—This project presents a simple, open loop approach to trajectory following on the PRG Husky. First, a simple calibration was done to relate user commands to drone movements. Two different trajectories were created - one for speed and one for accuracy, and the results were compared.**

I. Introduction/Problem Statement

The goal of this project was to implement several trajectories on the PRG Husky.

II. Calibrating the Husky

Even though we chose to use open loop control, we still needed to calibrate the physical response of the PRG Husky to a given input. This was done manually using the following approach.

A. Calibrate Single Step

First, we calibrated one step at a time. We would takeoff, give a single input in x or y, and then land. We noted that the input would saturate at 1, so we would give an input of 1 in either x or y. In other words, giving an input of 1, 10, or 100 would all result in the same physical movement. We repeated this process for $\pm$ 1 in both x and y 10 times each. For z, it was a little more difficult. We used our phone cameras to take pictures of the drone before and after climbing with a very distinguishable background. This allowed us to perform the test, and then measure the height change based on the picture. This process was again repeated 10 times. Based on these results, we found the conversion factor from input to output for our quadcopter.

B. Calibrate Multiple Steps

One thing we noticed when doing the single step calibration was that there was some drift when the quadcopter stopped. We believed this was because when the quadcopter moved forward and attempted to stop, it would tilt backwards and overcompensate. Our trajectories would not be single inputs (that would lead to choppy flight), but rather a publisher giving inputs at a certain rate.

The first step was developing a publisher and testing what rate worked best for our quadcopter. We found there was a tradeoff between the rate and the distance that the quad would move for a fixed input. If the rate was too high, the same input would have very little output. However, if the rate was too slow, we had a very choppy flight. After testing several different rates, we chose a modest publish rate of 5 Hz.

Next, we tuned a straight line in x, y, and z using a method similar to the single step. This time however, we published the commands at 5 Hz in such a way that it should have been going one meter. We initially used the scale factor obtained from the single step. This worked decently well, but we were able to run several tests in each direction to fine tune it.

III. Trajectory 1: Helix

To create the helix, the first step was to discretize a circle. Our inputs were the size of the circle, the time we wanted to complete the run in, and finally a scale factor for speed. This scale factor started as the one obtained above and was a tuning parameter to be changed.

At every position at the circle, the velocity in the xy plane was computed as a sine/cosine component.

Out of the three trajectories, the Helix was the most difficult to do using open loop control. One reason we realized is that without some closed loop, we had some drift occurring throughout the circle. We attributed this to the momentum of the quadcopter, but mostly due to the quad being not properly tuned.

To fix the problem of drift, we originally added in a 1 second hover after doing a quarter of the circle. Then to reduce time, we were able to reduce these pauses by changing the velocities throughout the circle. For example, when the x velocity changes from signs, the y velocity stays the same (positive or negative). We found that if we reduce the y velocity slightly at this point, it helps reduce the drift in the y direction. The same thing was done for the x direction

IV. Trajectory 2: 3D Diamond

For the 3D diamond, we started by creating a slow trajectory that moved point to point with a time delay between each leg. This allowed us verify that each leg was correct (this was done by taking pictures in the xz and yz planes and measuring).

Next, we took the pauses out, and the total trajectory still took a long time, however it was very accurate. We were able to increase the speed by increasing the size of each step. This was done by keeping the publishing rate at 5 Hz but increasing the size traveled each step. We were able to reduce the time by about 50 percent. However, this came at the expense of accuracy, as our turns became more rounded and less sharp. We especially had issues between legs 1 and 2 and legs 2 and 3. The y and z direction reverse completely during these changes, which required extra effort to overcome the momentum of the quad, mainly when the quad was moving faster.

V. Trajectory 3: Staircase

The staircase was done in a similar way as the Diamond. First, a very slow trajectory was created, to start with a very accurate solution. Small pauses were located between the steps to allow for easy calibration.

Next, the pauses were removed and the trajectory was improved by over 50 percent by increasing the size of each step. We didn't encounter as many issues moving to higher speeds compared to the 3D diamond, which we believe is because there aren't as many reversals in direction.

VI. Results

The videos of our PRG Husky flying the given trajectories are found in the attached folder. Below, the plots of the drone flying each trajectory in each of the three planes are shown. The actual drone movement, obtained using Vicon data, is compared to the input trajectory.

