UNIVERSITY OF
KWAZULU-NATAL

INYUVESI
YAKWAZULU-NATALI

DEPARTMENT OF MECHANICAL ENGINEERING

Software Integration for Human Detection in Mining UAV Systems

Sibonelo Motepe

BSc. Engineering (Electrical)

Supervisor: Dr Riaan Stopforth

Durban, May 2013

Submitted in fulfillment of the academic requirements for the degree of Masters of Science in Mechanical Engineering, College of Engineering and Science at the University of Kwa-Zulu Natal.

I, Dr. Riaan Stopforth, as the candidate's supervisor I have approved this dissertation for submission.

Signed: _____

DECLARATION

I, Sibonelo Motepe, declare that

1. The research reported in this dissertation, except where otherwise indicated, is my original research.
2. This dissertation has not been submitted for any degree or examination at any other university.
3. This dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
    a. Their words have been re-written but the general information attributed to them has been referenced
    b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References sections.

Signed: _____

To my late mother, thank you for loving and supporting me throughout my life.

Acknowledgements

I would like to thank my supervisor Dr. Riaan Stopforth for his support throughout the research, support from the research proposal phase and his encouragement to do MSc. He has been very inspirational and a source of knowledge. Dr Stopforth is always available for feedback and to share ideas at almost every hour of the day.

I would also like to thank my whole family for their support throughout my studies. Their undying support helped me continue even when conditions were not favourable. I would especially like to thank my fiancée, Miss. Nthabiseng Mohlakoana, who has stood by me through my studies. Her support and encouragement helped me to keep going.

I would like to thank God who has been with me from birth. Without the abilities he has granted me I would have not been able to complete this research.

I would also like to thank the CSIR and UKZN for funding this research.

Abstract

Mining is one of the main economic sectors in South Africa. Mining activity contains hazards such collapsing of structures, presence of dangerous gases, accidental explosions and fires. Even though most of these hazards are identified and minimized sometimes accidents occur. These accidents lead to human injuries, direct fatalities and fatalities resulting from delays in victims getting medical attention as a result of delays in search and rescue missions. The rescue missions in underground mines present challenges where rescuers are not sure which locations are victims in, what the area conditions like in the rescue path. A quad rotor unmanned aerial vehicle (UAV) for search and rescue missions is presented. The UAV is controlled from a remote location over Wi-Fi. The communication allows data relay to the ground control station. The communication system is tested on the university's Wi-Fi network. The UAV also contains a vision system that contains a human detection algorithm to give an indication of human presence to rescuers. The human detection system is based on Haar-Cascade classifiers. The model developed was found to have a false alarm rate of 5×10^{-3} % after training. The model was further tested on streaming data and the overall average positive human detection was found to be 97 %. In the same tests overall false average detection was found to be 2.5 %. The video feed is streamed from the UAV to the ground station (GS) and the flight control instructions are sent to the UAV from the GS via Wi-Fi.

Table of Contents

Table of Figures

List of Acronyms

APM	ArduPilotMega
ESC	Electronic Speed Controller
FAR	False Alarm Rate
GS	Ground Station
GUI	Graphical User Interface
IMU	Inertial Measurement Unit
PDB	Power Distribution Board
PI	Proportional Integral
UAV	Unmanned Aerial Vehicle

1. Introduction

South Africa is one of the major mining countries in the world. Mining accidents have caused multiple deaths in the past and in recent times. Search and rescue missions can become dangerous with rescuers having to go into unknown environments that are unstable. An underground mine collapsed on 21 January 1960, in Clydesdale Colliery at Coalbrook, resulting in 400 miners that died. Rescuers that went down the main shaft were blocked by collapsed ground and methane gas. Drilling had begun 1.5 km and 3.6 km away from where workers were suspected to be, with no success in finding them. This incident illustrates the importance of communication underground [1]. At Goldfields' KDC West mine a fire broke out on 30 June 2012. This led to 5 deaths and 14 people being hospitalized [2]. As of the 11 July 2012, 63 deaths had occurred already in mining incidents within that year [3]. These dangerous conditions during mine accidents and rescuers going into locations where the presence of victims has not been determined can delay the rescue mission. Search and rescue missions need to be made safer for human rescuers and also quicker for victims to be attended to. In this research unmanned aerial vehicle human detection vision system is developed for search and rescue applications in underground mines. The control methodology from a remote ground station is developed.

1.1. Mechatronics

Mechatronics is a field of engineering that integrates Mechanical, Electronics and Computer Engineering. One concept of Mechatronics Engineering is shown in Figure 1-1 [4].

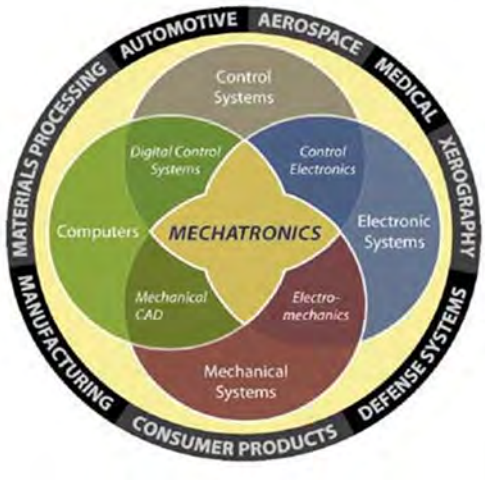

Figure 1-1: Graphical Representation of Systems Integrated Into Mechatronics

This integration has many applications in today's world, including applications in high voltage line inspection robots, urban search and rescue robots, line inspection robots, etc. This report presents the literature review and background research done relating to this mechatronics research topic. The report also presents the research break down. The research project topic is software integration for human detection in mining UAV systems. The research focused on UAV application for search and rescue missions in mine environments.

1.2. Literature Survey

Unmanned Aerial Vehicles (UAVs) are remotely piloted aircraft that can carry cameras, sensors, communications equipment or other payloads. These UAVs have been used in reconnaissance and intelligence-gathering applications since the 1950s and more applications are being researched. UAVs have multiple applications in recognition, environmental observation, maritime surveillance and mine removal activities [5].

UAVs have a low payload capacity and poor fuel/energy efficiency, but offer two main advantages when it comes to the manoeuvrability required in search and rescue applications. These UAVs have the ability to land almost anywhere and have the ability to fly at low altitudes and hover over disaster scenes. Unmanned helicopters are in use today and are usually flown using a remote interface, by either remote control or tele-operation. The aircraft is visible to the operator with remote control, while in tele-operation the aircraft is not visible to the operator. Here sensors and cameras on the aircraft send information to an operator on the ground through an interface, which is usually displayed on a computer screen or a similar interface. This interface usually requires a team to run and also requires a large amount of bandwidth to relay data to and from the aircraft in optimum time [6].

The current technological developments make it possible to have a combination of advanced sensor suites and ultra-fast and reliable hardware in a vehicle to achieve autonomous control. This characteristic can enable a UAV to have strong computational power, but there is still a challenge with passing and selecting relevant information and analysis of this information [6].

South Africa plays a major role in the international mining fraternity. An advantage of accurate location information is the capability to allow for robot navigation and guidance in the underground mining environment, which is applicable to safety monitoring robots, rescue robots and even production robots. Opencast mines use the global positioning system (GPS) to

obtain location information [7]. The problem with GPS is that the device must have a clear line of sight between itself and the satellite. Thus GPS systems cannot be used in confined spaces such as underground mines [8].

The unavailability of GPS technology in underground mining has given rise to research the possible alternatives. These attempts to exploit new sensors that measure inter-nodal ranges, signal strengths, acceleration or angles for location as well as research high sensitivity algorithms for signal acquisition and tracking in harsh environments [9]. The combination or integration of these sensors has also been investigated [7].

1.2.1. Unmanned Air Vehicles (UAVs)

A UAV has been described, in the introduction, as a remotely piloted aircraft that can carry cameras, sensors, communications equipment or other payloads. A UAV must have enough power to carry the payload. The payload depends on the blades, motors and battery. In order to maneuver in confined spaces for search and rescue missions the UAVs have to be properly sized. The UAV that will be dealt with in this research is a quad rotor flying vehicle. Research work has been done on the development of such vehicles, the stability and control [10]. Mathematical models, algorithms and computer simulations have been created and tested in this process [11]. Vijay Kumar illustrates the stability level that has been achieved on quad rotor UAVs [12]. Progress in the UAV control and stability have are evidence to the amount research contribution that have been made towards UAV stability and control. A lot of UAVs are available off the shelf for hobbyists and research purposes. These UAVs can be purchased and used to research other areas that can improve the implementation of UAVs in different situations, including, but not limited to, search and rescue, mining, surveillance, etc. [13].

A quad rotor UAV has four rotors with 2 vertically opposite rotor/blades acting in the same direction and the other pair in the opposite direction as shown Figure 1-2. These act to control the motion of the UAV (i.e. the yaw, roll and pitch). The quad rotor motor rotation configuration is shown in Figure 1-2 (a), where it is observed that the opposite rotors must rotate in the same direction [14]. To control the yaw of the quad rotor copter the speed of one opposite motor pair must be increased to be higher than of the other pair. Each pair leads to yaw in different direction as show Figure 1-2 (b) and (c) [10].

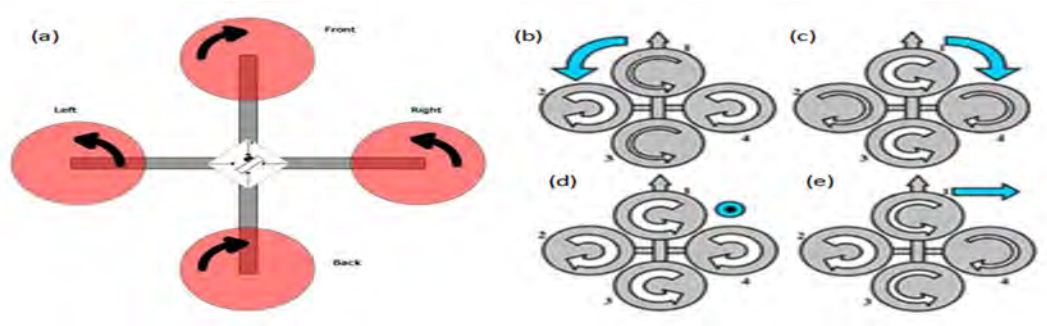

Figure 1-2: (a) Quad Rotor Copter Motor Rotation Configuration. (b) and (c) The Increase in Speed of a Pair of Motors Controls the Yaw of the Vehicle. (d) Vertical Motion and Hovering (Motors Speed Increasing, Decreasing Constantly or Remain Constant Value). (e) Manipulation of Thrust, Pitch and Roll by Difference in Thrust

Vertical motion and hovering is achieved by increasing the speed of all the motors at the same time as can be seen in Figure 1-2 (d) [10].

1.2.2. UAV Vision Systems

As stated, UAVs can be equipped with sensors and cameras. Cameras are used to view the UAVs surrounding, when being flown remotely or autonomously and for human detection. A colour camera and a thermal camera are used to detect human beings in emergency situations [15]. On board hardware is used to do the processing, and to determine the human positions of detected victims which are geolocated with the aid of GPS. The template method was tested on a still thermal camera. The method was still to be tested on a moving camera [15]. Literature reveals that in computer vision and human detection the best method in terms of speed and robustness is the Chamfer Matching [16]. Chamfer matching is a form of template matching, where a binary image of the template is distance transformed and imposed on to the distance transformed frame the search is performed on. Investigations of human detection on UAVs have been performed. One camera is used to decrease the payload the UAV has to support. The two methods that they investigate are monolithic models, with focus on the histograms of oriented gradients and part based models, with focus on the discriminately trained part based models. This system comprises of a GPS system as part of the navigation system. Histograms of oriented gradients (HOG) detectors are one of the most popular and effective methods proposed to date for people detection [17]. Background subtraction methods are also a common used method for detection. These methods apply to stationary cameras or cameras with restricted motion [18].

1.2.2.1. Template Matching

Template matching is a matching algorithm that matches an image portion (Template) against an input image by sliding the template over the input image. This function uses different methods to perform the matching. In OpenCV the following methods are used; square difference matching methods, correlation matching methods, correlation coefficient matching methods and normalized methods [19].

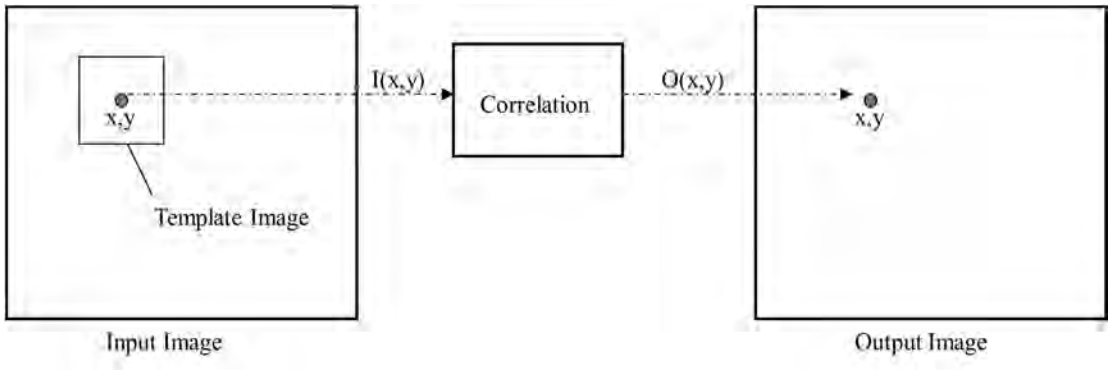

Figure 1-3: Template Matching

Template matching is illustrated in Figure 1-3 [20]. The square difference matching methods match the squared difference of the template and image as described by equation 1 [21].

$$R = \sum_{i}^{X} \sum_{j}^{y} |T_{ij} - I_{ij}| \qquad (1)$$

Where:

R is the Results

T is the template image

I is the input Image

A perfect match will give a result of zero, while a large number would not be a match. The correlation matching methods use multiplication to match the template to the image. Hence the results would be the opposite of the previous method. A zero represents a non-match and a large number represents a match. The correlation coefficient methods match a template and an image relative to their means. A perfect mismatch and match will be -1 and 1 respectively. The normalized methods help reduce the effects of lighting difference between

the image and template [19]. This method works best when the template is part of the input image that the object has to be found in. Template matching has been used to detect human beings from thermal images. The template used is an average template to capture variations in human shape [22].

1.2.2.2. Chamfer matching

Chamfer matching is a form of template matching that uses a binary image (I) and a binary edge template (T) to do the matching. The binary image's distance transform is found and the template matching is performed. The aim is to find a placement of the template on the distance image such that the sum of the distance transformed image (D) multiplied by the pixels values in the template image is minimized. The perfect match at a location is 0. A good match can still be found if the edges are slightly displaced. Match score is calculated using equation 2 below [23]:

$$R = \sum_{i}^{X} \sum_{j}^{n} T_{ij} \times D_{ij} \qquad (2)$$

This method has disadvantages, including when objects are slightly occluded, as the light and shading are different to the input image and template. Hussein et al. use chamfer distance matching as one of their algorithms to detect humans in a moving camera. A silhouette templates database is used to perform the search in input frames. A stabilization algorithm is used to get the foreground image from three consecutive input frames. The fore ground image is thresholded to get the binary image for the matching [24].

1.2.2.3. Haar Classifiers

The Haar Classifier is a visual object detection method developed by Viola and Jones. The method was originally intended for face detection, but can be used to detect other objects. The method allows for fast feature evaluations that select a number of important features using AdaBoost and combines more complex classifiers in a cascade structure to focusing on regions where objects are likely to be found [25]. Each classifier uses rectangular area to decide if the image region resembles the predefined object of interest or not. Figure 1-4 shows the Haar-like features types [26]. One of the types is selected based on the object shape and is extracted by calculating the integral data difference of specific regions [26]. The Haar Feature is a weak learner, and a large number of Haar-like features are necessary to accurately

describe an object. The Haar-like features are organised in a cascade to form a strong classifier. This is achieved by AdaBoost training, where the reliability of classifiers next to each other, affect each other [27].

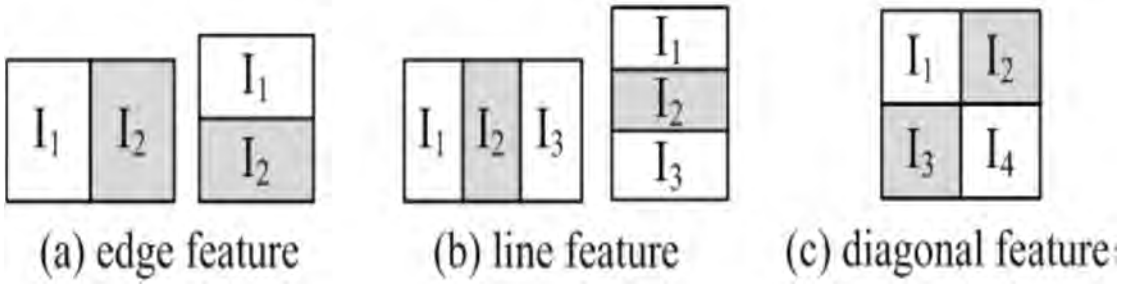

Figure 1-4: Types of Haar-Like Features

The Haar Classifier has been used in applications such as; object tracking [26], classifiers in automobile applications [28], pedestrian detection [28], people detection in complex scenes [29] and car theft detection [27]. The model detects the objects by applying a series of classifiers to every sub-window in the target image. The first stage eliminates non object containing sub-windows with minimal processing. The following stages require more computation to eliminate non object containing sub-windows. This is summarized in Figure 1-5 [26].

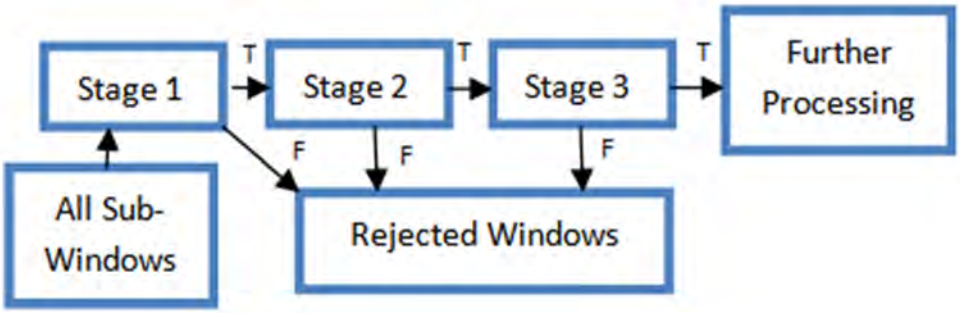

Figure 1-5: Cascade Classifier Illustration

1.3. Objectives and Specifications

The research aim is to focus primarily on the study of UAV vision systems, data processing and navigation. This research was to be used to:

a. Assemble a UAV to use as a platform for testing the vision system.
b. Develop a ground station for UAV control and human interface.
c. Develop a vision system that will be able to detect human beings on board the UAV. This would involve looking at the current available object detection methods and software used to implement these methods. The best method for the application is researched, and a model developed and tested to determine the best model.
d. Implement a communication system that interfaces the UAV with a ground station. This implementation will involve looking at current wireless communication methods and which can best be used on the UAV for the intended application of search and rescue. The communication must be simple to implement and improve on the current communication problems experienced by in UAVs.
e. Implement a control interface that will enable the interpretation of the control instruction from the remote ground station.
f. Develop a system that will integrate the UAV on board systems.

The overview of the research system is given in Figure 1-6.

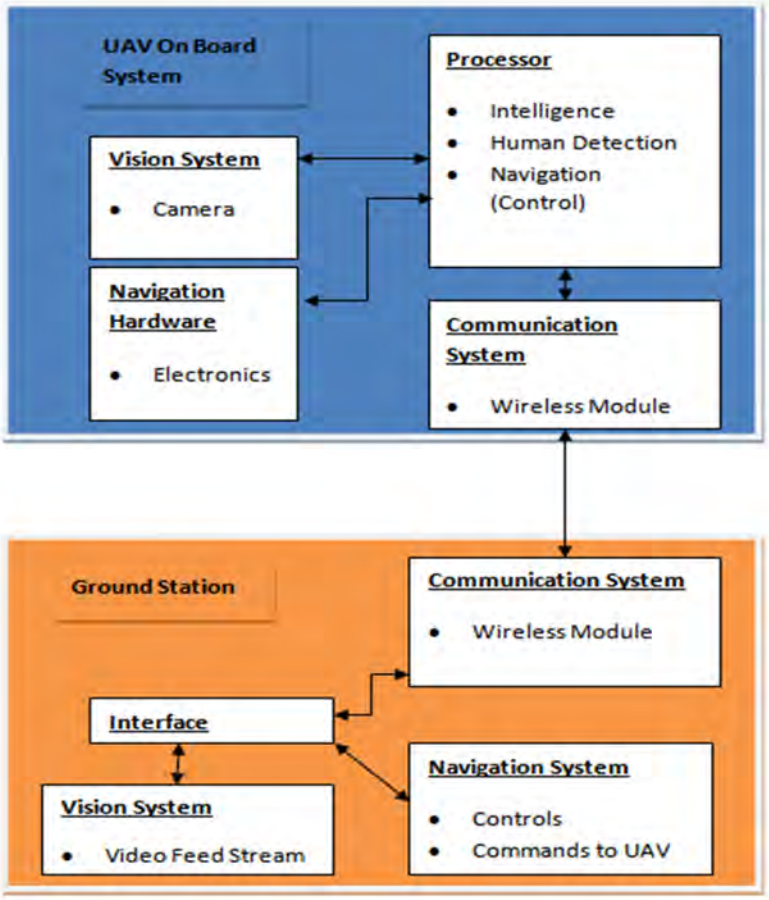

Figure 1-6: Intended System Overview

1.4. Published Work

During the course of the research the following papers were submitted to conferences and journals:

a. S.C. Motepe, A. Chikwanha, R. Stopforth, "Survey and Requirements for Search and Rescue Ground and Air Vehicles for Mining Applications", IEEE M2VIP, Auckland, New Zealand, 2012

b. S.C. Motepe, R. Stopforth, "UAV Vision System and Mechatronic Integration for Search and Rescue applications: Mining and Manufacturing Environments", IEEE Robmech, South Africa, 2013 (Submitted)

c. S.C. Motepe, R. Stopforth, "UAV Human Detection System for Search and Rescue Application", IEEE Robotics and Automation Magazine (Submitted)

1.5. Chapter Outline

Chapter 2 presents the electronic design and integration. The electronic design comprises of all the electronic circuitry and components that will be integrated to make the UAV achieve its intended functionality. This consists of the processor board, brushless motors, battery, electronic speed controllers (ESC's), communication modules and camera.

Chapter 3 presents the communication system between the UAV on board system and the UAV ground station. The UAV has to be controlled from a safe location. Thus the medium of control has to be a wireless one, which will allow the robot to easily maneuver. The wireless network has to also be able to handle image transfer between the UAV and the control station.

Chapter 4 presents the intelligence used by the UAV to interpret the control signals sent from the GS. The human detection system development is also presented in this chapter. The UAV system needs to be integrated. The UAV needs to be controlled from a safe distance. The ground station is developed for this purpose. The ground station (GS) sends control instruction from the rescuers to the UAV. These instructions need to be interpreted by the UAV. Rescuers need to dedicate most of their time in the rescue mission in locations where humans are located. The UAV has a vision system to detect humans.

Chapter 5 presents the experiments carried on the final mode to determine its behavior and accuracy. A comparison of the final model, preliminary model and the OpenCV distributed model is also presented. The comparison also provides the justification for using the final model as opposed to the distributed model and why the preliminary model had to be improved.

The conclusion summarizes the research work that was performed. The summary of the contributions is presented with respect to the research objectives. Future work recommendation for search and rescue application using UAVs and vision systems are presented.

2. Electronic Design

The electronic design comprises of all the electronic circuitry and components that will be integrated to make the UAV achieve its intended functionality. This consists of the processor board, brushless motors, battery, electronic speed controllers (ESC's), communication modules and camera. This chapter presents the electronic design and integration.

2.1. Data Processing (Processor Boards)

The UAV needs to have a controller that can handle and achieve the system's functionality. The UAV has the flight controller, ArduPilotMega, which controls all flight related tasks. The UAV then has the main controller, Odroid-X2, which interfaces the UAV to the controller, and handles all communications between the ground station and UAV. The Arduino Mega 2560 controls the communication between the main processor and the flight controller. These controllers are presented in this section.

2.1.1. ArduPilotMega

The ArduPilotMega (APM) is a specialised UAV flight control board by Sparkfun, containing an Atmega 2560 microcontroller. THE APM was used for the UAV flight control. The board is purchased of the shelf with headers to be soldered on for connection to the ESCs, input control signals from the Arduino Mega and the Inertial Measurement Unit. Figure 2-1 and Figure 2-2 [30] show APM Board and the typical APM setup, respectively.

