NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

VISION-BASED POSITION ESTIMATION UTILIZING AN EXTENDED KALMAN FILTER

by

Joseph B. Testa III

December 2016

Thesis Advisor: Vladimir Dobrokhodov
Co-Advisor: Brian Bingham

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704–0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2016	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE VISION-BASED POSITION ESTIMATION UTILIZING AN EXTENDED KALMAN FILTER	**5. FUNDING NUMBERS**
6. AUTHOR(S) Joseph B. Testa III	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB number ____N/A____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

The purpose of this project is to develop and integrate a prototype multicopter Unmanned Aerial Vehicle (UAV) with a vision-based algorithm to enable a relative position hold capability. The resulting solution will enable automatic operation of a UAV with respect to another visually detectable object without use of GPS receiver or when GPS signal is not available. Navigating a robot in a GPS-denied environment is a desired feature in many applications, including Maritime Interdiction Operations. While automatically maintaining its relative position with respect to a given target, the onboard system will also provide video coverage of "blind spots" and network relay between the boarding team and ship.

14. SUBJECT TERMS UAV, ROS, extended Kalman filter, Matlab, Simulink, position estimation	**15. NUMBER OF PAGES** 69
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540–01-280-5500 Standard Form 298 (Rev. 2–89)
Prescribed by ANSI Std. 239–18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

VISION-BASED POSITION ESTIMATION UTILIZING AN EXTENDED KALMAN FILTER

Joseph B Testa III
Lieutenant, United States Navy
B.S., Texas A&M University, 2010

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 2016

Approved by: Vladimir Dobrokhodov
Thesis Advisor

Brian Bingham
Co-Advisor

Garth V. Hobson
Chair, Department of Mechanical and Aerospace Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The purpose of this project is to develop and integrate a prototype multicopter Unmanned Aerial Vehicle (UAV) with a vision-based algorithm to enable a relative position hold capability. The resulting solution will enable automatic operation of a UAV with respect to another visually detectable object without use of GPS receiver or when GPS signal is not available. Navigating a robot in a GPS-denied environment is a desired feature in many applications, including Maritime Interdiction Operations. While automatically maintaining its relative position with respect to a given target, the onboard system will also provide video coverage of “blind spots” and network relay between the boarding team and ship.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

EKF	Extended Kalman Filter
KF	Kalman Filter
MIO	Maritime Interdiction Operation
ROS	Robot Operating Software
UAV	Unmanned Aerial Vehicle

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I dedicate this thesis to my children: Anthony, Sabrina, Monica, and Matthew. Their unquenchable curiosity and desire to learn is my inspiration.

I would like to thank my wife, Sarah, for her unwavering support as well as her patience while I “played with toys.” I would not be here without her support.

Thank you to my thesis advisor, Dr. Vlad Dobrokhodov, who gave me the opportunity to develop my own research topic. I will be forever grateful for his patience with explaining the Kalman filter. Also, I hope that he will remember our times troubleshooting MATLAB-UDP links with as much humor as I do. Thank you to my co-advisor, Brian Bingham, for his laptop, but more importantly for his ability to simplify complex concepts. I will miss our office chats.

Thank you to my brother, Salvatore, for his assistance with the coding. Also, thank you to the RoboDojo for providing a space to tinker and learn.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

In the Red Sea and off the Horn of Africa, warships patrol the seas looking for pirates and smugglers. If a ship is suspected to be conducting illicit activities, a warship can board the vessel, verify paperwork, and search for contraband. This is known as Maritime Interdiction Operations (MIO). During MIO, the warship provides cover for the boarding team; however, there are blind spots. The main blind spot is on the far side of the target vessel. The U.S. Navy uses helicopters to monitor the far side of the ship; however, there are over 50 warships that do not have the ability to deploy with helicopters. An Unmanned Aerial Vehicle (UAV) can provide similar support. A UAV mounted camera could provide valuable information regarding activities in a blind spot and relay audio and data between the war ship and boarding team. Finally, the cost of maintaining and operating a small UAV is orders of magnitude less expensive than a Navy helicopter.

One disadvantage that a UAV has compared to a manned helicopter is the UAV's reliance on outside sensors for determining position. Simple UAV are unable to determine their position. More complex ones can autonomously operate based on onboard GPS. There are many factors which can prevent the UAV from receiving GPS signal. Weather can prevent clear GPS signals. Inexpensive but effective GPS jammers are available for as little as $50 [1]. Even malfunctioning equipment can lead to a drone losing its position and thus aborting the mission.

A loss of a UAV's position could be dangerous to the drone as well as to others. Depending on the pre-programmed response, the drone could crash onto the target vessel. If the warship deploys multiple drones, a drone without its position could drift into another drone. To prevent such occurrences, the UAV should have an indigenous means of determining relative position. A UAV's camera could be used to augment a drone's positioning system as well as provide hold position capabilities if all other means of determining position are unavailable.

B. THESIS OVERVIEW

This thesis develops a vision-based tracking position estimation algorithm which enables a UAV to maintain station relative to a target using the UAV's camera and an onboard altimeter. We examine the scenario of a quadcopter UAV deployed off a Navy warship is inspecting a target vessel. During the inspection, the UAV loses GPS but must maintain a relative position to the target. The drone uses the algorithm to accomplish this task.

The exemplary scenario is divided onto a few phases to simplify the identification of required technologies. First, the drone processes the image and locates the target. For this experiment, the target was the "brightest point" in the camera frame; obtained either in normal daylight or illuminated with laser. The algorithm converted the image to an intensity map. Then, the algorithm locates the brightest point on the image and outputs that point's position in pixels of the camera frame. An experiment was developed to characterize the speed of the image acquisition and processing algorithm.

Next, the target position in pixels is converted into a usable measurement. The pixel position is converted into a distance from the camera center. An experiment was conducted to determine the correlation between the distance in pixels and the angle from the drone to the target. The drone was placed at different angles from the target. Since the longitudinal and vertical distances between the UAV and the target are known, the true angle can be calculated. For a range of angles, the pixel distance was recorded. Based on the results, a linear equation converts pixel distance to an angle to target thus establishing the calibration mapping.

A version of Kalman Filter was developed to minimize the impact of inaccuracies in the angle measurement as well as disturbances from the environment. The Kalman filter is a method of estimating values using an ideal model and noisy measurements. The filter uses the previous estimated position to predict its current position. Then, it estimates the current position using the prediction as well as the noisy measurement. This process is continuously applied. Since the angle measurement is nonlinear with respect to the states of UAV, an extended Kalman filter (EKF) is required. The EKF linearizes the

angular measurements when predicting position. The EKF's output is a noise-attenuated estimate of the drone's position and velocity relative to the target. The EKF was tuned using a stationary drone with respect to a simulated target.

The culminating experiment was a tuning of the EKF portion of the assembled algorithm. The result is an algorithm capable to locate a target, convert the raw data into a usable measurement and output a relative position to the target. The output position and velocity are compared to the actual position measured manually.

C. CONTRIBUTIONS

While the topic of UAV maneuvering based on vision-based tracking is not new, this thesis provides fundamental building blocks for continued research at Naval Postgraduate School and elsewhere. First, this thesis provides a systematic formulation and implementation of Simulink/ROS (Robot Operating Software) vision-based tracking using commercial off the shelf AR Drone quadcopter. Next, it quantifies tradeoffs between three different vision-based detection algorithms. Also, the thesis characterizes performance of vision-based tracking for feedback control. The final result is an experimental EKF detector which can be implemented in actual flight experiments.

THIS PAGE INTENTIONALLY LEFT BLANK

II. VISION-BASED DETECTION

The objective of the vision-based detector is to quickly and accurately locate a target within an image. The position of this target, in image-space, can then transformed into physical space and used as an input to the vehicle's position estimator. Stream of images is continually sent from the drone to the processing computer via Wi-Fi. This image is transmitted from onboard the drone to a remote computer, running MATLAB, for image processing. Three separate methods were tested to process the image stream in MATLAB:

- The first method compares the size of separate images to determine whether the drone moves closer to the target or farther away from the target.
- The second method locates a target in a raw image based on a given reference image.
- The third method finds the brightest spot on the image.

These methods were developed from examples on the MathWorks website. This third method was chosen because it is the fastest and most robust option. The image update rate for the third option is 10 Hz. Finally, an experiment was conducted to evaluate how the speed of the target affected the image detection, which determines maximum bandwidth for the target tracking loop. The test determined how fast the tracker could "follow" a chosen object in the camera frame. The UAV was able to track a target that with an angular velocity of 3 rad/sec.

A. PREVIOUS WORK

While vision-based tracking for range estimation has been studied and the individual challenges within this thesis have already been solved, the thesis undertakes a unique combination of these problems. Vision-based position estimation was demonstrated by Dobrokhodov for a fixed-wing UAV; however, operator input was required to select the target [2]. Yakimenko calculated and exhibited an algorithm for determining the position to a landing area using multiple targets with known relative position to the landing zone as well as each other [3]. Yakimenko's scenario differs from

the thesis scenario; because, that scenario requires multiple targets. This scenario also involved a fixed-wing UAV. Recently, Ramos compared multiple image tracking algorithms integrated with the Parrot A.R. Done, the same drone used in this thesis [4]. Ramos's research demonstrates the capabilities of more complex image identification algorithms; however, it does not take the next step of predicting the targets location.

B. REVIEW OF OPTIONS

Three different options were considered for the image detection algorithm. First, size differentiation compares an object's size in an image with its size in a previous image to determine direction of travel. Second, object comparison locates a target using a reference image. Third, intensity comparison converts the image into an intensity map and finds the brightest point in the image. After evaluating each algorithm as a foundation for distance estimation, intensity comparison was selected since it was fastest and most robust.

1. Size Differentiation

Size differentiation compares target sizes to the one defined in a reference image. The system is based on a pinhole camera model. The theory is from Dodd's notes on pinhole optics [5]. The pinhole camera is a closed box with a small hole in one end as seen in Figure 1.

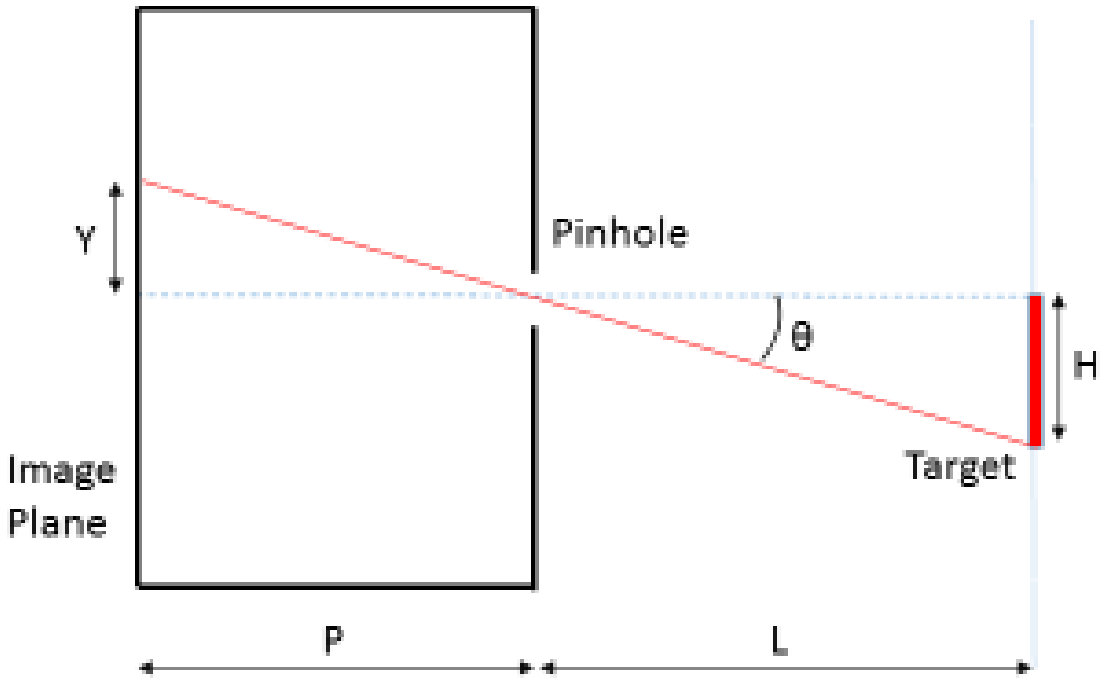


Figure 1. Theoretical Pinhole Camera

L is the distance from the target to the pinhole, P is the distance between the pinhole and the image plane, H is the height of the target, and Y is the height of the target's shadow on the image plane. P is also known as the focal length. Equation 1 relates these values.

$$\frac{Y}{P} = \tan(\theta) = \frac{H}{L} \tag{1}$$

The H and P are assumed to be constant. The size of the target on the shadow is inversely related to the distance L. If the distance between the target and the pinhole is decreased, the shadow on the image plane increases as seen in Figure 2.

