Autonomous Fixed-Point Landing for Quadrotor Aerial Vehicles

A Major Qualifying Project Report
submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science
by

_____

Brian Gallenstein

_____

Yael Rosenblum

In conjunction with Shanghai University students

_____

Jack Zhang

_____

Michael Chin

_____

Andrew Huang

Date: January 6, 2015

Advised by:
Professor Kevin Rong
Associate Professor ZhangXi

TABLE OF CONTENTS

TABLE OF FIGURES

ABSTRACT

Using a student-built quadcopter, the navigation capabilities of autonomous aerial vehicles were explored. The project implemented an artificially intelligent quadcopter that could autonomously identify, navigate to, and eventually land at a marked location. Through the use of an onboard camera and Raspberry Pi control board, image processing techniques were developed in Python in order to analyze and understand the quadcopter's environment. By binarizing and contouring the image feed from the camera, a red landing target was identified. Using this environmental model, control algorithms were developed in order to intelligently navigate to and land on the target autonomously.

ACKNOWLEDGEMENTS

The project team would like to express their sincerest appreciation to the students, faculty, and administration of Shanghai University (SHU) and Worcester Polytechnic Institute (WPI), who have made this collaborative experience possible. Additionally, the team would like to specially thank our Chinese teammates Jack, Michael, and Andrew for all their help in making both the project and our experience in China a success.

The project members would also like to express their deepest gratitude to Shanghai University for hosting and taking part in WPI's unique global projects program. Finally, the entire team would like to thank all of the advisors who offered their time and guidance to this project. Thank you to Professor Zhang Xi of Shanghai University for providing us with direction, resources, and knowledge especially with the image processing aspects of the project. Lastly, thank you to Professor Yiming Rong for ensuring the project ran smoothly.

INTRODUCTION

The Quadcopter Fixed-Point Landing project team consisted of two students from Worcester Polytechnic Institute Polytechnic Institute (WPI) and three additional team members from Shanghai University (SHU). The project team was based in the Zhabei district of Shanghai, China at SHU's Yanchang campus. Initial communication was established seven week prior to the project start date in order to make preparations and determine the scope and goals of the project. This time was used to outline specific objectives and research current applications of quadcopter and image processing technology.

This project sought to explore the navigation capabilities of aerial vehicles using an onboard camera for video capture and image processing. The main goal was to implement an artificially intelligent quadcopter that could autonomously identify, navigate to, and eventually land at a marked location.

The specific tasks necessary for a successful implementation are as follows:

i. Take off and explore with RC control

ii. Identify target on ground using image processing

iii. Switch to autonomous control of roll, pitch, yaw and throttle

iv. Navigate above target

v. Safely land

BACKGROUND

Unmanned Aerial Vehicles

Unmanned aerial vehicles (UAVs) are aircraft that have no human pilot aboard. This includes both simple unpiloted aerial vehicles, as well as remotely piloted aircraft. In the last few decades, they have become more commonly known as “drones”. Their flight is controlled either autonomously by onboard computers or via remote control from a pilot either on the ground or even in another vehicle.

Historically, the launch and recovery method of UAVs was strictly using remote human pilots. However, recent advances have led to an increase in the implementation of partially and completely autonomous systems. UAVs can serve many functions, but are primarily regarded by the public eye as tools for military and special operations applications. Despite this image, the past decade has seen a large growth in civilian and commercial uses of UAVs. These applications include firefighting, policing, security work, aerial photography, search and rescue, scientific research, maritime patrol, and even video capture for commercial motion pictures.

The diverse applications of aerial vehicles are mirrored in their equally diverse designs. The various configurations and constructions of these vehicles directly determine their in-flight capabilities and because of this relationship between form and function, quadcopters have emerged as the most popular UAV. The popularity of quadcopters stems from their adaptable and cost-effective design, which can be customized to excel at most general-purpose applications.

Introduction to Quadcopters

A quadcopter, also known as a quadrotor, is a multirotor helicopter that generates lift and propulsion using four rotors.[1] Quadcopters are classified as rotorcraft, as opposed to fixed-wing aircraft, because their lift is generated by a set of vertically oriented propellers.

Unlike most helicopters, quadcopters use two sets of paired propellers. One set is comprised of two clockwise (CW) oriented propellers, and the other consists of two counter-clockwise (CCW) oriented propellers shown in **Figure 1**. By alternating the direction that each propellor spins, the net torque on the quadcopter is zero. This allows the system to be stable without the use of a stabilizing rotor such as the tail rotor of conventional helicopters. Furthermore, the quadcopter can then be controlled through slight changes in the angular speed of each propeller. By changing the revolutions per minute (RPM) of the propellers, an imbalance of thrust and rotational momentum is introduced and a resultant roll, pitch, yaw or throttle is generated in the entire system.

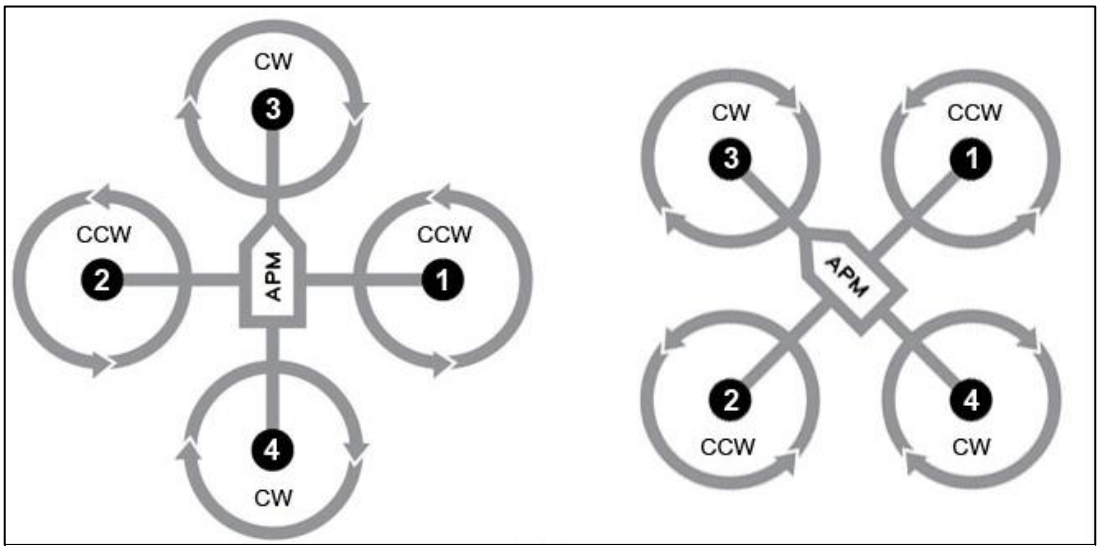

Figure 1: Common Quadcopter Configurations

[1] Hoffmann, G.M.; Rajnarayan, D.G.; Waslander, S.L.; Dostal, D.; Jang, J.S.; Tomlin, C.J. (November 2004)

Early in the history of flight, multirotor aircraft were explored in order to circumvent traditional stabilization problems of rotorcraft.[2] Configurations such as the helicopter have inherent control problems due to the torque generated by the large main rotor. In a helicopter, this problem is combated by a small vertical tail rotor that aims to prevent the main body of the rotorcraft from rotating. Beyond complicating the system, the use of a tail rotor has a downside in that it requires additional motors that do not contribute to lift. Because weight reduction and efficiency are vital in flying mechanisms, additional motors are often unsustainable or undesirable. As a result, multirotor designs emerged, using opposite rotor pairs to cancel out unwanted rotational energy while still providing lift.

Additionally, by using four rotors to generate lift, each rotor can be much smaller in diameter. This reduces the amount of kinetic energy present in each rotor, reducing the force of impact in the case of a crash. Due to these various advantages, a number of manned designs appeared in the 1920s and 1930s. These vehicles were some of the first successful heavier-than-air vertical takeoff and landing (VTOL) vehicles.[3] The main fallbacks to these pioneering designs were poor performance and too much pilot workload, the latter induced by primitive control systems.

Recently, quadcopter designs have become popular in the field of unmanned aerial vehicle research.[4] Generally small in size, quadcopters use a variety of sensors to achieve a high level of stability and control, allowing them to navigate even in narrow spaces. Additionally, because each rotor is small, they contain less energy during flight, which makes quadcopters much safer both to

---

[2] Büchi, Roland (2011). *Fascination Quadrocopter*.

[3] Leishman, J.G. (2000). *Principles of Helicopter Aerodynamics*. New York, NY: Cambridge University Press.

[4] Hoffmann, G.M.; Rajnarayan, D.G.; Waslander, S.L.; Dostal, D.; Jang, J.S.; Tomlin, C.J. (November 2004).

human operators and to the flight environment.[5] Lastly, quadcopters are generally low cost and easy to construct. All these factors contribute to making them the rotorcraft of choice for most academic and research purposes.

Current Quadcopter Implementations

Currently, there is a spurring market for personal and commercial quadcopters. These quadcopters are vested in different purposes and offer multiple functionality for its users. As it currently stands, UAV flight is legal for non-commercial purposes. However, with new regulations and the impeding popularity of autonomous flight vehicles, will result in the growing number of quadcopter uses in the near future.[6] Below are the most prominent current varieties of quadcopters along the different markets they serve.

Figure 2: The AR Drone

AR Drone

The AR Drone built by the French company Parrot is a prime example of the entertainment capabilities quadcopters are frequently associated with. The

[5] Hoffman, G.; Huang, H.; Waslander, S.L.; Tomlin, C.J. (20–23 August 2007).

