Autonomous Quadrotor Landing on a Moving Platform

Stan Brown & Chris Choi

1 Introduction

Over the years there has been been a growing interest in autonomous landing of quadrotors as landing is often one of the most dangerous and risk prone parts of flight. There are now several commercial and research based implementations of autonomous landing with the DJI Phantom and Pixhawk flight controller both being capable of automatic landing using a series of sensors and control algorithms. However there does not yet seem to be a reliable method of landing a quadrotor on either a moving platform or in cases where there is relatively large cross wind disturbance.

In recent years there have been at least least 5 academic publications on the topic [1, 2, 3, 4, 5, 6] that approach the problem from a variety of angles. Overall none of these approach have completely solved the problem and most approaches only work on very slow moving platforms with little or no wind disturbances.

One of the most challenging parts with autonomous landing is the difficulty of obtaining a reliable estimate of the landing target relative to the quadrotor, the landing target can often go out of the camera's field of view causing the quadrotor to lose track of the target, other vision based artifacts that include lighting, lens distortion, color, etc. In many cases GPS is not a viable option both the landing target and the quadrotor due to the lack of accuracy unless an RTK GPS system is used. In many cases the use of an RTK system is not a readily available option due to cost or quadrotor needs.

1.1 Related Work

The autonomous landing problem can be thought of as a set of three separate control problems or stages. First the quadrotor must detect the relative position and velocity of the landing target using either GPS of visual methods. Next the quadrotor must determine a rendezvous location with the landing target and plan the required flight trajectory. Once the quad has rendezvoused a final landing trajectory must be calculated between quadrotor and the landing pad.

For perception existing solutions use a variety of simple to complex techniques. In [2] a basic color threshold technique was to identify the landing target, while [6] used optical flow in images captured on board the quadrotor to obtain necessary relative information for control.

A comparison between a PID and Linear Quadratic Control (LQC) was explored in [4]. However the authors admit, even though LQC performed slightly better than the PID controller the difference was minor and may be attributed to the additional time spent tuning the LQC. In [6] a PID controller was developed to land on a oscillating platform in the vertical direction with no lateral movement. While interesting, the solution took over 1 minute to transition from hovering over the landing pad to landing. Additionally the experiments did not seem to account for pitch or roll of the platform.

In this project we focus developing a set of controllers that produce the final landing trajectory and using a fiducial marker called an AprilTags [7] to provide the estimated state of the quadrotor relative to the landing pad.

2 Methodology

This project can be separated into two individual but complementary components. First, a significant effort was spent on ensuring the quadrotor's state was estimated at a rate of at least 20Hz with AprilTags. Secondly, using information from the estimated quadrotor state, a state machine and a PID position controller were developed to track and lands on a desired target.

2.1 Measurement and State Estimation using AprilTags

One of the challenges we faced when estimating the quadrotor state using AprilTags was the computational resource required to sustain a rate of atleast

20hz. This problem is exacerbated when the image size increases, as do the computational time needed to extract the pose estimate of the AprilTag, from our initial experimentation the computational time grows exponentially as image size increases as shown in (add a figure here). This problem was also noted in the work of [5] where he addressed the issue by reducing the brightness of the image so that the majority of the image is black except for the AprilTag, thus reducing image processing requirements. An example of this implementation is shown in Figure 1 This allowed Ling [1] to calculate states at a rate of approximatly 10 - 15 fps but it came at the cost of requiring the brightness parameters to be set ahead of time and also the lost of robustness in estimating the AprilTags.

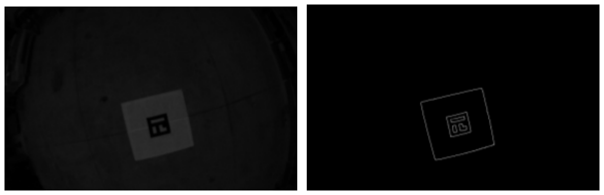


Figure 1: Ling's approach to optimizing AprilTags detection [5]

In this work, the slow rate of state estimation was addressed using two novel ideas. First, the image size and calibration parameters are set depending on the distance between the quadrotor's position relative to the AprilTag. Secondly, we introduce the use of a Region of Interest (ROI) window to focus only on the AprilTag to reduce computational time required to estimate the pose. These novel ideas will be discussed in the following subsections.

2.1.1 Adaptive Image Preprocessing

As discussed previously, the low rate of the AprilTag library when processing images of 640 by 480 pixels results in an update rate of approximately 3 to 5 fps on a Snapdragon 8 core processor, which is too low to be used effectively in a PID control loop. Building upon the work of [5], who improved the AprilTag estimation rate by reducing the brightness on the image such that the majority of the pixels are black, additionally a set of image preprocessing methods such as Canny-Edge thresholding were also used.

2.1.2 Adaptive Image Windowing

If one assumes that the quadrotor does not move to fast, there is relativity low rotation between image captures, and that the image update rate is quite fast (60 fps in this implementation), then the location the AprilTag in the following image can be estimated based off of the location it was last observed in the previous image. Therefore whenever an AprilTag is measured in an image, a bounding box around the AprilTag is calculated and then used in the following image to segment out portions of the image where the AprilTag is unlikely to be. An example of this implementation is highlighted in Figure 2. In cases where the AprilTag is not observed in the expected location (bounding box), the size of the bounding box is set to the size of the image and the entire image is processed.