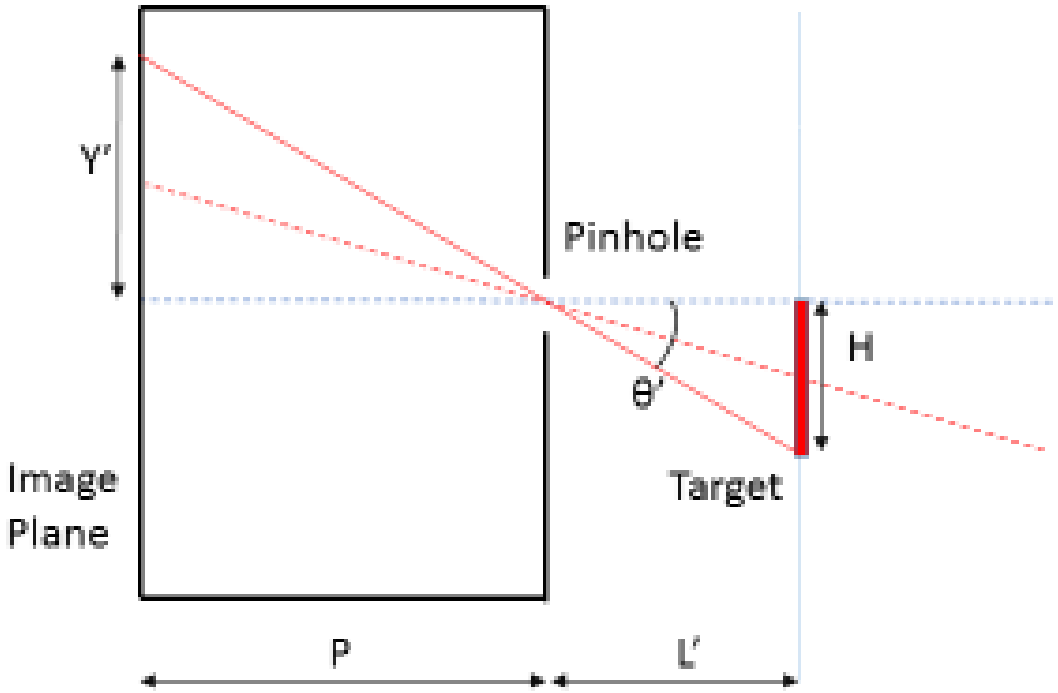


Figure 2. Pinhole Camera with Closer Target

L' is the new distance between the target and the pinhole, and θ' is the new angle. Y' is the new height to the target. Conversely, Y will decrease as L increases. The pinhole camera model can be applied to this problem as well as the pixel to angle calibration.

This method compares the size of a target in subsequent images. Size comparison determines only the dynamic of range from target; often the range itself does not matter as long the "reference" distance can be placed on hold. First, the camera captures an image of the target and a plain background. For this experiment, the target was a rectangle. Next, the algorithm grayscales the image, a conversion used in many image detection algorithms [6]. Grayscaling reduces the computational load as well as simplifies the algorithm [6]. Grayscaling is also necessary for using the edge detection function. Next, the algorithm detects the edges of the target that enables comparison of targets with a reference image. This method was based on the MathWorks example for edge detection [7].

After identifying the target, the target's size must be calculated. The algorithm determines the size by finding the maximum and minimum values of the bounds of the enclosing rectangle for both the X and Y dimensions as seen in Figure 3.

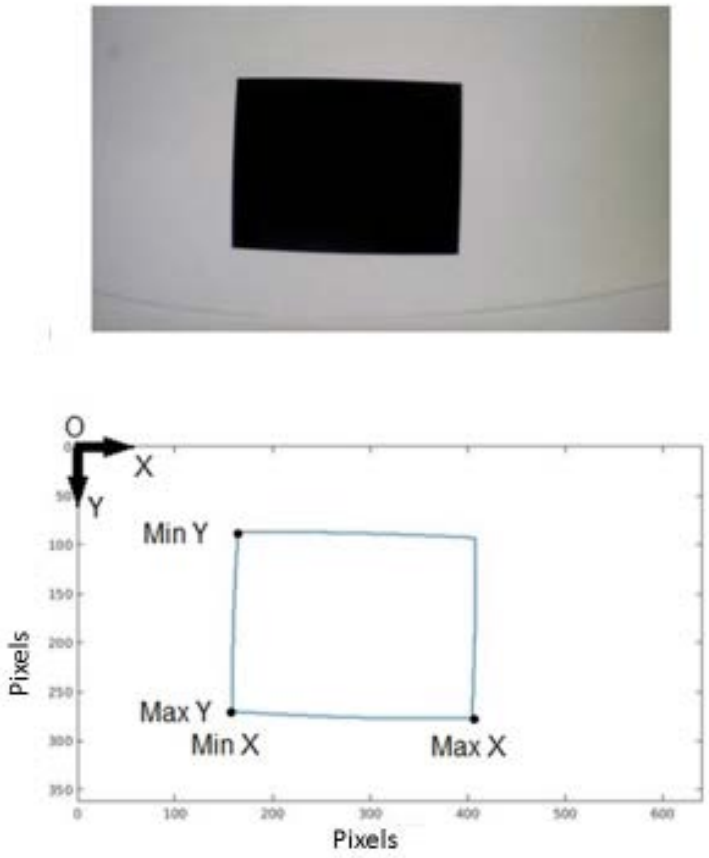
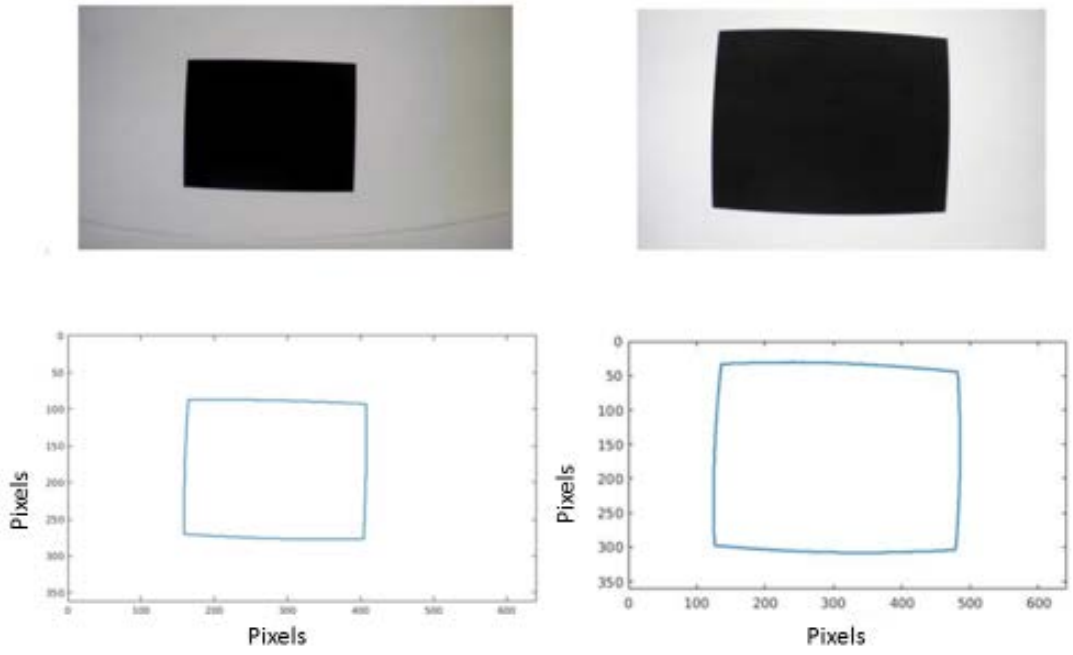


Figure 3. Maximum and Minimum Values of Edge Detection

Taking the difference of the maximum and minimum values for each dimension results in rough dimensions of the target in pixels. Multiplying these differences results in the target area, or its size. By comparing a target's size in a subsequent image, the drone can determine whether it is moving closer to the target or farther away from the target. Due to the distortion from the camera, the rectangular target appears rounded. While this affects the resultant calculations, the distortion affects the images equally and will not compromise the goal of the experiment. To equalize the affects, the target was placed in the center of the camera's field of view prior to capturing an image.

An experiment proves that size differentiation algorithm is feasible utilizing the pinhole camera model. The goal of the experiment is to use two images to determine the motion of the drone. First, the drone is placed in front of a black rectangular target with a solid white background. The white background ensures that there is an easily detectable contrast with the target as well as a lack of clutter for the algorithm to detect. At the controller's command, the drone captures a "Reference Image" of the target and calculates the target's size using the aforementioned algorithm. Next, the drone is placed at a different distance from the target while keeping the target in the camera's field of view. At the controller's command, the drone captures a "Variable Image" which is processed by the algorithm. Figure 4 shows an example run of the experiment.



The top frames are the captured images of the target. The upper left frame is the reference image while the upper right frame is the variable image. The bottom frames are the resultant edge detections from the captured images.

Figure 4. Size Differentiation of Reference Image and Variable Image

The top images are the captured images while the bottom images are the resultant edge detections. The algorithm then compares the sizes of the two images and thus estimates the direction of the UAV's travel. If the images are the same size within a certain

tolerance, the algorithm reports that the UAV is in the same position. If the Reference Image's size is larger than the Variable Image's size, than the algorithm reports that the UAV had been moved farther from the target. If the Variable Image's size is larger, than the algorithm reports that the UAV has been moved closer to the target. In the case of Figure 4, the UAV was moved closer to the target, resulting in a larger Variable image.

While feasible, this approach is not the best option. In order for the algorithm to work, the camera's field of view must be devoid of anything other than the target or an additional target search routine applied before the size calculation. While the MathWorks example filters everything except the target, it was unable filter out even a trivial amount of clutter in the background [7]. Additionally, more time would be required to find a correlation between the target's size in the image and the distance from the object. Overcoming these two issues would result in the size determination method becoming a feasible option.

2. Image Comparison

Image comparison locates an object within an image using a reference image, template. The algorithm uses metric values such as intensity to compare the template to overlapping parts of the image and locates the best match [8]. Similarly to the size determination method, the algorithm captures an image with a target from the camera and grayscaled. In Figure 5, the black circle is the target.

