16-662 ROBOT AUTONOMY

PROJECT 1A: REPORT

Team Evolutus
William Ku (wku), Yuhan Long (yuhanlon), Yu-Te Cheng (yutec), Dawei Wang (daweiwan).

January 29, 2015

1 System Diagram

Figure 1 displays the system diagram of the software pipeline (`simple_io_test.m`).

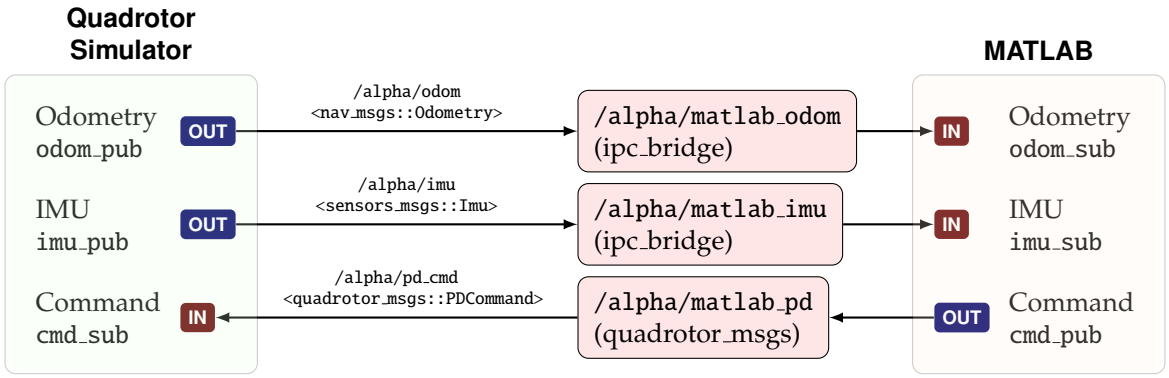


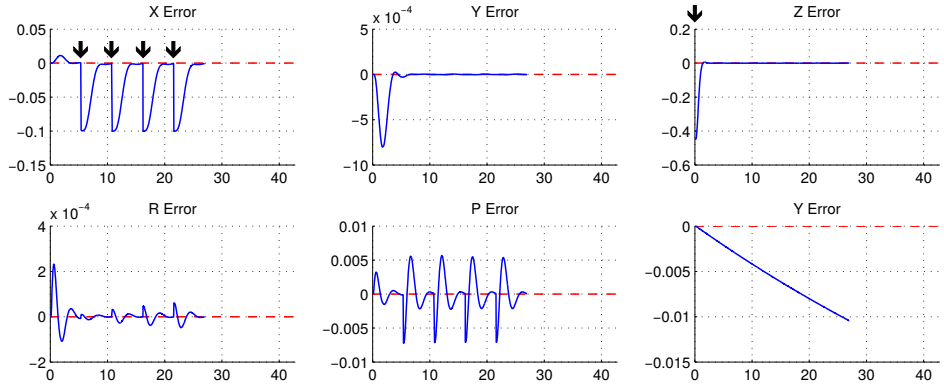
Figure 1: Interfacing Matlab and Quadrotor Simulator with IPC

The quadrotor simulator runs the simulation and periodically publishes the odometry and inerial measurement unit (IMU) data to the Robot Operating System (ROS). These data are subscribed by a group of bridging nodes that convert them into data structures interpretable by MATLAB, where the data are processed, and messages containing control commands are derived accordingly. These messages again go through a bridging node, and are subscribed by the simulator which closes the loop.

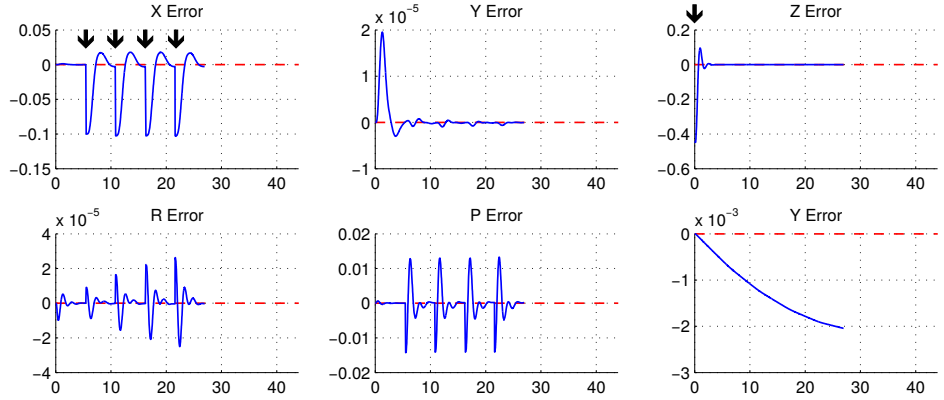
2 Hover Performance

The system oscillates between waypoints through a slight overshoot reaction. The system then converges to the desired position in a short amount of time (within two oscillation cycles). Modifying the gains associated with position control and attitude control will change the oscillation behavior of the system. Specifically, tuning the `Kp` gains will increase the oscillation of the system, while changing the `Kd` gains will damp the oscillation behavior, at the cost of slower system response.