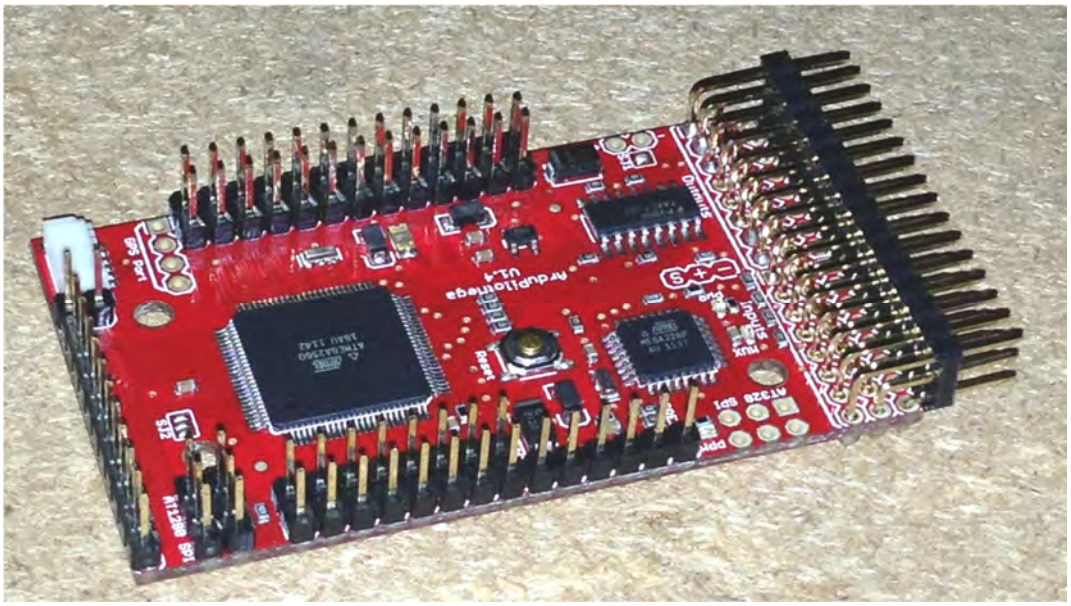

Figure 2-1: APM Board

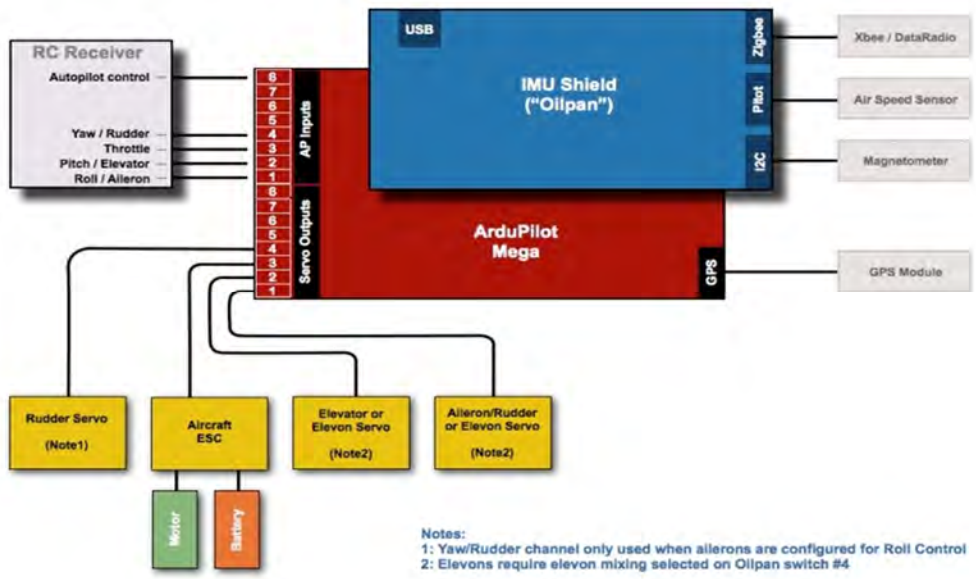

Figure 2-2: Typical APM Setup

2.1.2. Arduino Mega 2560

The Arduino Mega 2560 is used to interpret signals received by the UAV, via the Odroid-X2, to eliminate the traditional RC control that the APM is used for the UAV control and to enable the UAV to be controlled from the ground station. The Odroid-X2 receives control signals over Wi-Fi and sends them to the Arduino Mega via serial communication. The signals are interpreted in the Arduino Mega and 4 PWM signals are produced. The PWM outputs go to APM board.

2.1.3. Odroid-X2

The Odroid-X2 is a quad core, open source development platform. The Odroid-X2 was used as the main on-board UAV processor and has the possibility to integrate the whole on-board system. A Wi-Fi module connected to it for communication with the ground station, the camera for computer vision and the Arduino Mega board. Figure 2-3 shows the Odroid-X2 Developer Platform.

Figure 2-3: Open Development Platform, Odroid-X2

The processor runs the Linux operating system. The image processing is done on the board using OpenCV Libraries. Odroid-X2 has the following features, which make it ideal for the application [31]:

a. Low-cost
b. 1.7GHz Quad core ARM Cortex-A9 MPCore.
c. 2 GB RAM
d. 6 High speed USB2.0 Host Port , to connect peripherals such as a camera and a Wi-Fi
e. Runs Android or Ubuntu, which are open source operating systems
f. It is 90 mm × 94 mm in dimension, making it compact for the use on the UAV

The Raspberry Pi, which was considered to be used as the processor, has a processing speed of 700 MHz, 500 MB RAM and 2 USB inputs. The Gumstix Verdex Pro is also a processor that was considered to be use, but had a disadvantage of low processing speed (600 MHz) as opposed to ODROID-X2. Neither the Raspberry Pi, nor the Gumstix Verdex was therefore used. The ODROID-X2 was used, as it has a processing speed of 1.7 GHz, 2 GB RAM and 6 USB ports [33].

2.2. Inertial Measurement Unit

An inertial measurement unit is a device that measures the velocity, orientation and gravitational forces using gyroscopes and accelerometers. The ArduPilotMega IMU shield (Figure 2-4), which has three axis angular rotation and acceleration sensors, was used in this

research. This IMU is used for the UAV to determine its orientation and to help in maintaining flight level and stability [30].

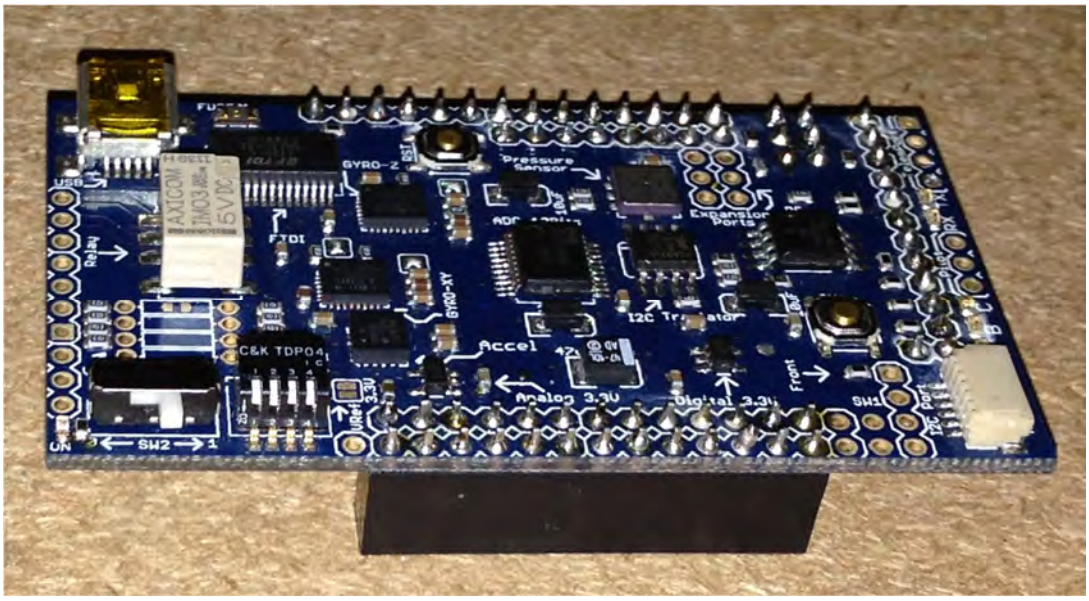

Figure 2-4: ArduPilotMega IMU Shield

2.3. Propeller Control

The UAV is a quad rotor copter. This requires 4 motors to control each of the propellers. Each motor had an electronic speed controller (ESC) connected to it.

2.3.1. Brushless Motors

Brushless motors were used to rotate the propellers. These were chosen due to the advantages they have over other motors (and AC motors). The advantages of DC brushless motors can be summarised as follows [34]:

a. Performance: Brushless systems offer the highest available dynamic accuracy
b. Size: They are the smallest available motors for a given power rating
c. Efficiency: The brushless motor is the most efficient motor for any industrial application
d. Safety: Brushless motors minimize explosions in dangerous gas environments

The brushless motors selected for our application have the following specifications

- Rotational Speed: 1000 (kv) RPM/V

- Continuous Current: 20 A
- Max. Current: 28 A
- Input Voltage: 6 - 11.1 V
- Max. Efficiency: 98 % No Load Current, 0.8 A
- Internal Resistance: 172 mΩ
- Power: 255 W
- Motor Weight (Motor only): 60 g
- Motor Dimensions (Diameter x Length): 28 mm x 30 mm
- Propeller Dimensions: 10" x 5 "/ 10 " x 6 "/9" x 4.5 "
- Input Battery Types: NiCd/ Nimh/ Li-po Battery

2.3.2. Electronic Speed Controllers (ESC)

An electronic speed controller is an electronic device that is used to control a motor's speed, direction and dynamic braking (Some ESCs). The choice of an ESC depends on the following factors [35]:

a. Type of motor used: Brushed or brushless motor
b. Current Rating: Should be equal or higher than that of motor to be controlled
c. Voltage Rating : Should be equal or higher than that of motor to be controlled
d. Low Voltage Cut-off: Automatic shut off when the battery power is low to save the battery from getting damaged
e. Price
f. Programmability

The Aeolian Motor Evolution ESCs, shown in Figure 2-5, were found to be suitable for the application and were used. They had the following properties [36]:

- Input voltage: 5.5 V-17 V (support 2-4S lithium batteries)
- Continue current: 30 A
- Burst current: 40 A (within 10 seconds)
- BEC output: 2 A/5 V (linear)
- ESC for Brushless motor
- Size: 50 mm x 28 mm x 12 mm
- Weight: 34 g

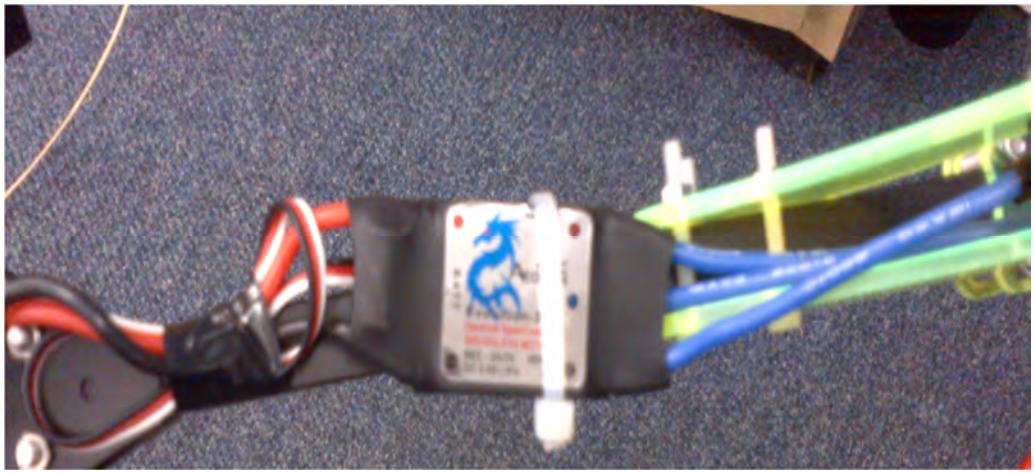

Figure 2-5: Aeolian Motor 30 ESCs Connected to UAV Arm

2.4. Power Supply

All the circuitry on the UAV needed a power source to power them up. A battery is the ideal power source since it is portable and mobile. Factors that need to be considered when selecting a battery are [37]:

- safety
- weight
- energy
- battery memory effect
- rechargeability

The properties of Lithium-ion Phosphate (Li-Po) batteries make them the most suitable for our application. Li-Po batteries are [38]:

- rechargeable
- safe
- weigh less for same size
- have no thermal runaway
- corrosion free on the terminals
- non explosive
- have no battery memory effect
- have more energy

– environmentally friendly

The Li-Po batteries to be used are the Win-Max Nano Platinum 5200 mAh, 11.1 V batteries. Initially the power was distributed by connecting the electronic components directly to the components in parallel to the battery. This configuration, shown in Figure 2-6, introduced a lot of wires and a difficulty in tracking where the components were connected. To make the circuit neater and easier to follow and to change components during test, a power distribution board was used. Headers and cables for the ESC power supply and output were soldered on to the board. The battery also had a cable soldered to the board that allows for the battery to be easily connected and disconnected to the distribution board (and the system). This also has a benefit of allowing easy disconnection for charging and removing power from the system.

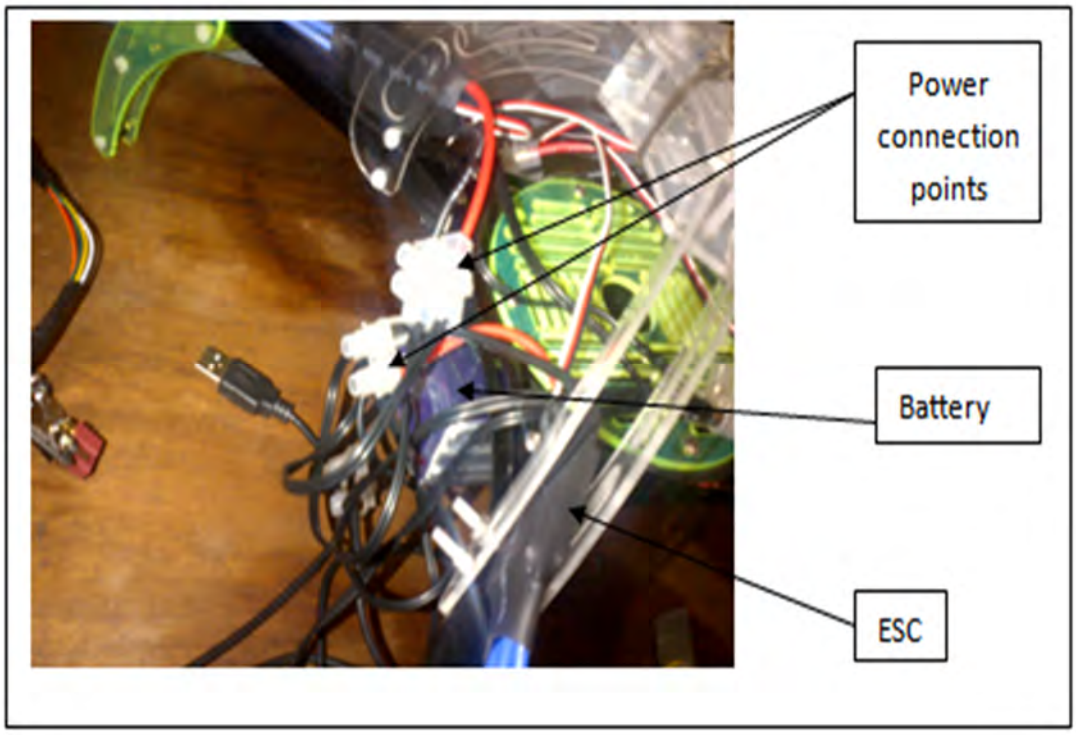

Figure 2-6: Power Distribution with ESC Connected Directly to the Battery

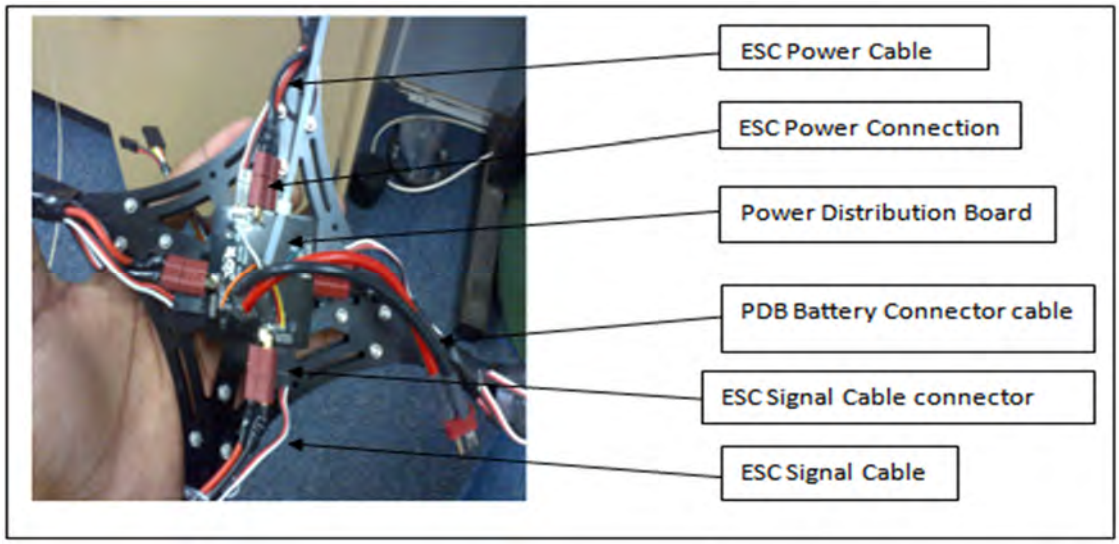

Figure 2-7: Power Distribution Board

2.5. UAV Mechanical Assembly

The UAV body structure has to be lightweight to decrease the payload. The structure must be strong enough to hold all the necessary components without getting damaged. A quad copter frame was assembled (Figure 2-8). The mechanical components used were laser cut. The metal components were supported with plastic structures for even landing, protecting the electronic components and to provide mounting points for the motors and propellers. The electronic components were assembled onto the UAV and wired as needed.

Figure 2-8: Assembled Quad Rotor UAV Body Frame on Left and Completely Assembled UAV

2.6. Chapter Summary

The electronic components as well as the UAV and mechanical structure integration were discussed. The processing boards were presented. The ArduPilotMega (APM) is the processor used for the flight control. The APM board has an inertial measurement unit (IMU) connected to it. The IMU measures the velocity, orientation and gravitational forces on the UAV using gyroscopes and accelerometers. The IMU helps the UAV maintain flight level and stability.

ODROID-X2 is the main processor onboard the UAV. ODROID-X2 has the communication interface to communicate with the ground station and, has the processing power to do the image processing and control signal interpretation. These control signals are sent to the APM via the Arduino Mega to the APM.

DC Brushless motors are used to drive the rotate the propellers. Electronic speed controllers (ESCs) are used to control the speed of the motors. Li-Po batteries are used to power the on-board system. The Li-Po batteries have a number of features which made them the suitable choice; these include being safe, weighing less for the same size and being environmental friendly.

3. Communications

The UAV has to be controlled from a safe location. Thus the medium of control has to be a wireless one, which will also not hinder the robot's maneuvering ability as opposed to tethering [37]. The wireless network has to also be able to handle image transfer between the UAV and the control station.

3.1. Preliminary Design

This subsection discusses the preliminary design and approach to the communication system.

3.1.1. Communication Modules

This depended on the communication protocol to be used. The communication protocols being considered were Wi-Fi (802.11 b/g/n) and X-bee's RF communication protocol (802.15.4 protocol). These protocols were being considered due to the hardware that would be used for the main processor, which was an Arduino Board or a mini computer. The communication module would also be determined by the processor hardware and protocol. The two protocols were considered because high frequencies are ideal for transmission through dense materials. Antenna size and frequency have an inversely proportional relationship [37]. The antenna used will therefore be small and fit onto the UAV.

3.1.2. Vision System

During a rescue mission the environment conditions of rescue environment needs to be known to the rescuers, therefore the information about the environment must be sent to the control station where the rescuers use it. Rescuers also need to know that there will be victims in the location where they will be deployed. The UAV had to be equipped with a camera that will be used to capture images. These would be used to get a visual of the rescue scene and for autonomous human detection. The camera used had to be able to take pictures (or videos). The videos must be able to be transmitted over a wireless network for to the control station for processing. The transmission could be done by the camera itself or using external circuitry. The camera therefore had to be able to be integrated into the processor used or connect directly to the wireless network. LinkSprite LS-T201 serial port camera is a camera that was found to be available in the market and was researched further. This camera is very compact (32 mm×32 mm) and can take pictures using the serial port [39]. The camera can be connected to take pictures over a wireless network [40]. The three previous features related to the

camera are what gave rise to it being researched further as opposed to other wireless or Wi-Fi cameras. This camera can be setup to take pictures and send them to a set path on a PC, using the LYS201. LYS201 is an executable software program that can be downloaded from LinkSprite interfaces to the camera and setup the camera. Figure 3-1 shows the LYS201 GUI.

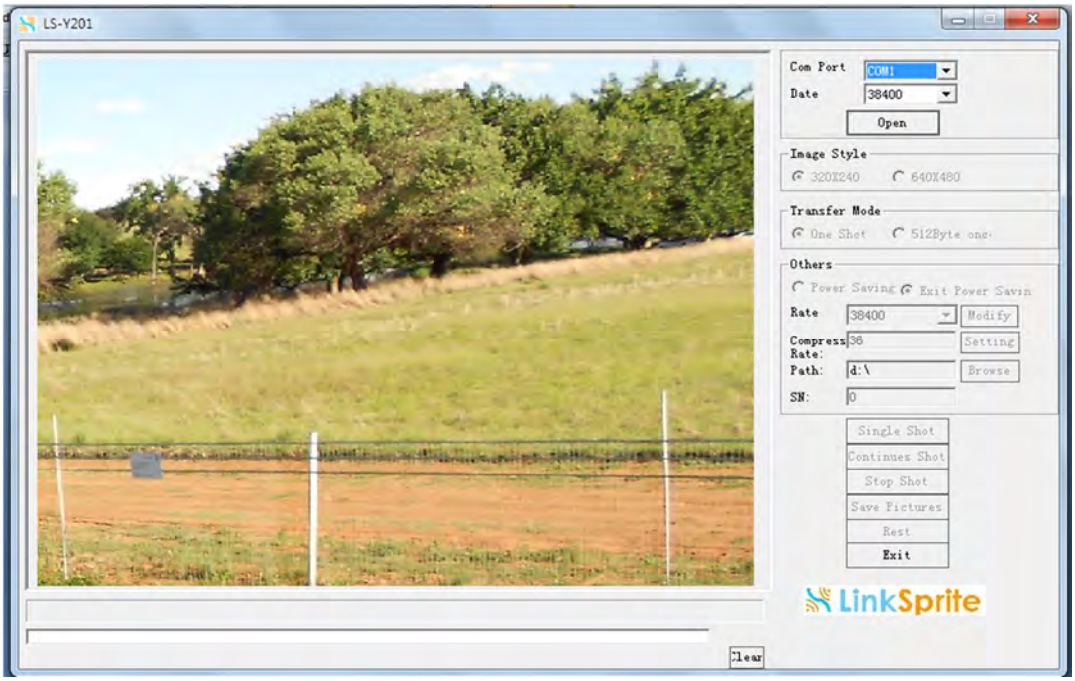

Figure 3-1: LYS201 User Interface

3.2. Final Design

This section discuses the final design and the steps taken from the preliminary design to get to the final design.

3.2.1. Wi-Fi Module

The main processor chosen was the ODROID-X2. The X-Bee protocol could therefore not be used. ODROID-X2 has Wi-Fi communication capabilities when a compatible module is connected to it. The Realtek RTL8191SU is the recommended, compatible Wi-Fi module by the manufactures of ODROID-X2. The Realtek RTL8191SU was used on the UAV for communication. This is an integrated single-chip MIMO wireless LAN USB 2.0 network interface controller. That complies with the IEEE 802.11n specification. It provides a high performance wireless client, with the combination of a MAC, IT2R capable baseband and RF in

a single chip. Figure 3-2 shows the RTL8191SU Wi-Fi module with an antenna [31]. The University of Kwa-Zulu Natal's Wi-Fi was used to test the communication between the UAV and the ground system.

Figure 3-2: RTL8191SU Wi-Fi Module

3.2.2. Vision System

The initial idea was to process the images on the ground station. This would require sending the images or video stream to the ground station. The images would be captured and the processor, ODROID-X would have a method of sending the images. A method of doing this was researched and client-server architecture was found to be a reliable and simple method to do this. This would be done using OpenCV and required a UNIX like OS to be implemented. Since the processor could use Ubuntu or Android, it could handle the image transfer. The system software needs a video grabbing part that grabs the video from the camera and a client connection part that sends videos to the client on connection. These two parts need to run simultaneously in loops at the same time. These therefore need to run as threads using multiple-threading. There also needs to be a thread for receiving the video stream and one for displaying the video on the client side. The human detection model processes the received frames and then displays them on a GUI. Figure 3-3 shows the server-client structure.