[6] Lowensohn, Josh. "FAA considering a Fast Track for Businesses That Want to Use Drones." *The Verge*. N.p., 16 May 2014. Web. 12 Oct. 2014.

major aspect that set Parrot's flying quadcopter vehicles apart, is the user control. The AR Drone is designed to be controlled by mobile devices, specifically smartphone software with iOS and Android.[7] During flight, users have the ability to hone in their photography skills with the aerial videos and images taken during in-flight control. These images are captured through the two mounted cameras on the quadcopter. Each of the cameras is USB connected with its own Wi-Fi capabilities, where the first is connected to the front and the second is vertically mounted[8]. There are a number of applications for the AR Drone, in the realm of entertainment and sport. The release of AR.Race, AR.Rescue, and AR.Hunter are piloting functions organized by the AR Drone community to compete in competition leaderboard races and obstacles.

Aeryon Scout

The main purpose of the Aeryon Scout drone is to bring the utilization of an intelligence gathering aerial vehicle to the local user. The main tagline for this

Figure 3: The Aeryon Scout

[7] "AR.Drone 2.0. Parrot New Wi-fi Quadricopter - AR.Drone.com - HD Camera - Civil Drone - Parrot." *AR.Drone 2.0.* Parrot, n.d. Web. 1 Oct. 2014.

[8] Brandon, Alan. "Control Your Own Augmented Reality Aerial Drone? There's an App for That." *Gizmag*. N.p., 6 Jan. 2010. Web. 13 Nov. 2014.

quadcopter is the ability of any user to be able to handle and control its motion. It is specifically categorized as a small Unmanned Aerial System (sUAS) that can be used in most terrains and under most conditions. This main military focus of the quadcopter is meant to be easily compacted into storage and taken on the road for image processing of geo-terrain.[9] The Aeryon Scout includes a 3-axis stabilized high-resolution still camera in color and infrared, a thermal infrared video camera, and a 10X optical video camera. This drone focuses on enabling industry-standard technology to give precise geo-location and post-processing tools in order to gather the most intelligent stitched images available on a consumer UAV.[9] With these features, the most common exercise applications for the Aeryon Scout is for real-time aerial surveillance and intelligence that can be used for military and policing detail. Another main focus from its perimeter and convoy security shines light to its usability of emergency and disaster response.[9] This is very important to be able to have a variety of preventative tools available on a consumer level.

Amazon Prime Air

A prime example of a more commercialized drone in the consumer industry is the Amazon Prime Air, also known as the “Amazon Drone.” Jeff Bezos,

Figure 4: Amazon Prime Air

[9] "Aeryon Small Unmanned Aerial Systems." *Aerial Vehicle Systems*. Aeryon, n.d. Web. 3 Dec. 2014.

the CEO of Amazon, unveiled this UAV. It was announced on a television program about the capabilities of package delivery in a matter of thirty minutes. Having such an announcement made about the functionality of a drone delivery system to the everyday consumer is a huge step to implementing more day-to-day usage from aerial vehicles. In the interview with Bezos, it was said to be projected to launch in close to four to five years.[10] Since this type of drone usage would affect a larger majority of people in terms of safety. In this case, Amazon is embracing their drone technology to ensure the operations are safer. Their planned utilized resources are geofencing to ensure the drones to do not exceed a specific height, currently no more than 400 feet. Another tool is GPS or radio frequency triangulation in order to better define geological obstacles for mapping.[11] Currently the only roadblocks are set by the regulations of the Federal Aviation Administration (FAA) for the specifics on unmanned aerial aircrafts

Digital Image Processing

Image processing is the analysis of visual data in the form of video or still images. There are a number of uses for image processing including visualization of objects that are otherwise difficult to view, image sharpening and restoration, image retrieval, image recognition and computer vision. Raw data is gathered from camera sources and then various techniques are applied to achieve different results. Most image processing techniques treat the input as a two-dimensional signal in the form of an array of pixels. This two-dimensional data represents a view of a three-dimensional environment, which incurs an inherent

---

[10] "Amazon Drones: Amazon Unveils Futuristic Delivery Plan." *CBSNews*. CBS Interactive, n.d. Web. 09 Dec. 2014.

[11] McNeal, Gregory S. "Six Things You Should Know About Amazon's Drones." *Forbes*. Forbes Magazine, 11 July 2014. Web. 6 Nov. 2015.

loss of data. Image processing attempts the difficult task of extracting information about the original environment from the simplified two-dimensional data.

Because of the limitation of conventional image data, depth cameras or multiple cameras are sometimes used to gather more information about the environment. However, these advanced approaches are outside of the scope of the project and, as a result, only two-dimensional image processing will be explored. Image processing can be broken down into three main processes comprising of a preprocessing stage, an analysis stage, and a display or data reading stage.

Image Processing Techniques

Preprocessing

Preprocessing is a blanket term for various methods with which data is prepared for analysis. In the realm of image processing, this is usually to clean up a noisy image and thus improve the performance of subsequently applied image processing algorithms. Often this is as simple as just blurring or filtering the image. Gaussian blurring and median filtering are the two most common types of filtering for image processing. These methods mainly act to smooth edges and remove “salt” from noisy images. Other preprocessing may include downsampling, where the image resolution is reduced for performance purposes, or various data conversions, such as converting a color image to grayscale or changing the convention of the pixel values.

Data Analysis

Most image processing techniques treat the input as a two-dimensional signal in the form of pixels. This two-dimensional data represents a view of a three-dimensional environment and the aim of image processing is to extract information about that original environment. This is inherently difficult due to the data loss in image capture. Because of this, depth cameras or multiple cameras are sometimes used to gather more information about the

environment. However, these advanced approaches are outside of the scope of the project and as a result, only two-dimensional image processing will be explored.

There are many uses for image processing, but many of the common tasks involve identifying and grouping distinct shapes in the image. This process is known as image segmentation. There are many ways of accomplishing image segmentation, but three of the most popular methods are thresholding, edge detection, and using data clustering.

Thresholding is the most basic technique and is the process of converting a grayscale image into a binary one. By assuming that most images can be divided into a distinct foreground and background, thresholding attempts to simplify an image by changing the foreground to white and everything else to black. The most intuitive way of thresholding is to pick a cutoff value (or range of values) and assign all of the pixels on one side of the cutoff to be the foreground, and any others to the background. Different algorithms are used to determine the optimal cutoff value, but balanced histogram thresholding and Otsu's method are some of the most popular.

Edge detection is the process of finding the outlines or contours of objects in an image. While being a sizable area of image processing in its own right, the contours generated by edge detection can be used for image segmentation. The aim of edge detection is to maintain the important structural information of an image while also filtering out any less relevant information. Contours are primarily identified by detecting sharp changes in the brightness or intensity of an image. It can be generally assumed that discontinuities in light correspond to sudden changes in depth or subject matter. However, determining the exact magnitude of discontinuity needed to identify an edge is a non-trivial problem. Many different approaches are used, but they can be grouped into either search-based techniques, which use first-order derivative expressions to find gradients in the image, and zero-crossing techniques, which use the zero points of second-order derivative expressions to directly find edges.

The most widely adopted algorithm is the Canny edge detector, which is still the most stable and accurate method, despite being created very early in the research of digital image processing. Generally, edge detection techniques are very sensitive to noise in the image, so they are typically paired with a smoothing or blurring preprocessing step. Lastly, the method of data clustering can be used to identify regions in an image. The most popular manifestation of data clustering is the k-means algorithm, which chooses K cluster centers and then averages pixel values based on the closest center. The algorithm is iterative in nature and simplifies images into a series of regions based on color. The number of regions generated depends on the K value supplied to the algorithm. K values can either be hardcoded or dynamically chosen depending on the application and preconditions.

Data Utilization

The final step of image processing is to intelligently display or utilize the results of the processing. For applications such as digital image editing, this is as simple as displaying the edited image and then saving it to a file. More complex applications such as computer vision may pipe the data from the image processing into higher-level control algorithms. This stage of image processing may include some post-processing, such as overlaying the image processing data over the original image, resizing the output, or even sharpening the image digitally.

METHODOLOGY

Research and Set-up

This project focused on the utilization of image processing techniques to allow an unmanned aerial vehicle to autonomously navigate within a space. Specifically, the main goal of the project was intelligently landing on a target location without a human pilot. Achieving this behavior was the driving force behind design and implementation choices. Because the project included distinct mechanical and software-based challenges, a split approach was designed in order to work on quadcopter control and image processing in parallel. For the first few weeks of the project, the team was split into two parts to handle each of these tasks separately. Later in the project lifespan, both teams combined and worked collaboratively to ensure system coherence and proper testing. The expected deliverable consisted of a custom-built quadcopter that could autonomously navigate to and land on a designated target through the use of computer vision.

In order to accomplish this objective, seven main tasks were formulated:

1. Conduct research and formulate plan
2. Assemble quadcopter
3. Implement object detection through image processing
4. Develop control algorithms
5. Implement autonomous landing
6. Conduct testing
7. Refine system performance

However, even before these steps could be taken, it was necessary to understand the objectives of the project and establish good lines of

communication between the students at WPI and the students at Shanghai University. Creating an initial problem statement, communicating effectively, and preparing for travel to Shanghai University were all vital operations that needed to be completed before the official start of the project.

