



UNIVERSITY of HAWAII®  
MAUI COLLEGE

# USCG SEARCH & RESCUE DRONE

Engineering Technology Capstone Project

Malatone Bouasym

Advisor: Jung Park, Ph.D.

Spring 2015

## **DISCLAIMER**

The recommendations, specifications, planning and assembly of systems in this report should not be used without consulting an experienced engineer. All use of information and possible duplication of the project from this report, such as assembling and flying a multi-copter, may be dangerous and done so at your own risk. Please familiarize yourself with all safety precautions prior to any experimentation as well as rules and regulations set forth by the FAA.

## **ACKNOWLEDGEMENTS**

---

Firstly, I would like to thank my family and friends for their encouragement and support throughout the duration of my studies. Furthermore, I thank all those who gave valuable advice, comments and willingness to share ideas which ultimately made this project possible.

<b><u>Point of Contact</u></b>	<b><u>Association</u></b>
Andrea & Fabian	3DRobotics
Joey Andrews	University of Hawai'i – Maui
Tomáš Báča	Czech Technical University in Prague
Christopher Boykin, LT	USCG Base Honolulu, C4IT Department
Judith Connelly	USCG C4ISR & Special Missions Branch
Guy Cranfill	USCG Research & Development Center
Harry Donenfeld	Drones & Cameras Maui
Stephen Dunn, LT	USCG Research & Development Center
Ashley Labarr, OS1	USCG Sector Honolulu
Brent Massey, OS1	USCG Sector Hampton Roads Command Center
John Meier	University of Hawai'i – Maui
Andrew Niccolai, Ph.D.	USCG Research & Development Center
Jung Park, Ph.D.	University of Hawai'i – Maui
Dany Thivierge	Canada Drones
Chanintorn Tongjan	jnMechanics Co., Ltd. – Bangkok, Thailand
Jeffrey Yopez, Ph.D.	University of Hawai'i – Manoa
Andrew Zeitler	University of Hawai'i – Maui

## **ABSTRACT**

---

The purpose of this project is to assemble a multi-copter that will successfully execute the waypoints of a software-generated search pattern, therefore performing the task usually assigned to helicopters, small-boats and cutters. By designing a capable drone and implementing a software solution to effectively detect objects that represent persons in the water, launching multiple automated systems can solve obstacles that the Coast Guard encounters. This will further support the mission statement of Coast Guard search and rescue missions:

“To prevent death or injury to persons and loss or damage to property in the marine environment.”

- COMDTINST M16130.2F [1]

# TABLE OF CONTENTS

---

Acknowledgements

Abstract

1	Introduction.....	7
1.1	Background.....	7
1.2	Statement of Purpose .....	7
1.3	Objectives .....	8
1.3.1	Goals .....	8
1.3.2	Functions.....	8
1.3.3	Benefits .....	8
1.3.4	Features .....	8
2	System Design .....	9
2.1	Hardware Overview .....	9
2.1.1	X8-M Multi-copter Platform.....	9
2.1.2	LiPo Battery .....	10
2.1.3	FlySky RC.....	10
2.1.4	Pixhawk Autopilot .....	11
2.1.5	Radio Telemetry Kit 915MHz Wireless Module .....	12
2.1.6	Raspberry Pi B+ Model & Camera Module.....	12
2.1.7	Gimbal, GoPro, FPV .....	13
2.2	Software Overview .....	14
2.2.1	Phoenix RC Simulator .....	14
2.2.2	Mission Planner.....	15
2.2.3	Image Processing .....	16
3	System Integration .....	19
3.1	Concept .....	19
4	SAR Simulation Results .....	20
4.1	Autonomous Flight .....	20
4.2	Object detection .....	20
4.3	Piloting Difficulties.....	21
4.4	Excessive Vibration .....	22
4.5	Battery Life VS Search Pattern Length.....	22

5	Logistics.....	23
5.1	Cost Analysis .....	23
5.1.1	Table 1 .....	23
6	Conclusion .....	24
6.1	Restating the Issue .....	24
6.2	USCG Search & Rescue Drone .....	24
6.3	Future Innovations .....	24

Appendix A: List of Abbreviations

Appendix B: List of Figures

Appendix C: Color Detection Code

References

Biography

# 1 INTRODUCTION

---

## 1.1 BACKGROUND

As a military, multi-mission, and maritime service, the United States Coast Guard (USCG) is obligated to 11 missions set forth by U.S. law. One of those missions is known as Search and Rescue (SAR). When SAR cases are in effect, assets such as helicopters and small-boats are launched from a high-endurance cutter to execute particular search patterns generated by CG software known as the Search and Rescue Optimal Planning System (SAROPS) [3].

Upon entering the last known coordinates of a victim in distress, SAROPS generates a search pattern. This pattern consists of waypoints that are calculated by an algorithm based on the weather and sea state. These waypoints are relayed to pilots and coxswains so that they can enter the information in to the navigational unit. After all this, the helicopter and small-boats are finally launched.

Logistically, it costs the Coast Guard a great deal of time, money, and manpower to launch such assets. This also affects asset longevity, thus hindering the limited amount of CG resources. The entire process is very inefficient, but the mission remains that lives must be saved and distress calls shall never go unanswered.

## 1.2 STATEMENT OF PURPOSE

An unmanned aerial vehicle (UAV) is an airborne automobile flown either manually or automatically from a ground control unit. Currently, vertical take-off and landing (VTOL) multirotor vehicles have an advantage over existing UAVs due to their fixed-pitch propellers and cost-efficiency to build. This means the entire aircraft is controlled solely by the varying the speed of its propellers [23].

Multi-copters are VTOL capable systems and their nomenclature depends on the amount of motors onboard the platform. Four motors would be called a quadcopter; six motors is a hexacopter; eight motors is an octocopter. Manually flown multi-copters keep their nomenclature, however with autonomous flight capabilities, they become known as a drone. Though classic helicopters have more maneuverability, their flight parts are more critical and costly. Furthermore, helicopters generate vast vibrations which would disturb any cameras and sensors, thus making multi-copters fit better for experimentation.

Currently, the Coast Guard does not make any official use of UAV/UAS/drone capabilities specifically for SAR yet, however research does continue. “The Coast Guard is preparing to employ UAS to augment the service’s land- and cutter-based aircraft, and expand the surveillance range of surface assets, such as the national security cutter.”[24] The purpose of this project will be to assemble a multi-copter that will successfully execute the waypoints of a SAROP-generated search pattern. In turn, the drone will perform the task usually assigned to helicopters and small-boats.

By choosing the best design for these objectives and integrating programmable software to detect objects in the water, a simulation for purpose of concept can bring dramatic changes to SAR missions. This will also preserve time, money, manpower, and assets, making the execution of a SAR case more efficient than ever.

### **1.3 OBJECTIVES**

Much planning must go into the steps needed to be taken to accomplish the objectives in this project. My initial thought was to first look at all the mistakes made by hobbyists who have crashed their multi-copters so I could raise questions for myself throughout the progress of the project. What is the best design? What is the purpose of each design? Will I be able to learn everything and then gain the skills to integrate them all into one whole working system? Here are the milestones I set for myself to make this project as realistic and successful as I could have planned in two semesters.

#### **1.3.1 Goals**

- Choose best design for mission
- Assemble multi-copter
- Self-train for piloting w/HubsanX4 prior to multi-copter manual flight trials
- Gain familiarity with flight controller software and RC
- Program drone to execute autonomous flight
- Develop image processing software w/Raspberry Pi
- Simulate a SAR case using multi-copter as one of the ‘assets’

#### **1.3.2 Functions**

- Automated flight based on GPS coordinates and waypoint input
- OpenCV object detection capability using color segmentation
- Wireless communications between multi-copter, ground control and RPi

#### **1.3.3 Benefits**

- Advancement from this project can proceed to an Automated SAR Case Performance
- Automated flight system quicker than training users for manual piloting
- Flexible and open source system design to fit with various operations & coding

#### **1.3.4 Features**

- Uses gimbal stabilizer with GoPro for first person view and recording capabilities
- RPi camera system in concert with computer vision system software for detection
- Pixhawk flight controller software highly customizable

## 2 SYSTEM DESIGN

---

### 2.1 HARDWARE OVERVIEW

There are various ways to acquire a multi-copter. The easiest and quickest method is to purchase a fully assembled, ready-to-fly, multi-copter, at hundreds or even thousands of dollars for a high-quality system. The most cost-effective way is to just build one from scratch. My skill level fell in between these two categories. I had to consider the time-consumption of this project and the fact that this was going to be the first multi-copter I would ever assemble. I initially looked at designing one from scratch. On the contrary, my external mentor, Tomáš Báča, recommended I focus more on flight training and software engineering as this can take up to two years of research and experiment alone.