For each trajectory, two different paths were created. One was a slower run that attempted to be as accurate as possible. The second run was much faster, which caused the accuracy to decrease. The tradeoff between speed and accuracy is something that must always be accounted for.

In the below plots, the red line is the Vicon data and the blue line is the desired trajectory. The times quoted were when we started the run to when we finished the run. We removed the takeoff time and the hover time after takeoff and before landing (to steady before and after the run).

A. Trajectory 1: Helix

For the Helix, we see that both the fast run and the slow run matched the desired trajectory decently well in the XY plane. Both had a uniform offset in the z direction (that started about a quarter of the way through), and is visible in the XZ and YZ planes.

1) Slow Run: The slower took 16 seconds, and matched very well in XY.

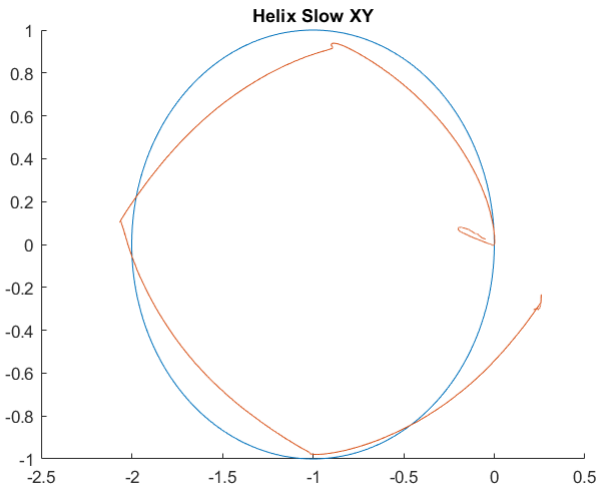

Fig. 1. Trajectory 1 Slow - XY Plane

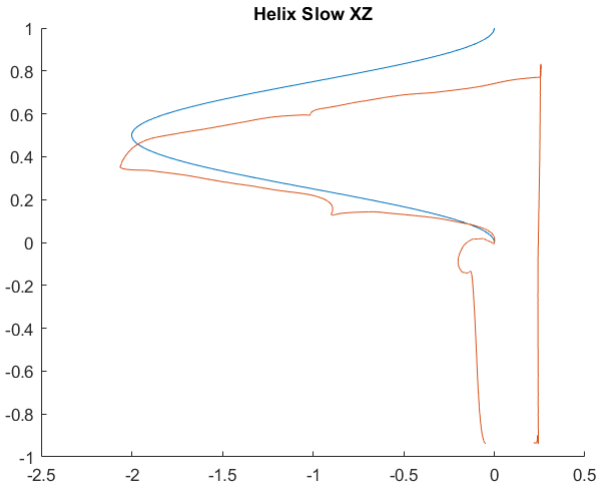

Fig. 2. Trajectory 1 Slow - XZ Plane

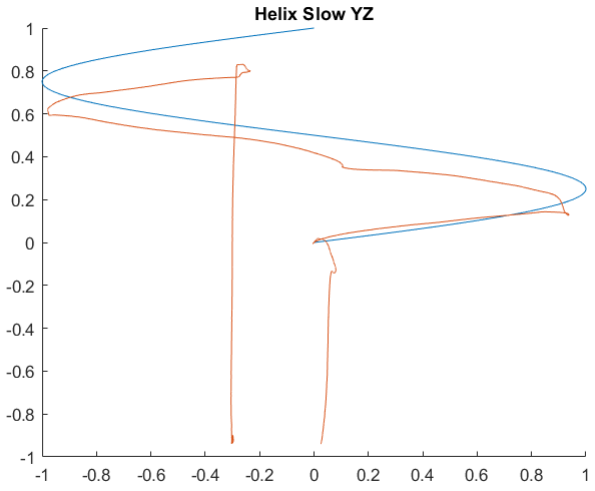

Fig. 3. Trajectory 1 Slow - YZ Plane

2) Fast Run: The fast took 13 seconds and matched in XY well, but less so compared to the slow run.