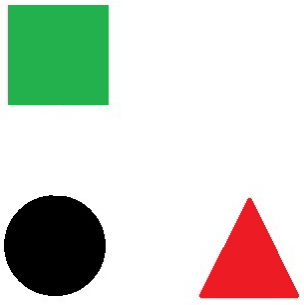


Figure 5. Example Target with Extraneous Images

In later iterations of the algorithm, the captured image is resized in order to improve processing speed. Next, the algorithm locates the target using a reference image; see Figure 6.

Figure 6. Reference Image

This location method involves comparing groups of pixels to find the closest match. A detailed description of the method can be found on the MathWorks website [9]. After reviewing the entire image, the algorithm returns the location of the center of the group of pixels which most closely resembles the reference image; see Figure 7.

Figure 7. Image Comparison

While this method was effective, it took too much computational time to find the target. Without resizing of a single frame, the algorithm took almost six seconds to compare all the different pixel groups and locate the target. As seen in Figure 8, image resizing can decrease the time taken to locate the target; however, this is not nearly fast enough for a position estimator. Additionally, resizing the image would decrease the detail of the image and make the object detector less accurate.

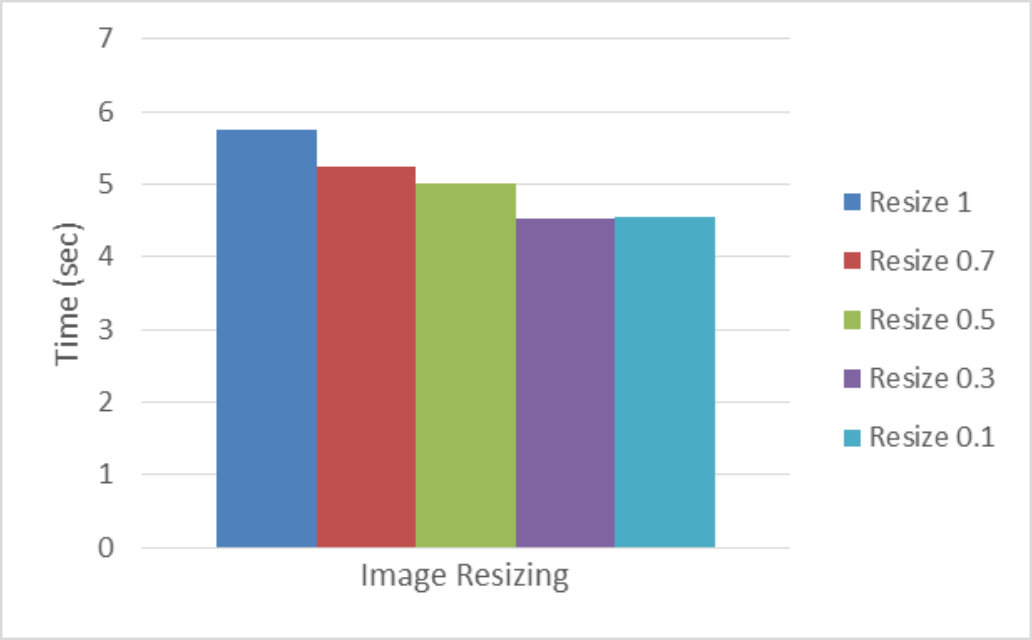


Figure 8. Algorithm Processing Times for Various Image Resizing

While this method of image comparison was too slow, a faster method of object comparison would be better for operational systems than the intensity comparison, the third option.

3. Intensity Comparison

Intensity comparison converts an image to an intensity map and locates the most intense point of the image. Images from the camera are made up of pixels. For a color image, each pixel has a value for red, blue, and green. An intensity map weights these values for each pixel and adds them together using equation (2) [10]:

$$Intensity = 0.299(red) + 0.587(green) + 0.114(blue) \quad (2)$$

These weighting values are the generally accepted values for calculating intensity [11]. The intensity comparison algorithm outputs the pixel position that has the highest intensity value. To prevent false detection, the intensity comparison has a minimum threshold. If the threshold is not met, the algorithm outputs the pixel position (0,0). The algorithm overlays this pixel position over the captured image as seen in Figure 9.

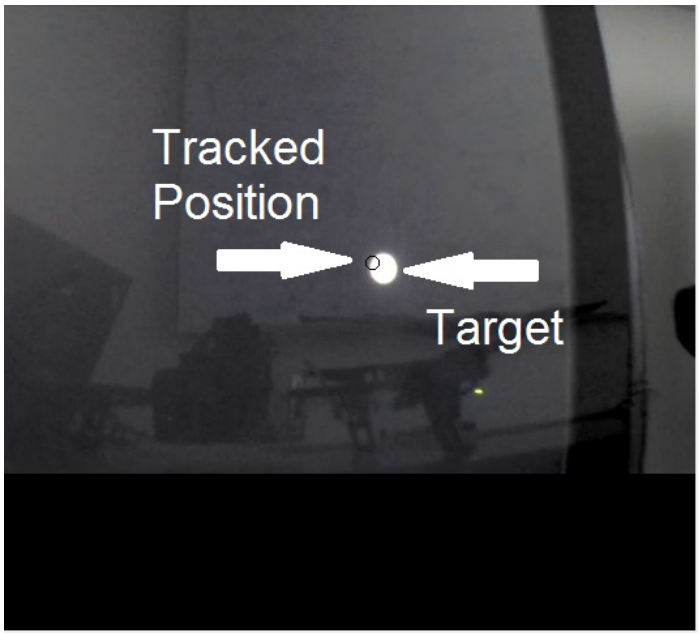


Figure 9. Intensity Comparison Experiment

Intensity comparison targets which were used in this thesis include a laser pointer, an electric candle, and white dot on a computer screen as seen in Figure 9.

Experimentation showed the intensity comparison was the best option of the three considered. The size differentiation algorithm requires a background devoid of anything other than a solid color. In contrast, the intensity comparison tracks a target within different environments as long as the target is the brightest object in the field of view. The image comparison algorithm updates every five seconds. The intensity comparison locates the target locates the target 50 times faster. While more work could be done to improve these methods, this is outside the scope of this thesis. The intensity comparison is the best option for this thesis.

C. EXPERIMENTAL ACQUISITION CHARACTERIZATION

The maximum target speed at which the algorithm tracks the target was determined experimentally. First, the camera is placed in front of a target computer with a rotating target as seen in Figure 10.

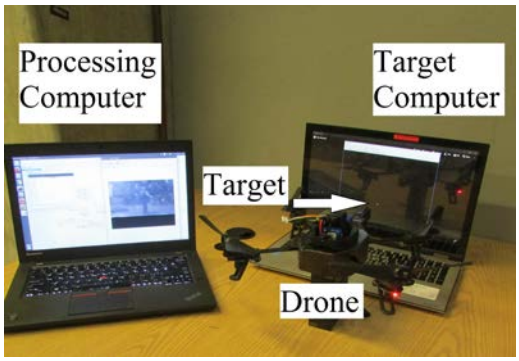


Figure 10. Experimental Setup to Characterize Image Target Acquisition

This target is rotating at a constant angular velocity; the algorithm has been developed and run in Simulink in soft real time. The UAV relays the image back to the image detection algorithm, which is running on the second processing computer also running MATLAB/Simulink. The image detection algorithm identifies the target and outputs the measured angular target position. Simultaneously, the target computer transmits the true angular target position to the processing computer. Figure 11 shows the architecture of the experiment.

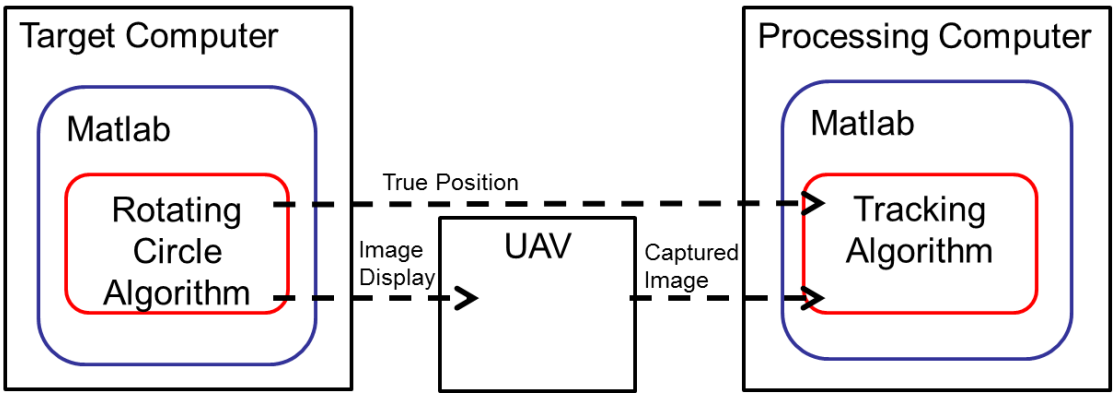


Figure 11. Experimental Architecture

The image detection algorithm calculates the difference between the true angular position and the measured angular position. This difference is the angular offset. The angular offset for each angular velocity was averaged. Negative offset occurs when the measured angular position lags behind the true angular position. While positive offset is physically

impossible, the positive spikes are caused by periodic lag between the target computer and the processing computer. This experiment was conducted for a range of angular velocities.

Based on the data, the camera's ability to track a target begins to degrade at three radians per second. Figure 12 shows the collected data as well as a curve fit of the data.

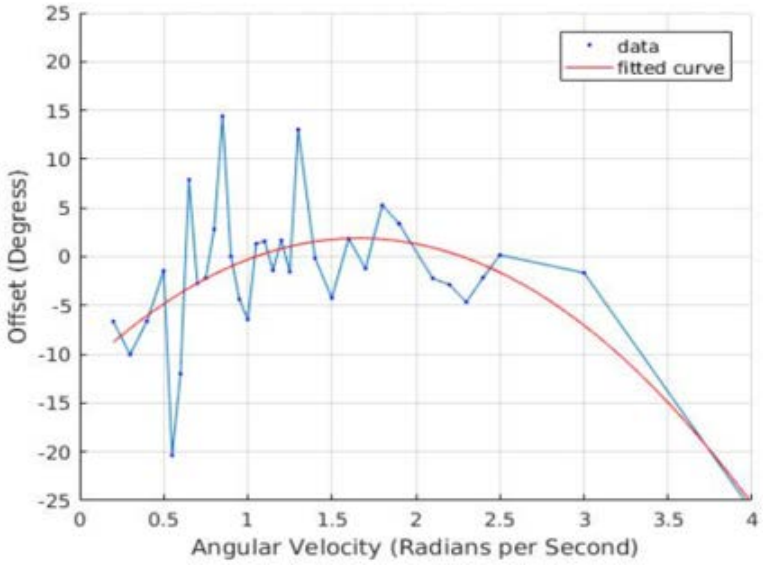


Figure 12. Average Offset for Various Angular Velocities

Between 0.2 and 3.0 rad/sec, the tracker stays within ten degrees of the target. At velocities greater than 3.0 rad/sec, the negative offset rapidly increases. Inconsistencies such as positive offsets imply that more work is needed to improve the experiment.

While not used in this thesis, this experiment can be useful for future work. This experiment determines the feasible rate of update for the target tracker. This threshold is the limit of the system's ability to track a visibly detectable target. According to the Nyquist Sampling Theorem, the sampling frequency must be at least twice the signal frequency [12]. When this estimator is implemented into a feedback controller, the responsiveness of the controller will be limited by the rate of feedback update.