Once communication and a basic understanding of the project goal was established, it was necessary to conduct some research on the capabilities of the technology. This step was mainly completed remotely before travel to China. Due to the fact that the quadcopter was in Shanghai, the WPI team focused mainly on capabilities of image processing. At this point it was decided to use an open source image processing library called OpenCV. Also, it was determined that the low-level control of the quadcopter would use a slightly modified version of an open source C library called MultiWii. Furthermore, Python was chosen for the high-level programming needs for several reasons. First, the team had some experience with Python and secondly, OpenCV has Python bindings that are arguably cleaner and easier to use than the native C library. On the Shanghai side, research was conducted on the physics of quadcopter flight. Additionally, the necessary components for construction were gathered. After research was finished, the Shanghai team began construction of the quadcopter. The main components of the quadcopter are outlined in **Figure 5**. Initially, the main development boards consisted of an Altera Cyclone IV FGPA for the low level

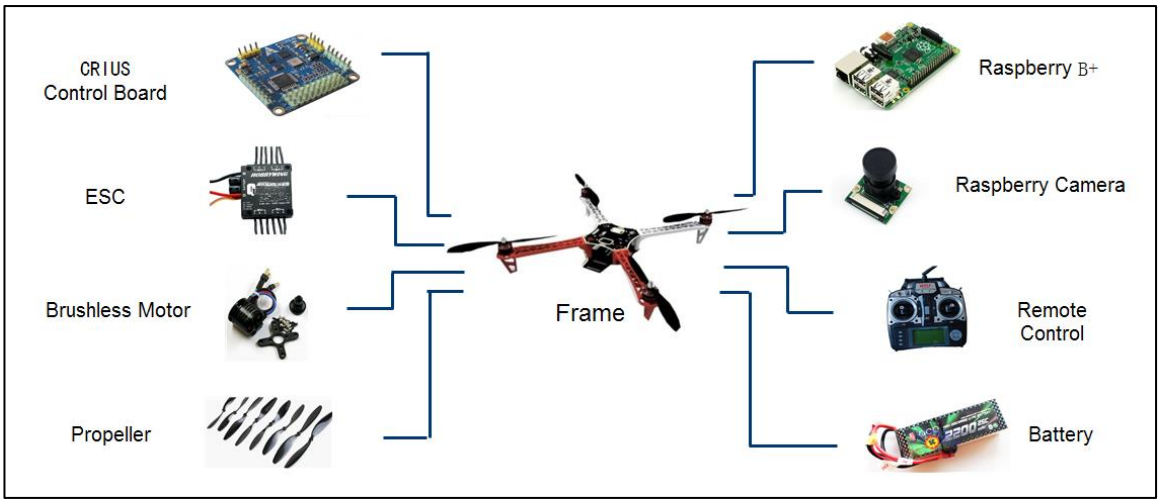

Figure 5: Basic Quadcopter System Layout

MultiWii code and a Raspberry Pi Model A board for implementation of the image processing and control algorithms. However, it was later determined that the Raspberry Pi Model B+ would perform better due to its lower power requirements, increased number of USB ports, and most importantly, its larger amount of memory.These early choices allowed the team to better define objectives week by week and helped shape how future work was divided and accomplished. By understanding the exact limitations and features of our system, more intelligent design choices could be made and a proper methodology could be followed.

System Architecture and Communication

As previously mentioned, the quadcopter control system is made up of two separate control boards, which interface with the various sensors and components of the quadcopter. However, it is important to understand how each component communicates with the rest of the system.

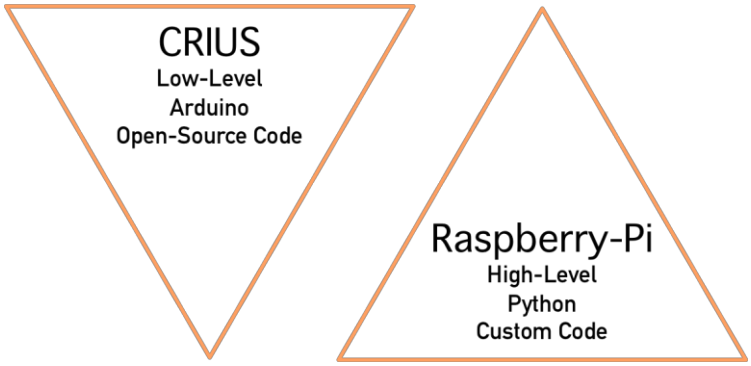

Figure 6: Division of Tasks between Boards

At the lowest level, the system is made up of static structural components. For this application, a 330mm glass fiber platform and four replaceable polyamide nylon arms were chosen. This frame was durable and easy to repair in the case of damage. Additionally, it was compact enough to work for both indoor and outdoor applications.

Next, the assortment of electrical and mechanical devices was needed. Separate from the quadcopter is the radio controller (RC), which allows a human operator to control the quadcopter remotely by controlling four channels: roll, pitch, yaw and throttle. For propulsion there are four brushless DC motors capable of spinning at 7100 revolutions per minute. Attached to these motors are 10 inch propellers that have a pitch measurement of 4.7 inches. Electrically, the motors are connected to the Altera Cyclone IV board through the interface of an electronic speed controller (ESC). The ESC is necessary because it converts the digital signal from the control board into an analog signal that powers the motors. The control board sends four separate signals to the ESC. Each line transmits a pulse-width modulated (PWM) signal that corresponds to the speed that each motor should spin at. PWM signals send data by varying the width of an electrical pulse. For example, if a 50Hz signal stays high for 1ms and low for 19ms per period, it might correspond to a minimum voltage sent to the motors. Conversely, if the signal stays high for 19ms and only goes low for 1ms, a maximum voltage might be sent to the motors. Another way of describing this relationship is by saying that the motor voltage varies linearly with the duty cycle of the input to the ESC.

For sensing, the Altera board comes with an inertial measurement unit (IMU), electronic compass and barometer. The IMU is a very common unit of sensors, which includes a combination of gyroscopes and accelerometers to determine the motion and pose of the quadcopter. This system was necessary for the MultiWii code to perform low-level stability control. The electronic compass is also used to determine orientation. For this project, it remained largely unused. The barometer determines altitude by monitoring atmospheric pressure. This sensor is notoriously unreliable, especially at low altitudes. These inaccuracies rendered the barometer mostly useless and other methods of determining altitude were explored instead. Aside from the aforementioned components, the quadcopter also had a Zigbee module for communicating with the radio controller. All these sensors interfaced with the Altera control board and helped with low-level control of the quadcopter.

For higher-level control, the Raspberry Pi board also has several peripherals. The two main components are the camera and a USB Bluetooth module. The camera is fairly self-explanatory. It points straight down from the quadcopter and provides the Raspberry Pi with data about what is directly below it. The Bluetooth module allows the Raspberry Pi to communicate with other devices using the Bluetooth protocol. This functionality proved invaluable for testing. By connecting with the Raspberry Pi using a Bluetooth interface for mobile phones, it was possible to send serial messages to the board. By writing

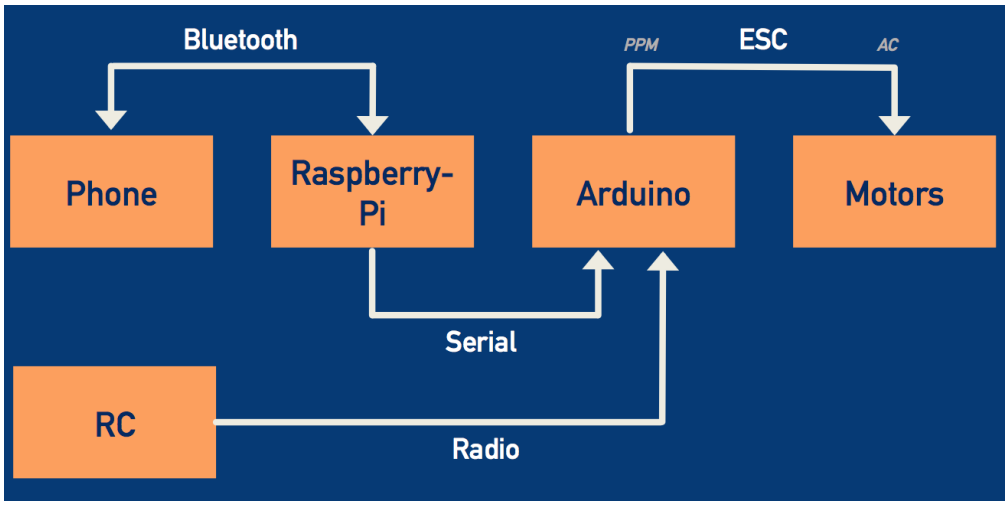

Figure 7: Communication Protocols

software to interpret the incoming Bluetooth messages, it became possible to control the quadcopter with a mobile phone. The exact methods used for testing are covered in a later section.

Lastly, it is important to understand how the two control boards communicated. The Altera board only had one serial port and because the control algorithms were going to run on the Raspberry Pi, it was necessary to use this port to send motor information from the Raspberry Pi to the Altera board. As a result, the Raspberry Pi did not have access to any of the sensors and peripherals connected to the Altera board. The serial connection between the two boards used the UART protocol to transmit data. When the Altera board receives messages on its serial port, it switches to autonomous mode and allows the Raspberry Pi to control the motors using the same four channels that the radio

controller normally uses. Any time the Altera board does not receive signals through its serial port, it defaults back to RC control. This is a safety feature that allows a human operator to control the quadcopter in case of error in the control systems. The communication between the various quadcopter components is summarized in **Figure 7**.

All these components come together to form a complicated yet powerful system. This system is the platform that software builds upon. The hardware provides the physical capabilities of the system, and the software provides the intelligence to utilize the hardware effectively.