In a time-sensitive environment, my research concluded with taking the open source design of a DIY Quad Kit [4] and converting it to replicate that of the X8-M [5] multi-copter sold by 3DRobotics, which usually costs \$5,400 on their website. I chose this style for many reasons, as it complemented my mission needs. For example, the ‘X’ style frame versus a ‘+’ style frame serves better purpose for the camera view. An extra set of motors and propellers also leaves me with confidence that I would not crash my drone or lose it if any of those devices were to fail in-flight. My approach thus made this system customizable to my project requirements and more cost efficient since I was not paying for a ready-to-fly multi-copter.

#### 2.1.1 X8-M Multi-copter Platform

As shown in Figure 1, the DIY Quad Kit came with the blue and black colored arms for the frame to ease orientation during flight. The body plates are made to house the Pixhawk autopilot card, the power distribution board, the electronic speed controllers (ESCs), u-blox GPS with compass and all loose wire that had room to be tucked in. To convert the system to mock the X8-M in Figure 2, I had to purchase four extra motors, four extra propellers and four extra ESCs.



*Figure 1: DIY Quad Kit*



*Figure 2: X8-M RTF Kit*

The assembly instructions were provided on the 3DRobotics website [7] with very straightforward instructions listing all tools needed and amount of hardware designated to each part. The instructions are to be followed with precision to include, in this order, the motor assembly, power system wiring, body plate assembly, Pixhawk plate assembly, leg assembly, calibration, and lastly,

the propeller assembly. It is very important to note that propellers are not mounted until calibration of ESCs is done on every single one. Calibrating with propellers attached can be very dangerous.

### 2.1.2 LiPo Battery

Most multi-copters use a lithium polymer (LiPo) battery to power the equipment onboard. I find this useful for the main reason that they abundantly distribute power. The downside is that they are very sensitive and can only take in voltage slowly. In other words, quick power draw versus a slow charge. When charging LiPo batteries, I would recommend a slow charge at one amp versus the option of four or six amps. The battery I used had a minimum capacity of 10000mAh. Its configuration is 14.8 volts with four cells [26]. For the power distribution board that came with the DIY Quad Kit, I chose this particular battery since it had enough power for my payload capacity and the XT90 connector (Figure 3) matched to that of the power distribution board. My battery also came with a fire-proof packet to charge the battery in. I would recommend that if there are multiple batteries in possession, to keep an ammo can or some type of fire-proof container to store the batteries in. From my research, LiPo batteries are highly flammable. I have watched videos [22] where the battery will catch on fire, so having a method of containment will be of great use in the future.



Figure 3: LiPo Battery w/XT90 Connector

### 2.1.3 FlySky RC

The remote controller (RC) for this project is called FlySky FS-TH9X 9CH 2.4G Transmitter. It contains nine channels with a radio frequency range of 2.40-2.48GHz. It takes a 12V battery, which I happened to possess a rechargeable 12V battery to power this unit. Otherwise, eight 1.5V AA batteries can be used. Keep in mind that the RC cannot work until the transmitter and receiver are bound together. I watch a tutorial video [8] to perform this process. Figure 4 on the following page displays what the function of each switch represents.

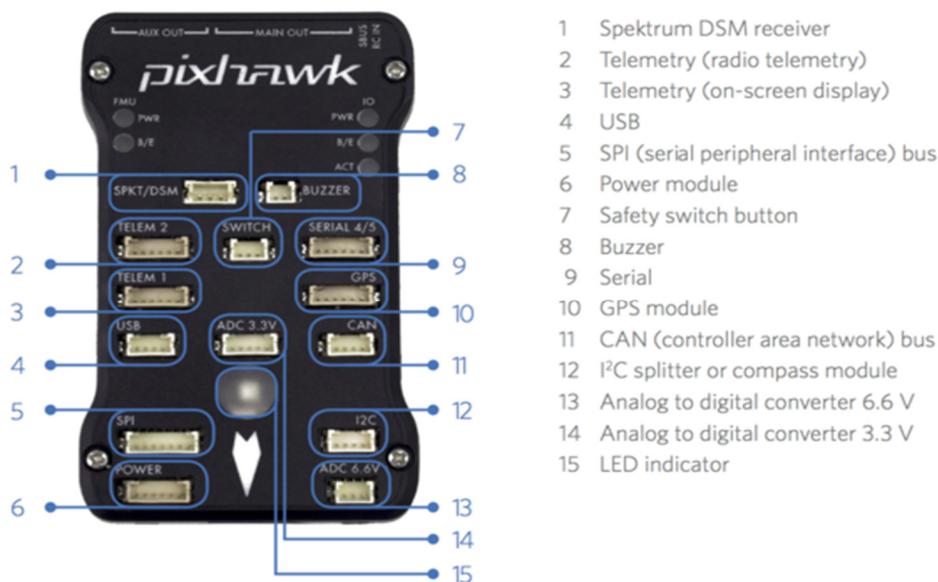
Manual flight with the RC is done by four independent signals. The throttle, which controls the thrust of all the motors; the elevator, which controls the forward motion (also known as pitch); the aileron, which controls the sideways motion (also known as roll); the rudder, which controls the rotation along the vertical axis (also known as yaw).



Figure 4: FlySky FS-TH9X

## 2.1.4 Pixhawk Autopilot

Pixhawk v2.4.5 [9] is one of the newer autopilots competing with others like the APM 2.5. It's a very useful piece of device due to the included sensors: gyro, accelerometer/magnetometer and barometer. There is always room for safety precautions and the Pixhawk has an option for attaching a buzzer to it other than just LED notifications. It can communicate via I2C, SPI, 2x CAN or USB. All of these features make it a very versatile system. There are even possibilities of connecting it to the RaspberryPi computer or Arduino so they can perform in sync with one another. In this project, its main purpose is to serve as the autonomous waypoint pilot through open source software via MAVLink. Figure 5 shows what the purpose of each port is, however, it is not just plug and play yet.



*Figure 5: Pixhawk v2.4.5*

To start with, the Pixhawk needs to be coupled with the free open source software called Mission Planner (for Windows) [17]. There are other free software programs for other platforms of choice. I initially plugged it straight into my laptop via USB but can later be used for piloting from the laptop by telemetry through a 915MHz radio. A quick start guide for the Pixhawk can be found on the Ardupilot website [10].

### **2.1.5 Radio Telemetry Kit 915MHz Wireless Module**

With the Pixhawk flight controller mounted, the motors calibrated and working, all sensors installed, and the RC bounded, the next step is to ensure the ground control can communicate with the multi-copter. To do this, a radio is needed to uses telemetry communications. The kit I decided to go with was very small in size so it was also light weight. The range was about one mile, which worked well for simulation purposes, however, a stronger, more expensive radio can always be purchased for longer ranges. This radio communicates with the Mission Planner software through MAVLink protocol framing and status reporting. Though mine came with a standard vertical antenna, I have read that the range can extend to several miles with an omni-directional antenna. A bi-directional amplifier can also be implemented for increase of range operability and this device contains open source firmware. The versatility of all chosen hardware makes this entire system highly upgradeable. The kit I purchased also came with a connector that goes directly to an Android phone, so I am also able to control the multi-copter and view the parameters from my Samsung Galaxy phone.