III. CAMERA CALIBRATION

For this application, camera calibration estimates the relationship between the pixel location, in 2D image coordinates, and the angle between the camera axis and the line between the camera and the target. This angle is one of two measurements needed to calculate the longitudinal distance to the target as discussed in Section I.B.

The camera calibration is a process of obtaining an analytical function connecting raw pixel measurements of the camera with the corresponding angular values and is done experimentally. The calibration process includes the camera placed in a fixed position relative to a target. Since the vertical and longitudinal distance to the target is known, the true angle is known. Using the pinhole camera model, the true angle is compared to the target's vertical distance to the camera center in the image as seen in Figure 13.

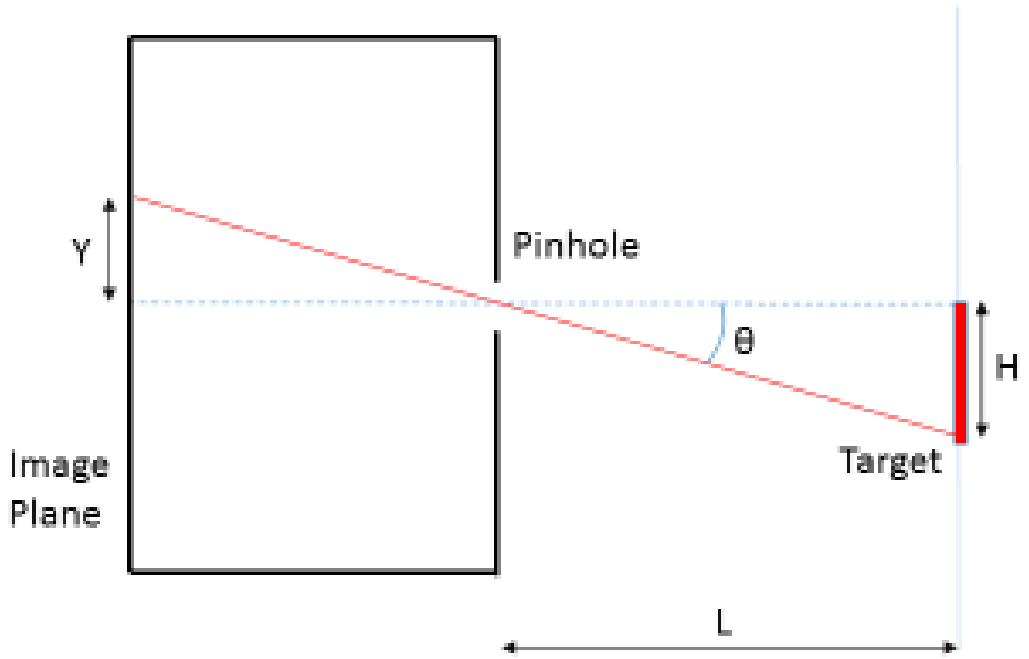


Figure 13. Pixel Size, Y, to Angle, θ, for Pinhole Camera Model

This experiment is done for a range of angles and at two separate longitudinal distances to the target. The result of the experiment is a linear equation.

A. ALGORITHM

After the image tracking algorithm locates the target and outputs the target's location in the image, this data in pixels must be converted into a useable angular measurement in radians. Recall that the approach is possible since the intrinsic parameters of the camera like focal length, size of the CCD chip, lens are known and constant. The coordinates of the target (y-value of the target's position) is referenced to the center of the image as seen in Figure 14 and Figure 15.

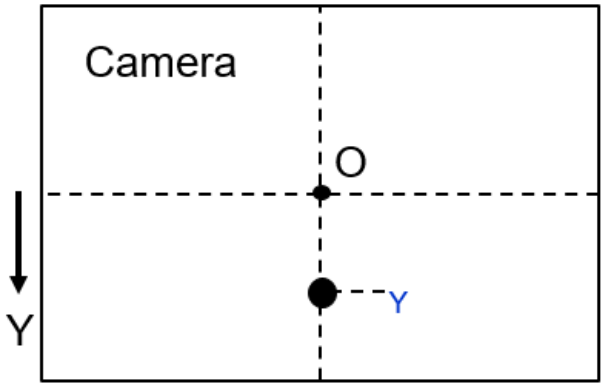


Figure 14. Target Position in Camera

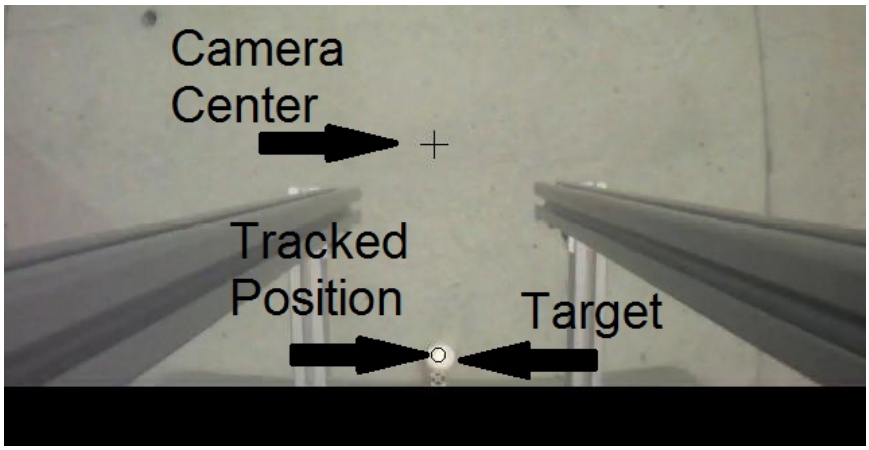


Figure 15. Captured Image from Drone during Experiment

The distance between the center of the image and the target position is the pixel distance. For this work, all targets were held near the y-axis of the image center, so the x-distance was ignored. The approach is reasonable as the camera is fixed in the body of UAV and for targeting the UAV would have to rotate to point to the target. This target in the image correlates to a physical target that has a relative position to the drone as seen in Figure 16.

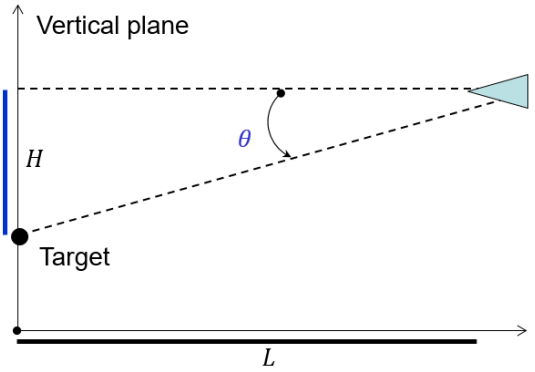


Figure 16. Target Location Relative to Drone

The angle between the drone and the target can be related to the vertical and longitudinal distance to the target using equation (3).

$$\theta = \tan^{-1}\frac{H}{L} \tag{3}$$

By comparing this angle to the pixel distance for varying angles, an analytical calibration equation can be derived. This conversion can then be used for further calculations.

B. CAMERA CALIBRATION EXPERIMENT

The goal of the experiment was to develop an equation which converts pixel measurements to angle. To do this, the drone was placed at different angles from a target as seen in Figure 17.

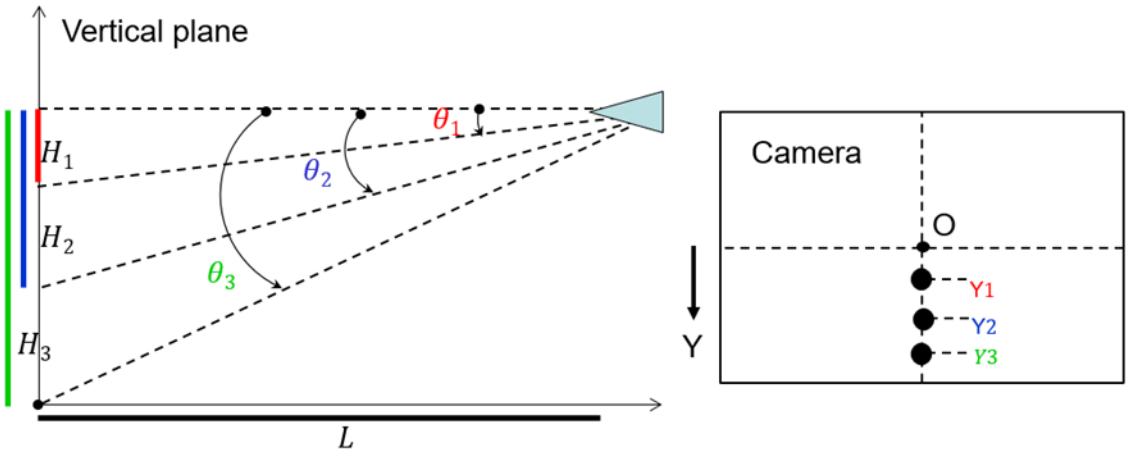


Figure 17. Target Position and Camera Image for Various Heights of Target

Since the vertical and longitudinal distances were known, the true angle from the drone to the target was also known. These values were compared to the position of the target in the image. Running the system resulted in the pixel distance. By repeating this experiment at different angles, the relationship between pixel distance and angle was characterized.

The experimental setup utilized a stationary drone with rigidly attached camera, a wall mounted target, and a laser pointer. The drone was positioned facing a wall with the camera one meter from the wall. The target was a piece of paper with crosses spaced two centimeters apart down the length of the paper. The top target was placed exactly in the center of the camera's field of view. Figure 18 shows the complete experimental setup.

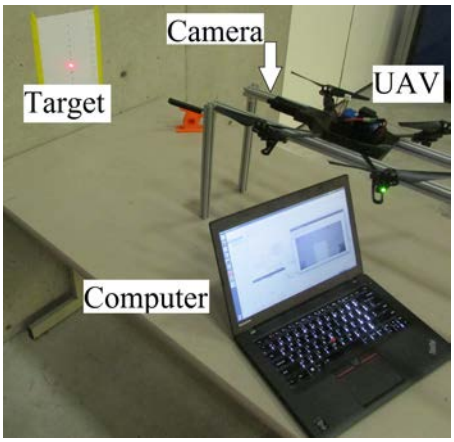


Figure 18. Angle Calibration Experiment Setup

Runs were conducted starting at the top target and proceeding down. For each run, the laser pointer illuminated at the center of a target. During a 30 sec run, MATLAB stored the target's position every 0.1 sec. After run completion, the subsequent target was illuminated and the experiment repeated. Once the target left the camera's field of view, the drone was moved to 2 m from the wall and the experiment repeated. After all the runs were completed, a MATLAB script filtered out occlusions for each run and determined the pixel distance from the camera. These occlusions occurred when the camera momentarily loses track of the target, because the intensity of the light in the image did not break the minimum threshold of the intensity detector. To prevent occlusions, the intensity detector's minimum threshold was reduced which fixed the problem. Figure 19 shows the results as well as the linear fit.