Image Processing

In order to navigate autonomously, a quadcopter must be able to gather data and learn about the characteristics of the environment. In our case, a camera was used to gather this information. Before any image processing code could even be written, a camera that could interface with the Raspberry Pi needed to be selected. Initially, a generic USB camera was used, however the resolution and size of the camera were not sufficient. A different USB camera was then purchased and used for early testing, but ultimately, the Raspberry Pi Camera Module was selected. The Raspberry Pi Camera Module was a vast improvement over USB cameras because it connected to the board via a ribbon cable. This interface was optimized to gather image data efficiently by using the Raspberry Pi's GPU to handle some of the processing. This gave some relief to the CPU, which was going to have difficulty handling the image processing, and control code anyways. Additionally, the Camera Module sported more features and customization natively. Once a camera was chosen, the objective was to be able to identify a target in the image.

With any sort of image processing technique, it is important to understand how to utilize the information from the camera, and manipulate efficiently. For the application of autonomous navigation, there are two main concerns that govern

design choices. First and most importantly, the image processing techniques need to sample at a very high rate or else the control algorithms will be starved of data and fail. Secondary to sampling speed, the algorithms need to be able to detect the target reliably and without false positives. These criteria were extremely important because, depending on the solutions chosen, they could change the architecture of the entire system.

One early concern of target detection was whether the processing would occur on or off the board. This decision was warranted by the nature of computer vision being very processing intensive. The Raspberry Pi has a somewhat limited processor compared to a full size desktop, so reaching acceptable sampling rates would most likely require sacrifices in reliability or control algorithm complexity. If an off-board processor was used, the image processing could be completed faster, but the latency of sending image data over a wireless connection might marginalize any gains.

In the two cases, having the processing sent via Bluetooth or other wireless communication on a stronger processor versus on-board processing to eliminate the delay time between transmission of data. Other solutions included downsampling the image data, so that the image processing algorithm needs to analyze less total pixels, and using a region of interest (ROI), which would allow the image processing only to act on parts of the image with relevant data. These options were tabulated and compared as shown in **Figure 8**. Eventually, only downsampling was necessary. Another potential issue was that the data received from the camera might be compromised by the vibration caused during in-flight capture. However, early testing determined this was not a significant concern.

Method	Pro	Con
Off-Board Processing	More processing power	Latency / Requires WiFi
Process every X frames	Can be adjusted in real time	Slower reaction time
Downsize camera frame	Not very processor intensive	Less accuracy / reliability
Use a ROI	Not very processor intensive	Finding ROI can be tricky

Figure 8: Potential Image Processing Optimizations

As previously mentioned, the procedure to process images usually follows three distinct steps: preprocessing, analysis and data utilization. In order to intelligently preprocess, the entire process needs to be planned out. Different analysis techniques require different preprocessing, so the first step was to compare different methods and select the best for our application. It was decided to use a combination of color-based thresholding and edge detection because of their simplicity to implement and solid balance of speed and reliability. This process would allow us to find a specific color target and extract useful information such as the size of the target in pixels and the center of the target within the camera frame.

Once the image processing techniques are selected, proper preprocessing can be conducted. When an image is captured from the mounted camera on the quadcopter, the OpenCV library reads it in as a two-dimensional array of RGB values. This means that the color of every pixel in the image is specified by a combination of red, green and blue. However, in order to do color-based thresholding later in the analysis step, it is generally easier to convert to the HSV convention. HSV represents each pixel as a hue (color), saturation, and value (brightness). After the data is converted to HSV, a Gaussian blur is usually applied to improve performance of later steps.

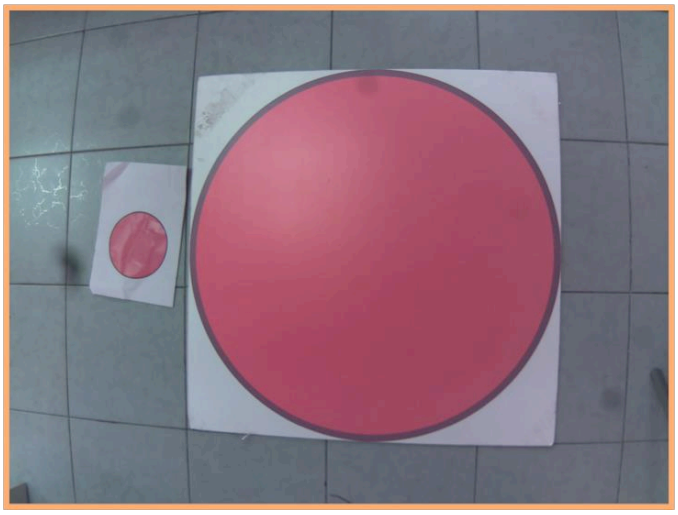

Figure 9: Unprocessed Camera Data

However, after testing, a Gaussian blur did not noticeably affect results and thus was removed to save processing time. An additional factor that strongly affected processing time was the resolution of the image. The desired conditions were to use the smallest resolution that still allowed acceptable reliability. After extensive testing and several camera changes, a resolution of 128x96 pixels was selected. This added an operation to the beginning of the preprocessing step. The image data was downsampled from its native 640x480 before any other operations. This downsampling was vastly important to achieving a reasonable sampling rate of around 3 Hz.

Following the preprocessing operations, the image data is directly analyzed to achieve computer vision. In this application, the analysis consisted of several steps. First, the image was binarized by thresholding between a range of HSV values. This range could be specified given the color of the target. The specified range needed to be fairly large in the saturation and value domains in order to account for different lighting or glares present in the environment, while still not allowing false positives. Because the environment and target was controlled, false positives could be reduced also through design choices. For

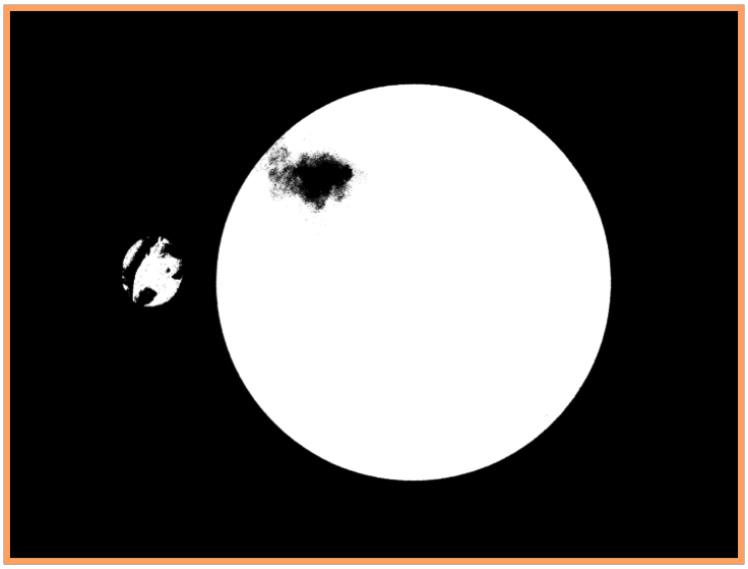

Figure 10: Binarized Image

example, a large red target was selected because our environment did not contain red and thus there was little chance of false positives. Despite these assumptions, qualitative testing was conducted to determine exactly how well different colors could be detected in outdoor environments. To do so, an in-flight video was recorded and the analyzed using a custom developed tool. This tool was designed so that the threshold range could be dynamically updated and applied to the previously recorded footage in real time. This testing demonstrated that the material of the target was more important than the color. Reflective surfaces would disrupt the image processing by appearing white when they reflected sunlight into the camera. Otherwise, the image processing performed well on a wide range of colors, and thus the target was kept red.

Once the HSV threshold is calibrated and applied to the captured image as seen in **Figure 10**, the next step was to implement edge detection. Because the image was already binarized from the thresholding step, Canny edge detection was able to very rapidly determine the contours of the image. Edge detection creates a boundary between light and dark areas using a set of connected curves. These curves create edges around each area of the image that is within the HSV range (red in this case). However, in order to account for

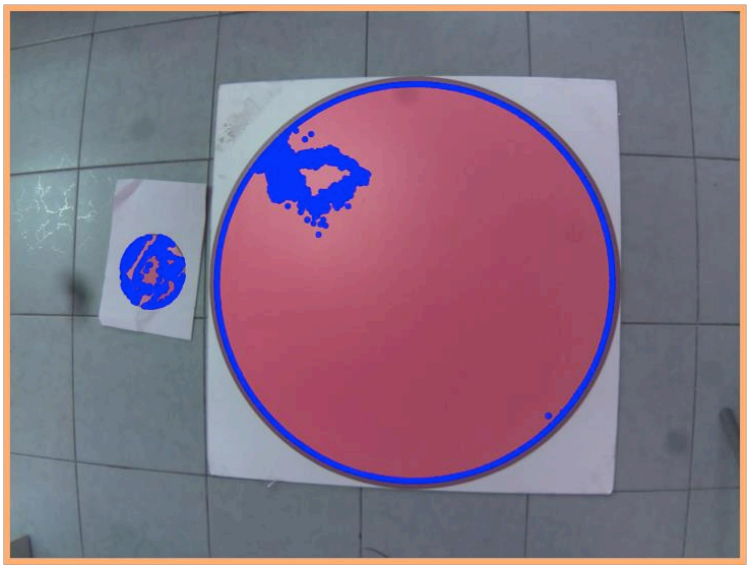

Figure 11: Image Data with Contour Overlay

the false positives or multiple detected objects within the camera view, it was important to select only the best target per image frame. It can be confidently assumed that the largest contour would correspond to the actual target because false positives often manifest themselves as small-detected areas in shadows or other low saturation regions. To improve the reliability of this assumption, any contours around improbably small regions were disregarded completely. This prevented the algorithms from simply selecting the largest false positive in the case when only false positives were detected. For debugging purposes, this contour could then be overlaid on top of the image feed as seen in **Figure 11**.