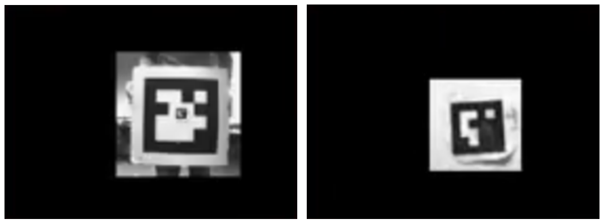


Figure 2: Adaptive Image Windowing

While the adaptive windowing procedure decreases the computational resources needed for extracting pose estimates, there are two other problems that windowing cannot solve. First, as the size of the observed Apriltag increases, windowing is no longer adequate in decreasing computational time, this is an issue in cases where a high update rate is required (e.g. quadrotor landing). Second, in our use case as the observed AprilTag gets larger it becomes easy for the camera to lose track of the AprilTag itself, increasing the risk of no estimation for that time.

2.1.3 AprilTag Inception

To address both the decreasing state update rate as a function of proximity and reduce the probability of losing sight of the AprilTag during landing a secondary AprilTag is embedded in the larger AprilTag, we term this technique AprilTag Inception (patent pending). This secondary AprilTag is assigned with a different family id to avoid confusion during detection, and is placed at the center of the larger primary AprilTag. Whenever the secondary AprilTag is captured in the image, the adaptive windowing method is set to track only the secondary AprilTag rather than the larger one, which reduces the portion of the image that must be processed in with each image capture. An example of this procedure is highlighted in Figure 3.

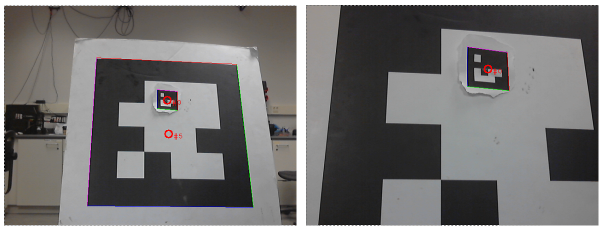

(a) 2 Apriltags Detected (b) 1 Apriltag Detected

Figure 3: Apriltag Inception

2.1.4 Adaptive Image Down-sampling

A final technique used to optimize the AprilTag detection was the introduction of down sampling the image size as the estimated distance between camera and AprilTag decreases. We devised 2 image down sampling sizes 320×280 (half resolution) and 160×140 (quarter resolution), when the distance between the camera and the AprilTag is less than 1.5 and 3 meters, the images are down sampled to the two resolutions respectively. If the camera is further than 3 meters from the AprilTag, the native resolution of 640×480 is used. This change only affected the image processing time when the quadrotor is within 3 meters of the quad and has the desired effect of maintaining a high AprilTag estimation rate.

In order to use down sampling effectively without corrupting the AprilTag estimations, 3 camera calibration files were also computed at the full, half and quarter camera resolutions, corresponding calibration files were then used with the AprilTag library during the state estimation procedure based on the current image resolution to obtain the correct pose estimates.

2.2 Controller and State Monitoring

In this project we assumed that the quadrotor has successfully rendezvoused with the target vehicle and has a clear view through the camera image of the landing target located on the target vehicle at all times. The first step towards autonomous landing is to position the quadrotor above the landing target such that the displacements in the x and y directions in the horizontal plane normal to the landing surface are near zero while simultaneously matching speed and aligning the axis with the target vehicle. Next, while maintaining the near zero displacements in the x and y directions, the quadrotor descends at a safe rate such that the vertical distance between the quadrotor and the landing target is reduced until landing is achieved. Ideally the quadrotor will descend rapidly when the vertical displacement between the qaudrotor and target is large, slowing down just above ($\approx 1m$) the landing target to minimize time and energy requirements of the landing manoeuvre.

To achieve the described behaviour, we developed a PID position controller that outputs attitude commands such as roll and pitch to reduce the horizontal and vertical distance between the quadrotor and the landing platform, as well as an altitude command to maintain a safe approach altitude such that the target does not go out of view during landing. We omit yaw control as we assume the quadrotor will be axis aligned with the direction of travel of the moving target.

The position controller takes the quadrotor's desired position and current position in x, y and z of the world frame as inputs, these inputs are mapped to attitude and altitude commands in the roll $\hat{\phi}_t$, pitch $\hat{\theta}_t$ and thrust $\hat{T}_z$ relative to the quadrotor frame as outlined in Equations 2, 3, 4, these equations however assume that yaw ψ is 0, if yaw is non-zero then roll and pitch need to be adjusted to account for the yaw changes as in Equation 5 and 6. The altitude command is adjusted when roll $\hat{\phi}$ and pitch $\hat{\theta}$ are non-zero to maintain altitude as in Equation 7.