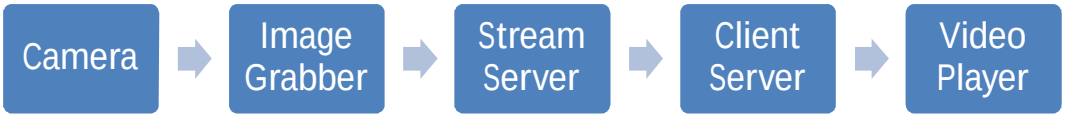

Figure 3-3 : The Video Stream Server-Client Structure

A server-client structure that uses OpenCV and works in this manner has been developed before [41]. The code for the server and client was downloaded and modified for the application. Before the system integration was tested the question of flight instructions transmission to the UAV from the ground station arose. This would need a structure similar to that of the image transfer and would have the data sent from the GS to the UAV. The possible solution to this was the researched. ODROID-X has capabilities to do the image processing for human detection. If the human detection was done on-board there would not be a need to send the image to the GS for processing, but the rescuers would still need to see the image detection results for verification and to see the environment where the detection was made. The processing speed on the ODROID-X is high enough to process videos in real time. This factor minimises any mission delays that can be introduced by the processing time. The confined space the UAV has to fly in during missions restricts the UAV flight speed. This restriction complements the time required for processing and further makes sure that the processing time does not delay the mission time. The communication problem could be summarised as shown in Figure 3-4.

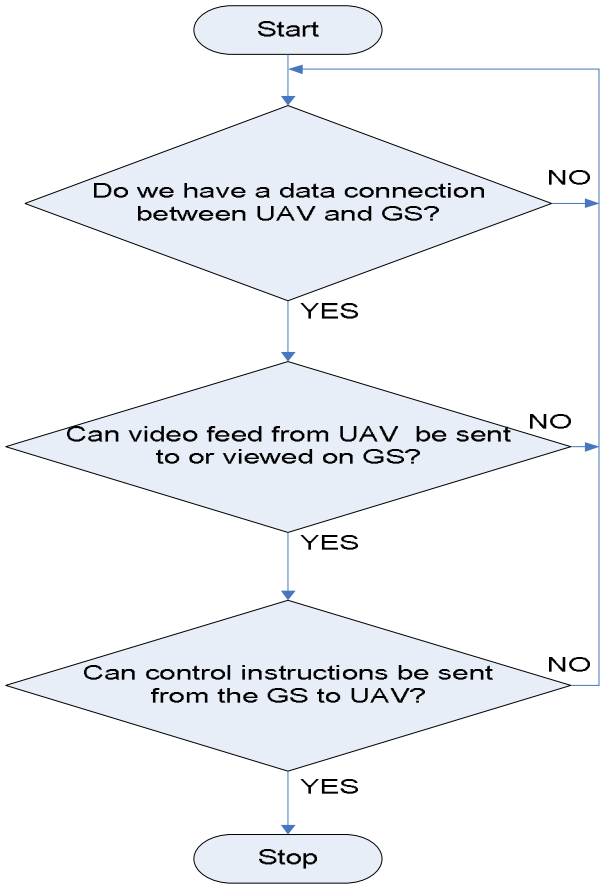

Figure 3-4 Data Transfer Problem

The complete communication system needed to integrate having the video stream available to the ground station and enable instruction to be sent to the GS. During the research a possible solution which was considered that would satisfy the above requirements was a Windows Remote Desktop Connection like application. Windows Remote Desktop allows a user to connect to a PC on the same network using its IP address and login password. This kind of application would provide access to the UAV. With the image processing happening on board and being displayed on the UAV's processor desktop, the human detection results can be viewed from the ground station. Remote Desktop Connection allows keyboard input, which could be used for flight control instructions. The login necessity would provide security, allowing a person who can fly the UAV to need login details. A number of applications were available for Ubuntu, to achieve remote desktop login. Some applications, like Remina Remote Desktop and Desktop Sharing, only allowed viewing of the desktop. TeamViewer was the software application found to allow keyboard input into the remotely logged on to desktop. The TeamViewer user interface is shown in Figure 3-5. The software that handles the keyboard

input is discussed in Chapter 4. Referring to Figure 3-4, once the UAV and GS are connected to the network, the video can be viewed on the GS and control instructions can be sent to the UAV. Thus a solution to the data transfer problem is found.

Figure 3-5: TeamViewer User Interface

The final system has the image processing performed onboard the UAV. This system eliminates the need to send the raw camera feed directly to the ground station for processing. After performing human detection on the camera input the processor handles the communication and data transfer between on board system and the ground station. The processed data is available on the ground station and can be recorded and stored on the GS hard drive. This enables the data to be viewed at a later time and for rescuers to be able to see if there is a false detection from the UAV. The Genius Webcam Eye 300 (Figure 3-6) was used to capture the video. This camera can take images with a resolution of up to 1.3 megapixels. The camera was calibrated using the OpenCV checker board calibration.

Figure 3-6: Genius Webcam Eye 300

3.3. Chapter Summary

The chapter presented the communication system between the UAV and the ground station. The communication protocol had to utilise high frequencies to transmit through dense walls in mining environments. The protocol also depended on the hardware being used. Wi-Fi was used as it utilised high frequencies and is available on the ODROID-X2. The UAV has a camera to take videos of environment. The communication system was used to send the videos to the ground station. The initial design was to send the videos to the ground station before being processed. The system needed to incorporate the sending of flight control instructions from the ground station to the UAV. The approach to have two server-client structures for video transmission and control instructions transmission respectively was not used for the complexity it will introduce. To decrease the complexity of the system the image processing for human detection was performed on board the UAV. TeamViewer was used to view the image processing on the UAV. TeamViewer allows keyboard input. Code was developed to detect the keyboard input, from the GS, on-board the UAV. This configuration allowed the UAV environment and the human detection to be accessible on the ground station and for control instructions to be sent to the UAV.

4. Artificial Intelligence

The UAV system needs to be a mechatronics integration, to allow it to be controlled from a safe distance. The ground station is developed for this purpose. The ground station (GS) sends control instruction from the rescuers to the UAV. These instructions need to be interpreted by the UAV. Rescuers need to dedicate most of their time in the rescue mission in locations where humans are located. The UAV has a vision system to detect humans. This chapter presents the intelligence used by the UAV to interpret the control signals sent from the GS. The human detection system development is also presented in this chapter.

4.1. Navigation

The UAV needs to be controlled to get to a disaster site. Rescue personnel must be able to control the UAV from a remote location (Ground Station). The rescuers have to send navigation commands to the UAV that must be interpreted and turned into actions. This section discusses the navigation control philosophy and the UAV on board intelligence for signal interpretation and flight control.

4.1.1. RC Controller

The ArduPilotMega can be loaded with software for the quad vehicle. This software can be downloaded directly on to the UAV from the internet using a software application called APM Mission Planner and is available for download on the Arducopter wiki Google code site. The software that was loaded on to the UAV, which is for flight control and stability, can be modified to suit the integrated system and required functionality. The modifications are to setup the parameters or change the behaviour of the UAV. Even though in this research the flight dynamics were not the primary focus, time was spend studying the flight dynamics code. This study was to have knowledge of how the code was structured, what each part of the code did, what parameters can be modified and which parts of the code may need to be altered or removed. This study was also important as the UAV would be controlled from a computer as opposed to a radio controller that the code was setup for. Parameters, such as radio controller, PI gains, etc. can be set-up the using the mission planner. The UAV was programmed and the parameters setup using APM Mission Planner. The system had to be adopted to use a computer to send flight instructions to the UAV's APM flight controller. An understanding of an RC controller was needed as an RC controller sends wireless signals to a receiver at a set frequency. The receiver must be set to the same frequency to receive the

control instructions (roll, yaw, throttle, pitch). These sent instructions are received by the RC receiver and the corresponding PWM is outputted on the receiver's different outputs. These signals go into the APM where they are manipulated to produce PWM output that go to the ESCs that run the motors. The software processes the instructions to get the desired motion of the quad copter UAV. To get an understanding of this, the hitec flash 5 RC controller (Figure 4-1) and Futaba RC receiver were used. Figure 4-2 show the RC receiver connected to the APM.

Figure 4-1: RC Controller

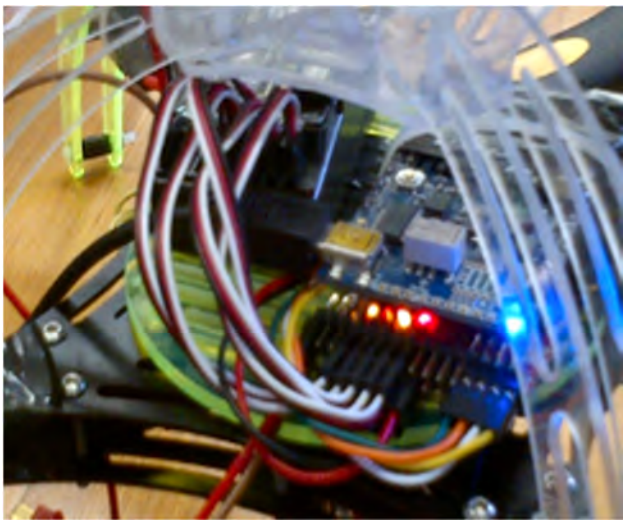

Figure 4-2: RC Receiver Connected to APM Board

The overview of the components integration and functionality is given in Figure 4-3.

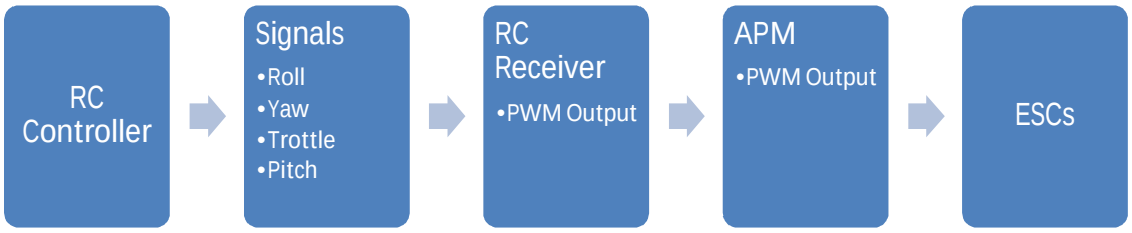

Figure 4-3: RC Controlled UAV Overview

The flight system was tested and worked. During further testing the system stopped working. The problem was investigated and the APM board was found to not arm the motors and to not send output signals on the output channels (ch1out to ch4out). The motor inputs and outputs are given in Figure 4-4. The minimum and maximum value for the input and output, are 1000 and 2000, respectively.

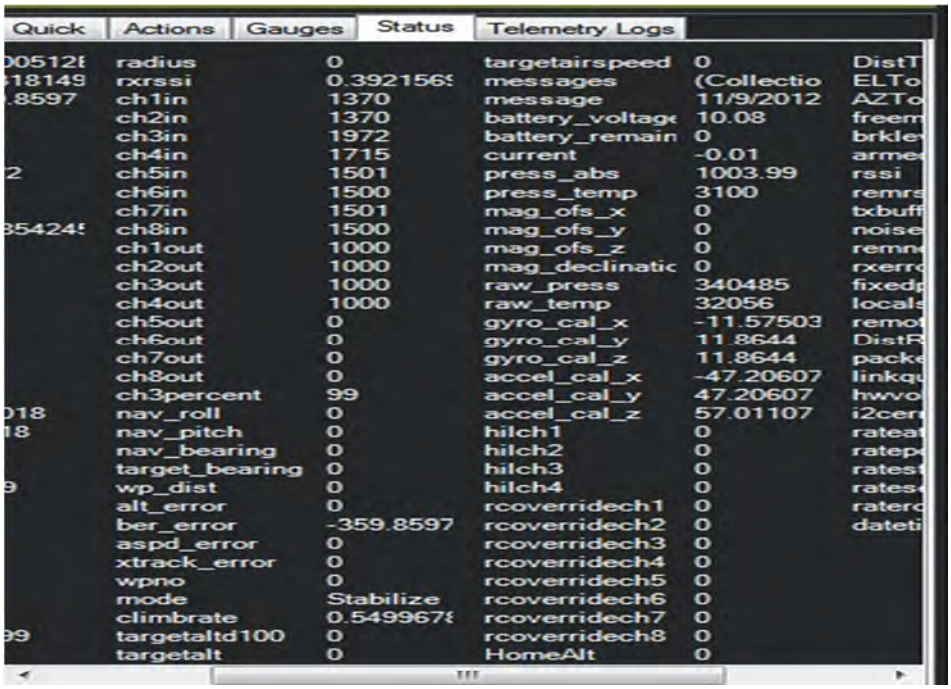

Figure 4-4: Copter Status as Viewed on the APM Mission Planner

Figure 4-5 shows the raw sensors for the RC input and the APM outputs on the mission planner. The motors were unable to arm, causing the input to not cause any change in the output to the motors.

Figure 4-5: Raw Sensors for the RC Input and APM Outputs

The APM board was tested and found to be faulty. The board was replaced with a new board and the new board was tested and was found to be working. To arm the motors the minimum throttle value and maximum roll value must be sent to the APM input for approximately 5 seconds. The UAV was then tested for flight and was found to be able to take off.

4.1.2. Ground Station Control

The test conducted in section 4.1.1 led to the conclusion that the UAV can be controlled and all the hardware related to the flight of the UAV is functioning. With this test completed, the aim was then to get the UAV to fly by sending the control instructions over Wi-Fi from the GS. This part was done with the following components:

i. Main Processor (ODROID-X2)
ii. Flight Controller (APM)
iii. Communication Method (Wi-Fi)
iv. Use of TeamViewer

The challenge was to research a method to interpret inputs from the ground station that are sent over the TeamViewer connection. The basic ground station has a keyboard and mouse. A joy stick could be added to be dedicated to the control. This addition would involve getting a joystick and then researching how it behaves over the connection, so the research was narrowed to the mouse and keyboard. The two input devices were then analysed to see which of the two to focus on and use. Table 4-1 and Table 4-2 summarize some advantages each input device has with respect to the other device.

Table 4-1: Mouse Advantage and Disadvantages as a UAV Input Controller

Pros	**Cons**
The mouse can be moved in different directions	The motion can be easily/accidentally triggered causing unwanted motion
Different motions can be used to trigger set motions of the UAV	Interpreting the actual motion direction can be a complex task
There are buttons on the mouse that can be used for further actions	

Table 4-2: Keyboard Advantage and Disadvantages as a UAV Input Controller

Pros	**Cons**
No accidental motion of triggering of UAV.	No motion inputs
Multiple keyboard key inputs	One key can be used and interpreted per time
Each character has a unique identifier(ASCII value)	
Interpretation of ASCII characters can be less complex as compared to mouse motion directions	

From the analysis above it was decided that using the keyboard inputs would be a better solution. It was found that OpenCV has keyboard input detection functionality. This input is achieved using the command cvWaitKey(). This function detects when a key is pressed and a variable can be assigned to this function to store the ASCII value of the pressed key. This meant that OpenCV running on the UAV can be used to detect which key is pressed and a certain key can be assigned to a specific flight command. These keys would then have to be made understandable to the APM and would need the characters to cause a PWM signal output into the APM. An ArduinoMega 2560 microprocessor board was used to interpret the characters and send a corresponding PWM instruction. Code was developed to detect keys pressed and to send characters over the serial port. Code was developed for the Arduino board to receive the character and produce PMW output corresponding to the desired control instruction. Table 4-3 summarizes the keys used and the corresponding instructions.

Table 4-3: Summary of Keyboard Keys Used to Control the UAV

Instruction	**Character**		**ASCII Value**	
	Decrease	**Increase**	**Decrease**	**Increase**
Pitch	a	d	97	100
Roll	z	w	122	119
Throttle	m	i	109	105
Yaw	j	l	106	108

Figure 4-6 shows the code algorithm for sending characters over the serial port once key detection had occurred. Select-case statements were used to check which key was detected, that the correct characters are sent. The select case options from 1 to *n*, is the ASCII values for the control instructions presented in Table 4-3. The code for the algorithm is presented in Appendix G.

```
Pressed key=cvWaitKey();       //Detect pressed key
Select Case (Pressed key ):
        Case 1
                Send character to serialport;
                break;
        Case 2
                Send character to serialport;
                break;
        .
        .
        Case n
                Send character to serialport;
                break;
```

Figure 4-6: Code Snippet for Sending Characters Over Serial Port From Main Processor to Arduino Board

Figure 4-7 shows a snippet algorithm of the code that is loaded to the Mega 2560 microcontroller chip. This code is used to receive characters and set the PWM duty cycle value to either maximum or minimum. The mapping was used in the initial stages and can be removed and values between 0-255 used directly instead. The tests were done with the minimum and maximum values but these were modified to have speed steps, instead of just maximum and minimum values. In the final design the algorithm was kept the same but the PWM value was either decreased or increased instead of being set between the maximum and minimum value. This was also motivated by the instability observed when the PWM values change drastically. Take off and landing functionality was added to the control. The instructions allow the plane PWM to gradually change automatically until the UAV has landed or taken off respectively. The code for the algorithm is presented in Appendix I.

```
Receive instruction;
Select Case (Receive instruction):
        Case 1
                Set PWM value for corresponding flight instruction;
                break;
        Case 2
                Set PWM value for corresponding flight instruction;
                break;
        .
        .
        Case n
                Set PWM value for corresponding flight instruction;
                break;
```

Figure 4-7: Arduino Code Snippet for Keys Detection and Mapping the PMW Duty Cycle Time

Figure 4-8 shows character values that are being received on the Arduino serial monitor, as keys are being pressed and detected by the code running on Eclipse, which has the video input being shown on "Capturing Image" window and runs the OpenCV library functions. The cvWaitKey() function is initiated when the GUI is open and will only detect the key if the GUI is active.

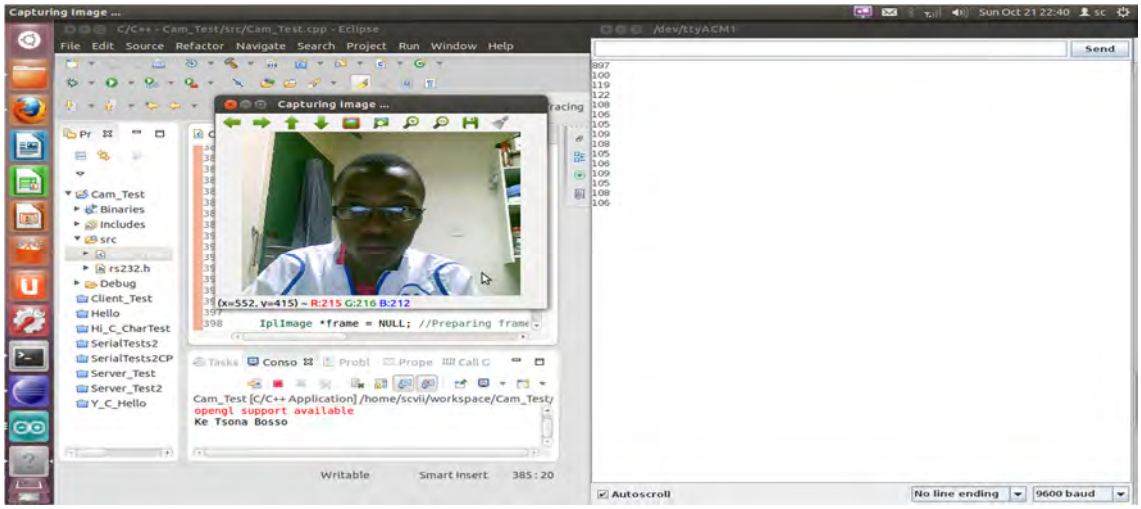

Figure 4-8: OpenCV Code Used to Detect and Send Character to Arduino Board and Arduino Serial Monitor Showing Received Characters

4.2. Human Detection

Rescuers need to control the UAV from a safe location. The rescuers need to have vision of the rescue environment. Victims need to be attended to in time. Most of the time in rescue missions can be lost trying to locate where humans are and the state of the of the rescue environment. This section presents UAV vision system. The system includes the human detection system which is used to identify the presence of victims in the search and rescue mission.

4.2.1. Preliminary Design

The aim was to have a reliable and simple to implement algorithm to do the human detection. The focus was on the implementation of the algorithm using open source software, with no discrimination to non-open source software. Open source is a software development method that utilizes the power of distributed peer review and transparency of processes. Its characteristics are better quality, higher reliability, more flexibility and lower costs [42]. Most computer vision researchers use either MATLAB or OpenCV. OpenCV is a library of programming functions developed by Intel aimed at real time computer vision. In comparing the two programming environments it was found that they both have their advantages and disadvantages. These characteristics are summarized in Table 4-4, where the italic text indicates an advantage of one environment over the other [43].

Table 4-4: Advantages and Disadvantages of MATLAB and OpenCV

MATLAB	**Open CV**
Installation Space Disk Space 3-4 GB RAM 1GB	*Installation Space 1-2 GB* *RAM <256 MB*
Licence costs	*Open Source*
Has strong image processing capabilities, but MATLAB is not developed solely for image processing	*Optimised for Computer Vision and Image Processing*
Can be slower, due to consuming a lot of RAM	*Consumes less than 100 MB of RAM during processing*
Not easily portable from prototype stage to final stage	*Can be easily ported to any device that runs C*
Skills learned are useful in MATLAB environment	*Can help in development of other programming languages skill such as C, C++ or JAVA, on different IDEs such as Eclipse, NetBeans, Visual Studio, etc*
Easier to use	More complicated to use
Automatically manages memory	The programmer must take into consideration to free up memory in the code

The Open Source Computer Vision (Open CV) library was chosen for research purposes due to its advantages which were ideal for the research. This was downloaded and setup and the methods researched are discussed.

4.2.1.1. Template Matching

Template matching was investigated before the other methods due to its simplicity. The main aim was to test the method and get familiar with OpenCV and Visual Studio. This investigation involved setting up OpenCV, learning to implement basic functions to open images, videos, etc. Template matching was first used to find heads in an image. The template is given in Figure 4-9, the input image in Figure 4-10.

Figure 4-9: Template Image

The input image had heads scattered and some slightly obstructed. This was also to test the model's ability to detect the target object when it is slightly obstructed. The results image in Figure 4-11 is the cvMatchTemplate function output, with the normalized correlation method used. Detections results of the method are shown by the areas that are brightly shining (light parts) in Figure 4-11. Figure 4-12 shows the target objects detected market on the input image. The result coefficient is set to 0.99 for a match.