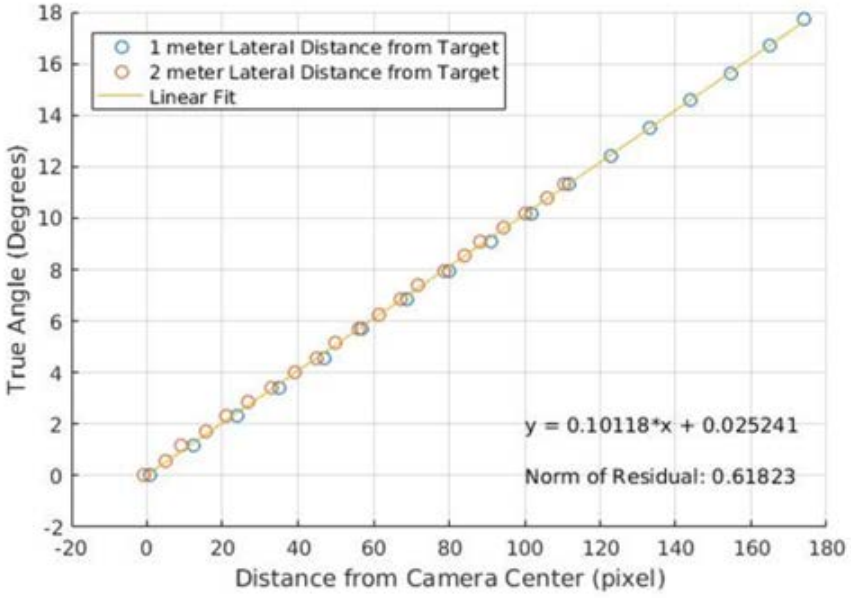


Figure 19. True Angle for Various Pixel Distances from Image Center

This linear fit estimates the relationship between the pixel distance and the angle between the drone and the target. Equation (4) is the resultant linear relationship between the pixel distance, Y, and the angle, θ:

$$\theta = 0.10118 * \mathrm{Y} + 0.025241 \tag{4}$$

The norm of the residual is 0.61623 degrees. This equation is then implemented into the Simulink model.

C. IMPLEMENTATION OF CALIBRATION RESULTS INTO SIMULINK ALGORITHM

The Camera Calibration block continually converts the pixel distance, Y, to angle, θ, utilizing the experimentally obtained linear fit equation. Figure 20 shows implementation of equation (4).

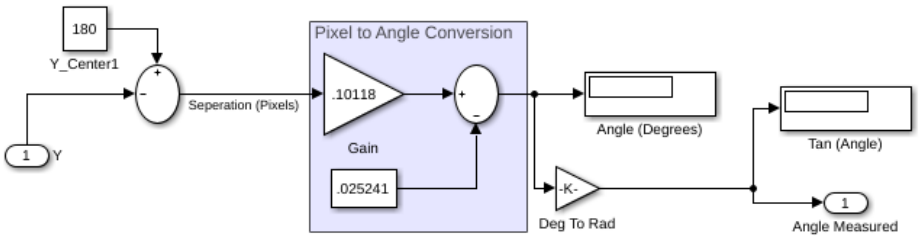


Figure 20. Pixel to Angle Linear Equation Implemented into Simulink

Once the pixel distance is converted to degrees, the angle is then converted to radians prior to inputting it into the Estimation Algorithm. This angle, as well as the altitude, is used to estimate the longitudinal distance from the target.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ESTIMATION ALGORITHM

The purpose of the estimation algorithm is to produce an estimate based on imperfect sensor measurements and a model of the system. I selected an extended Kalman filter (EKF). The EKF was built in iterative steps. First, a linear Kalman filter (KF) with one linear measurement was built which measures longitudinal distance and outputs estimated longitudinal distance as well as relative speed. Next, this system was expanded into a linear KF that measures longitudinal distance as well as vertical distance. This KF outputs estimated position and relative speed in both the longitudinal and vertical directions. Then, the linear KF was converted into an EKF that measured vertical distance and angle to target. The result was an EKF which outputs an estimated longitudinal position and relative speed in both the longitudinal and vertical directions.

A. PREVIOUS WORK

Since Rudolph Kalman introduced it in 1960, the Kalman filter has become one of the premier methods for obtaining "a recursive solution to the discrete-data linear filtering problem" [13]. The Kalman filter has been extensively utilized in autonomous navigation and human motion tracking as well as virtual reality and augmented reality [14]. One restriction on the Kalman Filter is that the relationships between the measurements and state space are linear. In cases where a relationship is nonlinear, an Extended Kalman Filter (EKF) can be applied. One EKF example is a bearing and range tracker, in which imperfect bearing and range measurements are used to estimate longitudinal vertical distance to the target [15]. For this application, an EKF is required because the relationships between angle and distances are nonlinear. This implementation of the KF, along with the subsequent EKF, are based the process and notation used by Welsh and Bishop [13].

B. KALMAN FILTER

The linear Kalman filter is an algorithm which recursively estimates a vector of states of an ideal model by means of recursive "blending" of linear measurement. Equation (5) is the equation of the true values of the system state which are unknown.

$$x_t = Ax_{t-1} + Bu_{t-1} + \omega_{t-1} \tag{5}$$

x is the true state, **A** is the difference matrix, **B** is the control matrix, u is a control input, and ω is the unknown process noise which are discussed later in this chapter. The future state is predicted utilizing the previously estimated state, the ideal system, and optional control inputs while assuming no noise entering the system. Equation (6) is the calculation of predicted state at time, t.

$$x_{pred,t} = Ax_{\text{est},t-1} + Bu_{t-1} \tag{6}$$

$x_{pred,t}$ is the predicted value and $x_{\text{est},t-1}$ is the estimated state from the previous step. A measurement is also taken to compare with the model as seen in equation (7).

$$z = \mathrm{H}\,x + v \tag{7}$$

z is the measurement, H is the relationship between the measurement and the state, and v is the unknown measurement noise. Like ω, v is an additive Gaussian noise. The algorithm blends this prediction with the imperfect measurement and outputs an estimated value. The blending is weighted by a gain that depends on the tunable values of the process model and the measurement noise covariances **Q** and **R**, which will be introduced later in this chapter. The estimated value is then used as the starting value to predict the next future value. This cycle of prediction and estimation can be broken up into one initiating step and four cycling steps.

1. Define Constant Values and Initial Inputs

Certain values do not change during the execution of the Kalman Filter.

- **A**—the process model matrix which relates the state at the previous time step, t-1, to the current time step, t.
- **B—**the control matrix that relates an optional control input, u_{t-1}, to the predicted state value.
- **H**—a matrix which relates the measurement vector, z, to the state vector, x_{est}.
- **I**—the identity matrix.

- $x_{est,0}$—an initial estimate of the states.
- $P_{est,0}$—the initial estimation error matrix.
- **Q**—the covariance matrix for the state model.
- **R**—the covariance matrix for the measurement model.

Q and **R** are weighting factors, and changing their values relative to each other affects the system's response. Increasing the **Q** value while holding the **R** value constant indicates that the ideal mode is less reliable. This results in the estimated state more closely tracking the measurements, including disturbances in these measurements. Conversely, decreasing the **Q** value while holding the **R** value constant indicates that the ideal mode is more reliable. This results in the estimated state more closely tracking the ideal model and slower responses to changes to the system. This also means that the estimated state will increase damping to disturbances. During experimentation, values of **Q** and **R** are adjusted to reach an optimized balance between responsiveness and disturbance rejection. To simplify the system, **R** was held constant for the final experiment; and **Q** was varied.

Additionally, **Q** was calculated two separate ways. First, **Q** was calculated using equation, (8).

$$Q = WqW^T \tag{8}$$

q is a scalar value. This equation describes a system in which the values are interdependent. For the KF with two inputs, the individual values of Q were adjusted. For the EKF and the final experiment, equation (8) was replaced with equation (9).

$$Q = q * \text{eye(n)} \tag{9}$$

eye(n) is an n by n identity matrix.

2. Predict the Future Value Using the Ideal System

The future state is predicted utilizing the previously estimated state, the ideal system, and optional control inputs while assuming no noise entering the system. Equation (10) is the calculation of predicted state at time, t.

$$x_{pred,t} = Ax_{\text{est},t-1} + Bu_{t-1} \tag{10}$$

$x_{pred,t}$ is the predicted value, A is the difference matrix, B is the control matrix, u_{t-1} is an optional control input, and $x_{\text{est},t-1}$ is the estimated state from the previous step. Control inputs include any supplemental information that will result in a more accurate predicted state. The B matrix transforms those control inputs to the state space. For this thesis, the control matrix and control inputs were not used.

3. Calculate "A Priori" Error Covariance Matrix and Kalman Gain

After calculating the predicted state, the A Priori Error Covariance Matrix, or prediction error matrix, is calculated using equation (11).

$$P_{pred,t} = AP_{\text{est},t-1}A^T + Q \tag{11}$$

$P_{pred,t}$ is the prediction error matrix and **Q** is the process noise covariance. $P_{\text{est},t-1}$ is the estimation error matrix from the previous time step. For the initial estimate, the estimation error matrix was defined by equation (12).

$$P_{est,0} = diag(Q) \tag{12}$$

Next, the Kalman gain, K, is calculated using equation (13).

$$K_t = \frac{P_{pred,t}H^T}{HP_{pred,t}H^T + R} \tag{13}$$

The Kalman gain is the weighting factor for the estimated state. As **Q** increases, K increases and the measurement is weighted more heavily. As **Q** decreases, K decreases and the ideal model is weighted more heavily.

4. Estimate Current Value Using Blend of Prediction and Measurement

The estimated value, $x_{\text{est,t}}$, is calculated using equation (14).

$$x_{est,t} = x_{\text{pred},t} + K_t(z_t - z_{\text{pred},t}) \tag{14}$$

z_t is the measurement vector and $z_{pred,t}$ is the prediction vector. For a linear system, the prediction vector is linearly related to the predicted state. Additionally, $(z_t - z_{\text{pred},t})$ is the difference between the measured values and the values based on the ideal system. This

difference is known as the residual and is the contribution of the measurement. As **Q** increases, **K** increases and the residual is weighted more. As **Q** decreases, **K** decreases and the residual is weighted less.

5. Calculate "A Posteriori" Error Covariance Matrix

After calculating the estimated state, the A Posteriori Error Covariance Matrix, or estimation error matrix, is calculated using equation (15).

$$P_{\text{est},t} = (\text{I} - \text{K}_t H) P_{\text{pred},t} \tag{15}$$

As the system cycles, the values within the $P_{\text{est},t}$ matrix should converge to a steady state value. Next, the estimated value and estimated error matrix are used to predict the future value using the ideal system, continuing the cycle.

C. EXTENDED KALMAN FILTER

The EKF accomplishes the same task with additional capability of accounting for non-linear relationships between the state and the measurements. For the EKF in this thesis, the estimation is linearized using the partial derivative of the measurement with respect to linear states. The prediction vector is a function of predicted states as seen in equation (16).

$$z_{pred,t} = f(x_{pred,t}) = \text{H}\, x_{pred,t} \tag{16}$$

For an EKF, the **H** matrix becomes the Jacobian and is modified to account for this nonlinear relationship as seen below.

D. IMPLEMENTATION AND RESULTS

The EKF was incrementally built to estimate the relative position of a target to a drone. First, a KF with one input was built in MATLAB. This KF was adjusted to receive two inputs. Next, the KF was modified into an EKF. This MATLAB EKF script was converted into a function which was tested in MATLAB. Finally, the EKF function was embedded into a Simulink block. This block completed the vision-based distance estimation algorithm.

The setup for each iteration was the same. A true longitudinal and vertical distance between the drone and the target was established. For the EKF, the true angle was calculated using the true distances. At the development and testing phase, the measured values were calculated by adding random values to the true values to represent noise. Next, constants and initial inputs were defined. Then, the MATLAB script runs. While the script runs, the estimated values, Kalman Gain, and error matrices are recorded. After the EKF completes its last iteration, the estimated, measured, and true values are compared and analyzed.

1. KF with One Input

The first step of the EKF's development was to develop a KF with one measurement. The measurement was the longitudinal distance with random noise added to it. Equation (17) is the ideal model matrix for a one dimensional KF.