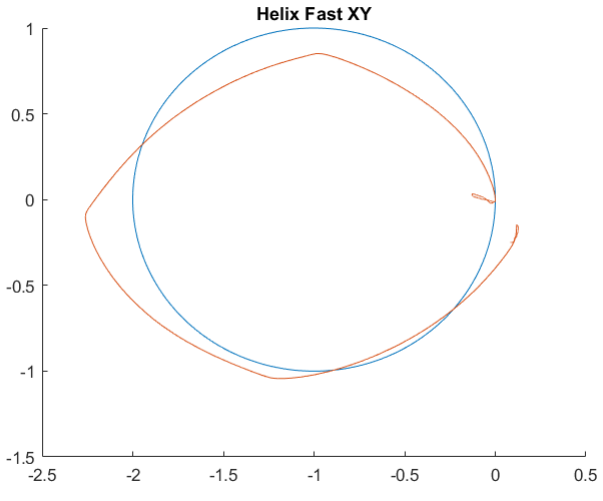

Fig. 4. Trajectory 1

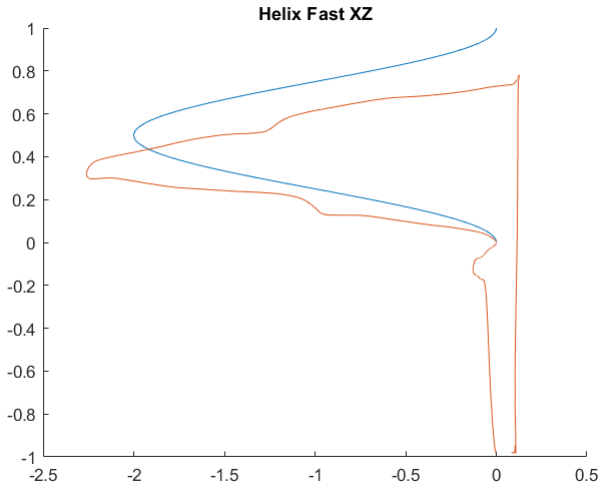

Fig. 5. Test 9 Roll

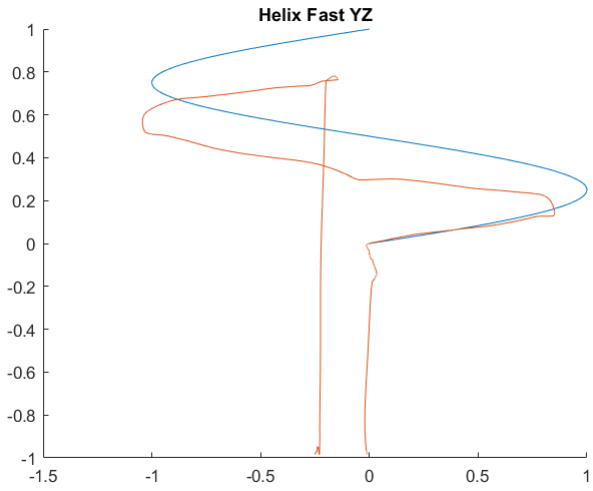

Fig. 6. Test 9 Roll

B. Trajectory 2: 3D Diamond

For the 3D diamond trajectory, the difference between the slow and fast run is noticeable. The fast run has curved turns due to the momentum of the quadcopter from the previous legs. Slowing the trajectory down, we see that the quadcopter matches the estimated path much better.

One thing we noticed when comparing the Vicon to the desired trajectory is that we lost Vicon at certain points. And when Vicon reappeared, it did not seem to be in the proper location (it seemed to just be right where the Vicon lost the signal). This was especially true for our fast run. We watched the videos of the flights, and for the slow run, there looked like there was a drift in y by roughly 20 cm from takeoff to landing, and Vicon said about 40 cm. This is close enough,

so we left the data as is. However, for our fast run, we again estimated about .2m of drift in the y direction but saw drifts mcuh larger (1m) from Vicon. Therefore, we will start by showing the raw data given by the Vicon. Then, we will show plots where we adjusted the data after the Vicon lost the signal based on the video that we took for our Fast Run.

1) Slow Run: The slow run took around 12 seconds.