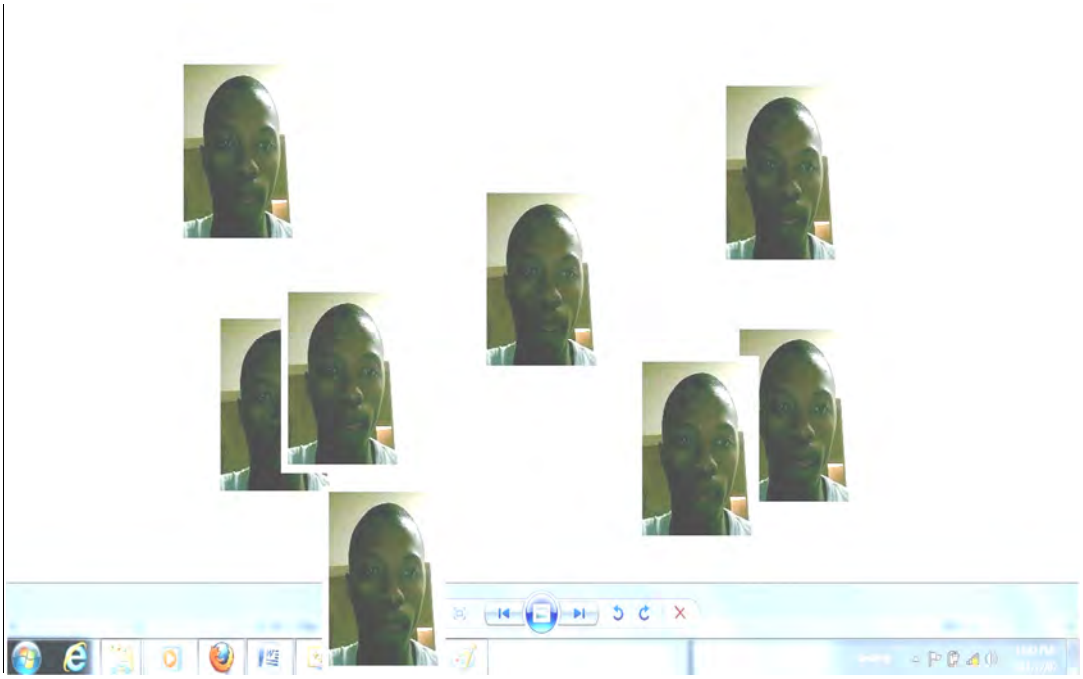

Figure 4-10: Target Input Image

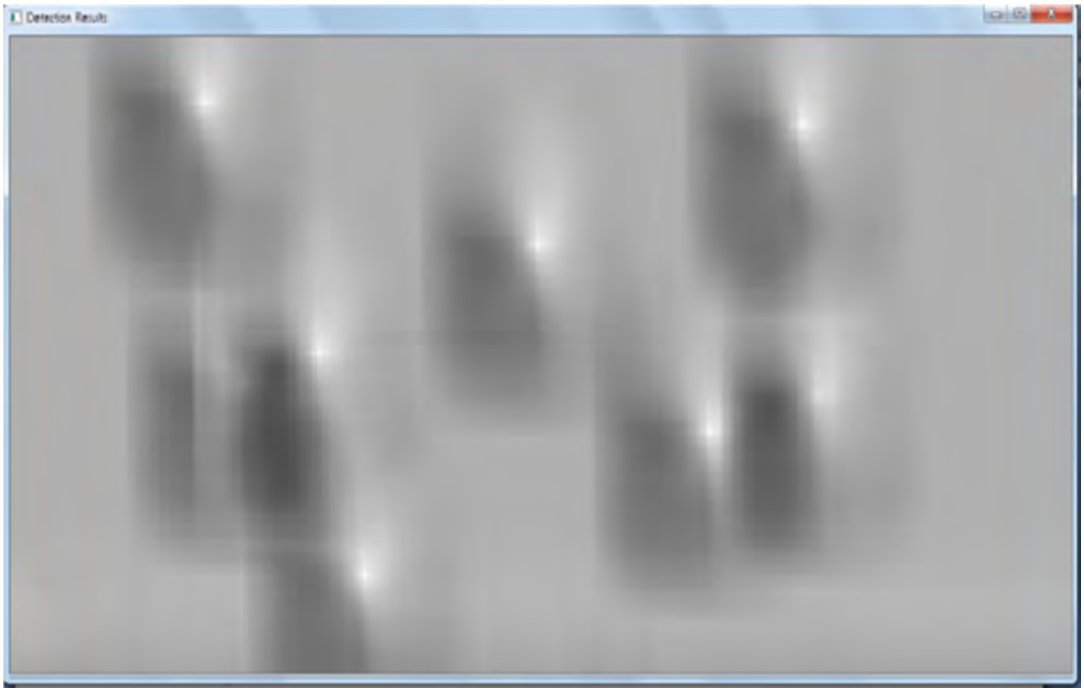

Figure 4-11: Correlation Normalized Method Results Image

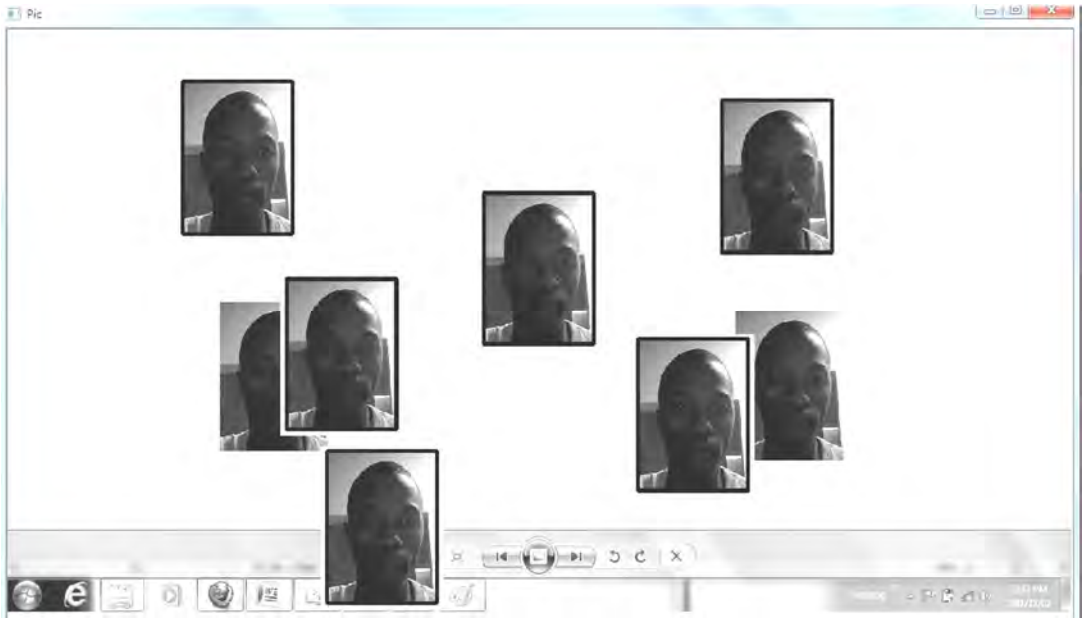

Figure 4-12: Results, Detected Target Images Marked on Input Image (Coefficient=0.99)

When the input image was changed to one that has faces, including that of the person in the template image, but not the same image as the template, no detections were made. Figure

4-13 and Figure 4-14 show the input image and correlation method results with the template image remaining the same.

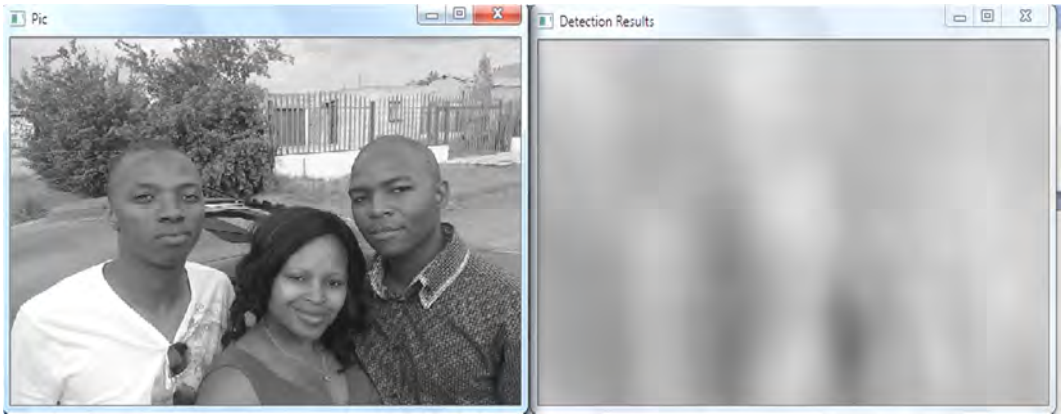

Figure 4-13: Input Picture and Correlation Normalized Method Results Image

Figure 4-14: Input Image Correlation Normalized Method Results Image

4.2.1.2. Chamfer matching

Chamfer matching was investigated on OpenCV. There were some difficulties in implementing the method in OpenCV. The application in MATLAB was investigated, and compared to OpenCV. The problem encountered was that it is similar to template matching, where a template used on an image that does not contain the object of interest and therefore yielded no detection. The implementation in MATLAB was not very robust and led to a number of MATLAB system crashes.

4.2.1.3. Haar Classifier

OpenCV has already pre-trained Haar Classifiers. These include the face detection classifier, upper body classifier, full body classifier and the lower body classifier, which were investigated. Code to test the latter three was written and they were tested. The aim was:

1. To test their human detection capabilities.
2. Their miss rate and false detection.
3. The common features that their detections are based on.

These classifiers were tested on 60 test images, web camera video feed and videos clips. The test on the 60 test images were done to see the hit rate, false hit and miss rate. The results are presented in Appendix A. The results using the different classifiers were the same for all three of the classifiers. It was observed that the classifiers depended on a clear frontal facial appearance for the detection. The figures below also demonstrate this. Figure 4-15 to Figure 4-17 show the results for the full body classifier. In Figure 4-15 there are positive detections (marked with a square). On TestImage11 there is a miss for the person in the foreground, facing away from the camera [44]. On TestImage13 the human is detected using the frontal face as the human is facing the camera [45].

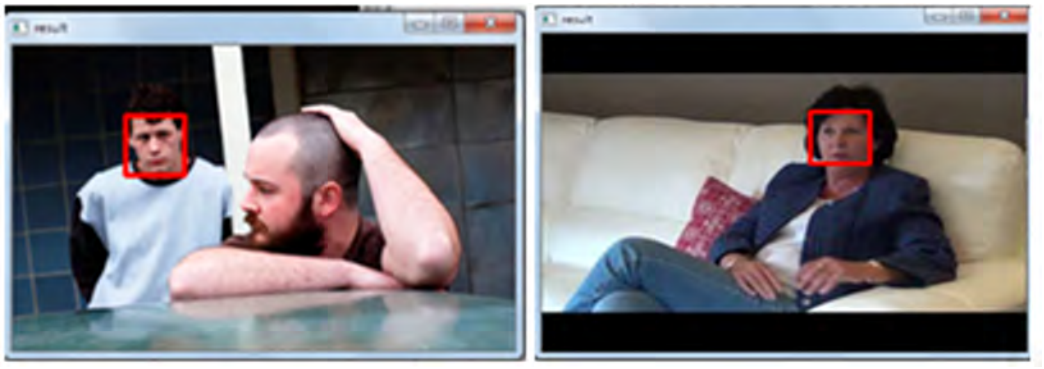

Figure 4-15: Full Body Haar Classifier Results on TestImage11 and TestImage13

In Figure 4-16 no detections were made. The faces in TestImage22 are obscured by the masks [46]. In TestImage15 the faces are not directly facing the camera, even though the whole bodies of the people in the image are visible, detections were not made [47].

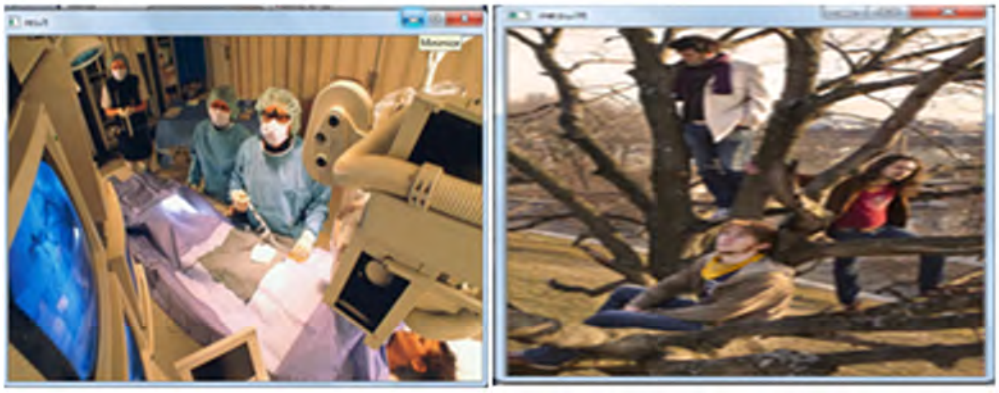

Figure 4-16 Full Body Haar Classifier Results on TestImage22 and TestImage15

In Figure 4-17 the human detections occur were the faces are facing the camera. The two people facing away from the camera and the person with the hidden head in TestImages23 and TestImage36 are not detected, respectively.

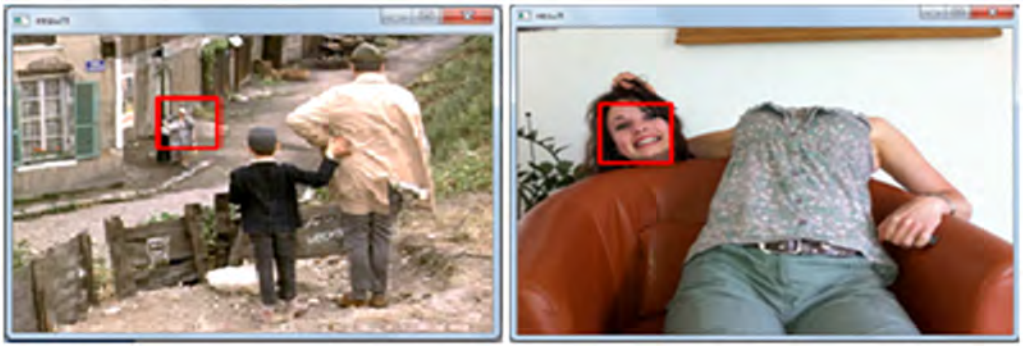

Figure 4-17 Full Body Haar Classifier Results on TestImage23 and TestImage36

The lower body Haar classifier was run on a video clip. Figure 4-18 shows that human detections were made despite there being no lower part of the body appearing in the frame [48].

Figure 4-18 Lower Body Classifier Results

From the results above it is observed that the model is capable of performing human detections. The models are based more on the face for detections. To use these methods a new classifier need to be re-trained. A legs detection model has previously been successfully trained [49]. A better human detection will thus be trained for human detection. The training process is summarized in Figure 4-19 below.

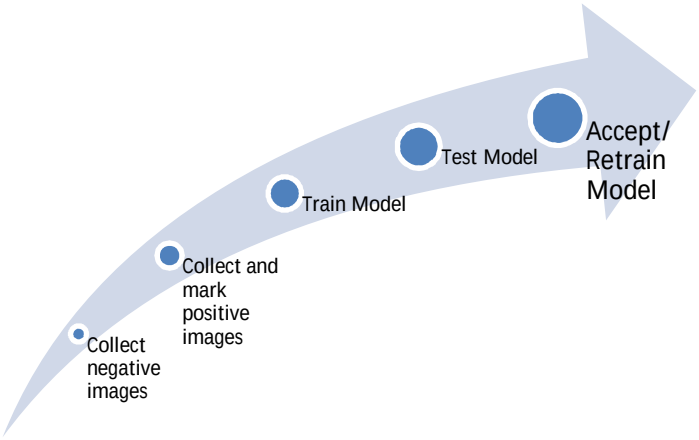

Figure 4-19: Haar Classifier Training Process

Thousands of images have to be collected for both the negative and positive training images. The negative images are those that do not contain the object of interest and the positive images are those that contain the object of interest. The object of interest has to be marked in positive images with object marker.

4.2.2. Final Design

The best approach to the problem was determined. Haar-Cascades classifier was the method to be used. This section presents the development of the Haar-Cascade model for human detection. The various stages of the model development are presented and the models developed in the process to get a model with a high positive detection and a low false negative detection.

4.2.2.1. Full Body Model

The model to be used as discussed before is the Haar-Cascade. Images had to be collected to train the human detection model. The negative images were collected first. Collecting these took some time. These images were jpeg images that had different backgrounds and did not contain any human part in them. A sample of the images used is shown in Figure 4-20 [50]. The negative image filenames are stored in a text file for use in training.

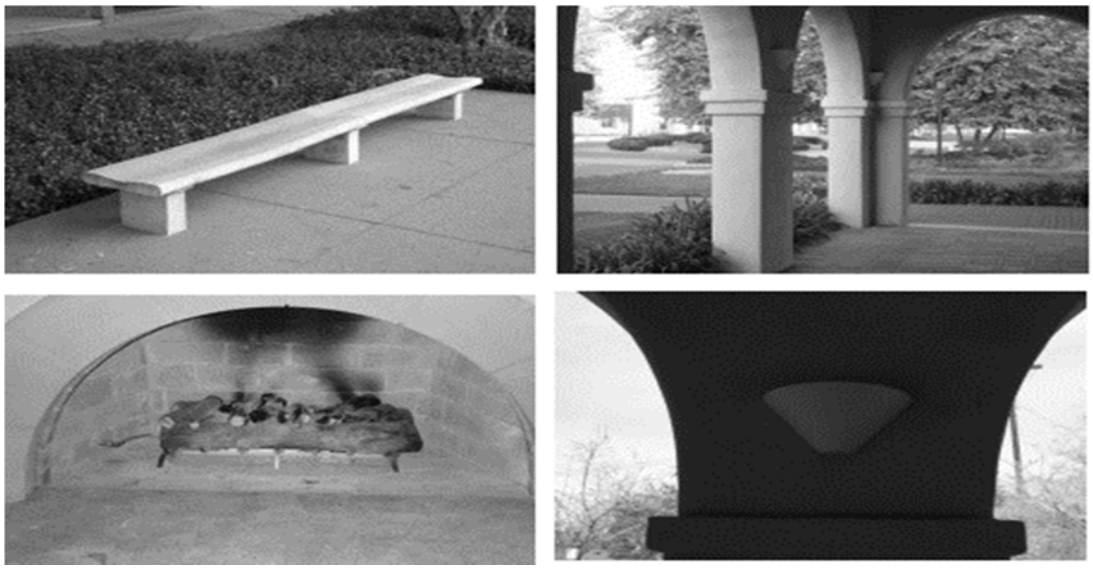

Figure 4-20: Sample of Negative Images Used in the Haar-Cascade Training

Positive images were then collected. These are the images that contained human bodies. In the initially collected data the aim was to collect images that contained people in more or less the similar poses you would find victims in the disaster scenes. A sample of image used in the preliminary model is in Figure 4-21 [51].

Figure 4-21: Sample of Positive Images Used in Training

The object of interest was marked using the ObjectMarker.cpp source code. The image file type was changed to .jpg and the fixed scale was changed to false. Object maker creates a .txt file with the object of interest enclosing rectangles and image filenames. This text file is used to form a vector file that is used in the training. Haar-Cascade model training usually have an optimum combination of negative images, positive images, number of stages and the size of vector images. One can try to find this for the particular model they are training, but will consume a lot of time as Haar-training can take anything from a couple of hours to weeks, depending on the data being used. The initial model was trained using 2024 positive images, with 5000 samples for the vector file, and 2032 negative images. The images in the vector file were 20 pixels × 20 pixels and the training was for 20 nodes. The model training was stuck on the 4th Node. To determine if the model was stuck the training process was observed over time and if the output was repeating a certain sequence continuously a conclusion was reached that the model was stuck. To validate this possible solution, some models were left for days while this repetitive pattern had started and were found to stay in that same node for that period, with no progress. Some parameters were changed and the training was restarted. This was done to different models until a model that completes the training was found. The models' false alarm rate (FAR) was checked and the results for all the models are tabulated in Table 4-5.

Table 4-5: Full Body Haar-Cascade Models Trained

Attempt	Model Characteristics related to the attempt	False Alarm Rate	Number of nodes attained	Number of positive images used	Number of negative images used
1	Possible cause of model getting stuck is pictures were human were not clear	0.0976893	5	2024	2030
2	Removed the unclear images from 1st attempt	0.0698266	10	1924	2030
3	Added more negative images (Grey Scale)	0.0310018	16	1924	2949
4	Added more negative images (Colour)	0.149952	4	1924	5719
5	Added more negative images (Colour)	0.214616	3	1924	6769
6	Converted all negative images to greyscale	0.0131064	14	1924	6769
7	Remove the rotated versions of positive images	0.268243	2	481	6769

8	Used data from attempt 6. Add more Negative images. Training successfully completed.	0.000443912	20 (Completed Training)	1924	8771
9.1.	Added more nodes to 8th attempt training data to try and get 40 nodes.	0.00022577	24	1924	8771
9.2.	Added more negative images	0.00215357	16	1924	10719
10	Used 7th attempt data with 9.2 negative images	0.0435	7	481	10719
11	Convert positive images to grey scale images and make a vector file of grey scale positive images using 8th attempt data	0.01606	12	1924	8771
12	Add more positive images to 8th attempt data	0.0474475	8	2912	8771
13	Add more positive images	0.522545	1	2912	10719

	to 9.2 data				
14	Used new positive image data set with 10^{th} attempt negative images	0.01743	9	988	10719
15	Used the new positive images dataset with 8^{th} attempt negative images. Used 24×24 for positive vector images and got stuck on node 0, Then changed back to 20×20. Used 24×24 to convert model training data to the html file used in performing detecetions	5×10^{-5}	23	988	8771
16	Used new positive image data set with 10^{th} attempt data	0.0293444	10	1469	10719

From Table 4-5 it is observed that the optimum number of negative images is 8771 with the first positive images dataset. Addition of more images had a negative impact on the false alarm rate, where the rate increased by a factor of 10 from 9.1 to the 9.2 model training attempt. Figure 4-22 shows the effects of grey scale negative images on the false alarm rate. The false alarm rate

decreased, with an increase of negative images, up to a certain point of 0.00044%. This corresponds to the lowest point, at 8871 images, on the graph in Figure 4-22.

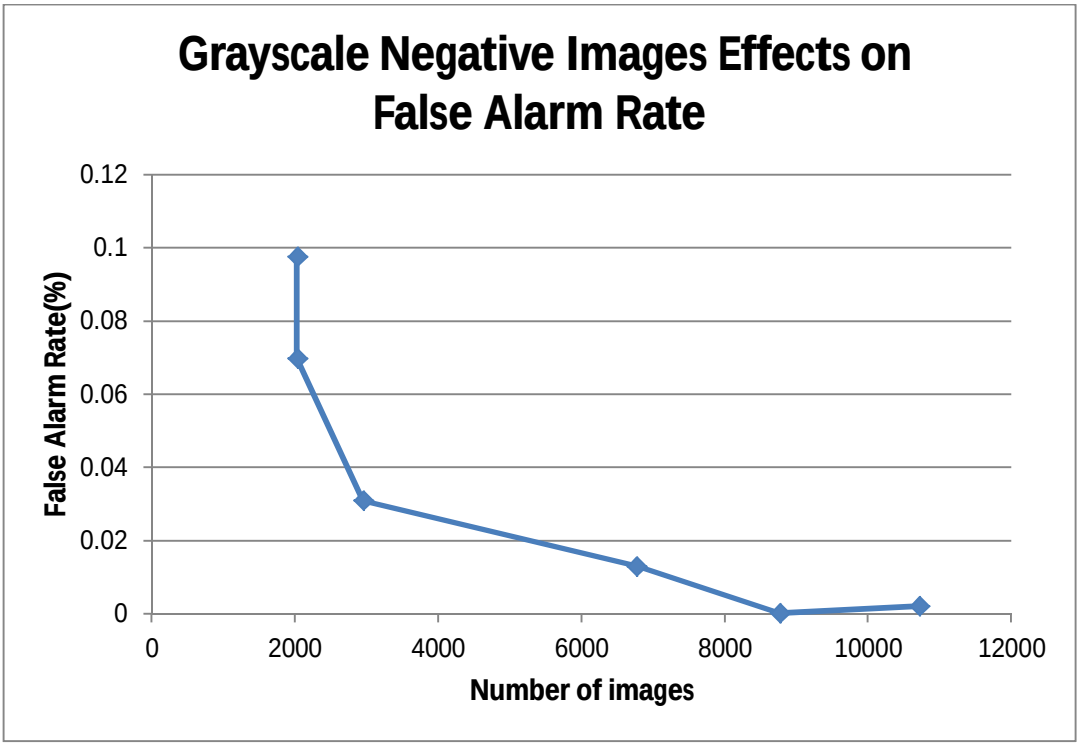

Figure 4-22: Effects of Negative Gray Scale Images on FAR

Grayscale negative images and color images have different effect on the training. The addition of color images to negative images drastically increases the FAR. Addition of the color images in the training resulted in the two biggest FAR values as shown in the graph in Figure 4-23. The rest of the data in the plot is the same that is shown in Figure 4-22. Attempt 5 and attempt 6 data show the effects on false alarm rates for the similar negative image data, where one model was trained with a combination of the approximately equal grey scale and color images and the other model with the same images all converted to grey scale images. The FAR dropped from 0.2 in attempt 5 model to 0.01 in attempt 6 model.

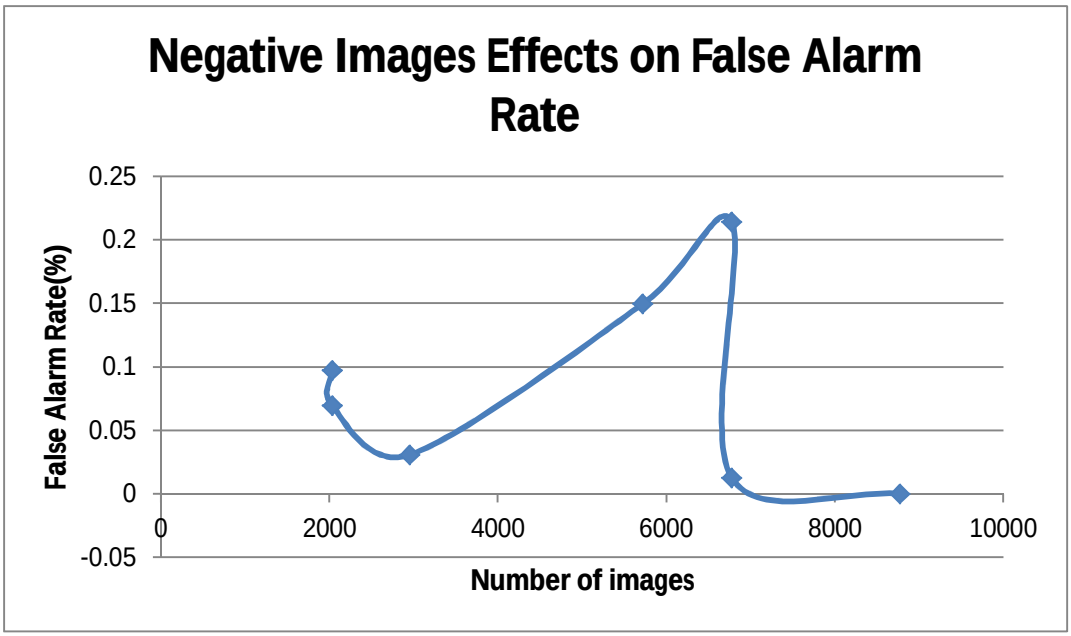

Figure 4-23: Effects of Negative Images on False Alarm Rate

Converting positive images from the 8th attempt data to greyscale and using them in attempt 11, increased the false alarm rate by a factor of 36 which is not a desired attribute in this training. New positive images were collected. With respect to the number of nodes achieved and the false alarm rate achieved, these new positive images seemed to have the same effect as the old positive images data set. This can be observed in attempt 16 and attempt 10, where the false alarm rates are within a 2% range from each other, with a false alarm rate of 1 is equal to 1×100 %. This observation is attained with the number of nodes being reasonably close to each other at 10 and 7 respectively. The new positive images dataset in this specific case were able to achieve more nodes in the training process and hence a better false alarm rate, even though the margin of the difference in the initial models was not big. The new positive images did bring a positive change, and hence the new images were viewed as a better dataset as opposed to the initial dataset. The 8th attempt cascade was tested and was found to have a high false alarm rate. This was observed when a human was present in the video frame. In the absence of a human in the video capture the false detections were present in a low percentage of the frames. Since the objective is to find human presence, this model can be used. The desired false alarm rate must therefore be greater than that in the 8th attempt. A low false alarm rate (1×10^{-5} or less) is desirable [52]. Some of the attributes that can be the cause to the complication in not completing most training attempts and to obtaining a high false alarm rate can be the number of features used. The human body has a

lot of different features. The clothes, worm by a person, have an impact on training. The human bodies used in training were in different poses. Features for two people facing forward, with one slightly facing to his left with his right arm raised and the other facing forward with his hand besides him are different. In training, the model tries to get features that make the object. With a lot of different features it is hard to get a good model. It is advised that as few features are used or a part that has features that can remain almost uniform be used [34]. The new positive image data set used, like the first one, had the whole human body in different poses. This data set however avoided the use of clustered human scenes where possible. Figure 4-24 [53], Figure 4-25 [54] and Figure 4-26 shows some new positive images dataset sample.