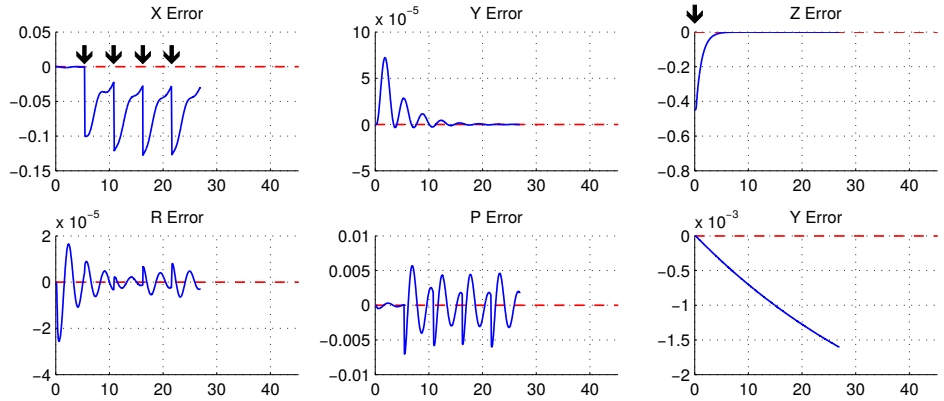
After tuning for quick response time and stability, we arrived at a set of gains that resulted in Figure 2a. The position of the system converges within one second and within two oscillation cycles to the desired values. The roll and pitch of the system are intentionally tuned for quick response. The quick response allows the system to react immediately to any error even though oscillations are often coupled with quick response time.



(a) Normal Behavior



(b) Behavior with Doubled `Kp`



(c) Behavior with Doubled `Kd`

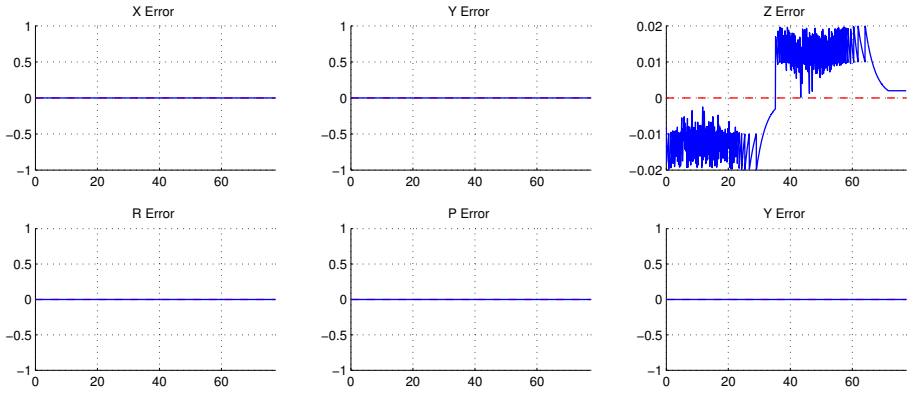
Figure 2: Position and Attitude (versus time [second]) with Different Gains
New waypoints are sent to the system at points indicated by arrows.

To observe the effects of modifying `Kp` and `Kd` gains, we simply doubled each gain and plotted the results. Figure 2b shows the oscillating effects from doubling `Kp` values for both position control and attitude control. More severe overshoots and oscillations in most fields resulted from higher `Kp` gains. Doubling `Kd` gains smoothed out the responses while adding a slight delay to the response time, as shown in Figure 2c. Due to this delay, the x-position is lagging behind the desired value.

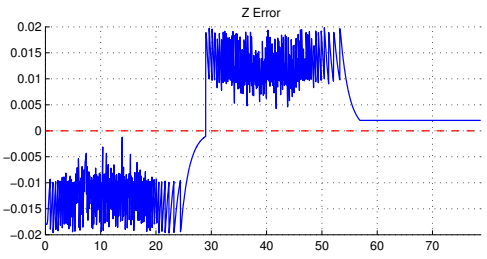
3 Line-Tracking Performance

To develop the line tracking controller, we produced desired position and velocity profiles for the system to follow. Each profile contains 300 waypoints. In our implementation, the system will take off, fly to a height of one meter in the z-direction, hover, and return to the ground. The velocity profile resembles a hill with a flat top. The system will accelerate to a fixed speed, remain at the desired speed, and decelerate as it approaches the final desired position. The tracking reference is updated to the next waypoint when the position error measurement - in Euclidean distance - is within a defined threshold of one centimeter.