*Figure 6: Radio Telemetry Kit 915MHz Wireless Module*

### **2.1.6 Raspberry Pi B+ Model & Camera Module**

In need of an image processing platform, I decided to go with the RaspberryPi (RPi). For one, I had no experience with it prior to this project, and two, it was very cost-efficient; I purchased an RPi+Camera combination on Amazon.com for \$40. This was only for convenience that it came in a package; I have also tried using a Logitech webcam on the RPi through one of the USB ports and with a little different coding, it still works just the same. Traditionally, the RPi does not come with a portable power supply. I zip-tied a portable phone charger that was light and small to the frame of my multi-copter and attached the RPi with Velcro to the battery housing. This gave it a perfect spot for the camera to see directly underneath the multi-copter. The RPi itself also takes a

few steps to get started. I began by going to the RaspberryPi website [12] to download an image to mount to the computer. I chose the Raspbian – Debian Wheezy distribution as it is commonly used amongst programmers. On the same website at a different location are also the instructions on installing the camera module [13].



*Figure 7: RaspberryPi B+ Model w/Camera Module*

### **2.1.7 Gimbal, GoPro, FPV**

The GoPro Hero 3 was only convenient to me because I owned one prior to this project so I did not include this in the costs analysis of my project. However, I have purchased a second camera for personal use that can also be used for this project in place of the GoPro at a more feasible price. It is called the AFUNTA SJ4000 [25] which I purchased from Amazon for \$70. I wanted to use the GoPro as a means of first-person-view (FPV). This way I am able to see where the multi-copter is flying when it gets to high altitudes and long distances. The purpose of the gimbal is to keep the camera in the same, stable position at all times, regardless of the maneuvers the multi-copter is making. The system I purchased for these functions was the Tarot T-2D V2 GoPro Brushless Gimbal [30]. Instructions on assembling the gimbal is also on the Ardupilot website [15].



*Figure 8: Tarot T-2D V2 GoPro Brushless Gimbal*

## 2.2 SOFTWARE OVERVIEW

The multi-copter assembly took a full semester to fabricate, to include parts delivery and testing before assembly. The software was also just as time-consuming, in my case, since I had to learn a few things from scratch. I first went with a flight simulator that was recommended a few times through blogs I read concerning manual flight training. A multi-copter is not easy to fly, so I wanted to prepare myself in preventing accidents and ensuring safety as much as possible.

The first thing I did was purchase a Hubsan x4 [16] on Amazon (free w/rebate so I did not include this in the cost analysis either). This is a good unit to practice with because it has no sensors so all controls are completely from the operator. I recommend flight of a small quadcopter and practice with the flight simulator before moving on to flying a large, more expensive multi-copter.

Mission Planner was very easy to work with, having tutorials documented and on video easily searchable on the web. The rest of the software highly involved work with the Raspberry Pi and OpenCV in which case I used yet another tutorial to familiarize myself with computer vision programming [21].

### 2.2.1 Phoenix RC Simulator

The flight simulator I used was called the Phoenix RC Simulator Version 5.0. This was a very useful system as it allowed me to deal with realistic situations for flight training. I can say that I crashed the simulated quadcopter several times; after using this program, I have yet to crash my actual multi-copter assembled for this project.



Figure 9: Phoenix R/C v5.0.v Flying Field

## 2.2.2 Mission Planner

The Mission Planner software is an open source made for Windows platform. I downloaded mine on the Ardupilot website [17]. For other platforms, I believe it is called APM Planner for Linux and MAC. With Mission Planner and the 915MHz radio telemetry, I am able to view all parameters of the multi-copter. I can see its altitude, GPS, compass, pitch, roll, etc. Mission Planner also allowed me to set up the controls on the RC so that I can choose which switches call for what functions. For example, I have Channel 7 set up to perform an automatic landing. With the instructions from the Ardupilot website [18], I followed the steps in performing procedures to be followed before the first flight ever performed upon assembling the multi-copter. In this order, I had to set up the flight modes, pre-arm safety check, and then arming and disarming. These are steps that cannot be skipped. There are ways to override safety checks, however, as a beginner, I would not recommend this.

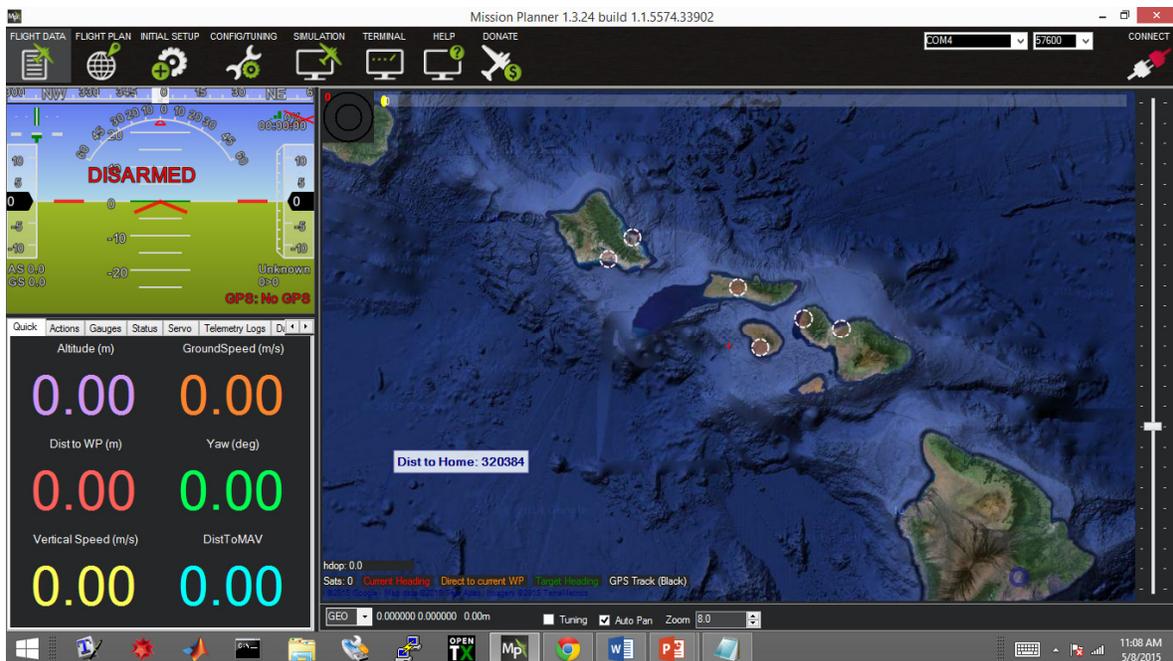


Figure 10: Mission Planner 1.3.24

My next purpose with Mission Planner was to use it for autonomous flight. In the flight plan tab of the software, there is space to enter waypoints. This is where the SAROPS-generated waypoints are entered and the speed, altitude and drone maneuvers can be adjusted here such as loiter, take-off, or return to launch (RTL). As you can see in Figure 11 and Figure 12 the comparison between what a SAROPS pattern looks like and my input of those same waypoints in Mission Planner.