Lastly, once the contours of the target were detected, it was necessary to convert that data into more useful forms. To accomplish this, a polygon approximation was applied to the contour. The target was circular in nature and thus a circular fit was applied. The contour was approximated by forming the smallest circle that could still enclose the entire contour. The power of this operation was that even when the target is viewed at an angle, the approximated circle would maintain the correct radius. This circular approximation can be viewed in green in **Figure 12**.

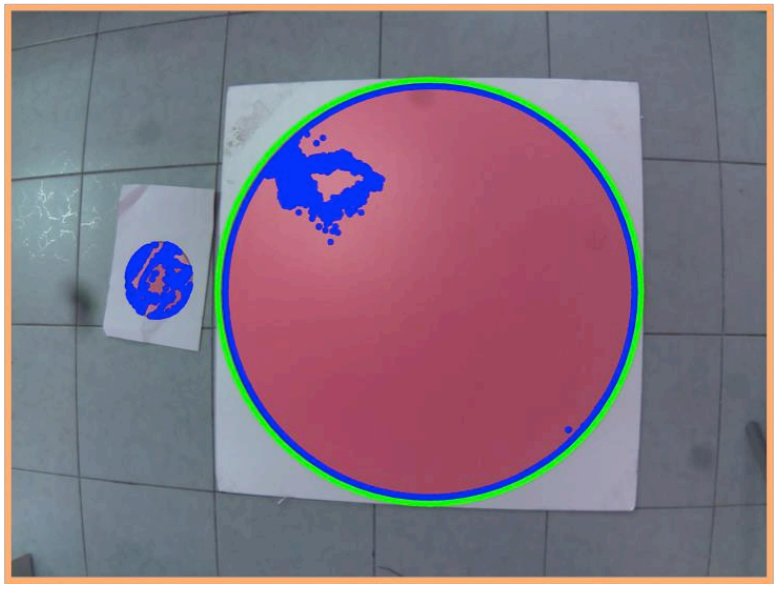

Figure 12: Circular Approximation of Largest Contour

The final step of image processing is displaying and using the data intelligently. The debugging displays have already been shown above to aid the reader in understanding what each step visually looks at, but the end usage of the data has still not been discussed. The challenge of this project was converting what primitive information the quadcopter has about its environment into useful inputs to a control algorithm. It was determined that only three pieces of information were necessary from the image processing for successful navigation. First, is the target even in view? This may seem like a silly question, but it is extremely important that the quadcopter reliably knows when it has the target within view. The other data is useless if there is any doubt the detection of the target in the first place. Once this criterion is satisfied, there are two values that help the quadcopter localize itself relative to the target. The location of the center of the target within the camera frame is important because it allows the quadcopter to determine which direction to fly in order to stabilize over top of the target. Additionally, the size of the target in pixels gives a measure of the height of the quadcopter.

To summarize, the quadcopter has a camera mounted on it that points straight down at the ground. By taking the image feed from this camera, red objects can be detected and approximated as a circle in the camera view. Information about this circular approximation can then be used to determine the location of the quadcopter relative the target. Once the quadcopter knows its location, control algorithms can be used to navigate.

Determining Height

Basic rules of perspective dictate that an object appears larger when it is close to the viewer, and inversely, it appears smaller when it is farther away. This relationship is easy to understand, but modeling the exact relationship between size and distance can be trickier. For the application of this quadcopter, rules of optics can be used to derive altitude using the size of a target in the view. Because the real world size of the target is controllable, it is possible to experimentally determine an equation that provides height in centimeters when provided the size of the target in pixels. In order to accomplish this, the target was viewed at ten-centimeter intervals and the corresponding size in pixels was recorded. Although testing had to be repeated with each change in camera and target, the general set up used can be seen in **Figure 13**.

Figure 13: Height Experiment Set-Up

After gathering data when viewing the target from between one and three meters away, a linear approximation can be applied. As shown in **Figure 14**, the relationship was observed to be inversely exponential. By extracting a mathematical equation of the line, it was possible to reasonably estimate the height of the quadcopter as long as the target was in view of the camera. This

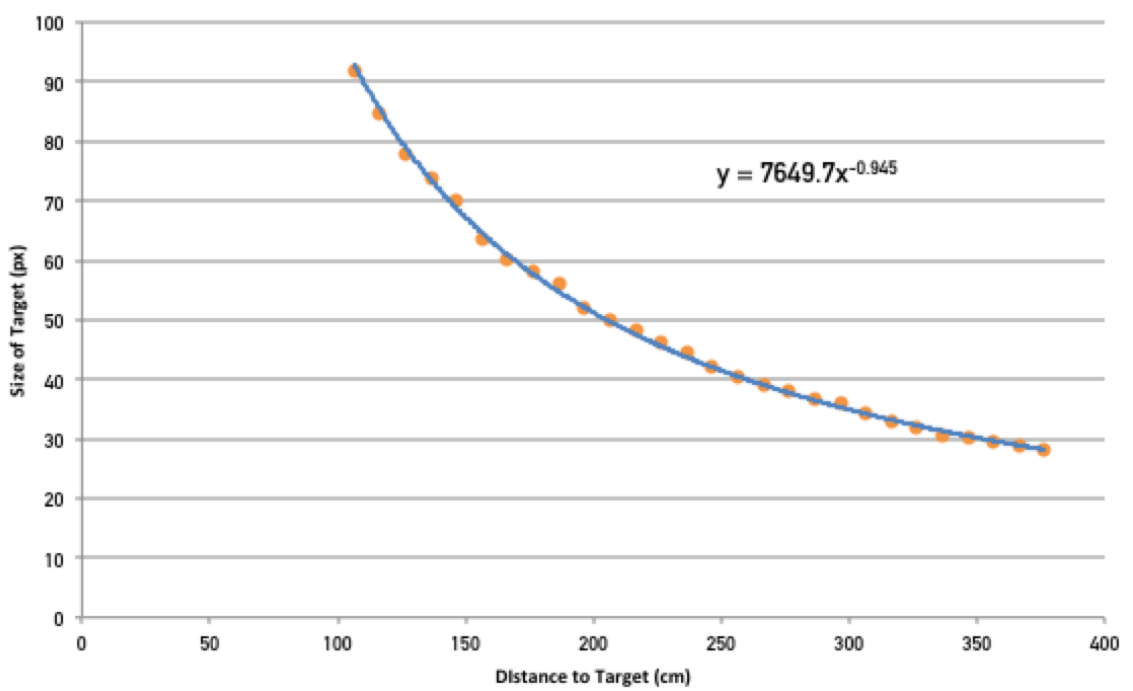

Figure 14: Linear Approximation of Height Data

experimental model had errors of up to twenty centimeters at the edges of its effective range; however, it boasted very little error at around two meters in altitude. The control algorithms could then be optimized to operate in areas where the algorithm was most accurate. These two factors combined to provide a much better estimation of altitude than the on-board barometer could provide. Additionally, the Raspberry Pi did not have access to the lower level sensors; therefore this experiment was vital in order to implement height control.

Control Algorithms

Once the quadcopter orients itself in the environment, algorithms are necessary in order navigate autonomously effectively. There are several ways of implementing control algorithms for flight. One solution is to preemptively calculate a flight trajectory as a set of parametric equations. Then the maneuver can be executed using time as the only control input. However, a static, computation-heavy solution was not optimal for this applications and design constraints, especially because it is highly dependent on accurate sensor data.

The Raspberry Pi board did not actually have access to low level sensors including the inertial measurement unit, barometer, and electronic compass.

Hypothetically this sensor data could be sent from the Altera board via a serial connection, but the only serial port on our board was needed by the Raspberry Pi to send motor controls. Without this sensor data, it is very difficult for a control system to determine the state of the quadcopter at any given time. The only data available to the Raspberry Pi was the image data from the camera. Due to these limitations, a simpler approach was chosen. Both the height of the quadcopter and the relative location of the target are known, therefore, it was intuitive to use a PID control loop to navigate.

Before the details of PID control can be explained, it is necessary to understand how the quadcopter operates when flown by a human using a radio control (RC). MultiWii provides an interface that only requires four values to control the behavior of the quadcopter. By specifying roll, pitch, yaw and throttle, MultiWii will individually control the speed of each motor. Roll is the motion of tilting sideways left or right. It allows the quadcopter to bank sideways from the direction the quad is facing. Pitch is the angle of the quadcopter when it tilts forward or backwards. This causes the quadcopter to fly forward or backwards relative to the front of the quad. Yaw is a twisting motion where the quadcopter rotates around its center. This reorients the front of the quadcopter to a new direction. Lastly, throttle corresponds to the overall speed of the motors. Throttle inherently affects the other three controls; if the throttle is high, changes in roll and pitch especially will result in more aggressive maneuvers. These four channels are mapped to control sticks on the remote control. By skillfully manipulating these analog sticks, a human operator can achieve very precise control over a quadcopter.