$$e = e_{\text{setpoint}} - e_{\text{actual}} \tag{1}$$

$$\hat{\phi}_t = K_p e_x + K_i \sum_{t=0}^{t} e_{x,t} dt + K_d \dot{e}_{x,t} \tag{2}$$

$$\hat{\theta}_t = K_p e_y + K_i \sum_{t=0}^{t} \dot{e}_{y,t} dt + K_d \dot{e}_{y,t} \tag{3}$$

$$\hat{T}_z = K_p e_z + K_i \sum_{t=0}^{t} \dot{e_z}_{,t} dt + K_d \dot{e_z}_{,t} \tag{4}$$

$$\phi_t = \cos(\psi_t)\hat{\phi}_t - \sin(\psi_t)\hat{\theta}_t \tag{5}$$

$$\theta_t = \sin(\psi_t)\hat{\phi}_t + \cos(\psi_t)\hat{\theta}_t \tag{6}$$

$$T = T_{\text{hover}} + \frac{\hat{T}_z}{cos(\phi_t)cos(\theta_t)} \tag{7}$$

3 Results

The methods outlined in sections 2.1 and 2.2 were implemented and tested on real quadrotor with an onboard computer and camera. The quadrotor is assembled from a DJI F450 quadrotor frame, 4 Emax 2213-935KV motors with complementary DJI E310 420S 20A electronic speed controllers. A Pixhawk v2.4 running the PX4 firmware stack was selected for the flight controller and an Odroid XU4 was used as an on-board computer to processes process video captured from a using a PointGrey Firefly 2.0 camera operating 60 frames per second (fps) with a resolution of 640 by 480 pixels through a FuijiFilm 135 degree FOV lens. The camera calibration was done using the ROS camera calibration package.

Images captured from the video stream are then processed using the AprilTag library after passing through the preprocessing steps outlined in 2.1 and the estimated state is used by the PID controller (outlined in 2.2 and also running on the Odroid) to produce attitude commands that cause the quadrotor to move towards a goal location. All attitude commands are send to the Pixhawk from the Odroid via usb and rely heavily on the Mavros package, which is a wrapper around the popular Mavlink communication protocol.

Using this testbed both the accuracy of the AprilTag state estimation was evaluated in the following sections. At this point in time a motion capture system was used in place of the AprilTag library to provide state estimates as it was found to be more reliable for initial testing purposes. In the coming week the PID controllers will also be tested using only the state estimates from the AprilTag library.

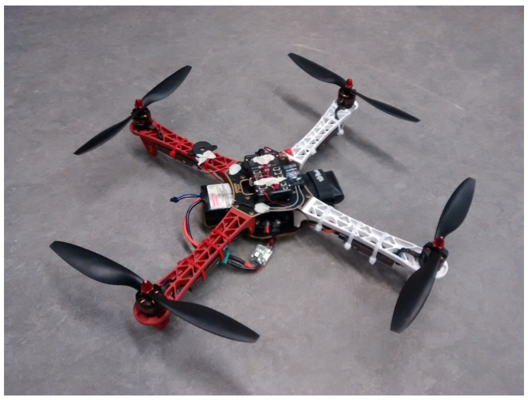
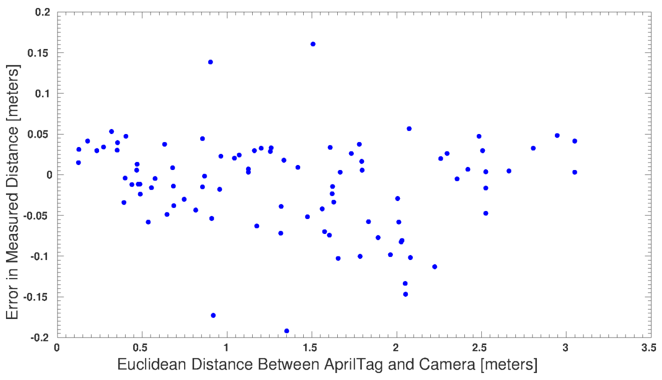
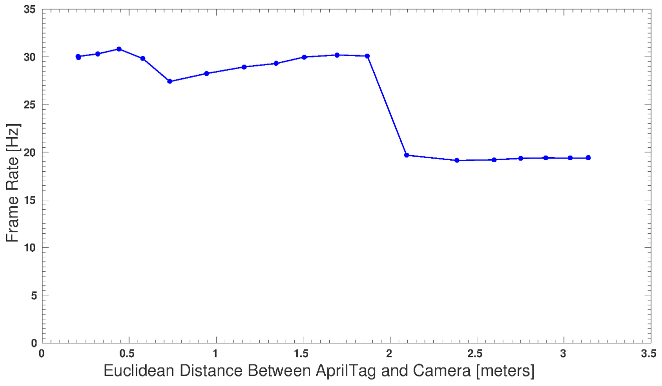


Figure 4: DJI F450 with Pixhawk v2.4



(a)



(b)

Figure 5: Evaluation of AprilTag state estimation and update rate as functions of distance

3.1 Evaluation of AprilTag Accuracy and Update Rate

To evaluate the accuracy of the AprilTag and effects that the adaptive windowing procedures have on the state estimations an Optitrack motion capture system (MOCAP) was used to provide a ground truth. By monitoring the pose of both the camera and the AprilTag in the MOCAP system, the series of measurements were taken using the MOCAP system to calculate the pose of the AprilTag relative to the camera from a variety of ranges and angles. A comparison between the estimated pose from the AprilTag and the MOCAP system as a function of distance is highlighted in Figure 5. Figure 5 also highlights the rate at which state estimates are received from the AprilTag library after being passed through the adaptive windowing procedure. A demo of the AprilTag pose estimation can also be seen in this video.