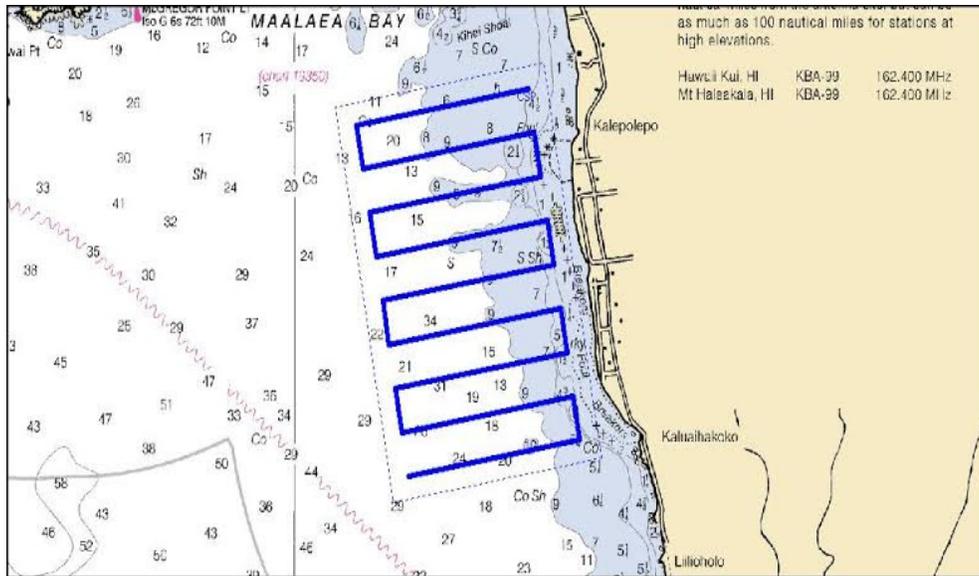


Figure 11: SAROPS Pattern Generated by OSI Brent Massey

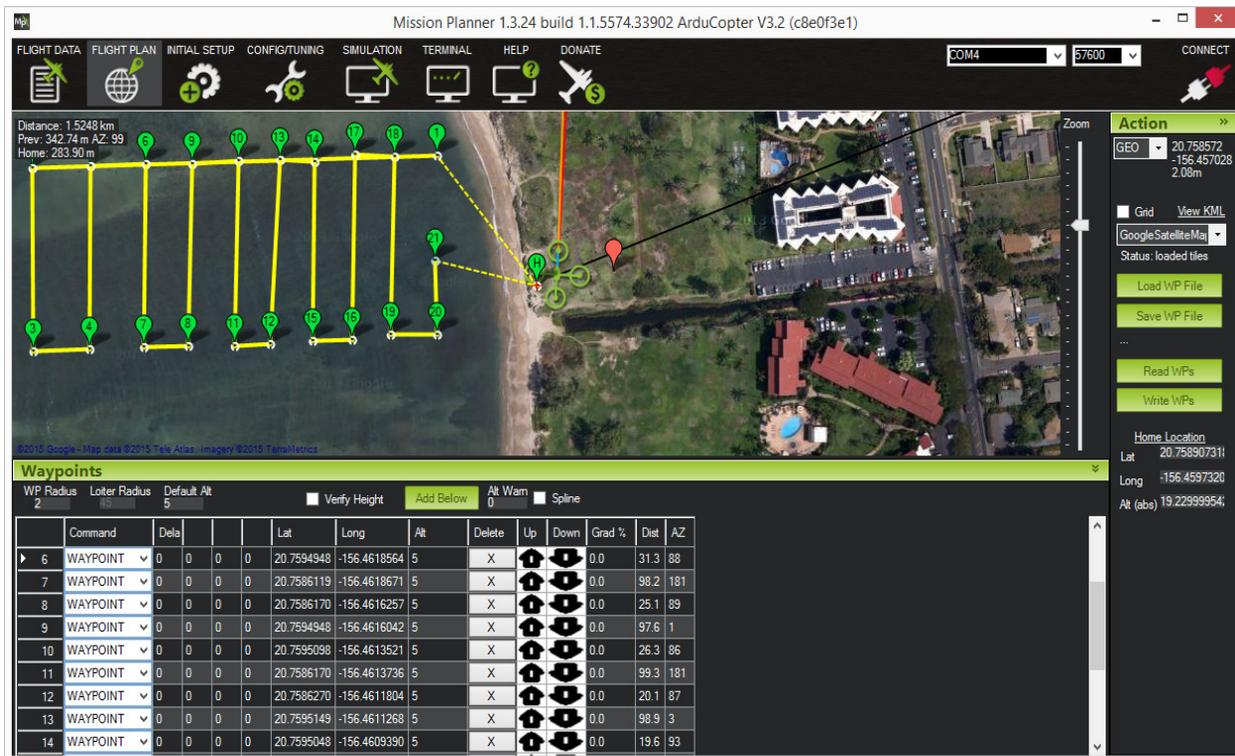


Figure 12: Mission Planner Flight Plan - Kihei

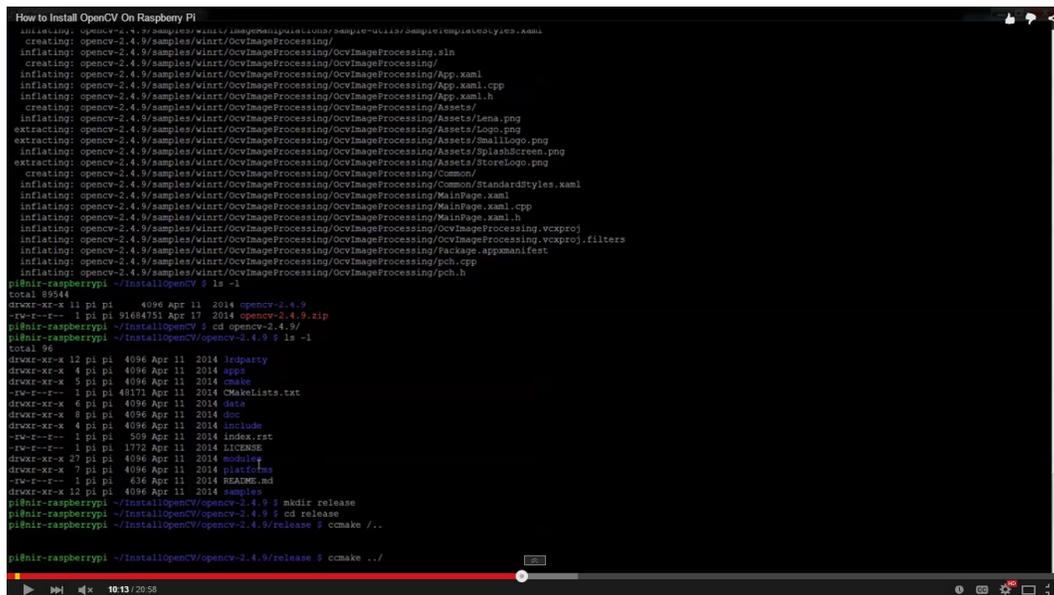
## 2.2.3 Image Processing

This process describes my method of approach for an object detection simulation. I found many tutorials that showed step-by-step instructions on how to detect even multiple faces of persons. This was very impressive yet complex and out of the realm of my current knowledge. I took a step

back and made my own milestone to simply just detecting colors. Since the Coast Guard requires that all vessels have life vests onboard, and those are usually orange in color, my goal was to detect objects of the color orange. This alone gave me much to research about and I learned a vast amount of information encompassing computer vision technology.

### 2.2.3.1 OpenCV

OpenCV contains libraries and software written to perform algorithms that would otherwise be extremely time consuming to calculate for each and every function to be called. It helps the program not just see, but understand the world. In a sense, it could be considered a branch off of remote sensing technology. OpenCV gave me the benefit of not having to learn about bit depth, buffer management, Eigen values, among other complicated computer vision parameters. The only downside to this was installing OpenCV on the RPi. It took me two trials. The first trial failed at seven hours into compiling, while the second trial was a success, at roughly 10 hours. I followed this tutorial to install OpenCV [19].



```
How to Install OpenCV On Raspberry Pi
~/installopencv:
  creating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing.sln
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing.sln
  creating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/App.xaml
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/App.xaml.cpp
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/App.xaml.h
  creating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Assets/
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Assets/Isma.png
  extracting: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Assets/Logo.png
  extracting: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Assets/SmallLogo.png
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Assets/SplashScreen.png
  extracting: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Assets/StoreLogo.png
  creating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Common/
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Common/StandardStyles.xaml
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/MainPage.xaml
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/MainPage.xaml.cpp
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/MainPage.xaml.h
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/OcvImageProcessing.vcxproj
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/OcvImageProcessing.vcxproj.filters
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/Package.appxmanifest
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/pch.cpp
  inflating: opencv-2.4.9/samples/wint/OcvImageProcessing/OcvImageProcessing/pch.h
pi@raspberrypi:~/installopencv$ ls -l
total 8944
drwxr-xr-x 11 pi pi 4096 Apr 11 2014 opencv-2.4.9
-rw-r--r-- 1 pi pi 91684351 Apr 17 2014 opencv-2.4.9.zip
pi@raspberrypi:~/installopencv$ cd opencv-2.4.9/
pi@raspberrypi:~/installopencv/opencv-2.4.9$ ls -l
total 96
drwxr-xr-x 12 pi pi 4096 Apr 11 2014 .
drwxr-xr-x 4 pi pi 4096 Apr 11 2014 ..
drwxr-xr-x 5 pi pi 4096 Apr 11 2014 cmake
-rw-r--r-- 1 pi pi 48171 Apr 11 2014 CMakeLists.txt
drwxr-xr-x 6 pi pi 4096 Apr 11 2014 data
drwxr-xr-x 8 pi pi 4096 Apr 11 2014 doc
drwxr-xr-x 4 pi pi 4096 Apr 11 2014 include
-rw-r--r-- 1 pi pi 589 Apr 11 2014 index.rst
-rw-r--r-- 1 pi pi 1772 Apr 11 2014 LICENSE
drwxr-xr-x 27 pi pi 4096 Apr 11 2014 modules
drwxr-xr-x 7 pi pi 4096 Apr 11 2014 platform
-rw-r--r-- 1 pi pi 636 Apr 11 2014 README.md
drwxr-xr-x 12 pi pi 4096 Apr 11 2014 samples
pi@raspberrypi:~/installopencv/opencv-2.4.9$ mkdir release
pi@raspberrypi:~/installopencv/opencv-2.4.9$ cd release
pi@raspberrypi:~/installopencv/opencv-2.4.9/release$ cmake ../
pi@raspberrypi:~/installopencv/opencv-2.4.9/release$ cmake ..
```