Looking back at autonomous navigation, a software control system can replace the RC and supply MultiWii with roll, pitch, yaw and throttle programmatically. MultiWii doesn't care where the data comes from, it can control the motors as long as it is provided data on those four channels. While

this interface works naturally, control is potentially difficult because the skill of a human operator is now replaced with pure data gathered from the camera. While a computer may have faster reaction times, it also cannot look ahead and evaluate situations outside the narrow field of view of it's camera. Additionally, when the quadcopter executes a maneuver, it is no longer parallel to the ground. This causes the camera to look slightly more in the direction opposite of motion. Essentially the camera is always looking away from where the quadcopter is going. As previously mentioned, the solution to some of these problems was PID control.

PID control is a fairly universal and popular control loop in robotics. As visualized in **Figure XX**, the basic premise is to create a closed feedback loop where the error in the system dictates the response. In the context of a flying quadcopter, if the distance between quadcopter and target is large, an aggressive maneuver will be made towards the target. Likewise, if the distance is small, a gentle adjustment will be made towards the target. PID achieves this control through three different calculations. First, proportional (P) control attempts to minimize the error by providing an instantaneous response to the current error. The integral (I) control aims to correct long term error by aggregating the total error. This is useful in situations such as a persistent northerly wind. The quadcopter will consistently be north of the target until the error accumulates enough to offset the wind. Lastly, derivative (D) control handles sudden impulses in the system. For example, if a very sudden gust of wind hit the quadcopter, the derivative control would allow it to recover rapidly. Together, these three factors allow a system to effectively reduce error quickly. The power of PID control is that only the target location needs to specified. The necessary response and path to realize that location is handled completely algorithmically.

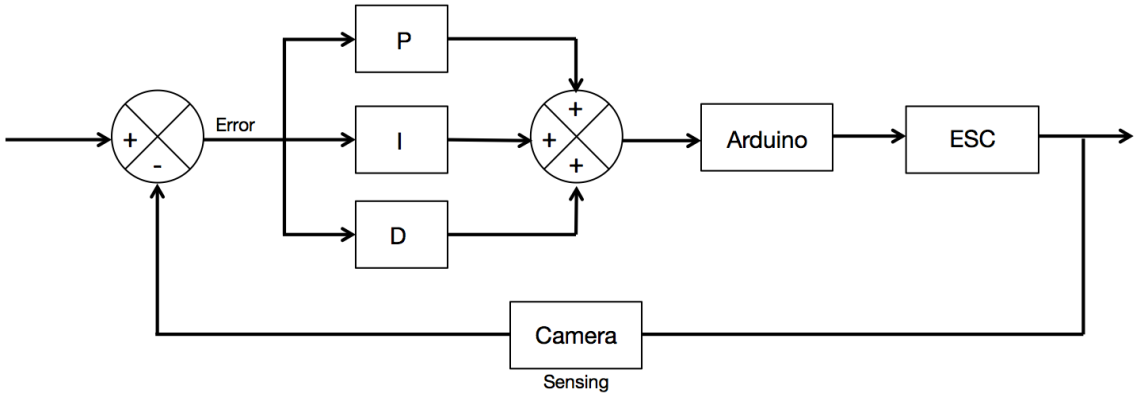

Figure 15: PID Feedback Loop

For this application, three separate PID controls were executed in parallel. Before any navigation could be completed, it was necessary for the quadcopter to maintain its height automatically. This was accomplished by experimentally determining the throttle at which the quadcopter would hover and maintain a steady state in the vertical axis. This value was highly dependent on battery charge, but slight errors in the steady state value were inconsequential once PID control was introduced.

For height control, the input to the PID system was the error in height. This error was calculated by subtracting the current height from the desired height. Using the equation derived in the early height experiments, the quadcopter's current height could be determined using only the size of the target. The desired height could be specified in real-time, but a default value of two meters was chosen because the height equation was most accurate in that neighborhood. Using this error, the PID control would add an offset to the steady state throttle value and a corresponding reaction could be seen in the behavior of the quadcopter. Once height control was implemented, the quadcopter could stabilize in an XY plane parallel to the ground. Navigating within this XY plane required two more PID channels to control the roll and pitch of the rotorcraft.

For control in these directions, a different error was used. This error corresponded to the distance from the center of the target to the center of the camera frame. Because the camera points straight down, the center of the image should be the point on the ground directly below the quadcopter. By comparing this point to the actual location of the target, an error can be calculated and inputted to the PID loop. A visualization of how the error is calculated can be seen in **Figure 16**. The control loop then allows the quadcopter to navigate directly above the target while maintaining its height.

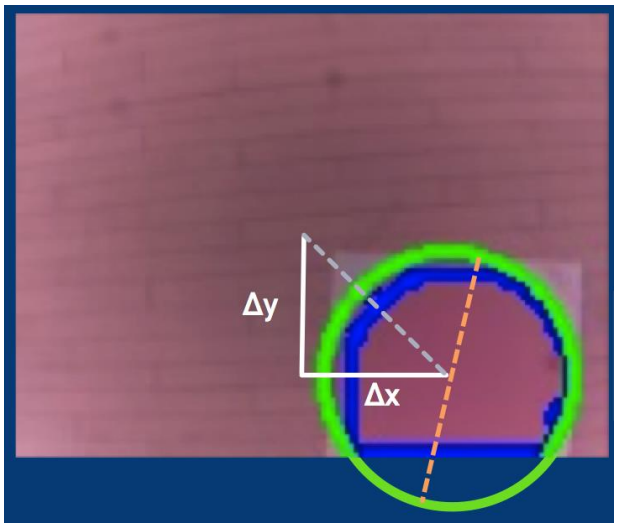

Figure 16: Visualization of Control Calculations

Performance Issues Encountered

There were several performance issues to overcome when testing PID control. Before any flight tests could be conducted, communication between the Raspberry Pi and Altera boards needed to be refined. The Raspberry Pi sent control instructions to the Altera board via a serial connection. However, if motor values were sent only once per main control loop, the motors would pulse and not operate smoothly. This was due to the fact that the image processing took too long and the motors required a faster sampling rate than the Raspberry Pi could provide. In order to handle this issue, the code was rewritten to be multithreaded. This allowed the code to transmit motor values at a very fast rate, even if these values were only updated once per control loop. Once race

conditions were properly handled using locks, this solution proved to be extremely effective.

Next, tweaking the PID constants proved to be difficult and time-consuming. The complex physics governing a rotorcraft in flight is difficult to model and thus simulation-based testing was impossible. Additionally, due to constraints in resources and time, a testing rig could not be build, so all testing needed to be performed during actual flight. This meant that when testing values, the quadcopter's stability could be compromised and crashes were inevitable. In order to overcome these roadblocks, more time was spent testing and any damage to the quadcopter was rapidly fixed using replaceable parts.

Another significant issue was the situation where the camera changed angles as the quadcopter maneuvered. This phenomenon caused the target to change locations in the camera frame, which in turn caused the quadcopter to overshoot and ultimately lose track of the target. This issue was worsened by the relatively slow sampling rate of the overloaded Raspberry Pi CPU. The solution to this issue was two part. First, the image feed was downsampled to be a fifth of its original size. This allowed the image processing stage to run much faster. Secondly, decreasing the PID constants reduced the response of the control system. This caused the quadcopter to move more slowly and kept the camera more parallel to the ground. Ultimately, extensive testing allowed both issues to be largely overcome.

Autonomous Landing

After successfully navigating to a location two meters above the target, it is still a non-trivial problem to land autonomously. Because limitations in the sensors available, there were several roadblocks to implementing autonomous landing. First, the quadcopter could not actually determine when it was on the ground. A simple bump sensor could have been used for this purpose, but time constraints prevented one from being obtained. Secondly, the height of the quadcopter could not be determined nor effectively controlled when the target was not in view of the camera. During landing, it is very difficult to keep the target visible. Additionally, at low altitudes, the target will actually fill the entire camera field of view, and once again height cannot be determined. These drawbacks necessitated a creative solution.

Three different ideas were implemented and tested. The most naive solution was to slowly descend using PID control and then simply cut motor power once the target filled the entire camera view. This solution failed because of lack of stability in flight. Environmental factors and small calculation errors would cause the quadcopter to drift slightly and often lose track of the target. Without vision of the target, the quadcopter could neither control its height with PID nor detect when to shut off its motors. Clearly a different solution was needed. A much more simplistic approach was then taken.

Because the steady-state throttle was already hardcoded into the code, the quadcopter should be able to descend slowly by using a throttle value just smaller than the steady state value. By maintaining this throttle value, the quadcopter would slowly lose height until it landed on the ground. This solution suffered both from not being able to detect the ground and also from the change in control physics at low altitudes due to the rotor airstream hitting the ground and causing turbulence. To solve the first issue, dead reckoning was chosen for simplicity. In essence, the control system would activate the landing procedure for a set amount of time and then cut motor power assuming the ground was reached during that time.

This solution is generally not very robust, however for the given environment, it could be very safely assumed that it would not take longer than five seconds for the quadcopter to land after beginning its landing procedure. This estimation was eventually reduced to 2.5 seconds during testing. If the time was accurate, it also solved the second problem of turbulence, because the motors would shut off anyways once the quadcopter reached the turbulent region. This solution was dependent on battery voltage, but given the constraints of the project and the lack of sensors, it proved entirely sufficient.

Testing

In order to properly test the quadcopter, a variety of approaches and tools were necessary. These tests were designed for different testing environments including both the outdoor quadrangle and an indoor gymnasium. It is obvious that the safest methods of testing are methods that do not require actual flight. Furthermore, when it is absolutely necessary to conduct flight tests, they must be done in a way that prioritizes safety. These two rules helped shape the how the quadcopter was tested.