From Figure 5 it appears that the average error of the estimated euclidean distance compared to MOCAP ground truth is within approximately

$\pm 20cm$ for any measurements between 0 and 3 meters. Due to the limited range of the MOCAP system distances greater than 3 m between the camera and the AprilTag were not tested. The system was also able to maintain a frame rate of approximately 30 fps in cases camera is less then 2 meters away from the AprilTag while producing estimates with an error within approximately $\pm 0.05cm$. In future, an analysis of the sources of error in Figure 5 will be conducted as well as evaluation of the error as a function of angle between the AprilTag and the camera.

It is suspected that the PointGrey Camera used in this test was not operating at the requested frame rate of 60 fps. This was not noted until after analysising the data and will be investigated this coming week. In the past there have been problems with the PointGrey camera changing its settings after several reboots, often resting the frame rate from 30 fps to 60 fps with no warning or indication. If this is the case, then the update rate of the AprilTag library on the Odroid may actually be faster than what is shown in Figure 5. In cases where a more powerful computer can be used, the adaptive image processing method results in the AprilTag library operating at 60 fps with no delay, making it extremely useful on machines that can leverage greater computational power.

3.2 PID and Controller Results

The PID controllers dervied in section 2.2 were tested on the quadrotor using the MOCAP system. Due to time constraints evaluation of the PID controller was only conducted using the MOCAP system to provide state estimates, but in the coming week further tests using the AprilTag system will be conducted. After a signifcant amount of testing the following values for the proportional, derivitave and integral gains (K_p, K_d, K_i) were determined for the roll, pitch and height controllers. Using state information from the MOCAP system, the yaw of the quadrotor was fixed at zero degrees for all of the following tests.

Roll Controller	
K_p	0.205
K_i	0.05
K_d	0.02

Pitch Controller	
K_p	0.205
K_i	0.05
K_d	0.02

Thrust Controller	
K_p	0.2
K_i	0.0
K_d	0.02

Figures 6 7, 8, 9, 10, 11 higlight the PID position control signal (roll, pitch and thrust) and corresponding proportional, integral and derivative error components while a quadrotor was set to hover at $(0, 0, 1)$ for approximately 65 seconds. Large disturbances were applied manually to the hovering quadrotor with a testing apparatus (a plank of wood) to test the correction behaviour of the position controller implemented. These disturbances are applied at the 15, 21, 37, and 57 second marks in all six Figures. In the case of the elevation controller, there was no integral term used due to the risk of a suck throttle occurring in cases where the quadrotor did not detect landing and tack off correctly.

From Figures 6 and 8 that the roll and pitch commands are bounded to $\pm 20^o$ further more when the quadrotor is perturbed in any direction the position controller has been able recover within 2-4 seconds. Same applies to the thrust commands, when altitude is perturbed the quadrotor is able to self correct to the desired altitude with time. A video of the plotted dataset can be viewed here.

In all cases there is a significant amount of noise on the derivative error term as highlighted in Figures 7, 9, and 11. It is suspected that this large error is due to the small time step of 0.01 seconds used in the calculation of the derivative term which has side effect of greatly amplifying any noise in the MOCAP state estimations. In cases where the MOCAP measurement did not correctly correspond to the current true state of the vehicle large spikes can be noted which appears around the 9 second mark in Figures 7, 9, and 11. The noise in the MOCAP system can be reduced by adding additional markers to the quad and by making a more rigid frame to mount the tracking markers.

These plots also highlight the need to place a limit on the commanded output as in the roll and pitch PID controllers there were cases where the sum of the derivative term with the proportional and integral error terms may lead to a commanded pitch or roll greater than 60 degrees which is considered far to aggressive for the quadrotor used in these experiments. Further tuning and testing of the PID controller will be carried out in the coming weeks along with tests that do not require the yaw to be stabilized at zero degrees. Theoretically it will be possible to directly apply the PID controllers to state estimates obtained by the AprilTag directly and in such cases it should be possible to use these same controllers to follow and land on a moving AprilTag. Such a test will be the focus of the majority of effort in the coming weeks.

4 Conclusions and Future Work

A set of algorithms were developed to provide fast, reliable and accurate estimations of state between a landing pad and a quadrotor using a series of image preprocessing methods and the MIT AprilTags library. These methods were tested and evaluated using a motion capture system to provide ground truth and were shown to provide update rates of greater than 20 Hz for a low power onboard, quadrotor mounted computer system. A set of PID controllers were developed, tested and evaluated using the same motion capture system to provide state estimates and a ground truth. A set of stable yet responsive PID gains were found and tested using this system and an evaluation of their behaviour in the presence of large disturbances was conducted.

In the coming weeks the state estimates used by the PID controllers will be provided solely by an AprilTag located on the ground and a series of procedures using these controllers will be derived such that the quadrotor will hover over the AprilTag before initializing a landing procedure. Once such a landing can be performed consistently in the face of minor disturbances it, the same procedure will be applied to quadrotor and landing system where the landing platform will be mounted to a moving vehicle.

References

[1] D. Lee, T. Ryan, and H. Kim, "Autonomous landing of a vtol uav on a moving platform using image-based visual servoing," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pp. 971–976, May 2012.

[2] J. Kim, Y. Jung, D. Lee, and D. Shim, "Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera," in *Unmanned Aircraft Systems (ICUAS), 2014 International Conference on*, pp. 1243–1252, May 2014.

[3] H. Voos and H. Bou-Ammar, "Nonlinear tracking and landing controller for quadrotor aerial robots," in *Control Applications (CCA), 2010 IEEE International Conference on*, pp. 2136–2141, Sept 2010.