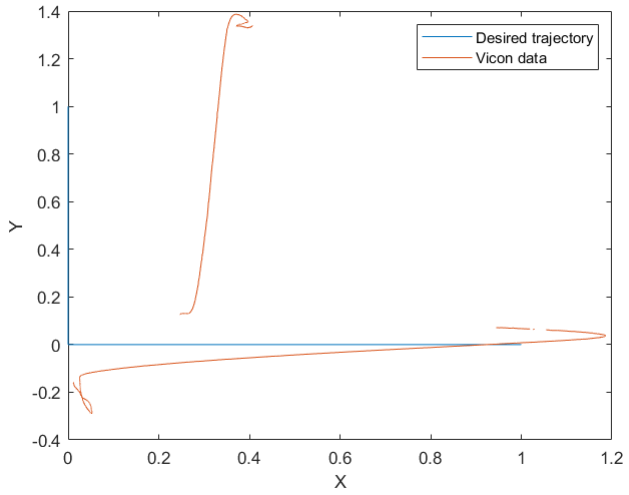

Fig. 7. Trajectory 1 Slow - XY Plane

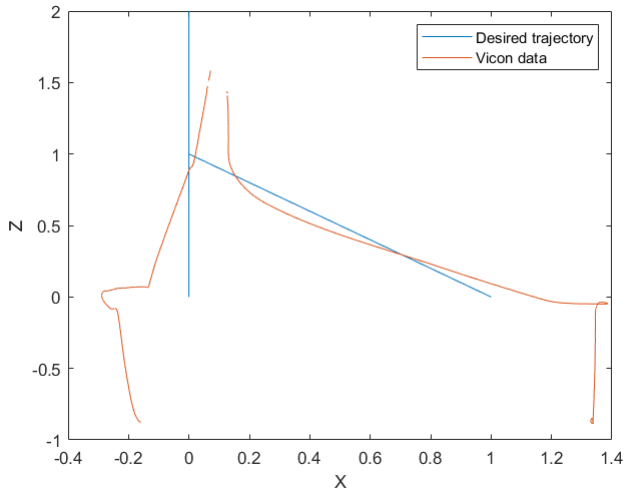

Fig. 8. Trajectory 1 Slow - XZ Plane

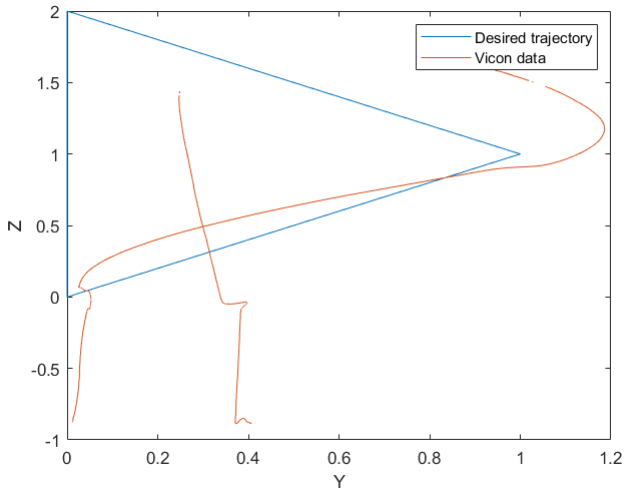

Fig. 9. Trajectory 1 Slow - YZ Plane

2) Fast Run: The fast run took a little over 6 seconds.