Figure 13: Installing OpenCV on RPi

### 2.2.3.2 RGB VS. HSV

To write the image processing code, I first needed to convert images from reading Red, Green, Blue (RGB) values to reading Hue, Saturation, Value (HSV) parameters. The advantage to this is that HSV is able to distinguish colors in an environment of various luminance [21]. This is something RGB does not offer the ability to do. Being that I want to simulate this in the ocean, luminance, reflections and highlights can be major factors, thus HSV is used. It is also simpler because now I can call for one value that represents many different shades of one color, thus easing the process of coding. In Figure 14 and Figure 15 below, you can see the differences between RGB and HSV color extractions.

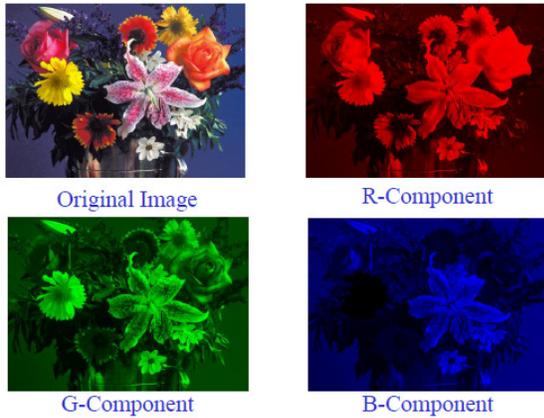


Figure 14: Example of RGB [21]



Figure 15: Example of HSV [21]

### 2.2.3.3 Blob Analysis

I used the idea of color segmentation [21] to analyze the type of object I was looking for. In this process, the program will take away pixels that are far from the target color while keeping those that are close to matching the targeted color. Then it provides a second screen to completely eliminate between light and dark. So if I were to see a set of orange colored marbles on a bed of blue blankets, the screen would show a black background with white circles. This concept then detects just blobs of a chosen color. As simple as it sounds, I believe the concept here works well when trying to detect objects in the ocean. The majority of the ocean is of a solid color and temperature so anything deviant from that would ideally be detected.

### 2.2.3.4 Image Moments

To create a ‘tracking’ effect, I used zero & first order moments [20] to calculate the area and position of the object. Then I layered each current frame on top of the prior frame to show the objects previous location and replaced the object area and position with the color yellow. The entire code took me about two months to figure out after extensive research. Learning the math and physics of image moments alone was quite time consuming and took repetition. I also used a tutorial by Ai Shack [20] on tracking colored objects in OpenCV as a thorough guide. The code can be viewed in Appendix D with a more thorough explanation of what each line is doing. This code was a modification of Ai Shack’s tutorial on tracking colored objects [20] in OpenCV in concert with the SimpleCV and RaspberryPi tutorial [21].

### 3 SYSTEM INTEGRATION

---

#### 3.1 CONCEPT

Now that all the hardware is assembled and the software is working, the challenge was to actually incorporate all systems together. This became another learning curve for me, to be able to see what is happening on the RPi all while running an image processing program and dealing with latency since it was attached to the multi-copter. My solution was to connect to the RPi by means of remote SSH access using the free software VNC in concert with PuTTY client, so that I can run OpenCV.



Figure 16: RPi + OpenCV + SimpleCV

Figure 17 shows the entire concept of how everything is communicating. Now that I could remote in to the RPi, my next issue was operating the entire system in a location that did not provide wireless internet access to route everything together. My closest, least time-consuming, and free solution was the iPhone6. I used it as a WiFi hotspot for my laptop and RPi to connect to and that is how they became interconnected. A tutorial for remote access via PuTTY and VNC can be found on the RaspberryPi website [14].



Figure 17: Complete Interconnection of all Systems

## 4 SAR SIMULATION RESULTS

---

### 4.1 AUTONOMOUS FLIGHT

With the weight of the gimbal, battery, and all parts connected to the multi-copter, it weighed approximately 6.5 pounds. My first flight trial, the multi-copter was able to be flown for 16 minutes without interruption. The only reason I stopped was because of the low battery warning. At 99% charge, the LiPo battery contained 16.8V. After 16 minutes of flight, I measured 14.1V. This was quite consistent throughout my flight trials since I only usually conducted them during calm days where the multi-copter did not have to fight the wind. With the weight of everything attached to the system, it decreased my flight time down to 14 minutes. In Figure 18 below is a snapshot of an autonomous flight I conducted behind my residency at Waipuilani Park.



*Figure 18: Autonomous Flight Trial w/Mission Planner – Waipuilani Park*

### 4.2 OBJECT DETECTION

I attempted to detect the orange boogie board in Figure 18 with the RPi image processing program. This was not a success because the connection between the RPi, the iPhone 6 hotspot and my

laptop was not reliable at a range further than 15 feet. It worked in close range, however the extreme latency made the system not serve its purpose. It is concluded that the RPi is not a high-performance piece of equipment, although, great for experimentation. My next plan was to perform this with my home network. I left the RPi remaining attached to the drone and placed an orange ball on a blue blanket. I wanted to simulate as close as possible to a life jacket in the ocean. Figure 19 shows a successful attempt to detect and track the ball.