Figure 17: Indoor Testing Environment

For image processing, the most conservative testing methods were observed. All image processing code featured debug modes that would display the output of the system on a monitor connected to the Raspberry Pi. This meant that the camera could be completely dismounted from the quadcopter during testing. In order to test the effects of in-flight motion on image quality, a similarly safe approach was

taken. Video footage would be captured while the quadcopter was operated by a human pilot. This video was then later analyzed using a tool that functioned almost identically to the actual image processing code. This tool was custom built to accept an image feed from a file and then analyze it using the same image processing techniques as the real control system. The advantages of this tool where that the user could pause or rewind the recording as needed. Once paused, the user could click areas of the image to extract data such as the HSV value of the selected pixels or information about any detected targets. Additionally, the algorithms could be updated in real time to test the performance and reliability of thresholding ranges without requiring more in-flight testing.

For basic implementations of motor control, in-flight testing was also unnecessary. By simply detaching the propellers from the motors, the quadcopter could be tested safely. Changes in motor speed could be monitored by listening to the sound of the motor. If more quantitative data was needed, an oscilloscope could be connected at the inputs to the ESC and the exact waveform could be monitored. This was sufficient to verify that the motors reacted correctly to changes in the motor signal sent from the Raspberry Pi. This allowed testing of all features of the control systems with the exception of the PID constants.

PID constants are scalars that when multiplied by the proportional, integral, and derivative error provides different outputs from the PID system. Baseline constants could be approximated by specifying the maximum response desired and then working backwards, but this approach is not guaranteed to provide optimal behavior during flight. As a result, these values were used simply to test if the PID was generally working.

As an example, pretend the control code is trying to stabilize at a height of two meters. If the quadcopter is manually held at a height of one meter (without its propellers attached) the motor will spin very quickly, attempting to gain altitude and achieve the desired height. Furthermore, the longer the quadcopter is kept below the set point, the faster the motors will spin. This is due to the integral component, which accumulates error. These basic tests were used to ensure that

each of the three PID channels was correctly implemented. However, in order to improve performance during flight, the PID constants needed to be fine-tuned during actual flight tests. In order to properly tweak the PID parameters, it was necessary to form safe testing procedures. At the lowest level, basic LED displays were used to signal when the target was in view or when the quadcopter was in autonomous mode. This was just a basic safety and convenience choice that allowed you to understand some basic information about the quadcopter's condition simply by looking at it. Aside from LEDs, there were three main testing sequences that were used to improve algorithm performance while minimizing the chance of damage to the quadcopter and its environment. The most basic and widely applicable test procedure was the "toggle test." This test was triggered by a Bluetooth command, as seen in Figure XX. Once the message was received, the quadcopter would enter an autonomous mode for a set duration of time. Because the duration of the test was predefined and fixed, the test would automatically end, even if Bluetooth connectivity failed. This was important because it allowed a human operator to safely recover and land the quadcopter, even if stability was lost during the test or other connection problems occurred. It is important to mention that the duration of the test could also be dynamically updated using additional Bluetooth commands. This allowed for more seamless testing as the duration could slowly be increased if performance was initially stable.

The next testing routine was known as a takeoff test. Once the Bluetooth start command was received, the quadcopter would slowly start increasing its throttle steadily. If the start command was sent a second time, it would then hold its current throttle and send back a Bluetooth message with that throttle value. Sending the start command a third and final time would then end the test and return control to the human operator. This test was primarily used to observe the takeoff behavior of the quadcopter. Additionally, it gave a rough estimate of the minimum throttle value necessary to achieve flight from the ground. However, it is important to note that flight physics behave differently at very low altitudes due to turbulence and other factors caused by the wind stream bouncing off the

ground. As a result, this test did not actually provide accurate information about the throttle necessary to achieve lift while actually in flight.

The final testing routine was the landing tests. This procedure once again could be initialized by a Bluetooth start command. The key difference is that there were actually three different methods for landing, which could be designated in the command send to start the routine. This was invaluable because it allowed for comparison of different algorithms in a real-world setting. Rather than designing a decision matrix to attempt to analyze the different methods in a lab, it became possible to just instruct the quadcopter to land then qualitatively evaluate each method's performance.

Aside from these three testing routines, over twenty other Bluetooth commands were implemented to control every aspect of flight and testing. The power in using Bluetooth communication for real-time updates is that multiple

Function	Command
Switch to Autonomous Control	auto
Switch to RC Control	rc
Exit	exit
Steady-State Display	ss
Steady-State Update Values	ss+ROLL+PITCH+YAW+THROTTLE
Take-off Test	to
Toggle Test	tt
Toggle Test (Pitch)	tt+p+VALUE+DURATION
Toggle Test (Roll)	tt+r+VALUE+DURATION
Record Video	rec
Stop Video	rec (while recording)
Drive Motors at Max Speed	hi
Drive Motors at Middle Speed	mi
Drive Motors at Minimum Speed	lo
Toggle PID Control for Roll	r
Toggle PID Control for Pitch	p
Toggle PID Control for Throttle	h
Toggle PID Control for All Channels	m
Change Desired Height	dh+##
Check if Target is Currently in View	t
Set PID Constants for Roll	x+Kp+Ki+Kd
Set PID Constants for Pitch	y+Kp+Ki+Kd
Set PID Constants for Throttle	h+Kp+Ki+Kd

Figure 18: Table of Bluetooth Commands

spent adjusting parameters at the lab and allows for better time management while at testing locations. This was especially important due to the limited availability of good testing locations.

Overall, by constructing a Bluetooth framework, it was possible to more efficiently test and adjust the behavior of the quadcopter. Additionally, by developing testing routines, it became possible to safely and properly test control algorithms without endangering the team members, the quadcopter, or the testing environment.

JUSTIFICATION

The growing market for unmanned vehicles and the utilization of more aerial modes of transportation for goods and packages calls for advances in quadcopter technology. This market revolves around the ability for quadcopters to behave intelligently. As technology improves, aerial vehicles will become more common in a variety of different consumer, military, and commercial applications. Quadcopters are especially versatile due to their cost effectiveness and ease of construction. These qualities are emphasized when viewed in terms of production value and the short turnaround time between creating a final design, manufacturing and finally marketing. This is especially relevant to current world markets, where increased accessibility, such as 3D printers, allows for almost anyone to rapidly prototype and construct products.

Intelligent flight control and navigation are required for any implementation of an autonomous quadcopter; the objectives that were achieved throughout this project demonstrated one solution for intelligent behavior. These results can be applied to other quadcopters in related fields. This determines that many of the key focuses and accomplishments of this project are applicable to different fields. In similar terms, the image processing capabilities are very modular in design. This allows any particular color-detecting feature to be reused for other various applications. The image processing techniques utilized by this project establish a framework for future implementations and features.

A major aspect of the project in terms of accessibility is its reproducibility and cost. The total cost of the quadcopter was kept at a minimal amount, even when accounting for any design changes needed to be completed throughout the entirety of the project timeline. Accounting the major costs on the main hardware components of the CIRUS board, Raspberry Pi, Raspberry Pi Camera Board, motors, propellers, quadcopter body skeleton, and minor circuit components and wiring, the overall cost was kept under $400. In this project, the quadcopter was constructed by hand without the use of fabrication machinery or other tools. It is

important to recognize that for commercial applications, different manufacturing processes would be used, so manufacturing costs could vary. This amount can be adjusted based on materials, design specifications and available facilities in different sectors of industry. Now, more than ever, it is relevant for a large jump in implementation of quadcopters, specifically due to the favorable cost-impact it exhibits.

RESULTS AND DISCUSSION

With the limited time of seven weeks to accomplish the given project goal and tasks, utilizing the time effectively was crucial. The allotted time gave perspective to the team on the most effective solutions for quadcopter control and autonomous flight with video capture of a specified target. These objectives were accomplished with individual focus as displayed on the corresponding Weekly Breakdown of the project and its progress.

Weekly Progress Breakdown

Week 1: Implemented edge detection and contouring

Week 2: Target detection with specific HSV ranges

Week 3: Height experimentation and redesign of the quadcopter body

Week 4: Control algorithms for height control

Week 5: PID calibration and Bluetooth capabilities

Week 6: Autonomous Roll and Pitch calibration with landing

Week 7: System collaboration and testing

Overall, the final objectives of the project were achieved over the course of the term. The quadcopter was able to reach stable-state conditions and utilize control algorithms for height control and navigation. Additionally, the image processing capabilities were realized through the detection of specified color-range objects in an environment. Once the target was detected and within a certain range in the camera's field of view, it would begin the sequence for autonomous landing.