[4] J. Friis, E. Nielsen, R. F. Andersen, J. Boending, A. Jochumsen, and A. Friis, "Autonomous landing on a moving platform," *Control Engineering, 8th Semester Project, Aalborg University, Denmark*, 2009.

[5] K. Ling, U. of Waterloo. Department of Mechanical, and M. Engineering, *Precision Landing of a Quadrotor UAV on a Moving Target Using Low-cost Sensors*. University of Waterloo, 2014.

[6] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto, "Landing a vtol unmanned aerial vehicle on a moving platform using optical flow," *Robotics, IEEE Transactions on*, vol. 28, pp. 77–89, Feb 2012.

[7] "AprilTags C++ Library." `https://people.csail.mit.edu/kaess/apriltags/`. Accessed: 2016-01-12.

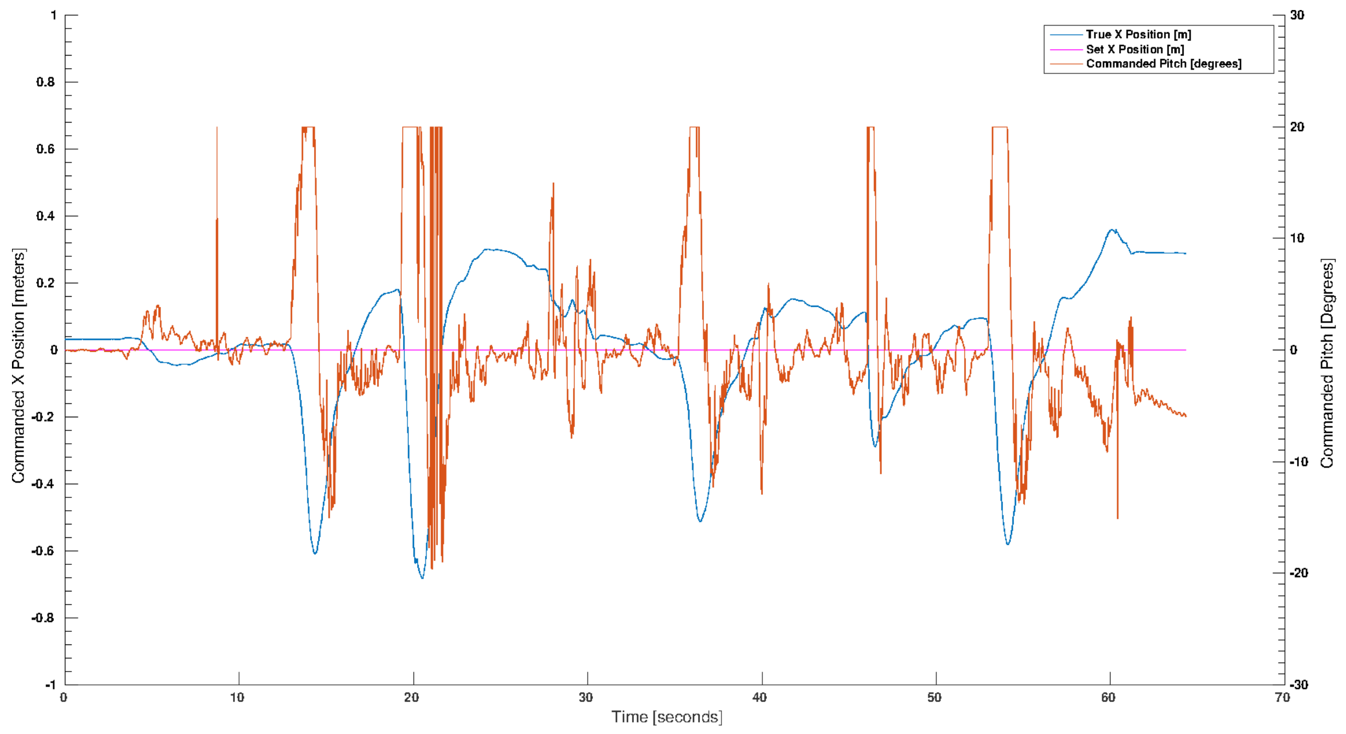


Figure 6: Position Controller - Pitch Command

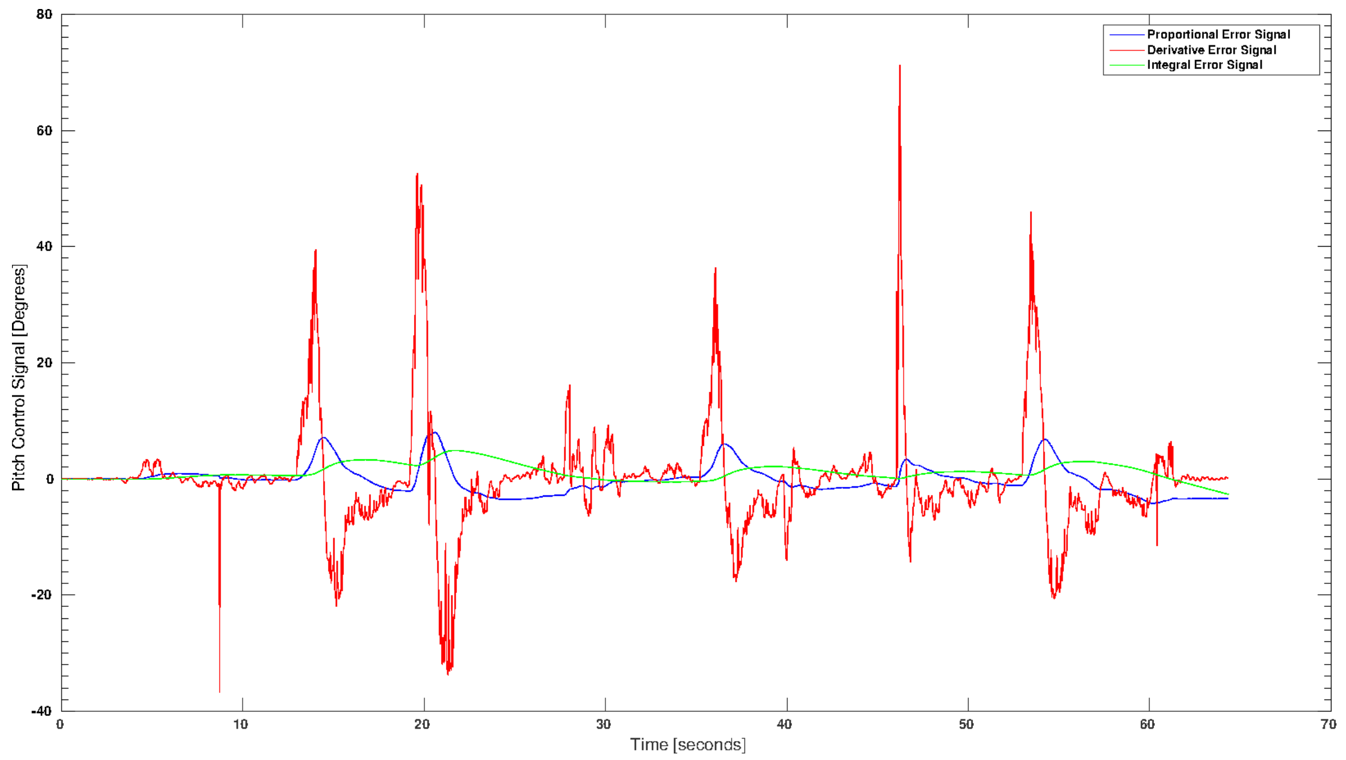


Figure 7: Position Controller - Pitch Control Signal Components

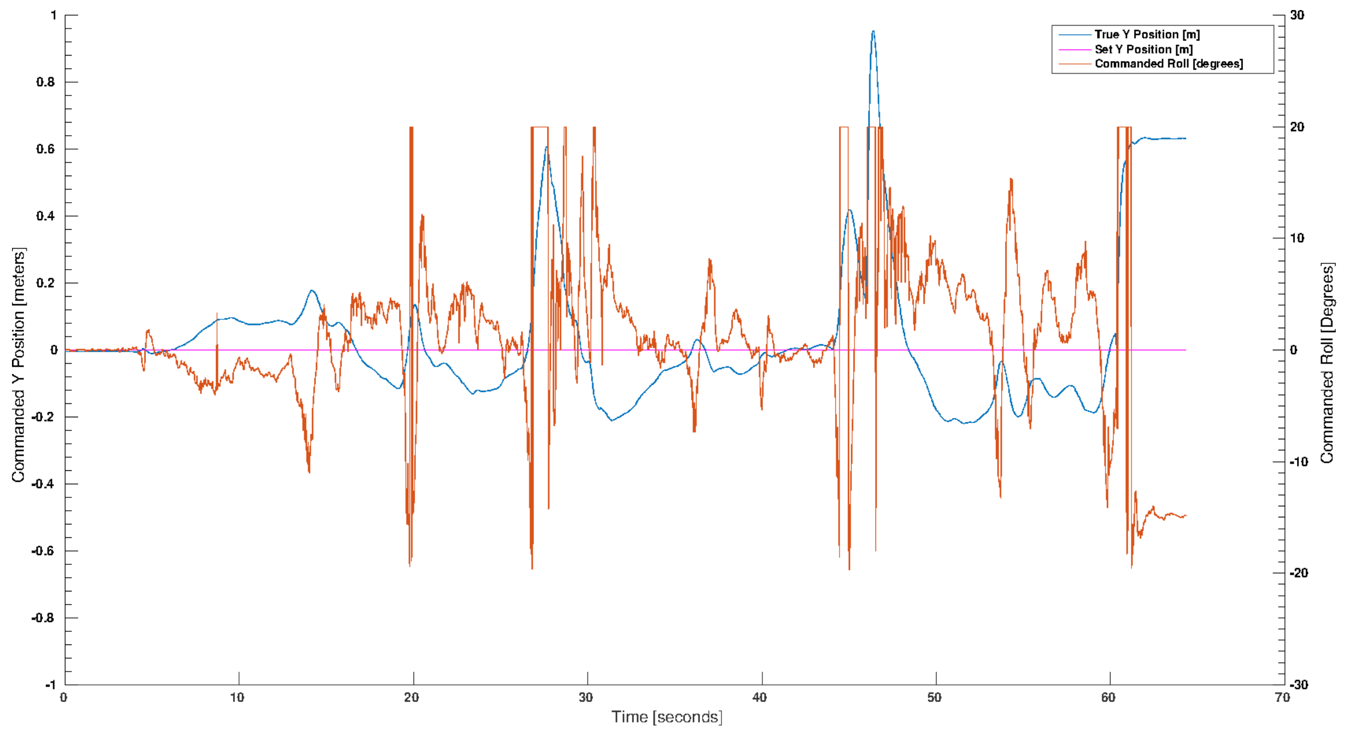