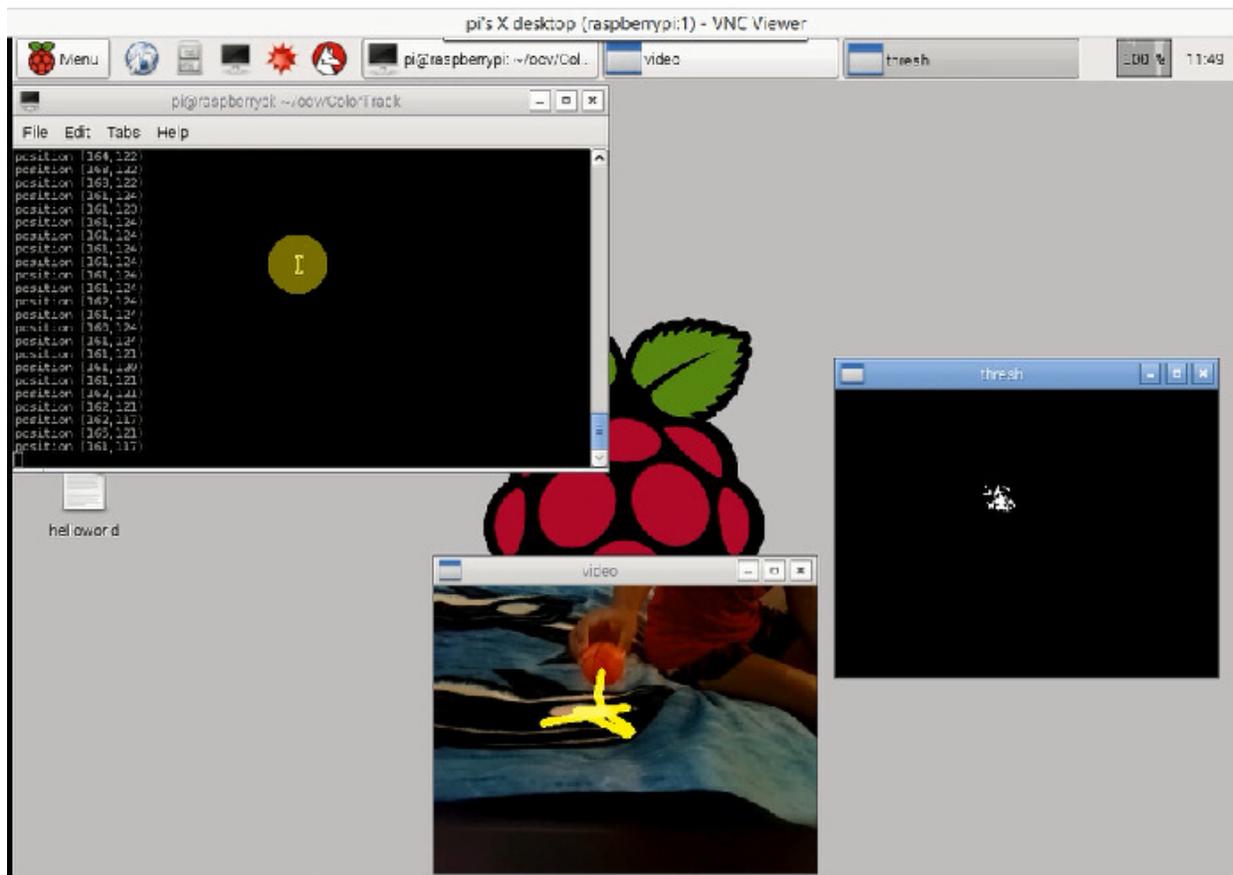


Figure 19: Remote in to RPi - Object Detect & Track

### 4.3 PILOTING DIFFICULTIES

Practicing and self-training for manual flight with the Hubsan x4 gave me great understanding to how multi-copters work. This is something I highly recommend, especially to those who cannot afford to crash something so costly. A major advantage I had was communicating with experienced drone pilots who gave me tips, tricks and advice as to how to deal with certain troubles most people go through in the process of learning to fly a drone. I had flight trials in windy weather just to test how much I had to over compensate for. Autonomous flight from the Mission Planner made everything run smoothly and it is actually the recommended method of flying drones. The only problem here was that the drone cannot sense obstacles. The human watching the FPV monitor would be considered the sensor in this case, because you can always stop the auto pilot mid-flight and take control.

#### **4.4 EXCESSIVE VIBRATION**

After using the FPV to record my flight trials, I noticed excessive vibration. I researched to find that this most likely came from the motors creating turbulence on the arms and possible propeller imbalances. I kept spare propellers around just to be cautious. I replaced the propellers and installed foam at the base of each of the motors. This helped drastically and now I can catch pristine video with my drone.

#### **4.5 BATTERY LIFE VS SEARCH PATTERN LENGTH**

The big question was how realistic this concept could be to apply to an actual SAR case. With more advanced equipment, I believe this kind of issue of low battery life could be mitigated. If anything, decreasing the speed so motor is not in need of high power could be one route or simply considering a larger battery if funds were available. However, with a larger battery, the system requires a heavier payload, and the costs continue to increase.

## 5 LOGISTICS

---

### 5.1 COST ANALYSIS

#### 5.1.1 Table 1

Part	Quantity	Cost /ea.
[26] <a href="#">DIY Quad Kit</a> (10% student discount)	1	\$495
[27] <a href="#">Motor</a>	4	\$18.00
[28] <a href="#">ESC</a>	4	\$18.00
[29] <a href="#">Propeller</a>	4	\$2.00
[30] <a href="#">LiPo Battery</a>	1	\$67.76
[31] <a href="#">FlySky FS-TH9X 9CH 2.4G Transmitter</a>	1	\$89.49
[32] <a href="#">Radio Telemetry Kit 915Mhz Wireless Module</a>	1	\$18.49
[33] <a href="#">RaspberryPi B+ &amp; Camera Module Combo</a>	1	\$39.95
[34] <a href="#">Tarot T-2D 2 Axis Camera Brushless Gimbal</a>	1	\$68.99
[35] <a href="#">FPV Transmitter</a>	1	\$29.99
<b>Total</b>	19 items	<b>\$961.67</b>

*\*\*Typical Ready-To-Fly assembled drones are \$1000 and the X8-M was \$5400*



*Figure 20: Actual Drone Built for this Project*

## **6 CONCLUSION**

---

### **6.1 RESTATING THE ISSUE**

In accordance with the COMDTINST 7310.1P, Coast Guard Reimbursable Standard Rates 2015 [2], each hour of any given SAR mission will cost \$14,237 per hour to operate a Coast Guard aircraft; \$8,083 per hour to operate a Coast Guard helicopter; \$6,290 per hour to operate a Coast Guard large ship; \$937 per hour to operate a Coast Guard small boat. That comes to a total of \$30,000 an hour spent on a SAR case when all assets are involved.

### **6.2 USCG SEARCH & RESCUE DRONE**

\$30,000 versus the near \$1,000 I spent on this project. Nearly \$1,000 I spent to assemble a multi-copter that replicates that of the X8-M which has a retail value of \$5,400. Imagine the possibilities of being able to launch 10 to 20 drones at once to conduct these same SAR patterns that the CG assets must do with actual man power and limited man hours. This project was a success, however major latency of FPS does not make it operational yet. This whole system proves a major concept and I hope to continue advancements with it. Improved, high-performance technology that can replace all of the more cost-conscious products I used would make this system of definite use. The amount of time it would take to launch drones could possibly save more lives than the service has ever thought possible.

### **6.3 FUTURE INNOVATIONS**

I am eager to continue advancing this project. I plan to actually use my drone voluntarily to help the community. Future additions to the system would be an infrared camera for night time operations; LEDs on the forward and aft of the drone so I can easily navigate in the dark; advanced image processing to implement face recognition technology – this could possibly be used for missing person searches; sensors to avoid obstacles, especially at night. With everything in mind, I can continue to make it a full service search and rescue drone.

## APPENDIX A: LIST OF ABBREVIATIONS

---

APM	ArduPilot Mega
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance & Reconnaissance
C4IT	Command, Control, Communications, Computers and Information Technology
CAN	Controller Area Network
COMDTINST	Commandant, United States Coast Guard Instruction
DIY	Do It Yourself
ESC	Electronic Speed Controller
FAA	Federal Aviation Administration
FPS	Frames Per Second
FPV	First Person View
GHz	GigaHertz
GPS	Global Positioning System
HSV	Hue, Saturation, Value
I2C	Inter-Integrated Circuit
LED	Light Emitting Diode
LiPo	Lithium Polymer
MAVLink	Micro Air Vehicle Link
MHz	MegaHertz
RC	Remote Control
RGB	Red, Green, Blue
RPi	RaspberryPi
RTF	Ready To Fly
RTL	Return To Launch
SAR	Search and Rescue

SAROPS	Search and Rescue Optimal Planning System
SPI	Service Provider Interface
SSH	Secure SHell
UAS	Unmanned Aircraft System
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
USCG	United States Coast Guard
VNC	Virtual Network Computing
VTOL	Vertical Take-Off & Landing

## APPENDIX B: LIST OF FIGURES

---

Figure 1	DIY Quad Kit
Figure 2	X8-M RTF Kit
Figure 3	LiPo Battery w/XT90 Connector
Figure 4	FlySky FS-TH9X
Figure 5	Pixhawk v2.4.5
Figure 6	Radio Telemetry Kit 915MHz Wireless Module
Figure 7	RaspberryPi B+ Model w/Camera Module
Figure 8	Tarot T-2D V2 GoPro Brushless Gimbal
Figure 9	Phoenix R/C v5.0.v Flying Field
Figure 10	Mission Planner 1.3.24
Figure 11	SAROPS Pattern Generated by OS1 Brent Massey
Figure 12	Mission Planner Flight Plan - Kihei
Figure 13	Installing OpenCV on RPi
Figure 14	Example of RGB
Figure 15	Example of HSV
Figure 16	RPi + OpenCV + SimpleCV
Figure 17	Complete Interconnection of all Systems
Figure 18	Autonomous Flight Trial w/Mission Planner – Waipuilani Park
Figure 19	Remote in to RPi - Object Detect & Track
Figure 20	Actual Drone Built for this Project