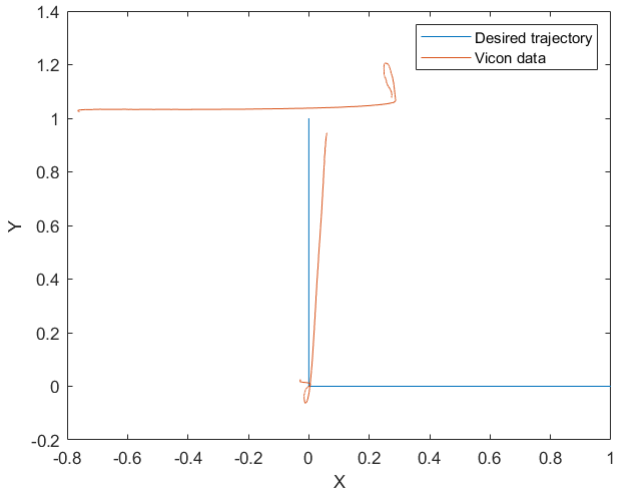

Fig. 10. Trajectory 1

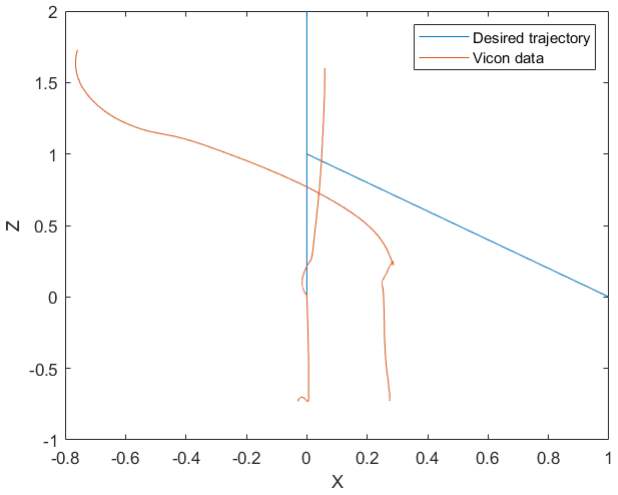

Fig. 11. Test 9 Roll

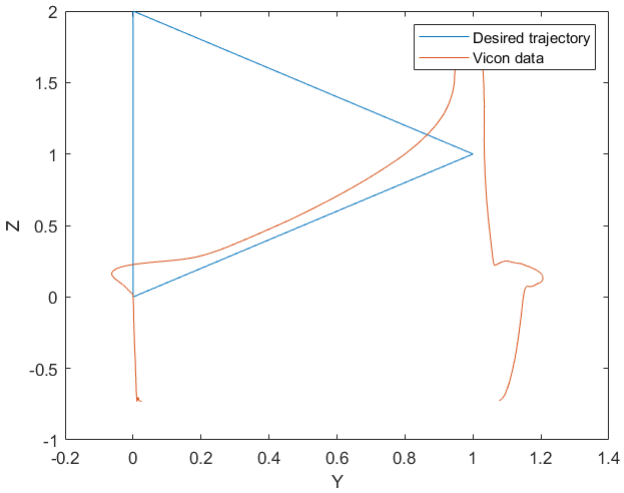

Fig. 12. Test 9 Roll

C. Trajectory 2: 3D Diamond - Adjusted

These plots are the same Vicon data as above, however the section after the Vicon lost the signal have been biased based on our observations from the video. This was just done for our Fast Run.

1) Fast Run: The fast run took a little over 6 seconds.