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{17}$$

The state consists of the longitudinal distance, x, as well as the longitudinal velocity, $\dot{x}$, as seen in equation (18).

$$x = \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \tag{18}$$

The KF was implemented in MATLAB. Multiple iterations were run with varying **Q** and **R** values. The effects of adjusting the **Q** and **R** values on the time to convergence and disturbance rejection were exactly as described in [13]. The resulting algorithm output the true longitudinal distance, measured longitudinal distance, and estimated longitudinal distance as seen in Figure 21.

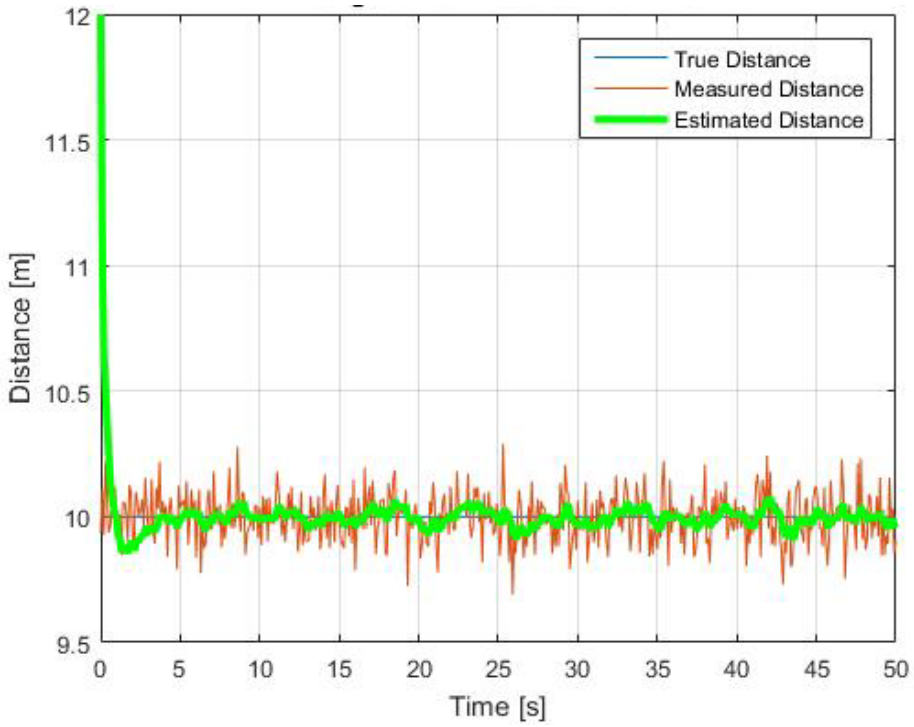


Figure 21. Longitudinal Distance for a KF with One Input

The estimated distance converged to the true distance in less than five seconds, while also minimizing the disturbance from the random noise. Equation (19) is the selected value of **Q**.

$$Q = \begin{bmatrix} 0.01 & 0.01 \\ 0.01 & 0.01 \end{bmatrix} \tag{19}$$

R is the 1-by-1 identity matrix.

The Kalman gain and estimation error matrix are indications of the data's validity. Figure 22 is a plot of the Kalman gain as a function of time.

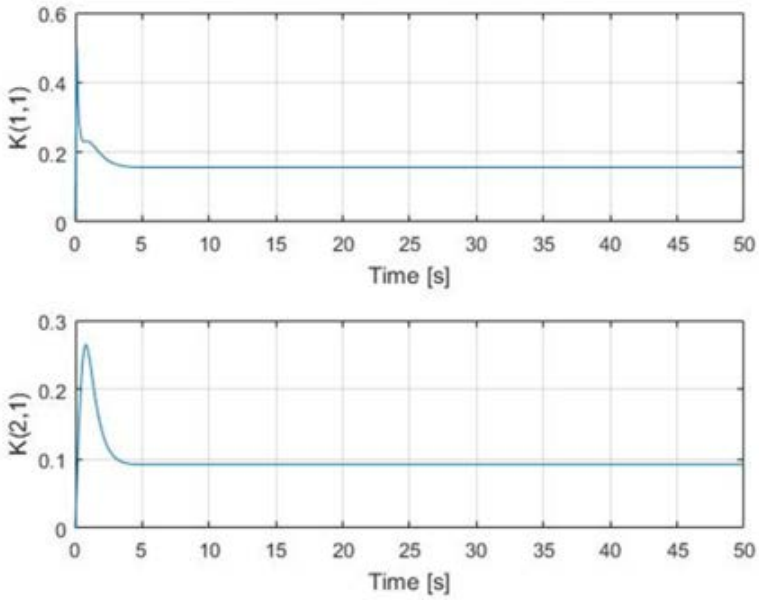


Figure 22. Kalman Gain for a KF with One Input

For a static system, the Kalman gain is expected to converge to a minimum steady state value. Due to the requirement to calculate a matrix inverse when calculating the Kalman gain, this calculation can be skipped once it reaches steady state. Similarly, the estimation error matrix converges in a similar manner as seen in Figure 23.

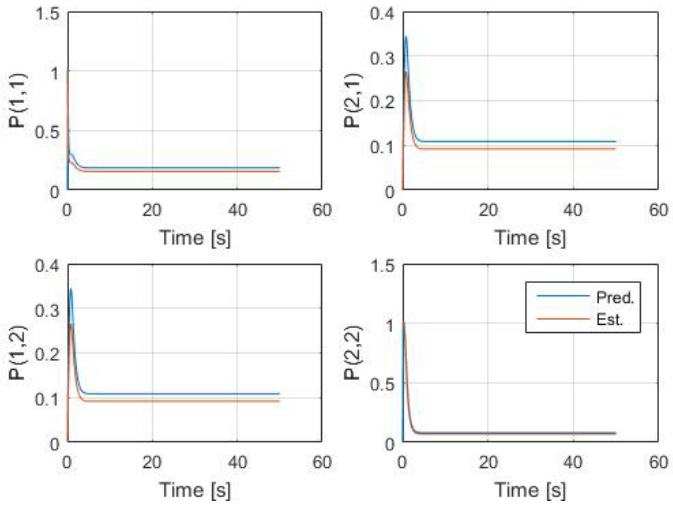


Figure 23. Estimation Error Matrix for a KF with One Input

The individual values should converge to a minimum steady state value. The Kalman gain and estimation error matrix plots for the subsequent KFs and EKFs were consistently similar in shape.

2. KF with Two Inputs

The KF was expanded to receive two measurements, the longitudinal and vertical distances. The A matrix became a 4x4 matrix as seen in equation (20).

$$A = \begin{bmatrix} 1 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & dt \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{20}$$

The KF output increased as well to include the vertical distance, y, as well as the velocity in the vertical direction, $\dot{y}$, as seen in equation (21).

$$x_{est} = \begin{bmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{bmatrix} \tag{21}$$

Similarly to the KF with one input, **Q** and **R** were adjusted. After adjusting **Q** and **R**, the system's behavior is identical to the preceding KF. Figure 24 is the resulting output for longitudinal distance.

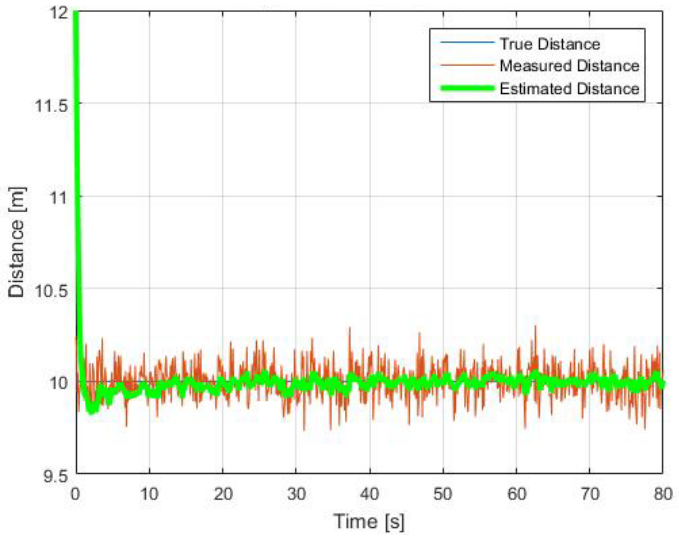


Figure 24. Longitudinal Distance for a KF with Two Inputs

Similarly, Figure 25 is the vertical distance estimation.

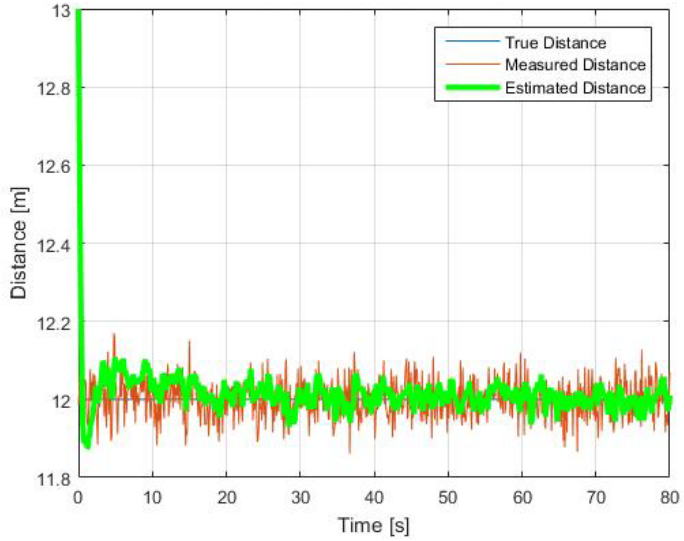


Figure 25. Vertical Distance for a KF with Two Inputs

Since the longitudinal measurements and the vertical measurements are independent of each other, the estimated values behave similarly to each other as well as to the estimated values in the one input KF as seen in Figure 21.

3. EKF with Two Inputs

The KF was then converted into an EKF. The longitudinal distance measurement was replaced with the angle measurement. This results in a non-linear relationship between the measurement and the state. The measurement is described using equation (22).

$$z_t = h(x_{est,t-1}) + v \tag{22}$$

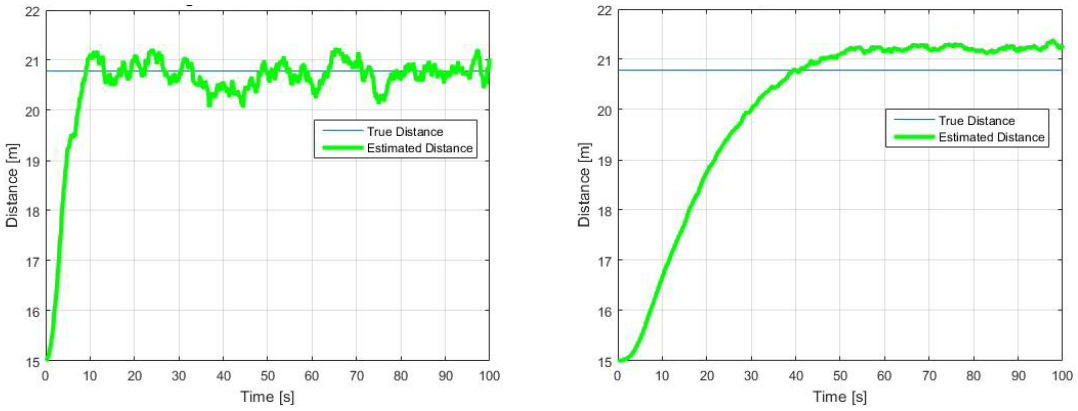
h relates the measurement to the state, and v is an unknown value that represents the measurement noise. H is linearized using equation (23).

$$H = \frac{\partial h}{\partial x} = \begin{bmatrix} \frac{\partial \theta}{\partial x} & \frac{\partial \theta}{\partial \dot{x}} & \frac{\partial \theta}{\partial y} & \frac{\partial \theta}{\partial \dot{y}} \\ \frac{\partial x}{\partial x} & \frac{\partial x}{\partial \dot{x}} & \frac{\partial x}{\partial y} & \frac{\partial x}{\partial \dot{y}} \end{bmatrix} = \begin{bmatrix} \frac{-y_{est}}{{y_{est}}^2 + {x_{est}}^2} & 0 & \frac{x_{est}}{{y_{est}}^2 + {x_{est}}^2} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{23}$$

The **A** matrix and EKF output are the same as the KF with two inputs. To optimize the system, the **Q** values were set separately with a constant **R**. Equation (24) is the resultant **Q** matrix.

$$Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 0.0001 & 0 & 0 \\ 0 & 0 & .001 & 0 \\ 0 & 0 & 0 & 0.0001 \end{bmatrix} \tag{24}$$

The zeros on the diagonal are **Q** values for the velocities. Since the system is stationary, setting the velocity **Q** values close to zero results in higher reliance on the ideal system. The longitudinal distance **Q** value is greater than the vertical distance **Q** value. While reducing the two values associated with the longitudinal distance **Q** value resulted in more noise rejection, it also increased the rate of conversion as seen in Figure 26.