## APPENDIX C: COLOR DETECTION CODE

---

```
// ColorDetect.cpp
//
#include <stdio.h>
#include <opencv2/opencv.hpp>
#include "RaspiCamCV.h"
IplImage* GetThresholdedImage(IplImage* img,CvScalar color, float percent)
{
    //RGB to HSV conversion
    IplImage* imgHSV = cvCreateImage(cvGetSize(img), 8, 3);
    cvCvtColor(img, imgHSV, CV_BGR2HSV);
    IplImage* imgThreshed = cvCreateImage(cvGetSize(img), 8, 1);
    CvScalar top=
cvScalar(color.val[0]*(1+percent),color.val[1]*(1+percent),color.val[2]*(1+percent));
    CvScalar bottom=cvScalar(color.val[0]*(1-percent),color.val[1]*(1-
percent),color.val[2]*(1-percent));
    for(int i;i<3;i++)
    {
        if(bottom.val[i]<0)
            bottom.val[i]=0;
        if(bottom.val[i]>255)
            bottom.val[i]=255;
        if(top.val[i]<0)
            top.val[i]=0;
        if(top.val[i]>255)
            top.val[i]=255;
    }
    //cvInRangeS(imgHSV, cvScalar(112, 100, 100), cvScalar(124, 255, 255), imgThreshed);
    cvInRangeS(imgHSV, bottom, top, imgThreshed);
}
```

```

        cvReleaseImage(&imgHSV);
        return imgThreshed;
    }
int main()
{
    //Open camera to start streaming live video
    RASPIVID_CONFIG config;
    config.width=320;
    config.height=240;
    config.bitrate=0;    // zero: leave as default
    config.framerate=0;
    config.monochrome=0;

    RaspiCamCvCapture * capture = (RaspiCamCvCapture *)
raspiCamCvCreateCameraCapture2(0, &config);

    int cal_win_size=20;
    CvScalar ColorCalMin=cvScalar(112, 100, 100);
    CvScalar ColorCalMax=cvScalar(0,0,0);
    bool calibrating=false;

    //Two windows: one for real-time view; the other is black and white to isolate object
    cvNamedWindow("video");
    cvNamedWindow("thresh");

    // This is the frame layering technique to simulate tracking
    IplImage* imgScribble = NULL;
    while(true)
    {
        IplImage* frame = 0;
        IplImage* imgOrangeThresh;

```

```

frame = raspiCamCvQueryFrame(capture);
//This is so the operator can manually set the threshold
if(calibrating)
{
    cvSetImageROI(frame,cvRect(config.width/2 - cal_win_size/2,
                                config.height/2 - cal_win_size/2,
                                cal_win_size,
                                cal_win_size));
    IplImage *img_cal = cvCreateImage(cvGetSize(frame),frame->depth,
                                       frame->nChannels);
    cvCopy(frame,img_cal,NULL);
    cvResetImageROI(frame);
    cvLine(frame,cvPoint(config.width/2 - cal_win_size/2, config.height/2 -
cal_win_size/2 ),
            cvPoint(config.width/2 + cal_win_size/2, config.height/2 -
cal_win_size/2 ),CV_RGB(255,0,0),1,8,0);
    cvLine(frame,cvPoint(config.width/2 + cal_win_size/2, config.height/2 -
cal_win_size/2 ),
            cvPoint(config.width/2 + cal_win_size/2, config.height/2 +
cal_win_size/2 ),CV_RGB(255,0,0),1,8,0);
    cvLine(frame,cvPoint(config.width/2 + cal_win_size/2, config.height/2 +
cal_win_size/2 ),
            cvPoint(config.width/2 - cal_win_size/2, config.height/2 +
cal_win_size/2 ),CV_RGB(255,0,0),1,8,0);
    cvLine(frame,cvPoint(config.width/2 - cal_win_size/2, config.height/2 +
cal_win_size/2 ),
            cvPoint(config.width/2 - cal_win_size/2, config.height/2 -
cal_win_size/2 ),CV_RGB(255,0,0),1,8,0);
    /*
    cvNamedWindow("calibration",CV_WINDOW_AUTOSIZE);
    cvResizeWindow("calibration",200,200);

```

```

cvShowImage("calibration",img_cal);
*/
IplImage* imgHSVCal = cvCreateImage(cvGetSize(img_cal), 8, 3);
cvCvtColor(img_cal, imgHSVCal, CV_BGR2HSV);
cv::vector<cv::Mat> channels;
cv::split(imgHSVCal,channels);
CvScalar a = cv::mean(channels[0]);
CvScalar b = cv::mean(channels[1]);
CvScalar c = cv::mean(channels[2]);
ColorCalMin = cvScalar(a.val[0],b.val[0],c.val[0]);
cvReleaseImage(&imgHSVCal);
cvReleaseImage(&img_cal);
}else
{
CvMoments *moments;
//If this is the initial moment, it must be started
if(imgScribble == NULL)
{
imgScribble = cvCreateImage(cvGetSize(frame), 8, 3);
}
//Color segmentation - holds the orange colors and blacks out everything
else
imgOrangeThresh = GetThresholdedImage(frame,ColorCalMin,0.20);
//Zero and first moment algorithms for object area and position
moments = (CvMoments*)malloc(sizeof(CvMoments));
cvMoments(imgOrangeThresh, moments, 1);
double moment10 = cvGetSpatialMoment(moments, 1, 0);
double moment01 = cvGetSpatialMoment(moments, 0, 1);
double area = cvGetCentralMoment(moments, 0, 0);

```

```

//Holds last known object position to create frame layers and tracking
static int posX = 0;
static int posY = 0;
int lastX = posX;
int lastY = posY;
posX = moment10/area;
posY = moment01/area;
{
    //This will leave track marks for the objects last known position
    cvLine(imgScribble, cvPoint(posX, posY), cvPoint(lastX, lastY),
cvScalar(0,255,255), 5);
}
//Combining the frames and track marks into the same live view feed
cvAdd(frame, imgScribble, frame);
cvShowImage("thresh", imgOrangeThresh);
cvReleaseImage(&imgOrangeThresh);
delete moments;
}
cvShowImage("video", frame);
//Key strike to quite program
int c = cvWaitKey(10);
if(c=='q')
{
    break;
}
if(c=='c')
{
    calibrating=!calibrating;
}

```

```
        }  
    }  
    //Exit the camera so it is available for use by other programs  
    raspiCamCvReleaseCapture(&capture);  
    return 0;  
}
```