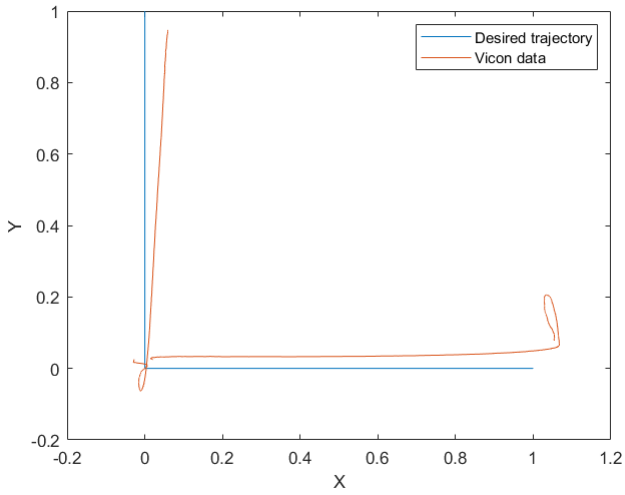

Fig. 13. Trajectory 1

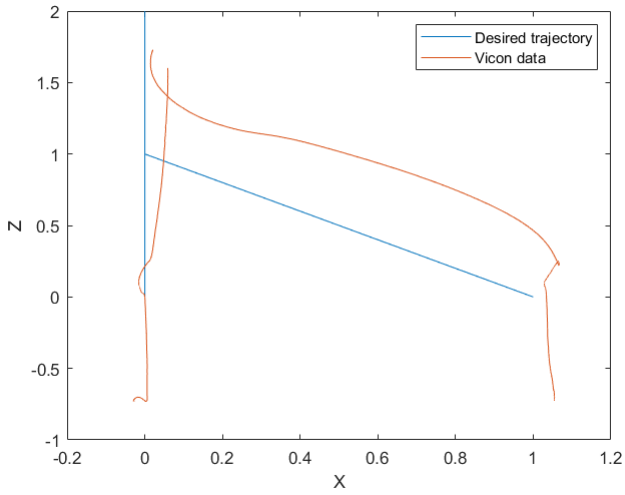

Fig. 14. Test 9 Roll

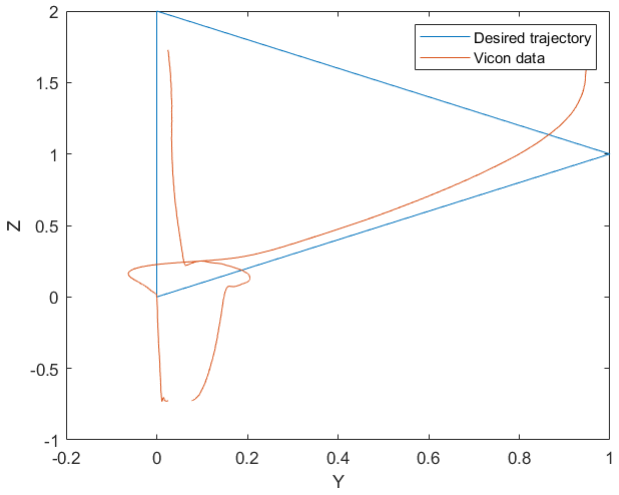

Fig. 15. Test 9 Roll

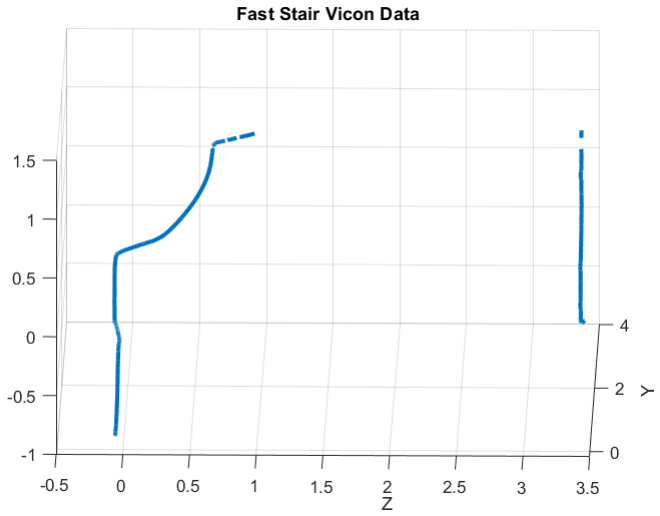

Fig. 17. Raw Vicon Data for Fast Stair Trajectory

D. Trajectory 3: Staircase

The Vicon data for the staircase was very inconsistent. Like the 3D diamond there were times when the Vicon would lose the drone. However, this occured at much greater frequency for the staircase making the Vicon data boarderline unusable. The 3D plots of the raw Vicon data are shown below for our fast and slow runs.

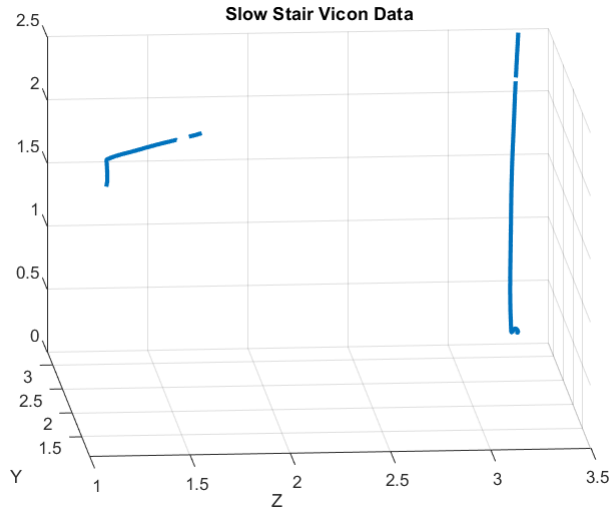

Fig. 16. Raw Vicon Data for Slow Stair Trajectory

As one can notice, for the slow run the Vicon captures only one half step (the in-plane portion) as well as the landing. The interesting thing is it only catches the second step and not the first step. A possible reason for this could be that the drone was started too far in the corner to be picked up on Vicon. The top step was around the same height as the Vicon, and might have even been slightly above the sensors, which explain while it was not capture.