The Q(1,1) value was adjusted to observe the change in time to convergence. Q(1,1) equal to 10 (Left) and Q(1,1) equal to 0.1 (Right)

Figure 26. Longitudinal Distance Estimation for Two Separate Values of Q(1,1)

The time to convergence increases from roughly 10 seconds to 60 seconds which is unacceptable. For this system, **R** equals a four by four identity matrix. The vertical distance estimation is the same algorithm as the previous KF. The complete EKF was converted into a MATLAB function file, and then an embedded Simulink block within the completed vision-based distance estimation algorithm.

V. TESTING AND ANALYSIS

After completing the three major parts of the algorithm, the algorithm was assembled for testing and tuning. The algorithm was tested on a stationary target utilizing different values of q and a constant **R**. The first test compares the convergence speed as a function of q. The second test compares the speed of divergence when an occlusion was introduced; occlusion is used to model intermittent loss of target by the onboard camera. Based on the experimentation, a q value of 10 is the optimal value.

A. EXPERIMENTAL CONFIGURATION

These experiments were conducted utilizing hardware and software that were optimal for the experiments as well as readily available.

1. Robot Operating System

The Robot Operating System (ROS) is a messaging framework which provides a common language for different hardware and software to interact with each other. ROS is open source and well-documented. Each piece of hardware or software within the ROS architecture is a node. These nodes publish data to a topic within the ROS network. Nodes receive data by subscribing to a topic which another node is publishing. The flexibility of this publish-subscribe architecture allows for almost seamless replacement of hardware or software within the ROS network [16].

2. UAV

Certain features are desirable when selecting a UAV. For vision-based tracking, the UAV must have a camera either rigidly fixed to the drone or attached on a gimbal. Also, the drone must be compatible with ROS, the chosen communication software. Additionally, there are other parameters which will affect the effectiveness of the UAV in an operational environment but not in an experimental environment. These parameters include: battery life, data transmission rate, additional sensors, and max speed.

The Parrot AR.Drone has all the characteristics for these experiments. The UAV has a 720p camera as well as an ultrasonic altitude sensor, built in Wi-Fi, and is made compatible with ROS [17]. Additionally, the UAV is easy to use, well-documented, and readily available. However, the UAV can only stay aloft for approximately 30 minutes on one battery. This would be enough time for a cursory reconnaissance of the target vessel but not for a continual watch. While this drone is ideal for experimental testing, the AR.Drone should be replaced for operational use with a UAV with a longer flight time.

B. INTEGRATION OF ALGORITHMS

The three separate algorithms were combined into an image-based position estimator. Before integration, each algorithm was tested using the same values that the previous algorithm had output. This resulted in an almost seamless integration. Next, critical values were tapped so that their data would be recorded for analysis. Figure 27 is the resultant Simulink algorithm.

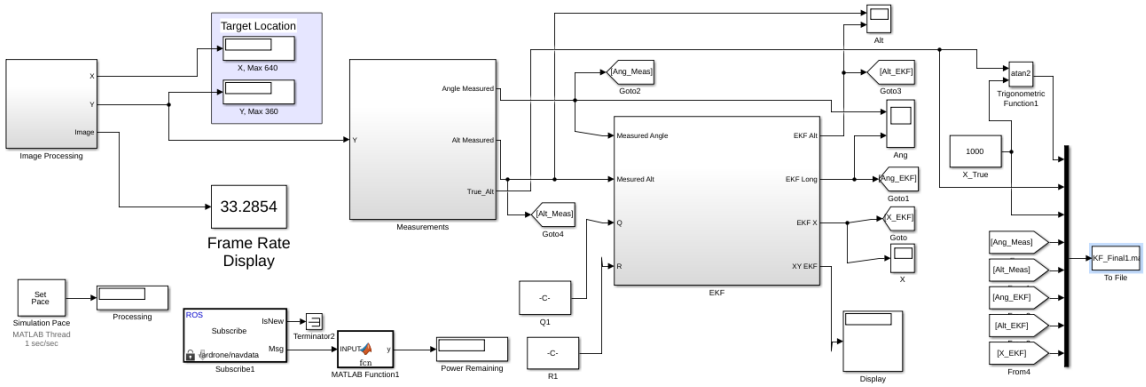


Figure 27. Assembled Algorithm in Simulink

C. EXPERIMENT AND RESULTS

The culminating experiment was a demonstration of the image tracking algorithm's effectiveness. The goal was to determine the optimal values of q and **R** for the entire system. For setup, the drone was placed on a rail in front of a lit target, as seen in Figure 28.

Figure 28. Culminating Experiment Setup

The rail allowed the drone to move longitudinally but restricted latitudinal motion as well rotational motion. The longitudinal and vertical distances between the camera and the target were measured, establishing true distances. Next, the **R** values were fixed into the system. See equation (25).

$$R = \begin{bmatrix} \sigma_\theta^{\,2} & 0 \\ 0 & \sigma_y^{\,2} \end{bmatrix} = \begin{bmatrix} (0.05rad)^2 & 0 \\ 0 & (0.1m)^2 \end{bmatrix} \tag{25}$$

σ_θ is the observed standard deviation of the angle, and σ_y is the observed standard deviation of the angle. Additionally, the vertical measurement was input into the system as a constant value with no noise.

The first part of the experiment, quantified the algorithm's speed to convergence for varying values of q. Thirty second runs were conducted with varying values of q, which ranged from 0.01to 1000. Additionally, the initial estimations of state were set away from the actual values in order to observe the speed of convergence to the steady state measurement. Figure 29 shows the resultant runs.

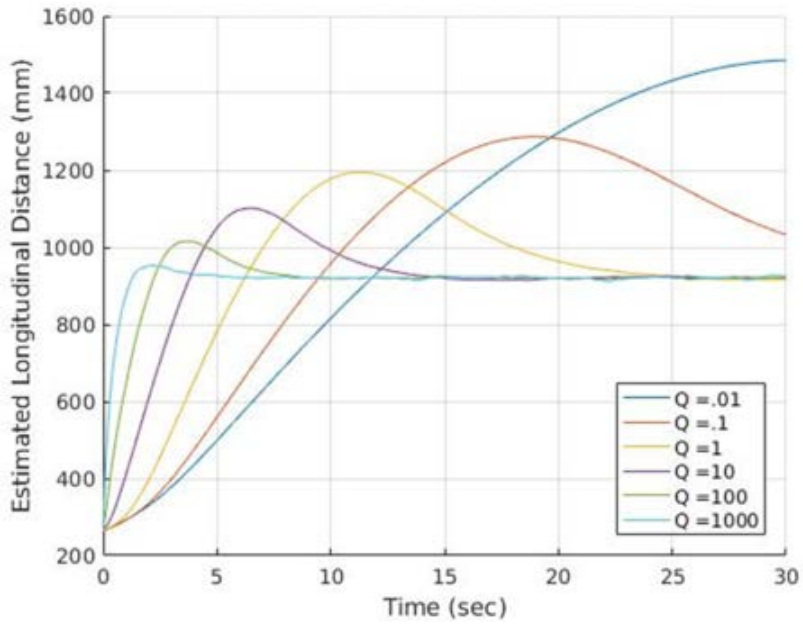


Figure 29. Position Estimation for Varying Q Values

All runs which required more than five seconds to converge to the steady-state value were eliminated as being too slow.

The second part of the experiment demonstrated the effects of disturbances to the system. The 30 sec runs were repeated but occlusions were momentarily introduced 10 and 20 sec into the run. When the target was lost, the tracker returned the position of its default pixel position, (0,0). This disturbance resulted in a measured longitudinal distance far greater than the true longitudinal distance. This occurrence could mimic the glare of the sun off the ocean. Figure 30 shows the response for q =100 and q = 1000.

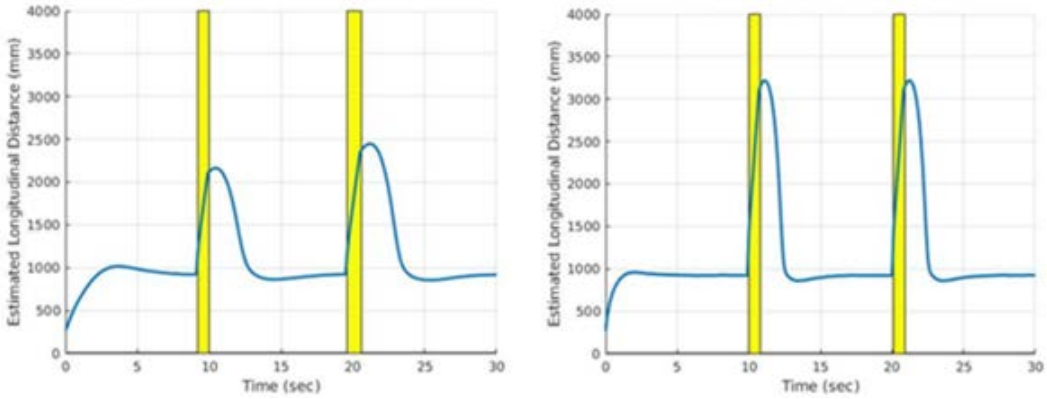


Figure 30. Position Estimation with Occlusions for q=100 (left) & 1000 (right)

Position estimations for q=100 and q=1000 both result in position estimation of over twice the actual distance when the momentary disturbance is introduced. In contrast, the position estimation for q=10 was much slower to diverge during the disturbance, as seen in Figure 31.