Final Quadcopter Assembly

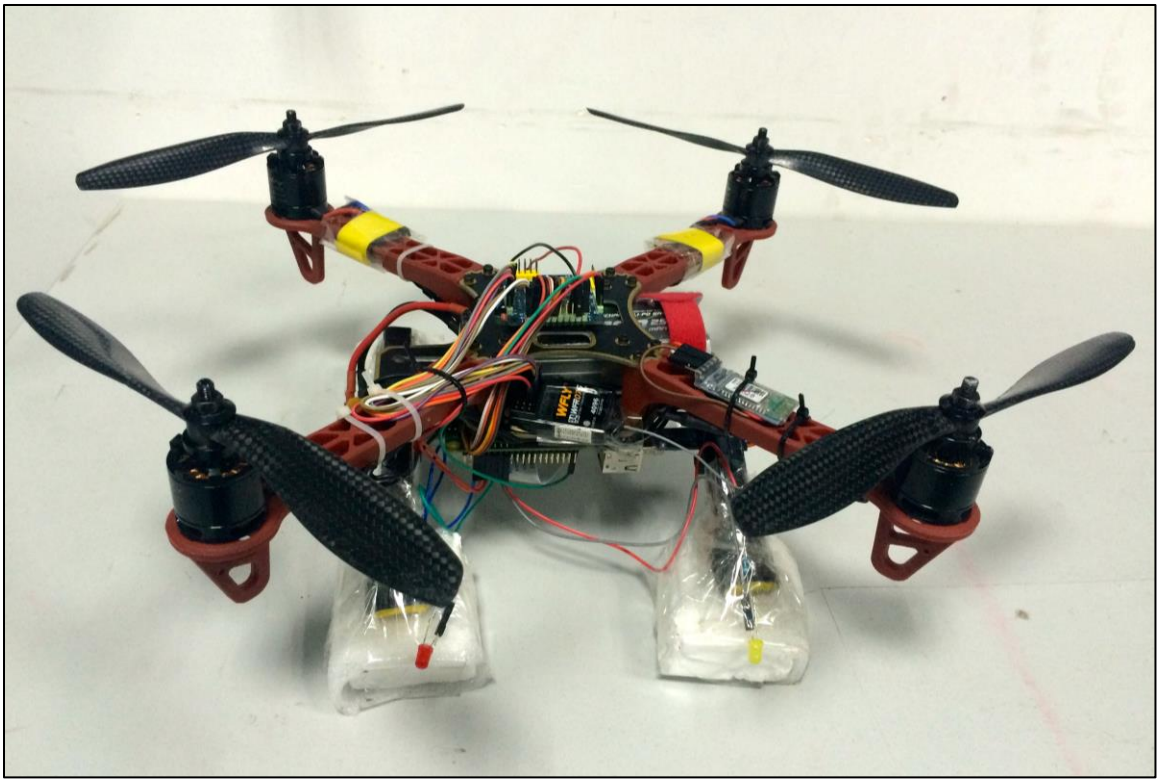

Figure 19: Quadcopter Assembly

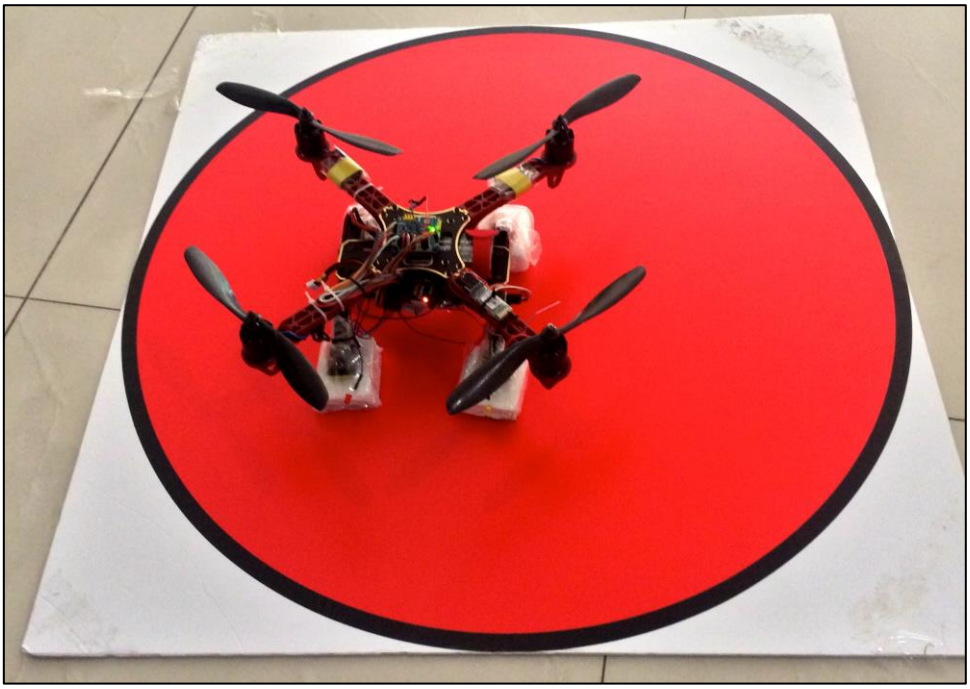

Figure 20: Quadcopter and Target

CONCLUSION AND RECOMMENDATIONS

Overall, the final results from the project yielded many successes. Despite the overall positive outcome of the project, there is still room for future research and improved methods which could build upon the foundation laid by this project Along with its successes, this project can offer several recommendations for future work, based on the team's experience.

When approaching a mechanical and software-based challenge, solutions are always dependent on the materials and systems utilized. Along these lines, a change in the hardware that was provided and utilized for the final quadcopter implementation could have allowed from more efficient and robust behaviors. Specifically, the dual-board approach for the quadcopter caused a difficulty when it was first implemented. This was caused due to communication constraints between the two boards. Each board had specific functionalities when it came to the control of the quadcopter. The Raspberry Pi board handled high-level control algorithms and then sent the results of these algorithms to the low-level CRIUS board, which, in turn, directly controlled the motors. This operation used the only serial port on the CRIUS board and thus restricted any other communication to or from the CRIUS.

The reason this was disadvantageous was that the CIRUS control board on the quadcopter came with a number of built-in and peripheral sensors that were essential for low-level stabilization. The board included an accelerometer, a gyroscope, an electric compass, and a barometer. This sensor data would have proved extremely useful to the control algorithms running on the Raspberry Pi board, however, the lack of additional serial ports made it difficult to transfer that sensor data to the Raspberry Pi from the CRIUS board. The gyroscope and accelerometer would have especially been vital, both for refining the accuracy of height measurements and for improving the control of the quadcopter. Additionally, this would have freed many parts of the control algorithm from dependency on camera data. In turn, this would have allowed for more stable and intelligent flight, even when the target was out of view of the camera.

However, based on the timeframe of the project and the provided hardware architecture, it was not possible to better integrate these sensors. A more drastic redesign of the system architecture would have taken far too long and rendered the project impossible in it's seven week timeframe.

Also, a number of hurdles were encountered with the camera used for the image processing aspects of the project. Most importantly, there were several different cameras used over the duration of the project. Due to the new camera hardware, the project timeline experienced minor stalls in progress, as the set-up and additional height testing for the new mounted cameras was required for proper functionality. These camera changes were caused by previous cameras' bulkiness on the quadcopter, the quality of the image, and the data transfer speed of the connection to the Raspberry Pi. The final camera used was the Raspberry-Pi Camera Board, which was easily integrated with the Raspberry-Pi and allowed for data transfer through a ribbon cable. However, the camera seemed to have a red tint and some desaturation to the camera view in certain conditions, which proved to be difficult when testing. This disrupted the reliability of the target detection in certain conditions and above certain heights, as the target created for detection was a highly saturated red circle.

Lastly, there were irregularities in the testing environment used throughout the project. Because the team lacked sufficient materials to construct a good testing rig for all but the most basic of tests, it was necessary to resort to testing locations around the SHU campus, which each had specific drawbacks and limitations. The team needed to test specific aspects of the quadcopter throughout the term, and having different constraints in the testing environments proved to not be an efficient process. Based on the available locations for testing and lab space, there were considerations that needed to be accounted for before any testing was possible. For example, indoor gymnasium space was available but only at very specific intervals and required the team to pay a fee to use the space. Eventually, the team was required to test both at indoor and outdoor locations based on availability. These various locations each had multiple constraints for height, weather conditions, and lighting differences. Having a

more consistent testing environment and schedule would have aided in achieving the project goals.

WORKS CITED

[1] Hoffmann, G.M.; Rajnarayan, D.G.; Waslander, S.L.; Dostal, D.; Jang, J.S.; Tomlin, C.J. (November 2004)

[2] Büchi, Roland (2011). *Fascination Quadrocopter*.

[3] Leishman, J.G. (2000). *Principles of Helicopter Aerodynamics*. New York, NY: Cambridge University Press.

[4] Hoffmann, G.M.; Rajnarayan, D.G.; Waslander, S.L.; Dostal, D.; Jang, J.S.; Tomlin, C.J. (November 2004).

[5] Hoffman, G.; Huang, H.; Waslander, S.L.; Tomlin, C.J. (20–23 August 2007).

[6] Lowensohn, Josh. "FAA considering a Fast Track for Businesses That Want to Use Drones." *The Verge*. N.p., 16 May 2014. Web. 12 Oct. 2014.

[7] "AR.Drone 2.0. Parrot New Wi-fi Quadricopter - AR.Drone.com - HD Camera - Civil Drone - Parrot." *AR.Drone 2.0.* Parrot, n.d. Web. 1 Oct. 2014.

[8] Brandon, Alan. "Control Your Own Augmented Reality Aerial Drone? There's an App for That." *Gizmag*. N.p., 6 Jan. 2010. Web. 13 Nov. 2014.

[9] “Aeryon Small Unmanned Aerial Systems." *Aerial Vehicle Systems*. Aeryon, n.d. Web. 3 Dec. 2014.

[10] "Amazon Drones: Amazon Unveils Futuristic Delivery Plan." *CBSNews*. CBS Interactive, n.d. Web. 09 Dec. 2014.

[11] McNeal, Gregory S. "Six Things You Should Know About Amazon's Drones." *Forbes*. Forbes Magazine, 11 July 2014. Web. 6 Nov. 2015.