For the fast run, over one complete step was captured, which is an improvement, but still doesn't leave too much to work off of.

The 3 view plots seem to indicate that we overshot our intended altitude for the slow trajectory, but it is difficult to draw too many conclusions from the Vicon data.

For the faster trajectory, it seems like we did not hit the required altitude or distance. This might be because we did not properly tune the quad for the faster speed.

1) Slow Run: The slow run took around XX seconds.

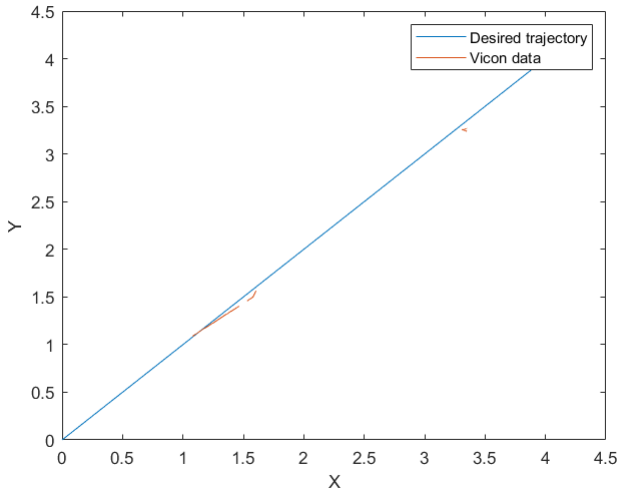

Fig. 18. Trajectory 1 Slow - XY Plane

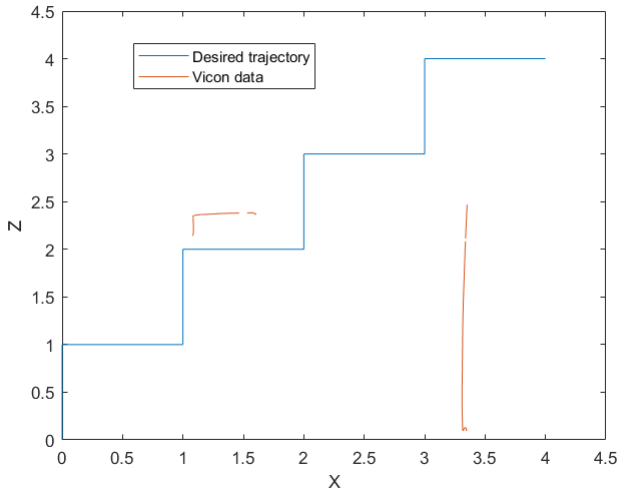

Fig. 19. Trajectory 1 Slow - XZ Plane

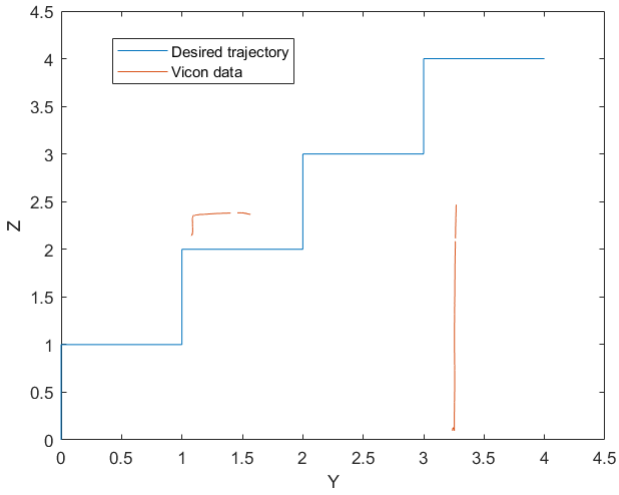

Fig. 20. Trajectory 1 Slow - YZ Plane

2) Fast Run: The fast run took a little over XX seconds.