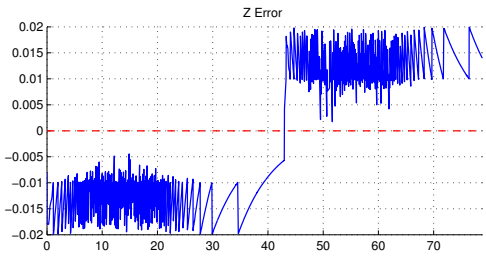
The system z-value shows significant oscillations about the waypoints, as shown in Figure 3a. This effect is caused by the alternating acceleration and deceleration of the system as it closes in at the waypoint before the target reference position is updated to the next waypoint. The position and velocity profiles and the odometer measurements are shown in Figure 4a. The tracking performance is perceivably satisfying. Noise is observed near the initial movements of the system, but this effect is expected as the system accelerates from the resting position.



(a) Normals Gains



(b) Doubled `Kp`

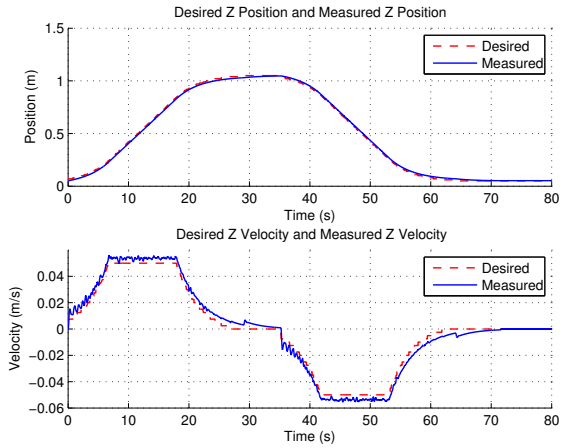


(c) Doubled `Kd`

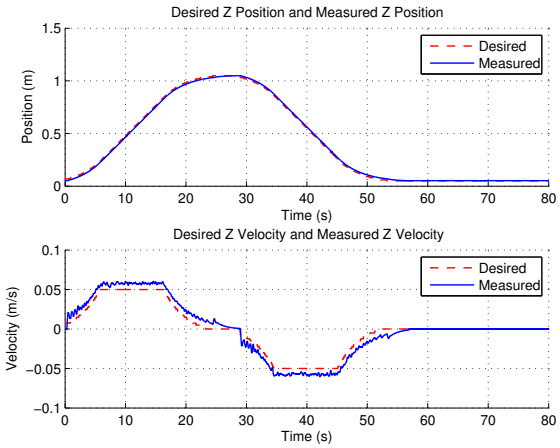
Figure 3: Errors with Different Sets of Gains
Errors on x, y, roll, pitch, yaw remain the same for different gains.

Modifying the gains associated with the position control (outer loop) changes the oscillation behavior of the system. The general behaviors of tuning `Kp` and `Kd` remain the same as described in the second question. One note is that, for this problem, we increased `Kd` value and lowered the `Kp` value in the z-direction to allow our controller to track the velocity closely. In another word, the increase in `Kd` value added more weights on velocity in our controller.

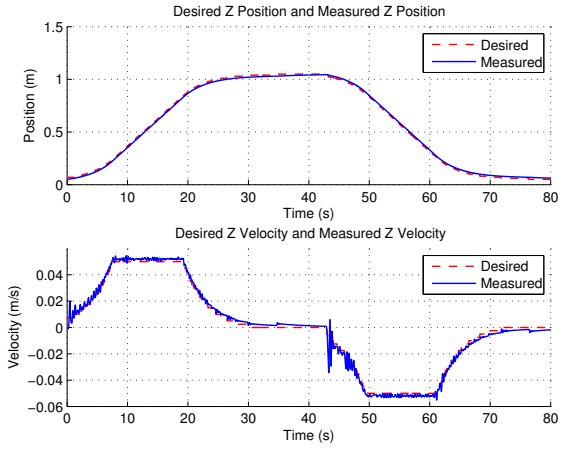
To observe the effects of modifying `Kp` and `Kd` gains, we simply doubled each gain and plotted the results. Figure 3b shows the oscillating effects from doubling `Kp` values for the position control. By doubling `Kp` values, the velocity tracking error increases. The increase in `Kp` values shifts the weights from velocity tracking towards position tracking in our controller. Doubling `Kd` gains allowed the controller to track the velocity more closely, at the cost of slower response to converge to the desired position, as shown in Figure 3c. The tracking performances for position and velocity for doubling `Kp` values and `Kd` values are shown in Figure 4b and Figure 4c, respectively. The system tracks the position and velocity closely. Noise is again observed near the initial movements of the system.



(a) Normal Gains



(b) Doubled `Kp`



(c) Doubled `Kd`

Figure 4: Carefully Tuned Gains Yield Optimum Performance