Figure 4-24: New Positive Images Dataset Sample Back View

Figure 4-25: New Positive Images Dataset Sample Side View

Figure 4-26: New Positive Images Dataset Sample

From Figure 4-24 it is observed that an image of someone facing away from a camera has different features, apart from the human body shape that can be used to classify them. The head cannot be used as it has no distinct features when someone is facing away from the camera. The upper body (arms or hands) model might be more useful in a case where a human is facing away from a camera. The legs can also be useful in such situations. Further research was done to find out if these models can provide a better solution. Figure 4-27 shows the factors that affect the model accuracy. These variables were manipulated in the training to achieve the optimum model.

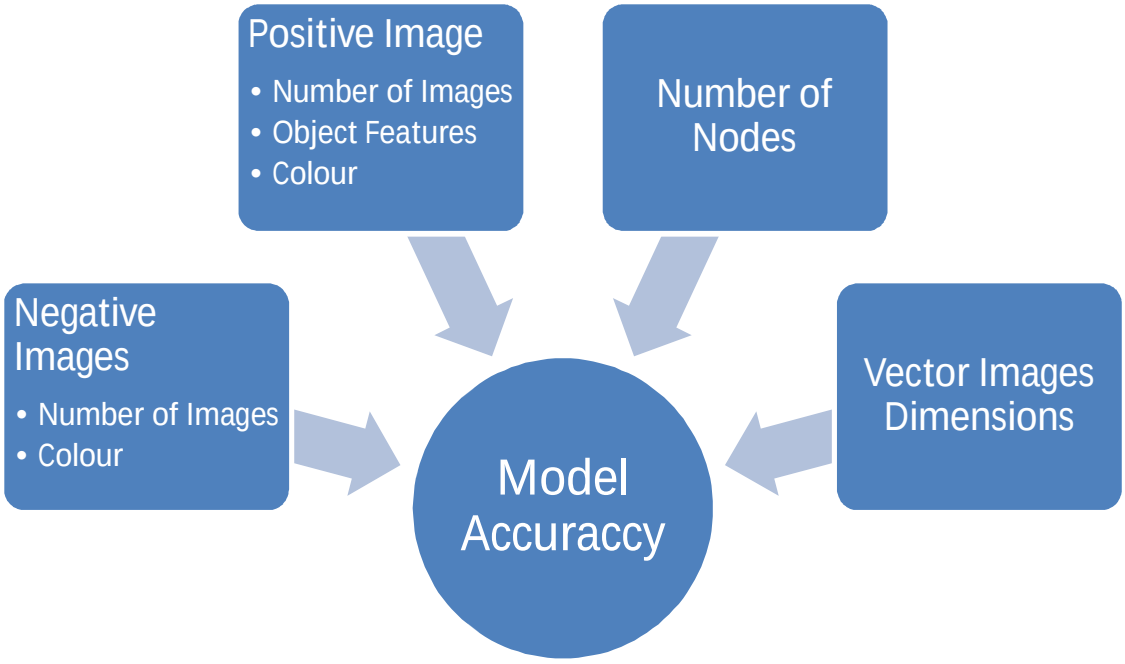

Figure 4-27: Factors that affect model accuracy

The model with the new positive data set was trained using the 8th attempt negative images. This was labeled the 15th attempt. While waiting for the model to complete the training, other alternatives were considered. Parallel models were considered as a solution (see section 4.2.2.2.). In the 15th attempt training, the vector file images were initially set to 24 pixels × 24 pixels. This training failed after the first node. The vector image sizes were then adjusted to 20 pixels × 20 pixels and the training continued. The model training went up to 23 nodes. On building the cascade html file an error occurred with the vector file sizes, as the first node had not been deleted. The cascade html file was build using 24 pixels × 24 pixels instead of 20 pixels × 20 pixels to accommodate the first node. This model had a false alarm rate of approximately 5×10^{-5}. The same model was retrained with 20 pixels × 20 pixels vector images from the 1st node and a false alarm rate of 4.9×10^{-5} was achieved. This model was labeled attempt 17 to distinguish from the 15th attempt 24 pixels × 24 pixels, but is not included in Table 4-5. The 24 pixels × 24 pixels, 15th attempt model was tested against 8th attempt model. The reason for this is that the 24 pixels × 24 pixels cascade had a lower false alarm rate when tested against the retrained 20 pixels × 20 pixels model. The models revealed that cascade models behaved better on video input than still images. The models were tested on the camera input. The models were able to detect legs, hands, the frontal and the side face. Figure 4-28 to Figure 4-30 show the results for 8th attempt model and 15th attempt data on the

left and right respectively. These tests were run after the development of the code that loads and runs two models in parallel (Section 4.2.2.2) and the models were tested in parallel on the same frame. This is because the parallel model code was developed during the 15th attempt training. Figure 4-29 shows the results on the frontal upper body.

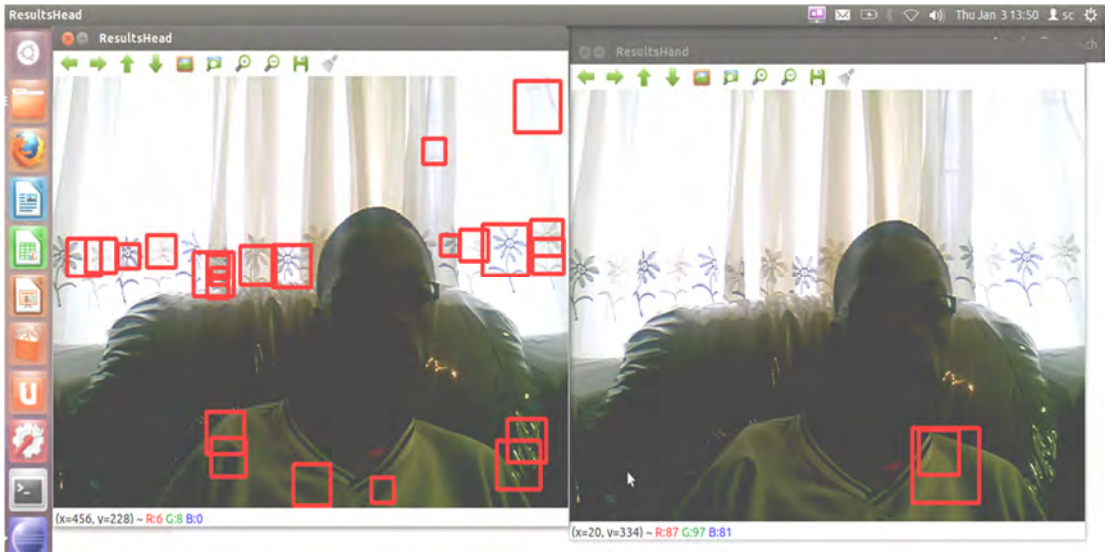

Figure 4-28 : 8th Attempt and 15th Attempt Frontal Upper Body Detection

Figure 4-29 shows the frontal detection with the camera from an angle where the camera is below the object. Figure 4-30 show the detection from the side view. The results show that the false alarm rate for the 8th attempt model is high as compared to that of 15th attempt data. This is observed from the amount of false negative in the 8th attempt model results. The 15th attempt model had a single false detection in Figure 4-29 and Figure 4-30.

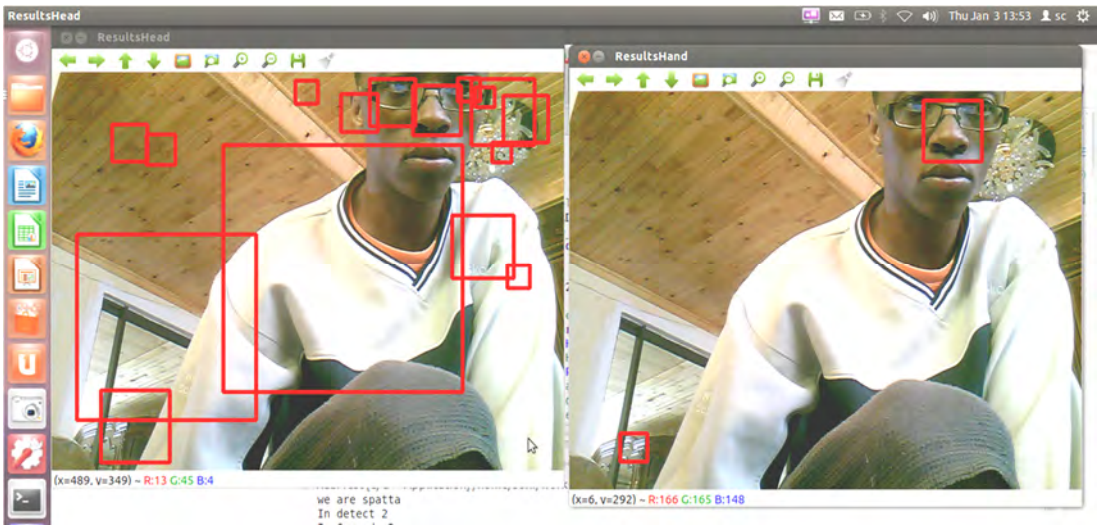

Figure 4-29 : 8th Attempt and 15th Attempt Frontal Upper Body Detection Camera From Below Angle

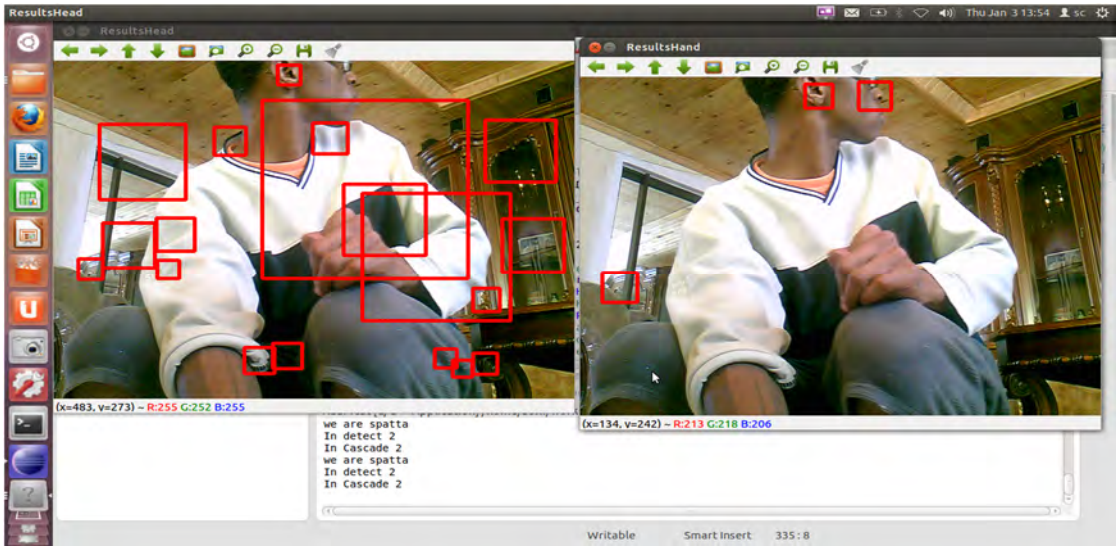

Figure 4-30 : 8th Attempt and 15th Attempt Side Upper Body Detection Camera From Below Angle

The test images used in section 4.2.1.2 were evaluated using the two models. The results are shown in Figure 4-31, Figure 4-32 and Figure 4-33. These figures show the results for attempt 15 and attempt 17 model on the left and right respectively. In Figure 4-31: Test Image 11 Results for Attempt 15 and 17 Respectively. Figure 4-31 attempt 15 data model is applied does not detect any human, whereas on Figure 4-30 the same model is able to detect a human facing away in a video feed frame. Attempt 17 is able to detect with multiple detections on the human facing away from the camera.

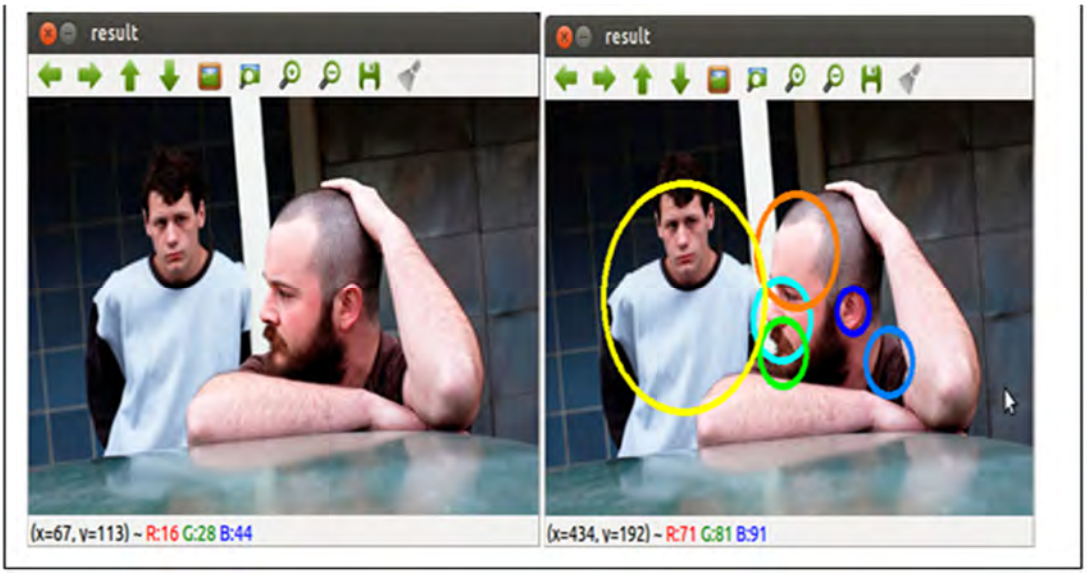

Figure 4-31: Test Image 11 Results for Attempt 15 and 17 Respectively.

In Figure 4-32 both models were able to detect the human, with attempt 17 model having multiple detections and 3 false detections. A similar case is observed in Figure 4-33, both models have a human detection and a false detection, with 17th attempt model having 2 false detections and a multiple detection on the second human. The 15th attempt model was taken as the final model as it had a lower false alarm rate and was able to perform human detections on videos with the human in different positions. The final model was further tested and the results are presented in chapter 5.

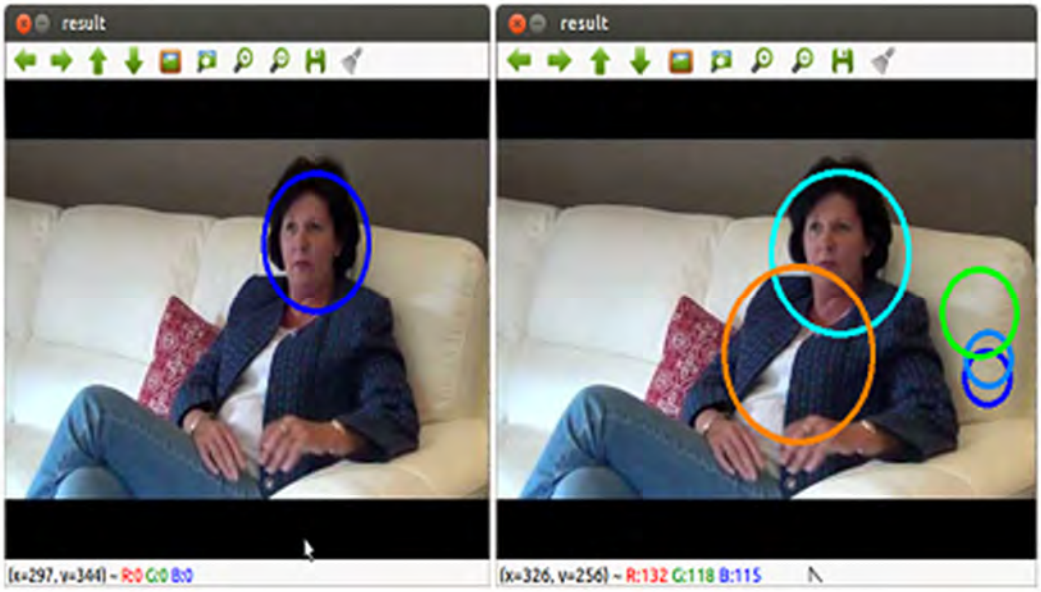

Figure 4-32: Test Image 13 Results for Attempt 15 and 17 Respectively.

Figure 4-33: Test Image 15 Results for Attempt 15 and 17 Respectively.

4.2.2.2. Parallel Models

This section looks at the possibility of using separate models in collaboration to perform human detection. This process consists of building models of different human body parts that are unique to humans. These models would be used to identify humans as opposed to a full body human. This process will make the training process better in terms of human feature that the model needs to use. The object has features that make it stand out. A model must be trained were a variation of the object is used without introducing variations that may introduce features that make the object difficult for the training model to identify as the same object as the rest of the training data objects. The data must be such that the features in the objects are not varied across the data, thus creating more features. For example in the previous model, the human may be restricted to facing forward only. This model would eliminate side and back views and their associated poses. The approach was to have one frame where different models could be loaded and then applied. The objects in mind were the head (frontal face), legs and hands. These were chosen because they are parts of the human body that are unique to human beings. Their presence in a frame can be used as a base to deduct that a human is present in the frame. The frontal face Haar-cascade model that is distributed with OpenCV would be used for the head, and then a model for the hands and legs would be trained. The initial step was to research how multiple Haar-cascade models could be used on the same frame. The initial approach was to load all the models and run them one after the other in series, in a similar manner to cascade models. The model would stop on detection of either of the objects as opposed to exiting on no detection in the early stage in Haar-Cascades. The other models (stages) are evaluated when there is no detection in the preceding models. Figure 4-34 shows this approach.

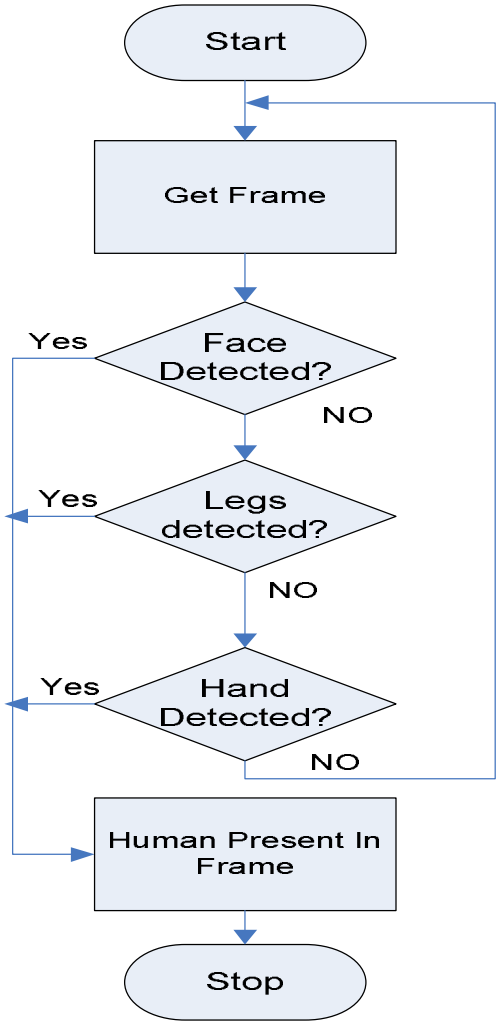

Figure 4-34: Algorithm for Using Multiple Models for Human Detections

The models would be loaded and then applied on the same frame. This concept was researched and tested. There were no publications found on multiple models being loaded and used on the same frame. Loading multiple models was attempted in code. This did not work and the model could not be tested on one frame. Multiple treading was used before in the research. A proposed concept was to use multithreading and run multiple interfaces of the same frame, where a separate model will be loaded and run in a separate thread. This was first tested by developing software for two models and loading them. For testing the principle the pre-built models were used. Figure 4-29, in Section 4.2.2.1 shows the software running two separate models for testing. The main program runs and uses a function to do the detection and marking of detected objects. The other thread loads the cascade and runs the detection in the directly. Function use was attempted in the second thread but caused errors.

4.3. Chapter Summary

This chapter presented the system artificial intelligence for the system integration and UAV control. The human detection system is also presented. The navigation had to be performed from a safe location. A ground station was developed to control the UAV. Wi-Fi was used as the medium of communication between the UAV and GS. The server-client architecture was proposed for sending video stream from the UAV to the GS, and to send control instructions to the UAV from the GS. The image processing was to be done on the ground station. To simplify the system the image processing was to be done on the UAV. The on board hardware has enough processing power for the image processing. The control instructions are sent from the GS via Wi-Fi. TeamViewer was used to access the UAV. TeamViewer has the capabilities to transmit keyboard inputs to the connected device. The keyboard characters are used to control the UAV. Code is written to detect the keyboard input on the UAV. The characters received by the main processor (ODROID-X2) are sent to the ArduinoMega 2560 board. This board is the interface between the main processor and the flight control board (ArduPilotMega). The ArduinoMega board interprets the characters and sets the PWM signals for controlling the roll, yaw, pitch and throttle of the UAV. These PWM signals are manipulated in the APM for UAV flight.

The human detection system design stages were presented. The design utilised open source computer vision library, OpenCV. Different matching methods were considered and tested. Template matching uses a template and scans the template image in the target image. This method was found to work best when the template is a sub-image of the target image. Chamfer matching was also researched. This method is a form of template matching that uses a binary image and binary edge template. The method has disadvantages when objects are slightly occluded and shading on the input image and template. Haar-cascade classifiers were researched. The OpenCV distributed models were tested and found to work for the application but were not able to detect humans from the side views and the back. A Haar-cascade model was developed to eliminate these short comings. The negative images used were found to have an effect on the model, where an optimum number of images for this model were found to be 8771 greyscale images. The initial positive dataset had to be changed as it contained a lot of clustered human images, distant humans in the image who in some cases were not clear and human in a lot of various postures. A model was established with a false alarm rate of 5×10^{-5}. This model was tested and found to detect human in different postures. Parallel models were also researched. In this

application models would be built for each human feature and the models run in parallel on the same stream in different windows. This required the use of threads. This approach would require development of many and was not pursued further on success of the full body model developed.

5. Experiments and Discussion

This section presents the experiments carried on the final mode to determine its behavior and accuracy. A comparison of the final model, preliminary model and the OpenCV distributed model is also presented. The comparison also provides the justification for using the final model as opposed to the distributed model and why the preliminary model had to be improved.

5.1. Final Model Testing

The final model was trained successfully and its application in real life situation was tested. The tests were to determine how well the model detects people in different poses. The test was also to check how well the model can detect people when parts of their bodies are obstructed. The model was implemented on the ODROID-X2 Board. This had the following properties which make it suitable for use on a UAV:

- Compact [90 mm × 94 mm]
- Enough Processing Power [1.7Ghz Quadcore ARM Cortex-A9 MPcore, 2GB Ram]
- Runs Open source Ubuntu or Android

The model was loaded and the tests performed with a person approaching the camera, and the camera (UAV) approaching the person. A live video stream from the camera was used. The experiment was done in a light indoor surrounding. The distance between the person and camera was kept between zero and 5 meters. The person rotated to all four sides (front, back, left and right) in each instance when he was getting close to the camera. The person remained in the same side for approximately 3 seconds. The detections were tabulated for four experiments (See Appendix J). The average detection for each side was determined. Figure 5-1 shows the average detection over the distance for the front and back view. For the front view the model was able to detect the person each time and gave an average of 100 percent. With the back view the model had better detections at 2 and 3 meters from the camera.