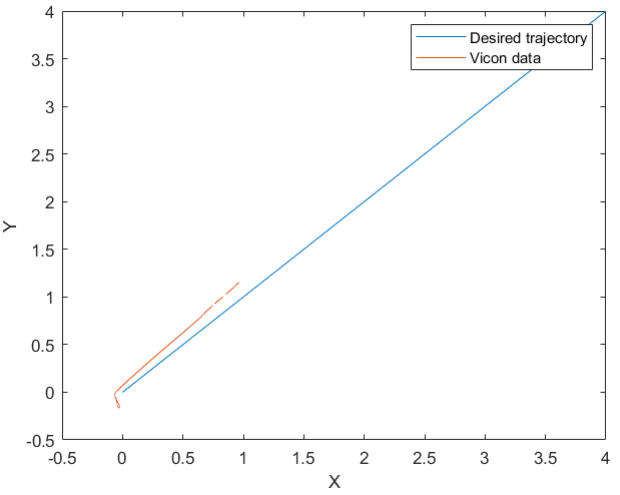

Fig. 21. Trajectory 1

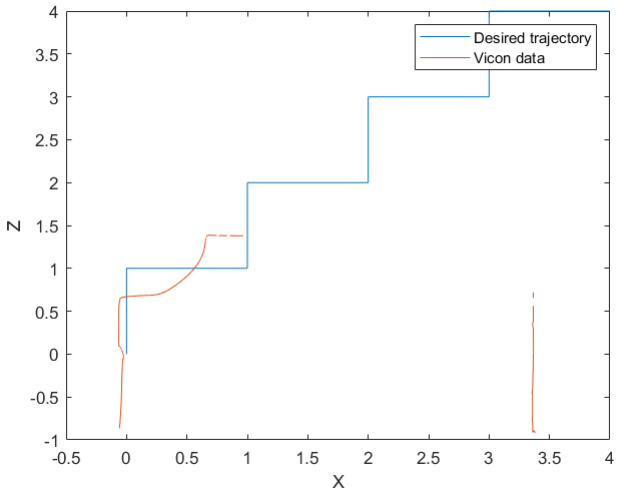

Fig. 22. Test 9 Roll

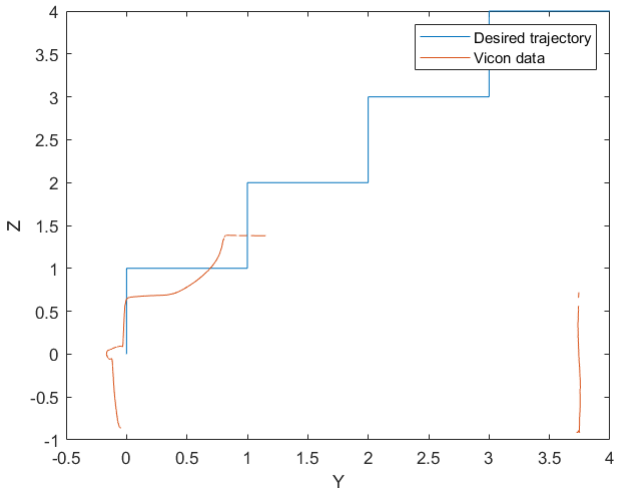

Fig. 23. Test 9 Roll

VII. Conclusion

The main conclusion that we drew from this work was that there will always be a tradeoff between speed and accuracy while path planning. Despite how much the quadcopter is tuned, at faster speeds the required change in momentum is large, which leads to rounder turns and more deviation from the straight line path. Although we certainly did not have our quad perfectly tuned, we know that these are issues that cannot be avoided.

A. Lessons Learned

One issue we encountered was that our quad's characteristics changed after we had one crash. This meant that we had to go and tune our parameters again. Another lesson we learned was that each time we took out the quad and started flying, there might be some small differences than the time before. We aren't sure exactly what the cause was (possibly using different propellers at different locations, or some other small issue), but this taught us that we should always do some general calibration checks every time we fly.

Another lesson that we learned was that open loop control can work for basic trajectories, but when doing realistic time-discretized trajectories, closed loop control is needed. For example, using an open loop controller for the 3D diamond and Staircase worked very reasonably well, with some tuning. However, using open loop control for the helix required much more tuning. If certain parameters were to change, such as the radius or pitch, there would be much more tuning required to be accurate. Therefore, while closed loop controllers might take more effort in tuning up front, they are better and more robust in the long run as more complicated trajectories can be implemented with less effort.

References

[1] ENAE788 Class 5 Slides
[2] Some Code taken from learnopencv.com/rotation-matrix-to-euler-angles/