## REFERENCES

---

- [1] Commandant Instruction M16130.2F. U. S. Coast Guard Addendum to the United States National Search and Rescue Supplement (NSS) to the International Aeronautical and Maritime Search and Rescue Manual (IAMSAR)  
[https://www.uscg.mil/hq/cg5/cg534/manuals/COMDTINST M16130.2F.pdf](https://www.uscg.mil/hq/cg5/cg534/manuals/COMDTINST_M16130.2F.pdf)
- [2] Commandant Instruction 7310.1P. Coast Guard Reimbursable Standard Rates  
[http://www.uscg.mil/directives/ci/7000-7999/CI\\_7310\\_1P.pdf](http://www.uscg.mil/directives/ci/7000-7999/CI_7310_1P.pdf)
- [3] Search and Rescue Optimal Planning System (SAROPS)  
<https://www.uscg.mil/hq/cg5/cg534/SARfactsInfo/SAROPSInforSheet.pdf>
- [4] DIY Quad Kit  
[https://store.3drobotics.com/products/diy-quad-kit?taxon\\_id=32](https://store.3drobotics.com/products/diy-quad-kit?taxon_id=32)
- [5] X8-M  
[https://store.3drobotics.com/products/x8-m?taxon\\_id=32](https://store.3drobotics.com/products/x8-m?taxon_id=32)
- [6] 3DRobotics  
[http://3drobotics.com/diy-quad-kit/?\\_ga=1.50258854.148655665.1413535538](http://3drobotics.com/diy-quad-kit/?_ga=1.50258854.148655665.1413535538)
- [7] 3DRobotics. “DIY Quad Kit Build Manual”  
<http://3drobotics.com/wp-content/uploads/2014/05/3DR-DIY-Quad-Build-Manual-vA.pdf>
- [8] "RCHAH. Binding process for FlySky or Turnigy 9x radio system."  
<https://www.youtube.com/watch?v=D-1Pp6eHHEQ>
- [9] Pixhawk v2.4.5  
<http://copter.ardupilot.com/wiki/common-autopilots/common-pixhawk-overview/>
- [10] 3DRobotics. “Pixhawk Quick Start Guide”  
<http://3drobotics.com/wp-content/uploads/2014/03/pixhawk-manual-rev7.pdf>
- [11] RaspberryPi  
<https://www.raspberrypi.org/downloads/>
- [12] RaspberryPi. “Installing Operating System Images”  
<https://www.raspberrypi.org/documentation/installation/installing-images/README.md>
- [13] RaspberryPi. “Connecting the Camera Module”  
<https://www.raspberrypi.org/documentation/usage/camera/>
- [14] RaspberryPi. “Connecting to a Pi over VNC Using Windows”.  
<https://www.raspberrypi.org/documentation/remote-access/vnc/windows.md>
- [15] Tarot Gimbal Assembly  
<http://copter.ardupilot.com/wiki/common-optional-hardware/common-cameras-and-gimbals/common-tarot-gimbal/>

- [16] Hubsan x4  
[http://www.hubsan.com/productinfo\\_14.html](http://www.hubsan.com/productinfo_14.html)
- [17] Mission Planner  
<http://ardupilot.com/downloads/>
- [18] Ardupilot First Flight Procedures  
<http://copter.ardupilot.com/wiki/flying-arducopter/>
- [19] Installing OpenCV  
[https://www.youtube.com/watch?v=\\_DMqGTsEo30](https://www.youtube.com/watch?v=_DMqGTsEo30)
- [20] Ai Schack  
<http://www.aishack.in/>
- [21] Cuauhtemoc Carbajal. “Computer Vision using SimpleCV and the RaspberryPi”  
[http://www.researchgate.net/publications.PublicPostFileLoader.html?id=54aada77d039b1666d8b45a5&key=5077bcd7-30c9-4be9-a926-939b7f23f85e](http://www.researchgate.net/publications/PublicPostFileLoader.html?id=54aada77d039b1666d8b45a5&key=5077bcd7-30c9-4be9-a926-939b7f23f85e)
- [22] “LiPo Battery Fire”  
<https://www.youtube.com/watch?v=QjKW3KUz5uo>
- [23] Celine Teuliere, Laurent Eck, and Eric Marchand. Chasing a Moving Target from a Flying UAV. *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference*. IEEE, 2011.
- [24] US Coast Guard Research and Development Center. Unmanned Aircraft System.  
<http://www.uscg.mil/acquisition/uas/default.asp>
- [25] AFUNTA SJ4000  
[http://www.amazon.com/dp/B00KX1YSOA/ref=pe\\_385040\\_121528360\\_TE\\_dp\\_1](http://www.amazon.com/dp/B00KX1YSOA/ref=pe_385040_121528360_TE_dp_1)

**Purchases** (*NOTE: Not all products may remain at the same price nor may any companies give the same discounts I received*)

- [26] DIY Quad Kit  
[https://store.3drobotics.com/products/diy-quad-kit?taxon\\_id=32](https://store.3drobotics.com/products/diy-quad-kit?taxon_id=32)
- [27] Motor  
[http://store.jdrones.com/AC2830\\_358\\_Motors\\_p/ac2830358.htm](http://store.jdrones.com/AC2830_358_Motors_p/ac2830358.htm)
- [28] ESC  
[http://store.jdrones.com/jD\\_UltraPWM\\_ESC\\_20\\_AMP\\_p/jdultra20amp.htm](http://store.jdrones.com/jD_UltraPWM_ESC_20_AMP_p/jdultra20amp.htm)
- [29] Propeller  
[http://store.jdrones.com/product\\_p/ac1045bl.htm](http://store.jdrones.com/product_p/ac1045bl.htm)
- [30] LiPo Battery

[http://www.hobbyking.com/hobbyking/store/\\_\\_64438\\_\\_Multistar\\_High\\_Capacity\\_4S\\_10000mAh\\_Multi\\_Rotor\\_Lipo\\_Pack\\_US\\_Warehouse\\_.html](http://www.hobbyking.com/hobbyking/store/__64438__Multistar_High_Capacity_4S_10000mAh_Multi_Rotor_Lipo_Pack_US_Warehouse_.html)

- [31] FlySky FS-TH9X 9CH 2.4G  
[http://www.tmart.com/FlySky-FS-TH9X-9CH-2.4G-Transmitter-for-RC-Helicopter-Airplane-Mode-2\\_p264894.html?](http://www.tmart.com/FlySky-FS-TH9X-9CH-2.4G-Transmitter-for-RC-Helicopter-Airplane-Mode-2_p264894.html?)
- [32] Telemetry Kit 915Mhz  
<http://www.ebay.com/itm/915Mhz-100MW-Telemetry-Radio-Module-3DR-3DRobotics-for-APM2-5-2-6-Pixhawk2-4-6/121539912659?>
- [33] RPi+Camera  
<http://www.amazon.com/Raspberry-5MP-Camera-Board-Module/dp/B00E1GGE40>
- [34] Tarot T-2D Gimbal  
<http://www.ebay.com/itm/like/161465734123?lpid=82&chn=ps>
- [35] FPV Transmitter  
[http://www.banggood.com/Eachine-Light-L250-5\\_8G-250mW-VTX-FPV-Transmitter-For-Gopro-3-p-933197.html?currency=USD&createTmp=1&utm\\_source=google&utm\\_medium=shopping&utm\\_content=saul&utm\\_campaign=Rc-Quad-us&gclid=CjwKEAjwvbGqBRCs3eH4o5C74CYSJAB3TODsDgZ-5geQsMsKICNhGOIBqIBAyNjT-ypn0-evnTBNbxoCYEHw\\_wcB](http://www.banggood.com/Eachine-Light-L250-5_8G-250mW-VTX-FPV-Transmitter-For-Gopro-3-p-933197.html?currency=USD&createTmp=1&utm_source=google&utm_medium=shopping&utm_content=saul&utm_campaign=Rc-Quad-us&gclid=CjwKEAjwvbGqBRCs3eH4o5C74CYSJAB3TODsDgZ-5geQsMsKICNhGOIBqIBAyNjT-ypn0-evnTBNbxoCYEHw_wcB)

## ***BIOGRAPHY***

Malatone Bouasym was born in Grand Island, Nebraska. She enlisted in the United States Coast Guard in 2004 and has been serving on active duty for over ten years. Her most recent work has been aboard Coast Guard cutters rated as an Electronics Technician Supervisor in the Operations Department.

Malatone received her Associate of Applied Science degree in Electronics Communications from Thomas Edison State College and is now a senior at the University of Hawai'i – Maui College. She is expected to graduate in May 2015 with a Bachelor of Applied Science in Engineering Technology. After graduation, Malatone will continue on to Alameda, California to work in the C4IT Department in support of the Coast Guard Rescue 21 program.

During her time at Maui College, Malatone has become a member of the Institute of Electrical and Electronics Engineers (IEEE), UH Maui College Veterans Club, IEEE Women in Engineering, and the Phi Theta Kappa Honor Society. She has been awarded the President's Silver Volunteer Service Award for 500+ hours of volunteer work including Toys for Tots Campaign, Rescue Mission soup kitchens, Salvation Army, Partnership in Education, playground renovation in Palau, Coast Guard Law Enforcement demonstrations in Majuro.

Malatone's goal is to become a USCG Chief Warrant Officer in the Electronics Specialty to apply her subject matter expertise to operational missions. She plans on continuing her career in the Coast Guard until retirement.