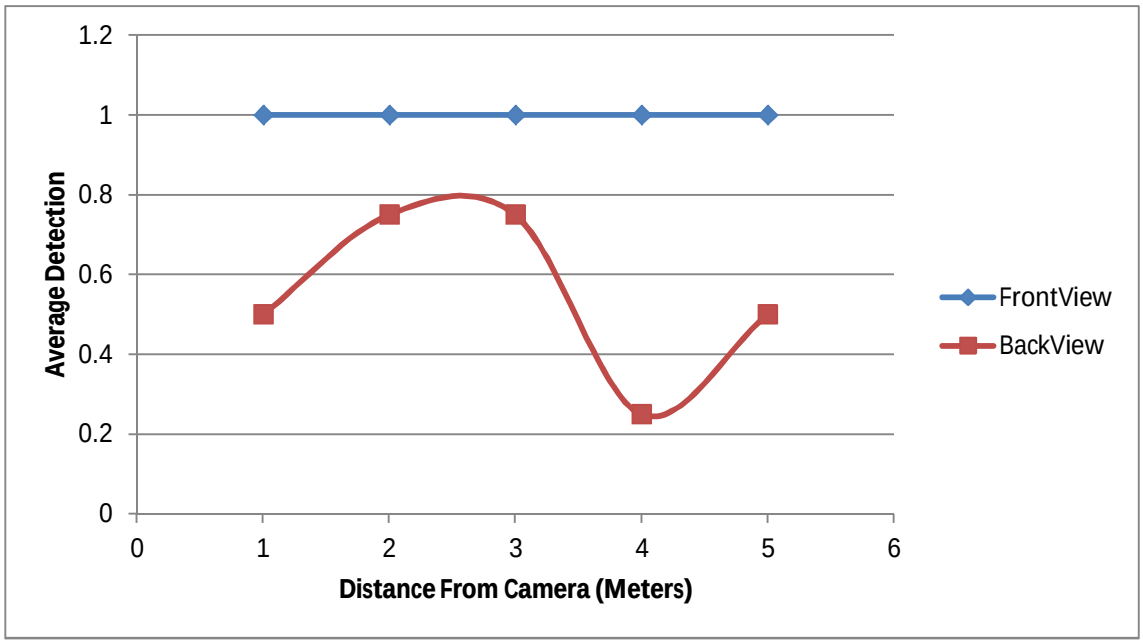

Figure 5-1: Average Detection Against Distance

The front view used mainly the face for the detection apart from using the hands and legs. The back view model used the arm posture (upper body) and hands more to detect the person. In Figure 5-2 the model uses the hand and the head to detect a person from the back. The model detects the upper body and face in the frontal model. The model can detect frontal view better than the back view. The side view detections by the model are good when the camera is closer to the object. The software is configured to mark the detected object using red squares. The detection drops from 100% when the object's distance from the camera becomes greater than four meters. The average detection rate drops to 75% and 25% for the left and right side respectively. Hence the detection rate will increase as the UAV approaches a victim in a rescue mission.

Figure 5-2: Back and Frontal Detections

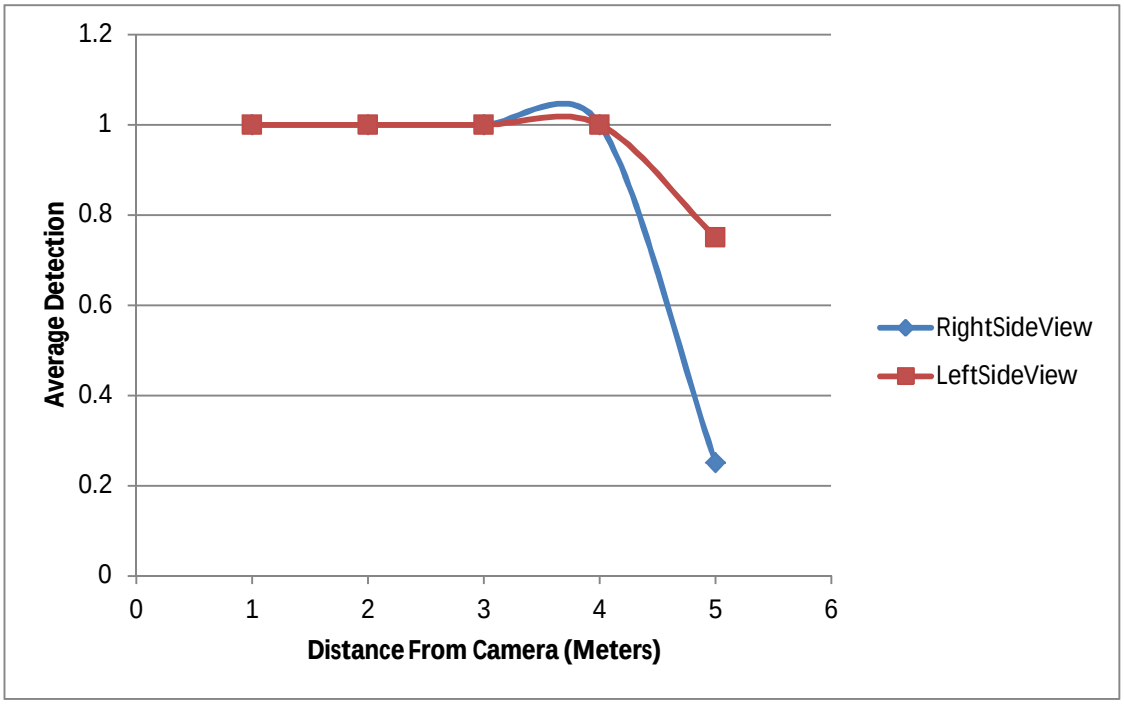

Figure 5-3: Average Detection Against Distance

Figure 5-4 shows the detections when the person is 1 meter away from the camera and when the person is 4 meters away from both sides. The model can therefore be used in real life detection and has an acceptable combined detection rate.

Figure 5-4: Side Detections at 1 Meter (Top) and 4 Meters (Bottom) From the Camera

Figure 5-5 shows the model detecting a human hand. The results below are from consecutives frames captured every second from the screen video recorded while experiment was in progress. The frames follow each from left to right, top to bottom.

Figure 5-5: Final Full Body Model Detecting Human Hand

The full body model was found to also be able to detect a person approximately 3 meters away from the camera when obstructed. The model detected the foot and hand. This represents a case where a human can be trapped under a rock with a limb showing. The frames follow each from left

to right, top to bottom are from a record of the results onscreen during the experiment. These frames were taken a second apart.

Figure 5-6: Final Full Body Model Detecting Human Hand and Foot

5.2. Model Comparison

OpenCV has a full body human detection Haar Cascade model that is distributed with it. This model was initially tested before the development of the final model tested above and was found to have challenges in side view and back detections. A new model was developed and tested throughout the different stages of development. This section presents a comparison between the OpenCV distributed model (Generic model), the developed model in its preliminary stages as discussed in Section 5.1, and final model. The experiment was carried out with 4 videos for each of the human body side (left, front, right, rear). Videos were used so that the models are tested on the same input data, making the results comparable and the comparison reasonable. The videos allowed for 10 experiments on each side, with the person's distance from the camera varying from one meter to 5 meters. The lighting conditions were changed as opposed to the lighting conditions in Section 5 a). The lighting conditions were non uniform, darker toward the camera and lighter further from the camera. The later part represented a case where there could be sum light source that makes the rescue environment lighting non uniform. The videos were then processed in Ubuntu and the detection and false detection percentage recorded. The average detection and

false detection percentage at the different distances was then calculated and plotted. The videos contained one person and a positive detection was recorded as one (hundred percent). A false detection was counted as one, where one false positive detection was counted as 100% false detection. The aim is to get the lowest false detection and the highest positive detection, which is one (100%). The results for the average detection and average false detection are presented in Table 5-1-1 and Table 5-2 respectively.

Table 5-1: Average Detection

Average Detection												
	Rear			Front			Left			Right		
Distance	Generic	Prelim	Final	Generic	Prelim	Final	Generic	Prelim	Final	Generic	Prelim	Final
1	0.1	0.9	1	0.2	1	1	0	1	1	0.3	1	1
2	0.6	1	0.8	0.4	1	0.8	0.7	1	0.9	0.5	1	1
3	0.4	1	1	0.8	1	1	0.5	1	1	0.7	1	1
4	0.9	1	1	1	1	1	0.9	1	1	0.9	1	1
5	0.8	1	0.9	0.9	1	1	0.2	1	1	0.5	1	1

Table 5-2: Average False Detection

Average False Detection												
	Rear			Front			Left			Right		
Distance	Generic	Prelim	Final	Generic	Prelim	Final	Generic	Prelim	Final	Generic	Prelim	Final
1	0.4	2.5	0	0.6	2.7	0.1	0.6	2.7	0	0.9	3.6	0
2	0.7	4.2	0.1	0.8	5.6	0	0.7	4.7	0	0.6	5.6	0.1
3	1.1	5.4	0.1	0.9	8.1	0	0.7	3.9	0	0.8	7.3	0
4	0.9	5.5	0	1	6	0	1.2	5.9	0	1.2	5.7	0
5	0.8	4.9	0.1	1.1	5	0	1.2	4.7	0	0.8	5.5	0

The results in Table 5-1-1 and Table 5-2 were plotted. From the results it is observed that the average false detection is higher for the preliminary model, followed by the generic model and the final model has the lowest average false detection. The generic model's average detection is lower than that of the other two models. The preliminary model has a better average detection in comparison to the final model at 2 meters for all the sides except the right side, where the average

detections are equal at all distances. On the rear side the average detection of the prelim model is higher than that of the final model at 5 meters. From the results we can thus see that the low average detection and high false detection of the generic model does not make it ideal for the application. The preliminary model detection is the best, but the high false detection it has makes the model impractical for the application, as it is bound to give false impressions to the rescuers. The final model has a low average false detection. The final models average detection is high enough for practical use, with the minimum average detection of 80 percent at two meters on the rear and front side. The rear view average detection and average false detection results for the different models discussed is given in Figure 5-7.

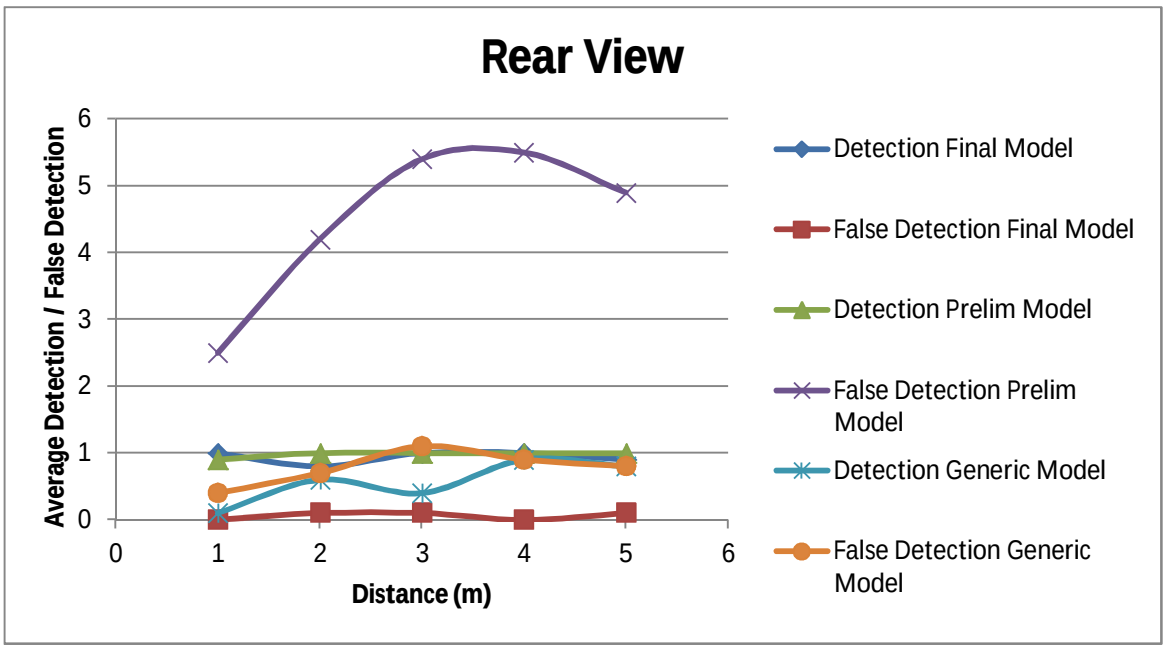

Figure 5-7: Rear View Models Results

Figure 5-8 shows the front view results for the average detection and average false detection for the three models discussed. The average detection and average false detection results of the left side is presented on Figure 5-9. Figure 5-10 presents the image processing results of the three models on the right side view.

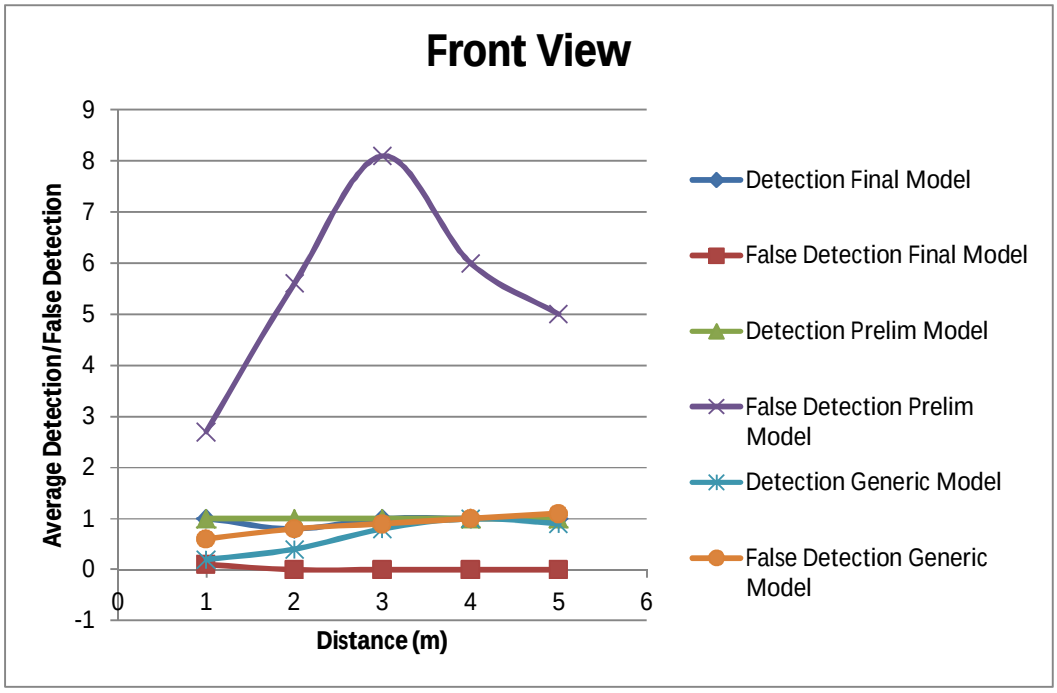

Figure 5-8: Front View Models Results

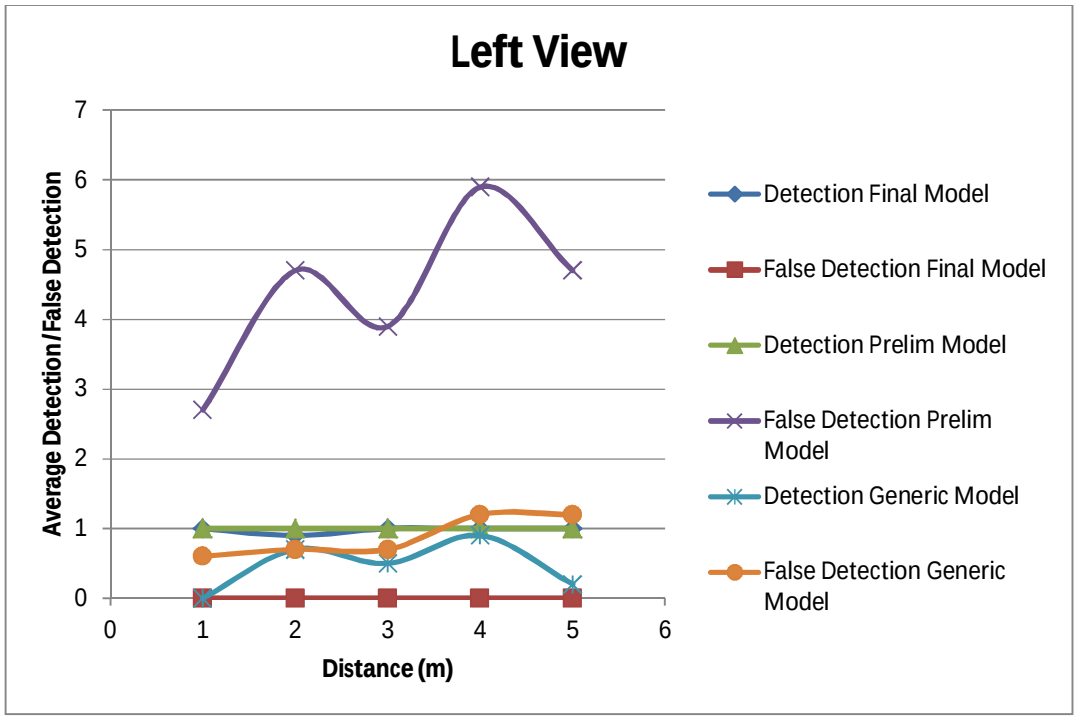

Figure 5-9: Left View Models Results

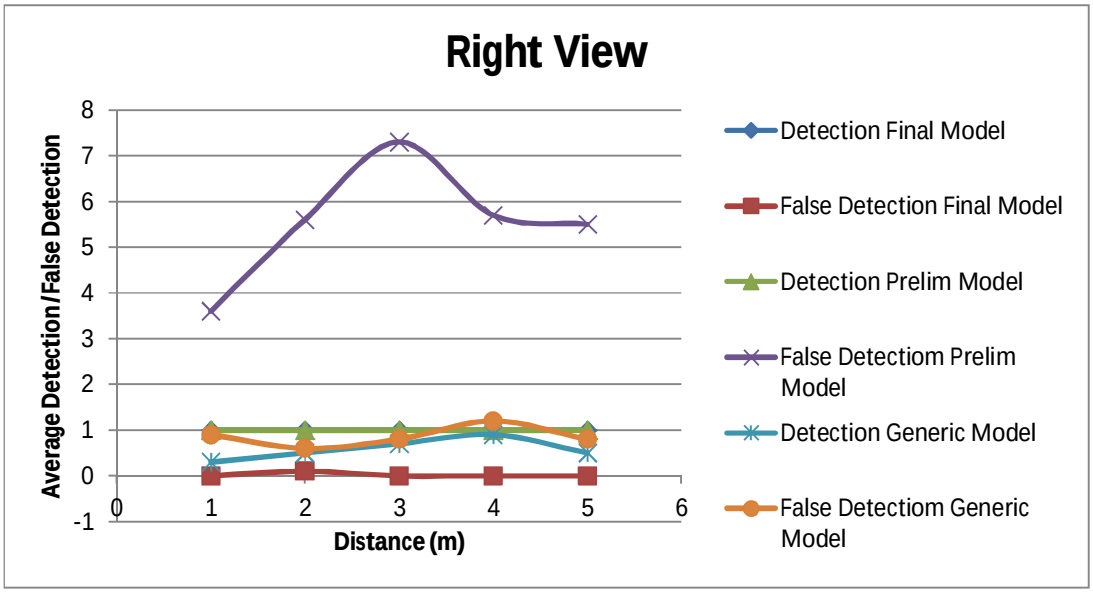

Figure 5-10: Right View Models Results

Figure 5-11 shows the results from experiment one on the left side view, with the object four meters away from the camera. The top left image shows the generic model's positive detection and a false detection. The top left image in Figure 5-11 shows the results from the final model where a positive detection is observed. The bottom image shows the preliminary model results. The preliminary model has six false detections and a positive detection. Multiple detections are seen on the object due to the model evaluating different parts of the image the entire human features within the model.

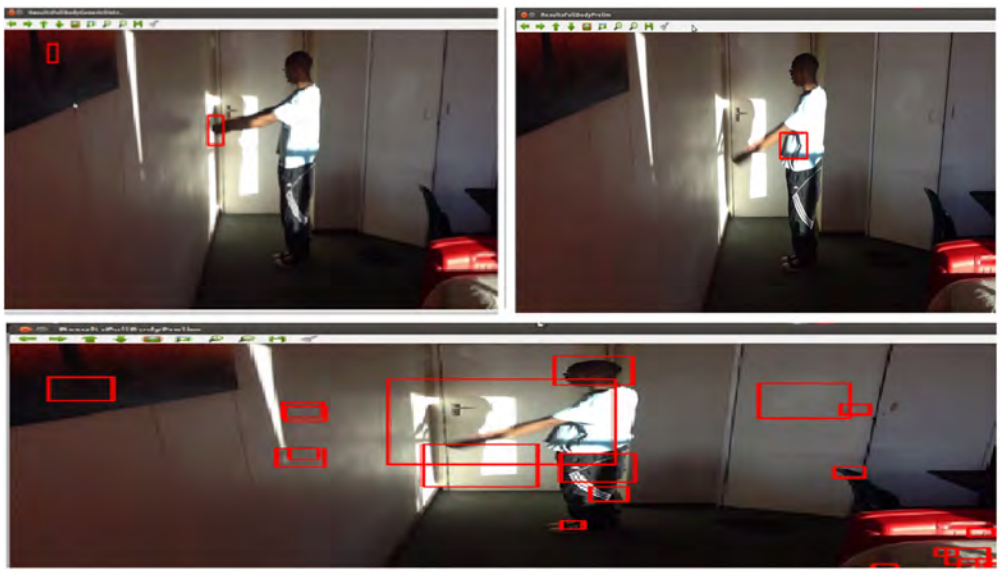

Figure 5-11: Left Side Detection of Different Models, Clockwise from Top Left; Generic Model, Final Model and Preliminary Model

The sample positive figure was used to test the three different models. Figure 5-12 shows the results of the generic model, while Figure 5-13 and Figure 5-14 show the results of the preliminary and final models respectively.