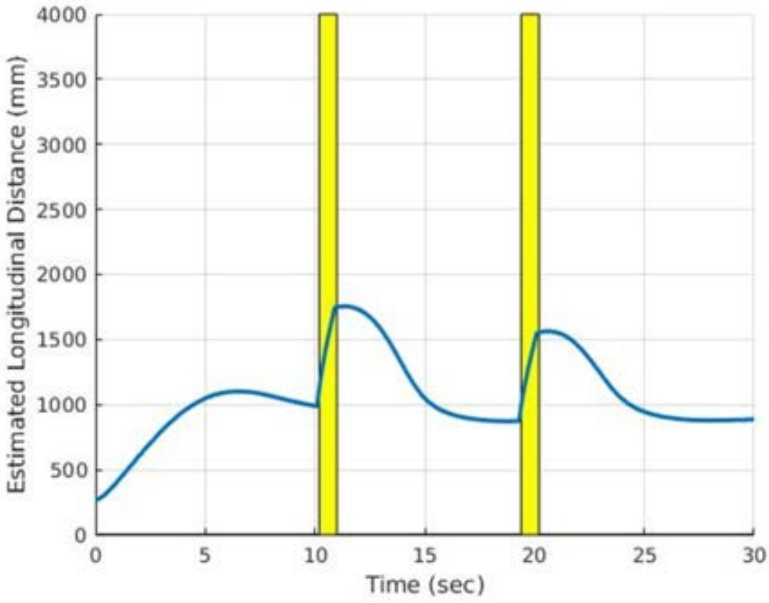


Figure 31. Position Estimation with Occlusions for q=10

Due to the algorithm's ability to quickly converge to the measured value while damping the effects of momentary occlusions, the optimal value of q is 10 for the fixed values of **R**.

D. ANALYSIS

1. Effects of Varying Q Values

Selecting a **Q** value is a balance of system responsiveness and noise rejection. As seen in Figure 26 as well as Figure 29, smaller values of **Q** result in slower responses to measurements. This is due to the algorithm weighting the ideal system more than the measurements. When disturbances such as noise or sudden changes are introduced to the system, the output will more closely track the ideal system. This results in a damping effect to the noise. Conversely, increasing the **Q** value results in a faster response to the measurement. This is due to the algorithm weighting the measurement more than the ideal system. This faster response also results in a faster tracking of noise and less damping to the measurement.

2. Steady State Error from Ground Truth

There are a few possibilities for the steady state error observed in the distance estimator. The measured longitudinal distance between the target and the camera was one meter. The estimated steady state distance of the camera was 0.92 meter. The UAV was held in place throughout the experiment. One possibility is that there is an error in pixel to angle calibration. A discrepancy between the pixel to angle conversion could result in the steady state distance error. This error could be corrected by rerunning the pixel to angle calibration experiment or by modifying the estimation algorithm. Additionally, the error could be caused by errors in experimental setup. Millimeter discrepancies in measurements can compound to become a larger discrepancy. Re-testing the system would help in identifying these errors. Finally, fluctuations within the target's location in the image could affect the resultant estimated position. While the chosen targets were selected for their small size, their position in the camera occupies multiple pixels. Discrepancy between the spot the camera selected on the image and the measured position could result in an error in estimated position. Most likely, the system's steady-

state error is a combination of these possibilities. The application and its acceptable level of error will determine whether this error is acceptable.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSIONS AND FUTURE WORK

A. CONTRIBUTIONS

At the conclusion of this thesis, we developed a vision-based algorithm which estimates position using only passive measurements without using GPS. The algorithm receives an image from the drone. It locates the target in the image and outputs an angle. The tuned EKF estimates the drone's longitudinal position using this angle as well as an altitude provided by the UAV. These pieces of the algorithm were individually optimized then combined and implemented into an algorithm. The algorithm was characterized using various experiments. The result is a working estimator which can be the foundation for future research. The work described in this thesis can be found on the NPS Gitlab server in the project: Vision Based Tracking EKF Thesis https://gitlab.nps.edu/jtesta/vision-based-tracking-ekf-thesis.

B. FUTURE WORK

There are many ways in which this work must be expanded before it can be applied outside of an experimental environment. This future work can be categorized into four groups: extension, disturbance characterization, assumption elimination, and integrate estimator with controller.

1. Extension

More complexities could be added to the system to make it more effective at estimating its position, including:

- Replace constant altitude value with ultrasonic barometer measurement. This measurement would introduce more noise into the system and the EKF would require retuning.
- Replace the intensity detector with an object recognition algorithm. While the intensity detector works in an experimental setup, it would be suboptimal in an operational environment. It would require that the target be the brightest object in the image. An object recognition algorithm would allow for a more flexible tracker.

- Replace the zero order hold with a first order hold. The current algorithm assumes the target holds position if the algorithm loses track on the target. Predicting the future position of the target based on its velocity gives the tracker a better chance of finding the target.

- Replace the linear pixel to angle algorithm with a nonlinear algorithm. While a linear calculation works well for the combination of distances and camera in the experiment, this correlation may not be the best fit at longer distances and/or with a different camera.

- Increase the update rate of the EKF and retune the Q and R values. The EKF updates at the same rate as the image acquisition portion of the algorithm. The EKF could run significantly faster and provide a more accurate output.

The goal of these optimizations would be to make the system faster and more accurate as well as making the estimator more robust.

2. Disturbance Characterization

This thesis proves that the algorithm works in an experimental environment. An operational system must be able to operate in different environments. Some disturbances which can be applied to the algorithm include:

- Tracking with glare from the sun (either reflecting off the ocean or dawn/dusk)

- Tracking in fog or reduced visibility

- Tracking with intermittent loss of target

These disturbances will occur in an operational environment. Algorithm modifications will be required to account for them.

3. Assumption Elimination

In order to narrow the scope of the thesis as well as simplify the problem, many assumptions were made regarding the system. These assumptions include:

- Camera is rigidly installed on UAV and at the center of the drone. While applicable for this UAV, more complex drones have gimbal cameras.

- Attitude of camera is known and constant. Since the drone was not flying, it was always parallel to the ground. When flying, the drone's attitude will change and must be accounted for in calculations.

- Altimeter provides relative height to target. This assumes that the target is always on the ground. In the fleet, the drone will not know the height of the target from the water.
- System is two dimensional. To simplify the problem, the target is always on the y-axis of the camera. In the fleet, the target will have different aspects, unless it's a sphere. Additionally, the angle calibration will be need to be redone at different distances off the y-axis.

While these assumptions were acceptable for the limited scope of this thesis, changing parameters such as the distance to target or the type of drone could render an assumption unrealistic. Removing any one assumption makes the problem incrementally more complex, but also makes the problem more realistic.

4. Integrate Estimator with Controller

The next step in this research is to implement the estimation algorithm in a controller and then experiment with a flying UAV. The goal of the first experiment would be to maintain a relative position to a fixed target. Afterwards, there are numerous experiments which can be performed to characterize and optimize the system. With ROS, interchanging UAVs is mainly an exercise in retuning the pixel to camera ratio and retuning the Q and R values.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX. EXTENDED KALMAN FILTER

```
function [XY_out, K_out, P_out]= EKF_fn(Ang_In, Alt_In,  XY_In, P_In, Q, R)
% Extended Kalman Filter
% Joseph Testa
% Naval Postgraduate School
% Adapted from the process used by Welch and Bishop

% Knobs
%Q Process Noise Covariance, (increase Q) = trust process (aka model) trusted less
%R % Measurement Noise Covariance, (increase R) = trust measurement less

dt= .1; %Time Interval

% Initial Estimates
        XY_est = XY_In;
        P_est = P_In;

% Matrix relates previous step to next step
Amat = [1 dt 0 0; 0 1 0 0; 0 0 1 dt; 0 0 0 1];

% Predict
XY_proj= Amat*XY_est; % Prediction (Project), using estimate from last step
P_proj =Amat*P_est*Amat'+ Q; % Predict/project error covariance ahead

Z = [Ang_In; Alt_In]; % Measurements

% Update Jacobian
h11 = -XY_est(3,1)/(XY_est(1,1)^2+XY_est(3,1)^2);
h13 = XY_est(1,1)/(XY_est(1,1)^2+XY_est(3,1)^2);
H = [h11 0 h13 0; 0 0 1 0];

Z_proj(1) = atan2(XY_proj(3),XY_proj(1));  % theta - predicted
Z_proj(2) = XY_proj(3);

RESID = Z-Z_proj'; %Residual

% Kalman Gain
K = P_proj*H'*inv(H*P_proj*H'+R); % Calculate Kalman Gain

% Estimation
XY_est = XY_proj+K*RESID; % Update position with measurement

% Errror Covariance Matrix
P_est=(eye(4)-K*H)*P_proj; % Estimated error covariance

% Output
XY_out = XY_est; % Estimated x,x_dot,y, y_dot
P_out = P_est;
```

```
K_out = K(1:2,1); % Kalman Gain

end
```

Published with MATLAB® R2016a

LIST OF REFERENCES

[1] J. Brandon, “GPS jammers illegal, dangerous, and very easy to buy,” 17 March 2010. [Online]. Available: http://www.foxnews.com/tech/2010/03/17/gps-jammers-easily-accessible-potentially-dangerous-risk.html. [Accessed 21 November 2016].

[2] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones and R. Ghabchelo, “Vision-based tracking and motion estimation for moving targets using small UAVs,” 2011.

[3] O. A. Yakimenko, I. I. Kaminer, W. Lentz and P. Ghyzel, “Unmanned aircraft navigation for shipboard landing using infrared vision,” *IEEE Transactions on Aerospace and Electronic Systems* , vol. 38, no. 4, 2002.

[4] N. R. Ramos, “Assessment of vision-based target detection and classification solutions using an indoor aerial robot,” September 2014. [Online]. Available: http://calhoun.nps.edu/handle/10945/43984.

[5] S. Dodds, “Pinhole optics,” [Online]. Available: http://www.owlnet.rice.edu/ ~dodds/Files231/pinhole.pdf. [Accessed 6 December 2016].

[6] C. Kanan and G. W. Cottrell, “Color-to-grayscale: Does the method matter in image recognition?,” 10 January 2012. [Online]. doi: 10.1371/journal.pone.0029740.

[7] MathWorks, “Detecting a cell using image segmentation,” [Online]. Available: https://www.mathworks.com/help/images/examples/detecting-a-cell-using-image-segmentation.html. [Accessed 29 November 2016].

[8] MathWorks, “Template matching,” [Online]. Available: https://www.mathworks.com/help/vision/ref/templatematching.html. [Accessed 12 December 2016].

[9] MahtWorks, “vision.TemplateMatcher System object,” [Online]. Available: https://www.mathworks.com/help/vision/ref/vision.templatematcher-class.html. [Accessed 29 November 2016].

[10] MathWorks, “Color space conversion,” [Online]. Available: https://www.mathworks.com/help/vision/ref/colorspaceconversion.html. [Accessed 29 November 2016].

[11] S. Dikbas, T. Arici and Y. Altunbasak, “Chrominance edge preserving grayscale transformation with approximate first principal component for color edge detection,” in *IEEE International Conference on Image Processing*, San Antonio, 2007.

[12] B. A. Olshausen, “Aliasing,” 10 October 2000. [Online]. Available: http://redwood.berkeley.edu/bruno/npb261/aliasing.pdf.

[13] G. Welch and G. Bishop, “An introduction to the Kalman filter,” 24 July 2006. [Online]. Available: http://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf.

[14] G. F. Welch, “HISTORY: The use of the Kalman filter for human motion tracking in virtual reality,” *Presence*, vol. 18, no. 1, pp. 72–91, 2009.

[15] V. N. Dobrokhodov, *Model Parameterization and Parameter Estimation*, 2013.

[16] Open Source Robotics Foundation, “About ROS,” [Online]. Available: http://www.ros.org/about-ros/. [Accessed 30 November 2016].

[17] Parrot, “Parrot AR.DRONE 2.0 Power Edition,” [Online]. Available: https://www.parrot.com/us/drones/parrot-ardrone-20-power-%C3%A9dition#technicals. [Accessed 30 November 2016].

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California