Figure 8: Position Controller - Roll Command

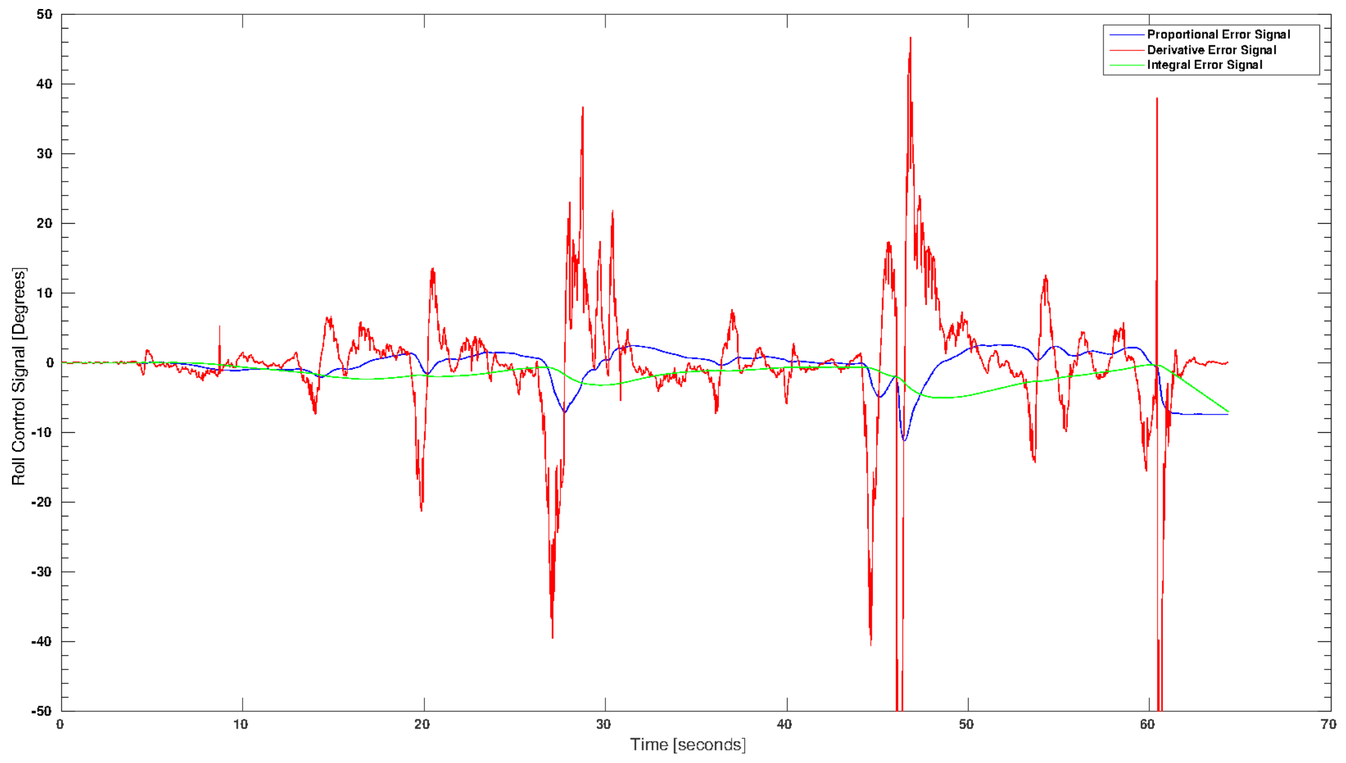


Figure 9: Position Controller - Roll Control Signal Components

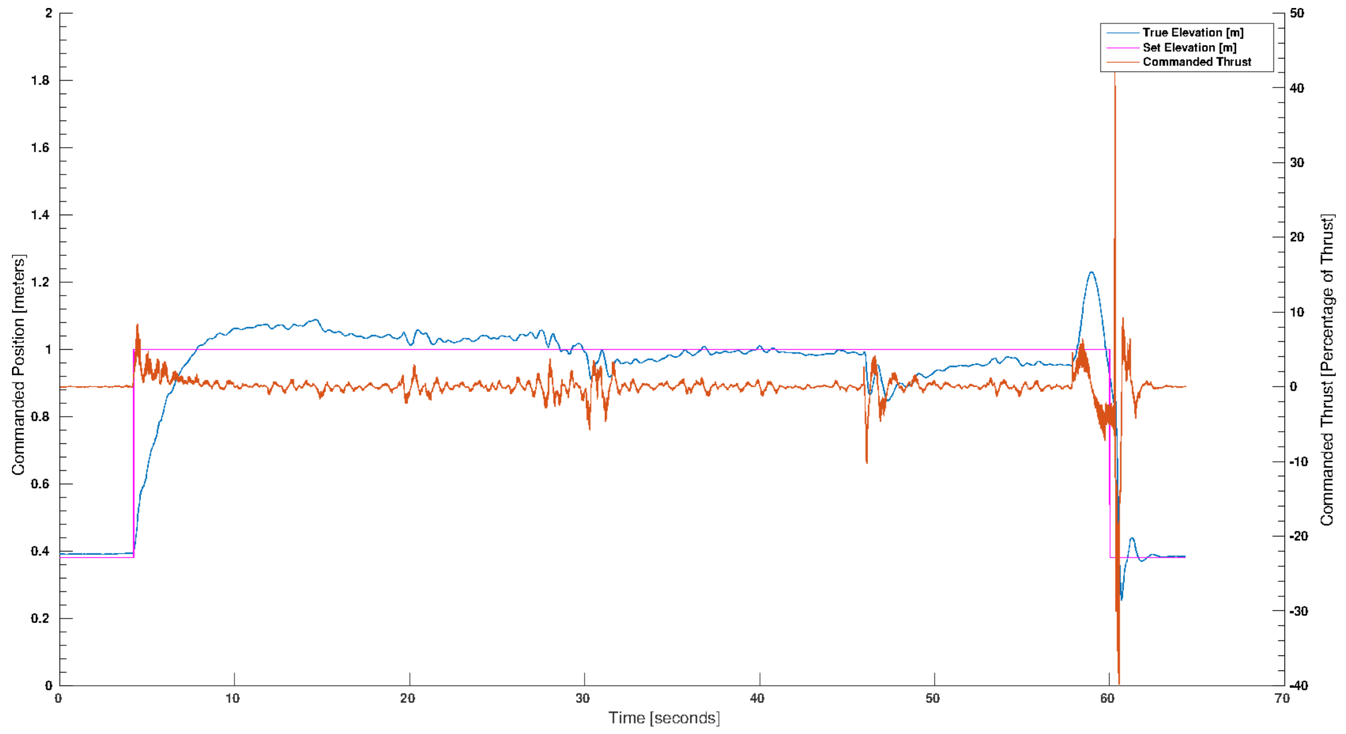


Figure 10: Position Controller - Thrust Command

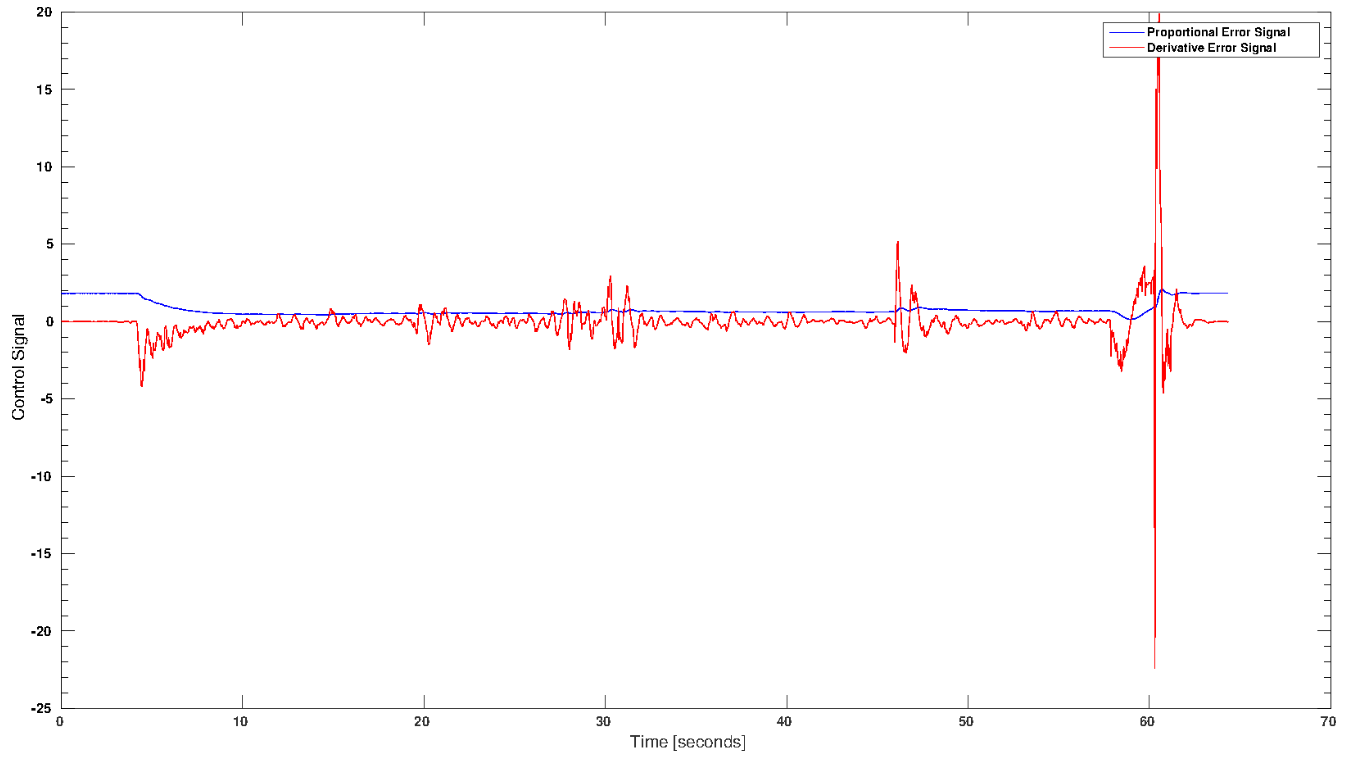


Figure 11: Position Controller - Thrust Signal Components