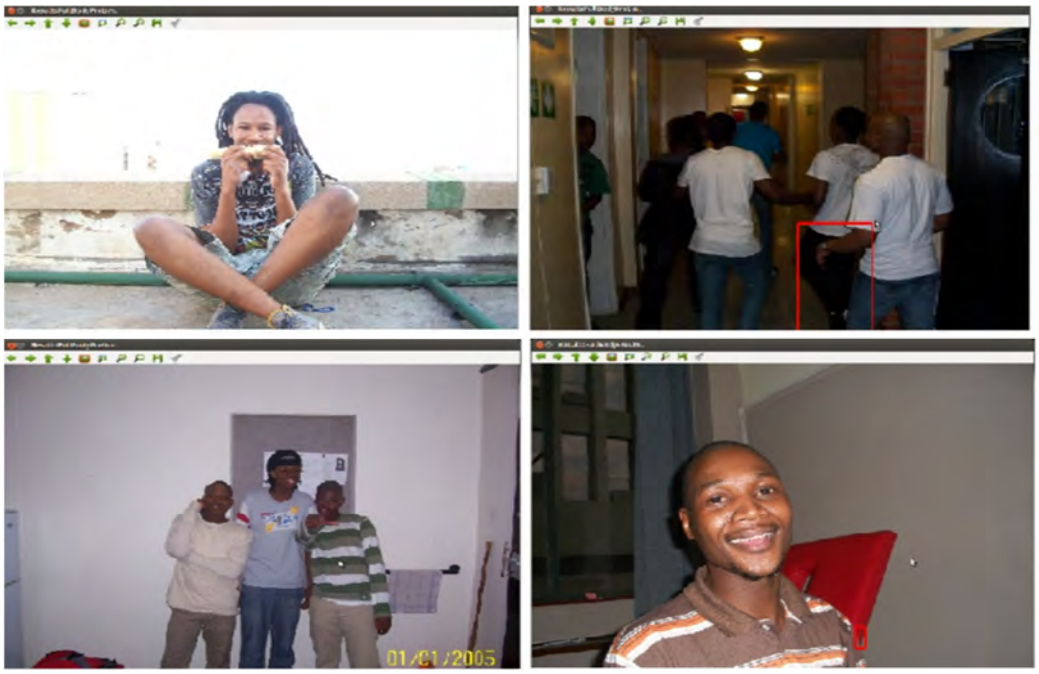

Figure 5-12: The Results of the Generic Model on the Positive Sample Images

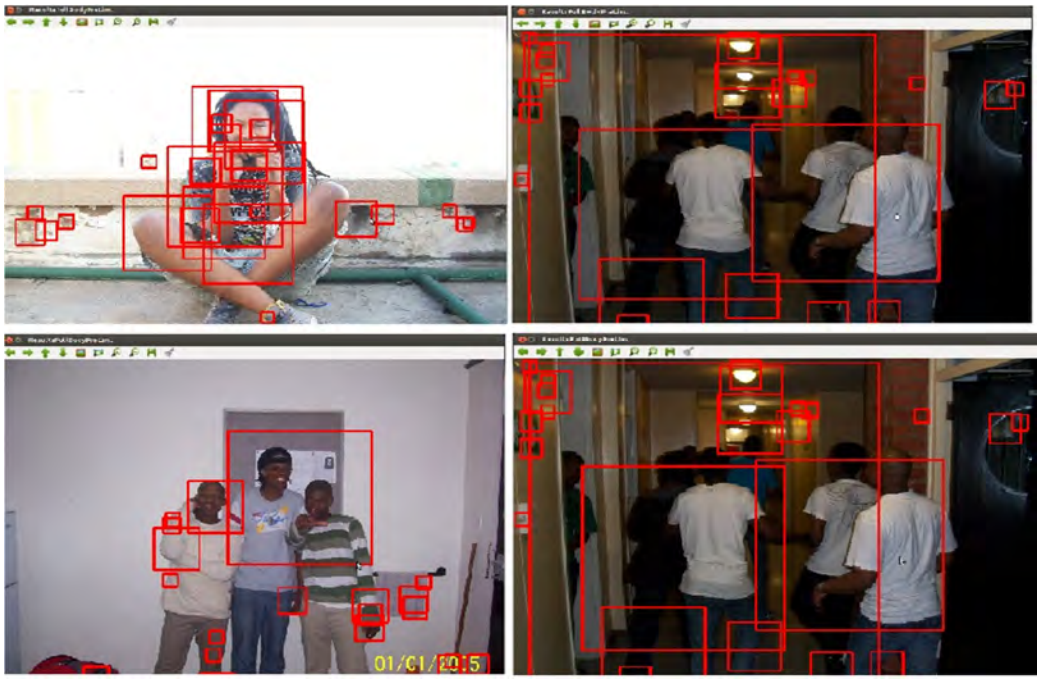

Figure 5-13: The Results of the Preliminary Model on the Positive Sample Images

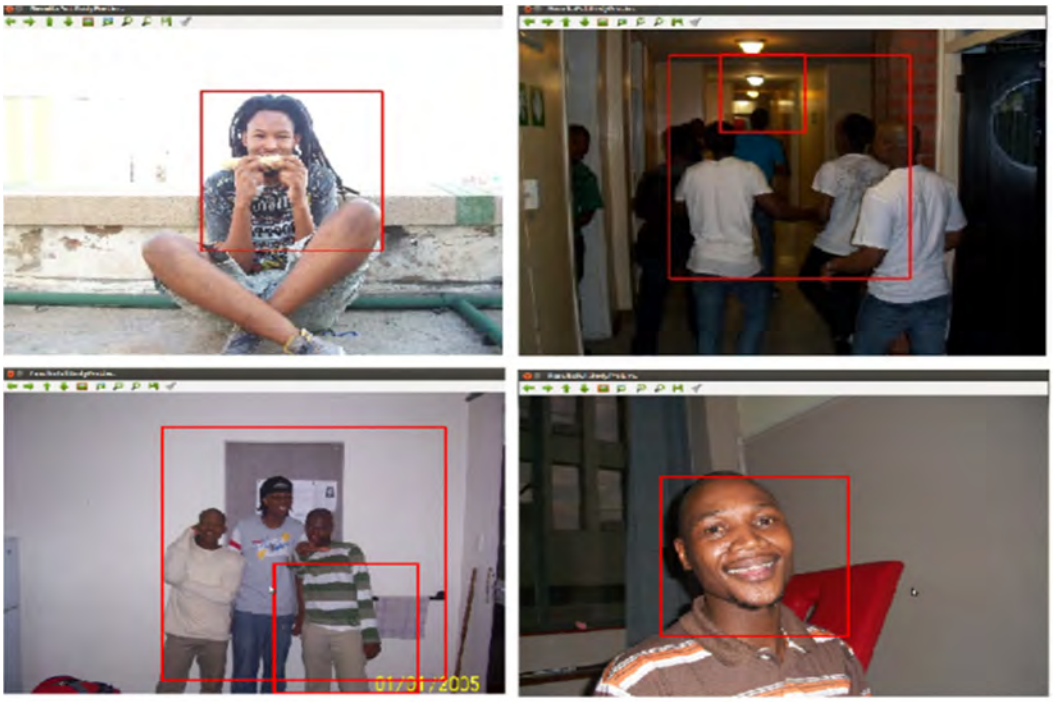

Figure 5-14: The Results of the Final Model on the Positive Sample Images

5.3. Chapter Summary

This chapter presented the experiments on the vision system and the developed human full body detection. The generic model has an average false detection that is higher than its average detection. This high average false detection that the generic model has makes the model impractical in search and rescue missions, as it will generate a large number of false alarms to rescuers. A similar behavior is observed with the preliminary model, where the model has the highest average false detection as compared to the other two models. The high average false detection the preliminary model exhibits is the price for its high average detection. The final model has a low false average detection and a high average positive detection. The average detection is greater or equal to ninety percent in all experiments except the two cases at two meters on the right and left side view.

The final model has a maximum false average detection of ten percent (10%) as opposed to the eight hundred and ten percent (810%) and one hundred and twenty percent (120%) maximum values from the preliminary and generic model respectively. Thus the final model was the best model to use for the application. This performance was shown experimentally on videos and images.

6. Conclusion

The UAV system was presented. The UAV had a quad rotor configuration, which is symmetrical and helps with stability. The hardware for the system integration was presented. The UAV had a main processor with quad core 1.7 GHz processors and 2 GB RAM. The systems computational power was utilized for the UAV vision system, which contains a human detection system. The processor also integrated the UAV system. The processor ran Ubuntu 12.04 operating system. The intelligence for UAV navigation and human detection was run on the main processor. Instructions were sent from a ground station to the on-board system via Wi-Fi. The RTL8191SU Wi-Fi module was found to be compact and compatible with the main processor and was used to connect to the wireless network. The control instructions received from the ground station are interpreted by the main processor and sent to the Arduino Mega 2560. The Arduino Mega 2560 takes the received instructions and generates control PWM signals. These signals go to the ArduPilotMega where they are used to control the UAV propellers. The propellers were connected to brushless motors which were controlled by the APM through the electronic speed controllers. APM had an Inertial Measurement Unit connected for the UAV for maintaining flight level and stability. The camera for the vision system was also connected to the main processor.

The development of the vision system was discussed. The need for the development of a model is shown by the comparison of the generic model and final model. The need to improve the preliminary model was shown in the results. The generic model has an average false detection that is higher than its average detection. This high average false detection will generate a large number of false alarms to rescuers. A similar behavior is observed with the preliminary model, where this model had the highest average false detection as compared to the other two models. The preliminary model exhibits a high average detection and a high average false detection, which makes this model not practical due to the false alarms it can raise to the rescuers. The final model has a low false average detection and a high average positive detection. The average detection is greater or equal to ninety percent in all experiments except the two cases at two meters on the right and left side view. It was observed during the model training that different factors played different roles. The selection of positive images samples had to be clear and the human features clearly differentiable. This characteristic was observed with a better model (final model) resulting from a positive sample with clearly visible people, in fewer complex poses, as opposed to the sample with people in complex poses and in clustered scenes.

The negative image sample used for training had an optimum value, which resulted that if the sample size was increased any further, it had a negative impact on the model. For model training, the positive data needed to be easily interpretable, to allow for the model to learn without confusing it what a human looked like. The Image processing was done on board the UAV and the results were available on the ground station.

The final model had a maximum false average detection of ten percent (10 %) as opposed to the eight hundred and ten percent (810 %) and one hundred and twenty percent (120 %) maximum values from the preliminary and generic model respectively. Thus the final model was the best model to use for the application.

6.1. Summary of Contributions

The contributions to the research can be summarized as follows:

a. An overview of the South African mining fatalities and the need for research on search and rescue was presented. An overview of current UAVs and UAV research was presented. The different applications for UAV were presented. A UAV was assembled as a platform for applications in mining search and rescue applications. A UAV is used as opposed to ground vehicles as similar research work is already being undertaken in the author's research group. A UAV has advantages of not needing a surface to travel on.
b. A ground station was developed using a PC running Linux Ubuntu 12.04. The operating system was based on the develop software running requirements. This was to cater for features like multi-threading which require a UNIX structure to run.
c. Review of available object detection models for human detection. Development and testing of a human detection system based on Haar Cascade Classifier with the ability to detect humans in different poses with a low false detection. The model has the ability to detect humans using different human features from different sides of the human body.
d. Development and establishment of a communication interface between the ground station and the UAV, utilizing Wi-Fi as the medium of communication. The complexity of the interpretation of signal is simplified by having access to the UAV through TeamViewer and developed software onboard the UAV to interpret GS instructions. Testing of the communication system was performed with the use of the local Wi-Fi network.

e. The control interface that enabled the interpretation of the flight control instructions was developed. This was done in by hardware and software mechatronics integration. The Odroid-X2 connected the onboard system to the UAV onboard system to the GS. The Odroid-X2 was further connected to the Arduino Mega 2560 board which was connected to the ArduPilotMega. Software was developed and run on the Odroid-X2 board to receive control instruction. These instructions were transferred to the Arduino board, where software was developed to covert the instructions to PWM signals that are used by the APM to send speed command to the ESCs and then the motors.
f. The integration of the UAV on board system for optimization of hardware utilization for search and rescue applications. This was achieved by using the Odroid-X2 board and utilizing its computational power for image processing and handling communications. The Odroid-X2 has a Wi-Fi module for wireless communication and USB for connection to other peripherals, which allowed connection to the Arduino board for transferring flight control instructions to the APM.

6.2. Future Work

Future work related to UAV vision systems and, search and rescue missions can include the following:

a. Applying the human detection model in other types of search and rescue robots such unmanned ground vehicles (UGV).
b. Researching the application of human detection models in game reserves where a signal can be sent to game rangers if there is a person that is in an area without authorization. This application can help reduce rhino horn pouching.
c. The application of 3D mapping on UAVs in mines to give the path map to rescuers as mentioned by S. Motepe et al. [55].
d. Researching and developing inexpensive micro sized Wi-Fi repeaters for use to extend the network in mining environments. UAV can then carry these repeaters and extend the network as they fly in the mining/rescue environment.
e. Researching the optimal placement of Wi-Fi repeaters to get the optimal coverage for UAV connection to the ground station

References

[1] H. van de Groenendaal, "Control and communication in mining applications", http://eepublishers.co.za/article/hans-profitek-06-control-and-communication-in-mining-applications.html, EngineerIT, last accessed 18 July 2012

[2] The Citizen, 02 July 2012, "Cosatu bemoans miner deaths", http://goo.gl/BSj8mN, last accessed 25 July 2012

[3] Fin24, "Goldfields lambasted over mine deaths", 11 July 2012, http://m.news24.com/fin24/Companies/Mining/Gold-Fields-lambasted-over-mine-deaths-20120711, last accessed 24 July 2012

[4] YASKAWA, "The Straight Dope on Mechatronics", April 2008, http://www.yaskawa.com/site/products.nsf/SearchV/86256EB0005E781A86257301006E3CEE?OpenDocument&Source=SearchResultPage, last accessed 24 June 2012

[5] AA Wahab, R Mamat and SS Shamsudin, "The Development of Autopilot System for an Autonomous UAV Helicopter Model", Universiti Teknologi Malaysia, 2006

[6] N Aldawoodi, R Perez, W Alvis and K Valavanis, "Developing Automated Helicopter Models Using Simulated Annealing and Genetic Search", Genetic and Evolutionary Computation Conference, Washington, USA, June 2004

[7] K. Hlophe, "GPS Deprived Localisation for underground mines", CSIR Centre for mining, 2010

[8] WiBorne, "Long-Range Active RFID Systems for Underground Mines", California, USA, 2006

[9] R. Mautz, "Overview of Current Indoor Positioning Systems", Swiss Federal University of Technology, ETH Zurich, Switzerland, 2009

[10] Y Naidoo, R Stopforth and G Bright, "Development of an UAV for Search & Rescue Application Mechatronics Integration for a Quadrotor Helicopter", IEEE Africon 2011 - The Falls Resort and Conference Centre, Livingstone, Zambia, 13 - 15 September 2011

[11] Z Ye, P Bhattacharya, H Mohamadian, H Majlesein, Y Ye, "Equational Dynamic Modeling and Adaptive Control of UAV", Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering, Los Angeles, CA, USA - April 2006

[12] TED Ideas worth spreading, "Vijay Kumar: Robots that fly … and cooperate" http://goo.gl/kLQFZh, last accessed 04 April 2012

[13] DIY Drones, http://diydrones.com/profiles/blog/show?id=705844:BlogPost:17789, last accessed 05 April 2012

[14] F. Zhao, “RC Quadrotor Helicopter”, http://www.instructables.com/id/RC-Quadrotor-Helicopter/, last accessed 09 April 2012

[15] P. Rudol and P. Doherty, “Human Body Detection and Geolocalization for UAV Search and Rescue Missions Using Color and Thermal Imagery”, Department of Computer and information science

[16] M. Liu, O. Tuzel, A Veeraghavan and R. Chellappa, “Fast Directional Chamfer Matching”, Mitsubishi electric research laboratories, July 2010

[17] M. Andriluka, P. Schnitzspan, J. Meyer, S. Kohlbrecher, K. Petersen, O. von Stryk, S. Roth and B. Schiele, “Vision Based Victim Detection from Unmanned Aerial Vehicles”, The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, 18-22 October 2010, Taipei, Taiwan

[18] S. Montabone, A. Soto, “Human Detection Using a Mobile Plartform and Novel Features Derived From a Visual Saliency Mechanism”, Image and Vision Computing, vol. 28, issue 3, March 2010, pages 391-402

[19] G. Bradski and A.Kaehler, “Learning OpenCV”, September 2008

[20] L.J. Latecki, “Template Matching”, Temple University, Lecture Slides, March 2005

[21] G. Bradski and A Kaehler, , “Chapter7: Histograms and Matching”, Learning OpenCV, O’Reilly Media, USA, 2008

[22] H. Nanda and L. Davis, “Probabilistic template based pedestrian detection in infrared videos", Intelligent Vehicle Symposium, 2002. IEEE, vol. 1, June 2002, pp. 15 -20

[23] J. Kosecka, “Shape based matching”, Lecture notes, March 2006

[24] M. Hussein, W. Abd-Almageed, Y. Ran and L.Davis, “Real Time Human detection in Uncontrolled Camera Motion Environments”, IEEE International Conference on Computer Vision Systems, 2006

[25] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features”, Conference on Computer Vision and pattern Recognition, 2001

[26] S. Park, S. Lee,S, Kim and K Cho, “Design of AdaBoost Classifier Circuit using Haar-Like Features for automobile Applications”, International SoC Design Conference (ISOCC), 2011, Pages 262 -265

[27] A. Ahilan and E.James, **“**Design and Implementation of Real time Car Theft Detection in FPGA**”**, 3rd International Conference on Advanced Computing (ICoAC), 2011, Pages 353 – 358

[28] Z. Yang,L Weiming, M Chen and L. Zilong, “Pedestrian Detection in Complex Scene Using Full Binary Tree Classifiers Based on Locally Assembled Binary Haar-like

features", 9th World Congress on Intelligent Control and Automation (WCICA), 2011, Pages 1180 – 1184

[29] M. Sialat, N. Khlifat, F. Bremond and K. Hamrouni, "People detection in complex scene using a cascade of Boosted classifiers based on Haar-like-features", IEEE Intelligent Vehicles Symposium, 2009 , Pages 83 – 87

[30] ArduPilot-Mega Official Arduplane repository, "Connecting your RC equipment", http://code.google.com/p/ardupilot-mega/wiki/RC , last accessed 06-July-2012

[31] Hardkernel, Odroid Platform Developer, http://goo.gl/TZhPLu, last accessed 06-Dec-2012

[32] Netram, "Raspberry Pi – Model B", http://www.netram.co.za/Electronics/Dev-Boards/Raspberry-Pi/Raspberry-Pi-Model-B.html, last accessed 19-August-2013

[33] S. K. Phang, J. J. Ong, R. T. C. Yeo, B. M. Chen and T.H. Lee, "Autonomous Mini-UAV for Indoor Flight with Embedded On-board Vision Processing as Navigation System", 2010 IEEE Region 8 International Conference on Computational Technologies in Electrical and Electronics Engineering (SIBIRCON), 2010

[34] E. Lee, "Large Brushless drive systems: Is there one in your future?", Article for Drive System Group, 2001

[35] DraganFly Innovation Inc, "Choosing the Right Speed Controller For Your Electric RC Airplane", http://goo.gl/IqLm18, last accessed 7-July- 2012

[36] Hobby B, "Aeolian Evolution 30A Programmable ESC 2-4S with 2A/5V BEC", http://goo.gl/erV540, last accessed 6 May 2013

[37] R. Stopforth, "CAESAR – Improvement and Modifications for USAR Robots", PhD Thesis, University of Kwa-Zulu Natal, 2010

[38] J. Nguyen, Performance of Phosphate Lithium-ion Batteries in Motive Applications, Valence Technology, Austin, Texas, USA

[39] LinkSprite Technologies, Inc, "Linksprite JPEG Color Camera, Serial UART Interface", User Manual, March 2012

[40] Wireless Image Transmission using ZigBee Protocol, http://goo.gl/klHoAU, accessed 23-May-2012

[41] A. Nashruddin, Streaming OpenCV Videos over a network, http://goo.gl/Lvm2El, last accessed 16 Dec 2012

[42] The Open Source Initiative, http://www.opensource.org/ , last accessed 28 June 2012

[43] Fixational, "OpenCV vs Matlab", http://goo.gl/WRqeR8 , Last Accessed 06 September 2013

[44] G. Dunks, “Review Snowtown”, 18 May 2011, http://goo.gl/1GWRTP , last accessed 03 May 2013

[45] Untitled, http://i.ytimg.com/vi/KylhIvceR14/0.jpg , last accessed 03 May 2013

[46] P. Assanti, “Brescia, concorso per 3 Tecnici di Radiologia” 23 May 2011, http://www.gazzettadellavoro.com/tecnico-radiologia-concorso-brescia/66437/, last accessed 03 May 2013

[47] Boston GreenFest 2013, “EcoStage Performers 2012 Friday August 2012”, August 2012, http://www.bostongreenfest.org/ecostage.html , last accessed 03 May 2013

[48] K. Williamson, J. Plec, “Vampire Diaries”, Season 1, Episode 1, 2009

[49] Ousam2010, “Haar-like feature object detection”, 25 April 2011, http://www.youtube.com/watch?v=yYXsvC0O0wE&feature=results_video&playnext=1&list=PL37F7D225146EE69E, last accessed 02 July 2012

[50] M. Weber, Background image dataset, California Institute of Technology

[51] XAXOR, “Funny: Another selection of drunk people pics {Part 2}”,27 December 2010, http://xaxor.com/funny-pics/21598-funny-another-selection-of-drunk-people-pics-part-2.html, last accessed 18 March 2013

[52] J. Domanski,”READ_THIS_FIRST”, 2006, http://goo.gl/KUdc84, last accessed 6 May 2013

[53] Southcoasting, untitled, http://www.flickr.com/photos/yihuisu0830/8124212962/, 26 Octoberr 2012

[54] Just Jared, “David Beckham is Milan Mohawk Man”, 29 December 2009, http://www.justjared.com/photo-gallery/2404656/david-beckham-mohawk-man-07/, last accessed 2 May 2013

[55] S.C. Motepe, A. Chikwanha, R. Stopforth, “Survey and Requirements for Search and Rescue Ground and Air Vehicles for Mining Applications”, IEEE M2VIP, Auckland, New Zealand, 2012

[56] T. van Beelen, “RS-232 for Linux and Windows”, http://www.teuniz.net/RS-232/, last accessed, 03 Jan 2013

Appendix A

The table below presents test results for the OpenCV upper body, lower body and full body Haar classifiers that are distributed with the software.

Table A1. Haar Classifier Test and Comparison

	Upper Body			Lower Body			Full Body		
	Correct Hit	False Hit	Miss	Correct Hit	False Hit	Miss	Correct Hit	False Hit	Miss
TestImage1	1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00
TestImage2	0.00	0.22	1.00	0.00	0.22	1.00	0.00	0.22	1.00
TestImage3	0.80	0.00	0.20	0.80	0.00	0.20	0.80	0.00	0.20
TestImage4	1.00	0.20	0.00	1.00	0.20	0.00	1.00	0.20	0.00
TestImage5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage6	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage7	0.38	0.00	0.63	0.38	0.00	0.63	0.38	0.00	0.63
TestImage8	0.67	0.33	0.33	0.67	0.33	0.33	0.67	0.33	0.33
TestImage9	1.00	3.00	0.00	1.00	3.00	0.00	1.00	3.00	0.00
TestImage10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage11	0.50	0.00	0.50	0.50	0.00	0.50	0.50	0.00	0.50
TestImage12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage13	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
TestImage14	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage15	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage16	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage17	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage18	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage19	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage20	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage21	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage22	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage23	0.33	0.00	0.67	0.33	0.00	0.67	0.33	0.00	0.67
TestImage24	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage25	0.33	0.00	0.67	0.33	0.00	0.67	0.33	0.00	0.67

TestImage26	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage27	0.33	0.00	0.67	0.33	0.00	0.67	0.33	0.00	0.67
TestImage28	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
TestImage29	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage30	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage31	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TestImage33	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage34	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage35	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
TestImage36	0.50	0.00	0.50	0.50	0.00	0.50	0.50	0.00	0.50
TestImage37	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage38	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage39	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage40	0.50	0.00	0.50	0.50	0.00	0.50	0.50	0.00	0.50
TestImage41	0.50	0.00	0.50	0.50	0.00	0.50	0.50	0.00	0.50
TestImage42	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage43	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage44	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage45	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage46	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage47	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage48	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage49	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage50	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage51	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage52	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage53	0.50	1.50	0.50	0.50	1.50	0.50	0.50	1.50	0.50
TestImage54	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage55	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
TestImage56	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage57	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00	1.00
TestImage58	1.00	0.80	0.00	1.00	0.80	0.00	1.00	0.80	0.00

TestImage59	1.00	0.00	0.00	1.00	0.00	0.00	1.00	0.00	0.00
TestImage60	0.00	0.25	1.00	0.00	0.25	1.00	0.00	0.25	1.00

Appendix B

OpenCV can take time to install, due to the complexity/confusion in the installation instructions. These are the installation instruction for OpenCV 2.2 on Windows to assist in the installation for a new OpenCV user.

1. Install Microsoft Visual C++
    a. The Express Edition can be downloaded from the Microsoft website: http://www.microsoft.com/visualstudio/eng/downloads#d-2012-express
    b. The 2010 edition was used in this research
2. Download and install CMAKE 2.8.3 installer.
    a. This was downloaded from : http://code.google.com/p/itkvisualizations/downloads/detail?name=cmake-2.8.3-win32-x86.exe&can=2&q=
3. Download OpenCV2.2.exe and install OpenCV2.2
    a. This was downloaded from http://sourceforge.net/projects/opencvlibrary/files/opencv-win/2.2/
    b. The installation destination was chosen to be the operating system home drive C:\
    c. A new folder was created in the OpenCV2.2 folder and named build
4. Then CMake was launched, then the source code and the binaries folder selected. These are shown on the top two text fields on Figure B 1.

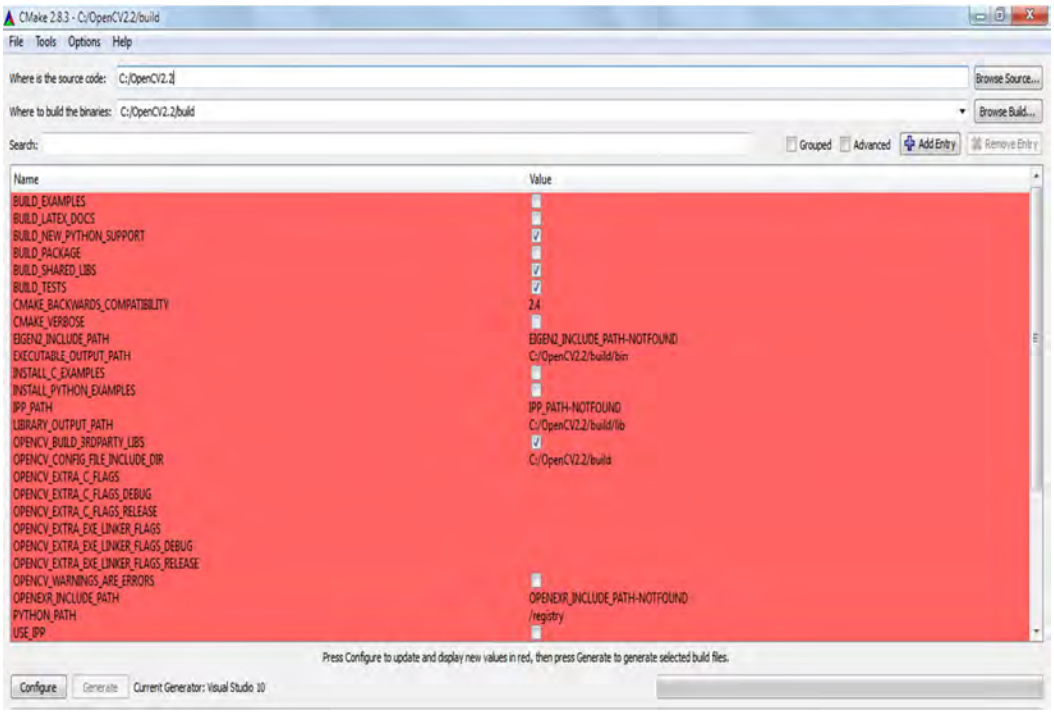

Figure B 1: CMake OpenCV Setup before generating binaries

a. Click configure, on first setup the pop up to select the programming tool to be used with OpenCV appears, then Visual Studio 10 was selected.
b. Then on configure OpenCV will generate the binaries and the screen turns white (Figure B 1).

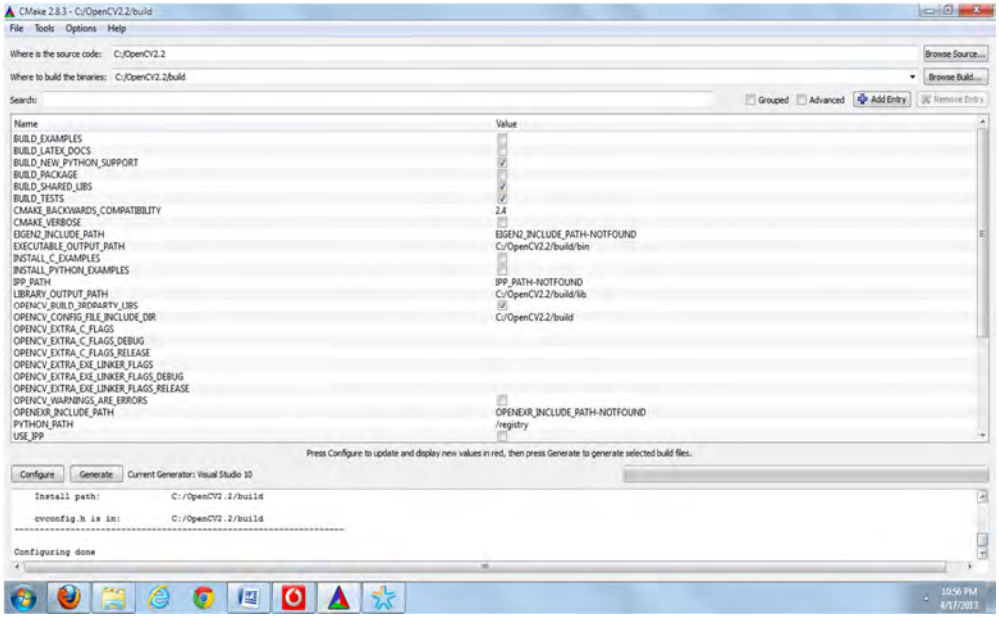

Figure B 2: CMake after generating binaries

5. OpenCV libraries have to be specified for every project in Visual Studio.
    a. Make new WIN32 Console Application
    b. Click on Project, then Properties
    c. The properties window pops up. Select VC++ Directories as shown in Figure B 3. The bold fields have to be specified.

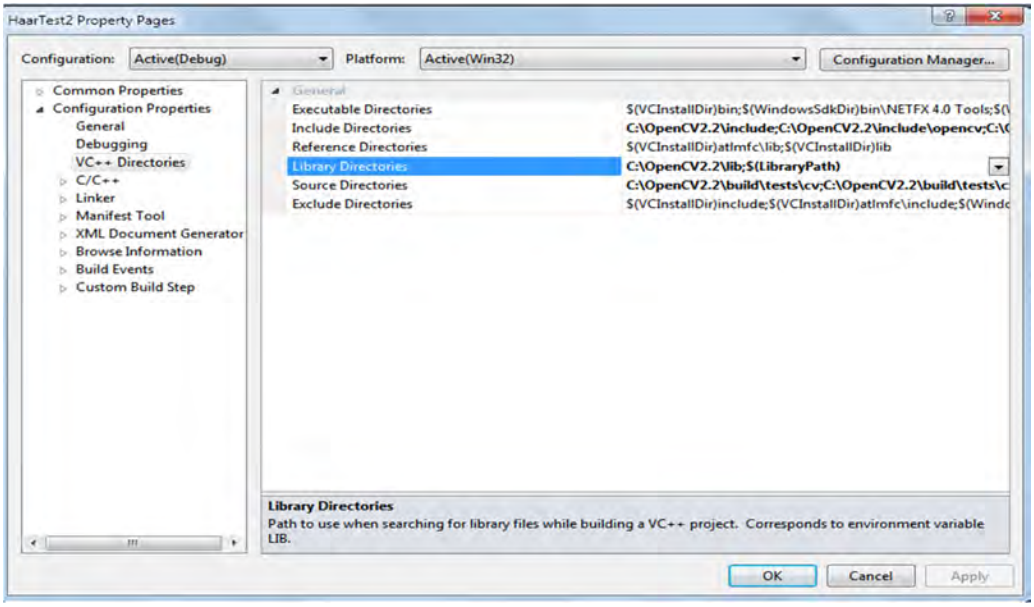

Figure B 3: Project properties window

    d. The bold fields in c. are Include Directories, Libraries Directories and Source Directories. These directories are specified respectively as show

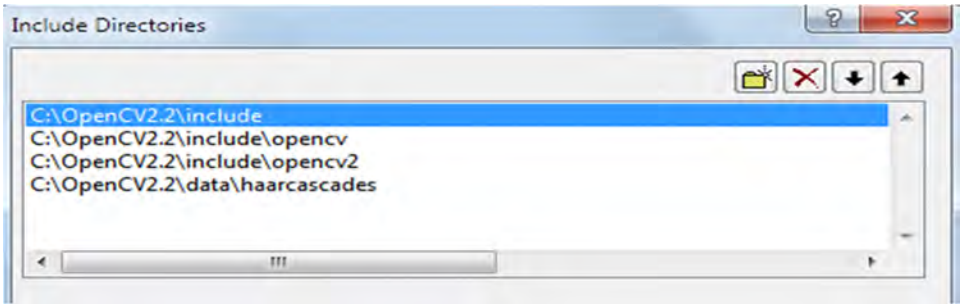

Figure B 4: Directories to be entered into the Include Directories field

Figure B 5: Directories to be entered into the Library Directories field

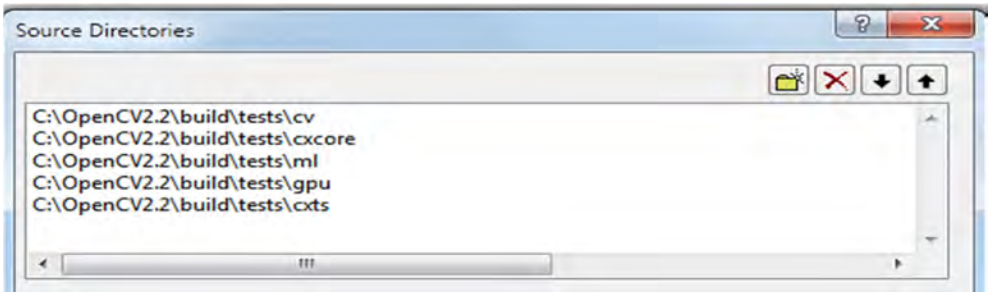

Figure B 6: Directories to be entered into the Source Directories field

e. Open the Linker menu, select Input and then add the directories shown in Figure B 7 under additional dependencies. After this your OpenCV codes should work.

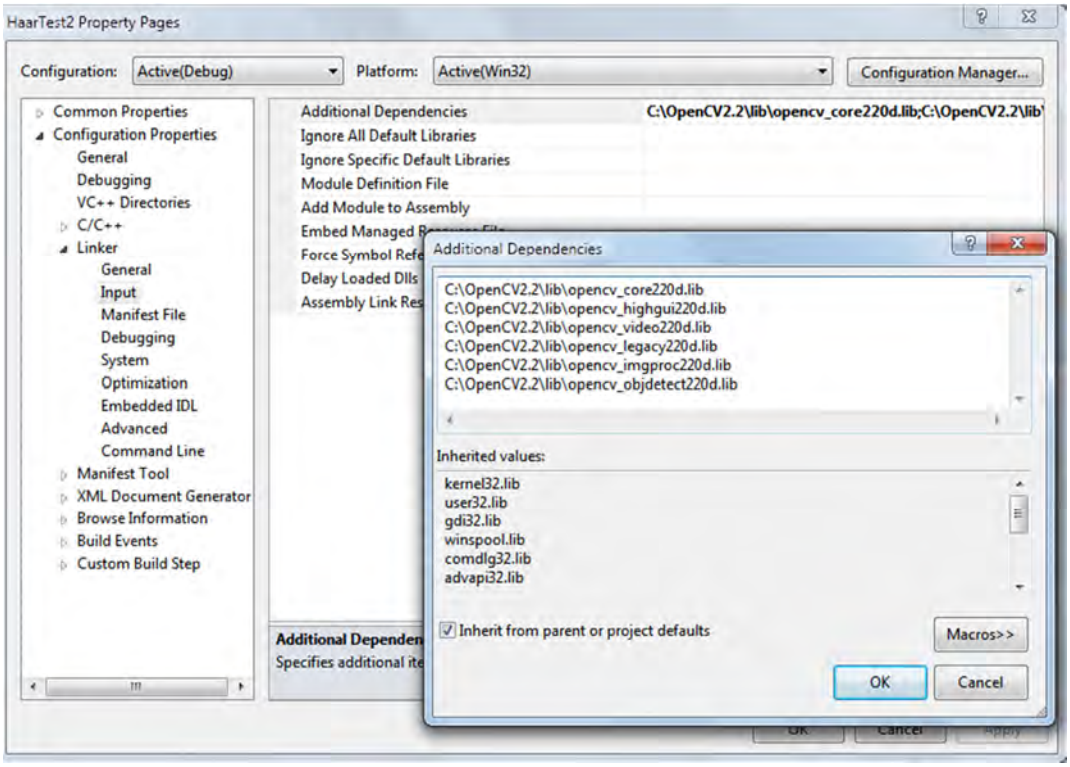

Figure B 7: Linker input additional dependencies directories addition

Appendix C

This section presents the installation instruction of Ubuntu 12.04

1. The simplest method to use is to download the installer and launch it from windows desktop. This can be downloaded from: http://www.ubuntu.com/download/desktop/windows-installer?distro=wubi&release=&bits=
2. For new PC installations, that don't have an existing operating system, the Ubuntu Image Can be downloaded from http://www.ubuntu.com/download/desktop
3. The installation instructions are available here: http://www.ubuntu.com/download/desktop/install-desktop-long-term-support

Appendix D

This section presents installation instructions for OpenCV 2.4.0 on Ubuntu 12.04 (Desktop). Eclipse was used as a programming tool.

1. Installing eclipse
    a. Download eclipse for Linux
    b. Unzip into the home directory
2. Install java
    a. Install from the software center or
    b. Use the instruction sudo apt-get install sun-java6-jdk
3. Installing OpenCV
    a. Install OpenCV from the software center and then use the follow the statements below

sudo apt-get install build-essential checkinstall git cmake ffmpeg libfaac-dev libjack-jackd2-dev libmp3lame-dev libopencore-amrnb-dev libopencore-amrwb-dev libsdl1.2-dev libtheora-dev libva-dev libvdpau-dev libvorbis-dev libx11-dev libxfixes-dev libxvidcore-dev texi2html yasm zlib1g-dev libtbb-dev libv4l-dev libopenexr-dev libunicap2-dev libavformat-dev libswscale-dev libdc1394-22-dev libgstreamer0.10-0 libgstreamer0.10-dev gstreamer0.10-tools gstreamer0.10-plugins-base libgstreamer-plugins-base0.10-dev gstreamer0.10-plugins-good gstreamer0.10-plugins-ugly gstreamer0.10-plugins-bad gstreamer0.10-ffmpeg python-numpy

sudo apt-get install pkg-config

sudo apt-get install libpng12-0 libpng12-dev libpng++-dev libpng3

sudo apt-get install libpnglite-dev libpngwriter0-dev libpngwriter0c2

sudo apt-get install zlib1g-dbg zlib1g zlib1g-dev

sudo apt-get install libjasper-dev libjasper-runtime libjasper1

sudo apt-get install pngtools libtiff4-dev libtiff4 libtiffxx0c2 libtiff-tools

sudo apt-get install ffmpeg libavcodec-dev libavcodec52 libavformat52 libavformat-dev

sudo apt-get install libgstreamer0.10-0-dbg libgstreamer0.10-0 libgstreamer0.10-dev

sudo apt-get install libxine1-ffmpeg libxine-dev libxine1-bin

sudo apt-get install libunicap2 libunicap2-dev

sudo apt-get install libdc1394-22-dev libdc1394-22 libdc1394-utils

sudo apt-get install swig

sudo apt-get install libv4l-0 libv4l-dev

sudo apt-get install python-numpy

sudo apt-get install build-essential libgtk2.0-dev libjpeg62-dev libtiff4-dev libjasper-dev libopenexr-dev cmake python-dev python-numpy libtbb-dev libeigen2-dev yasm libfaac-dev libopencore-amrnb-dev libopencore-amrwb-dev libtheora-dev libvorbis-dev libxvidcore-dev

sudo apt-get install python-dev

tar -xvf OpenCV-2.3.1a.tar.bz2

cd OpenCV-2.3.1/

mkdir build

cd build

cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON ..

make

sudo apt-get install libavformat-dev libswscale-dev

cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON WITH_FFMPEG=ON WITH_GSTREAMER=ON ..

make

sudo apt-get install libv4l-dev

cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON WITH_FFMPEG=ON WITH_GSTREAMER=ON ..

make

sudo make uninstall ffmpeg

sudo apt-get uninstall ffmpeg libavcodec-dev libavcodec52 libavformat52 libavformat-dev

sudo make uninstall ffmpeg libavcodec-dev libavcodec52 libavformat52 libavformat-dev

sudo make uninstall ffmpeg

sudo make remove ffmpeg

sudo apt-get remove ffmpeg

cd OpenCV-2.3.1/

cd build

cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON ..

make

cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON ..

make

cd

sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl libc6-dev libncurses5-dev:i386 x11proto-core-dev libx11-dev:i386 libreadline6-dev:i386 libgl1-mesa-glx:i386 libgl1-mesa-dev g++-multilib mingw32 openjdk-6-jdk tofrodos python-markdown libxml2-utils xsltproc zlib1g-dev:i386D

cd OpenCV-2.3.1/

cd build

cmake -D WITH_TBB=ON -D BUILD_NEW_PYTHON_SUPPORT=ON -D WITH_V4L=OFF -D INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D BUILD_EXAMPLES=ON ..

make

cd

sudo apt-get install libopencv-dev

sudo apt-get install build-essential checkinstall cmake pkg-config yasm

sudo apt-get install libtiff4-dev libjpeg-dev libjasper-dev

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev

sudo apt-get install python-dev python-numpy

sudo apt-get install libtbb-dev

sudo apt-get install libqt4-dev libgtk2.0-dev

clear

tar -xvf OpenCV-2.4.0.tar.bz2

cd OpenCV-2.4.0/

mkdir build

cd build

cmake -D WITH_QT=ON -D WITH_XINE=ON -D WITH_OPENGL=ON -D WITH_TBB=ON -D BUILD_EXAMPLES=ON ..

make

sudo make install

sudo gedit /etc/ld.so.conf.d/opencv.conf

sudo ldconfig

Appendix E

The installation instructions for Linaro Ubuntu 12.04 (ODROID-X2) are presented here. There are different images of Linaro Ubuntu distributions and a number of them were tested but did not work. The images tested included images from Linaro (http://releases.linaro.org/).

1. Download the Linaro Ubuntu 12.04 image from :
   http://www.mdrjr.net/odroid/mirror/ODROID-X2/
   - Other images from other sites were tried before this but didn't work
   - The LCD image used as the development was done using an LCD display.
2. Unzip the image
3. Download Win32DiskImager
4. Use to write the image to the SD card. Make sure the correct drive is selected and the MD5 Hash is checked

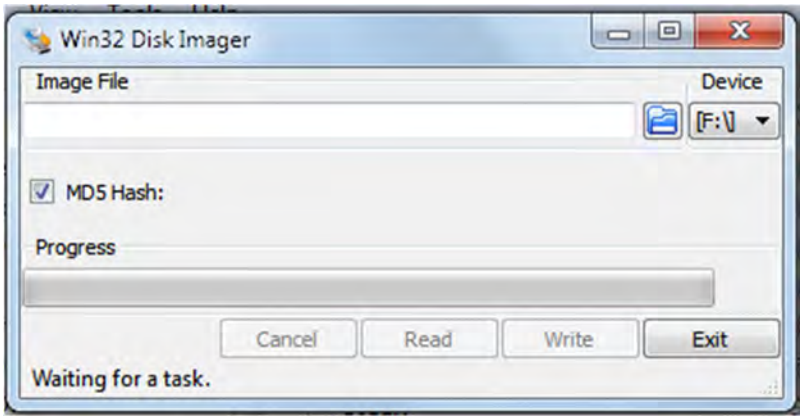

Figure E 1: Win32 Disk Imager

5. After writing the image the image ODROID-X2 should be able to boot from the SD Card

Appendix F

In Appendix B the installation instructions for windows platform were given, due to the difficulty that can arise in the installation. The installation instructions for OpenCV on Linaro Ubuntu 12.04 are presented here. These are presented as they are not the same as the OpenCV Ubuntu Desktop installation instructions. These instructions were used in this research and are based on a script by .

```
if [ "$arch" == "i686" -o "$arch" == "i386" -o "$arch" == "i486" -o "$arch" == "i586" ]; then

flag=1

else

flag=0

fi

echo "Installing OpenCV 2.4.0"

mkdir OpenCV

cd OpenCV

echo "Removing any pre-installed ffmpeg and x264"

sudo apt-get remove remove ffmpeg x264 libx264-dev

echo "Installing Dependenices"

sudo apt-get install libopencv-dev

sudo apt-get install build-essential checkinstall cmake pkg-config yasm

sudo apt-get install libtiff4-dev libjpeg-dev libjasper-dev

sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libdc1394-22-dev libxine-dev libgstreamer0.10-dev libgstreamer-plugins-base0.10-dev libv4l-dev

sudo apt-get install python-dev python-numpy

sudo apt-get install libtbb-dev

sudo apt-get install libqt4-dev libgtk2.0-dev

echo "Downloading x264"
```

```
wget ftp://ftp.videolan.org/pub/videolan/x264/snapshots/x264-snapshot-20120528-2245-stable.tar.bz2

tar -xvf x264-snapshot-20120528-2245-stable.tar.bz2

cd x264-snapshot-20120528-2245-stable/

echo "Installing x264"

if [ $flag -eq 1 ]; then

./configure --enable-static

else

./configure --enable-shared --enable-pic

fi

make

sudo make install

cd ..

echo "Downloading ffmpeg"

wget http://ffmpeg.org/releases/ffmpeg-0.8.12.tar.bz2

echo "Installing ffmpeg"

tar -xvf ffmpeg-0.8.12.tar.bz2

cd ffmpeg-0.8.12/

echo "note change here, no if statememt used"

./configure  --enable-shared

fi

make

sudo make install

cd ..

echo "Downloading v4l"
```

```
wget http://www.linuxtv.org/downloads/v4l-utils/v4l-utils-0.8.4.tar.bz2

echo "Installing v4l"

tar -xvf v4l-utils-0.8.4.tar.bz2

cd v4l-utils-0.8.4/

make

sudo make install

cd ..

echo "Downloading OpenCV 2.4.0"

wget -O OpenCV-2.4.0.tar.bz2 http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.0/OpenCV-2.4.0.tar.bz2/download

echo "Installing OpenCV 2.4.0"

tar -xvf OpenCV-2.4.0.tar.bz2

cd OpenCV-2.4.0/

mkdir build

cd build

cmake -D CMAKE_BUILD_TYPE=RELEASE ..

make

sudo make install

sudo echo “/usr/local/lib” >> /etc/ld.so.conf

sudo ldconfig
```

Appendix G

This section presents the software for detecting character on-board the UAV and transmitting them to the Arduino board. The serial communication setup is from the open source code [56]. The OpenCV functionalities, including character detection, and the data transfer were developed by the author.

```
//============================================================================
// Name        : Cam_Test.cpp
// Author      : SCVII
// Version     :
// Copyright   : Open Source
// Description : Hello World in C++, Ansi-style
//============================================================================


#include <iostream>
#include <stdio.h>
#include <stdlib.h>

#include <cv.h>
#include <highgui.h>

#ifdef _WIN32
#include <Windows.h>
#else
#include <unistd.h>
#endif

#include "rs232.h"

using namespace std;

#include "rs232.h"


#ifdef __linux__   /* Linux */
```

```
int Cport[22],
   error;

struct termios new_port_settings,
      old_port_settings[22];
//SCVII Changed ttyS[n] to ttyACM[n]
char
comports[22][15]={"/dev/ttyACM0","/dev/ttyACM1","/dev/ttyACM2","/dev/ttyS3","/dev/ttyS4","/dev/t
tyS5",
      "/dev/ttyS6","/dev/ttyS7","/dev/ttyS8","/dev/ttyS9","/dev/ttyS10","/dev/ttyS11",
      "/dev/ttyS12","/dev/ttyS13","/dev/ttyS14","/dev/ttyS15","/dev/ttyUSB0",
      "/dev/ttyUSB1","/dev/ttyUSB2","/dev/ttyUSB3","/dev/ttyUSB4","/dev/ttyUSB5"};


int OpenComport(int comport_number, int baudrate)
{
 int baudr;

 if((comport_number>21)||(comport_number<0))
 {
  printf("illegal comport number\n");
  return(1);
 }

 switch(baudrate)
 {
  case     50 : baudr = B50;
               break;
  case     75 : baudr = B75;
               break;
  case    110 : baudr = B110;
               break;
  case    134 : baudr = B134;
               break;
  case    150 : baudr = B150;
               break;
  case    200 : baudr = B200;
               break;
```

```
  case     300 : baudr = B300;
                 break;
  case     600 : baudr = B600;
                 break;
  case    1200 : baudr = B1200;
                 break;
  case    1800 : baudr = B1800;
                 break;
  case    2400 : baudr = B2400;
                 break;
  case    4800 : baudr = B4800;
                 break;
  case    9600 : baudr = B9600;
                 break;
  case   19200 : baudr = B19200;
                 break;
  case   38400 : baudr = B38400;
                 break;
  case   57600 : baudr = B57600;
                 break;
  case  115200 : baudr = B115200;
                 break;
  case  230400 : baudr = B230400;
                 break;
  case  460800 : baudr = B460800;
                 break;
  case  500000 : baudr = B500000;
                 break;
  case  576000 : baudr = B576000;
                 break;
  case  921600 : baudr = B921600;
                 break;
  case 1000000 : baudr = B1000000;
                 break;
  default      : printf("invalid baudrate\n");
                 return(1);
                 break;
}
```

```
  Cport[comport_number] = open(comports[comport_number], O_RDWR | O_NOCTTY | O_NDELAY);
  if(Cport[comport_number]==-1)
  {
    perror("unable to open comport ");
    return(1);
  }

  error = tcgetattr(Cport[comport_number], old_port_settings + comport_number);
  if(error==-1)
  {
    close(Cport[comport_number]);
    perror("unable to read portsettings ");
    return(1);
  }
  memset(&new_port_settings, 0, sizeof(new_port_settings));  /* clear the new struct */

  new_port_settings.c_cflag = baudr | CS8 | CLOCAL | CREAD;
  new_port_settings.c_iflag = IGNPAR;
  new_port_settings.c_oflag = 0;
  new_port_settings.c_lflag = 0;
  new_port_settings.c_cc[VMIN] = 0;      /* block untill n bytes are received */
  new_port_settings.c_cc[VTIME] = 0;     /* block untill a timer expires (n * 100 mSec.) */
  error = tcsetattr(Cport[comport_number], TCSANOW, &new_port_settings);
  if(error==-1)
  {
    close(Cport[comport_number]);
    perror("unable to adjust portsettings ");
    return(1);
  }

  return(0);
}


int PollComport(int comport_number, unsigned char *buf, int size)
{
  int n;
```

```
#ifndef __STRICT_ANSI__                  /* __STRICT_ANSI__ is defined when the -ansi option is used for
gcc */
  if(size>SSIZE_MAX)  size = (int)SSIZE_MAX;  /* SSIZE_MAX is defined in limits.h */
#else
  if(size>4096)  size = 4096;
#endif

  n = read(Cport[comport_number], buf, size);

  return(n);
}


int SendByte(int comport_number, unsigned char byte)
{
  int n;

  n = write(Cport[comport_number], &byte, 1);
  if(n<0)  return(1);

  return(0);
}


int SendBuf(int comport_number, unsigned char *buf, int size)
{
  return(write(Cport[comport_number], buf, size));
}


void CloseComport(int comport_number)
{
  close(Cport[comport_number]);
  tcsetattr(Cport[comport_number], TCSANOW, old_port_settings + comport_number);
}

/*
```

```
Constant  Description
TIOCM_LE  DSR (data set ready/line enable)
TIOCM_DTR DTR (data terminal ready)
TIOCM_RTS RTS (request to send)
TIOCM_ST  Secondary TXD (transmit)
TIOCM_SR  Secondary RXD (receive)
TIOCM_CTS CTS (clear to send)
TIOCM_CAR DCD (data carrier detect)
TIOCM_CD  Synonym for TIOCM_CAR
TIOCM_RNG RNG (ring)
TIOCM_RI  Synonym for TIOCM_RNG
TIOCM_DSR DSR (data set ready)
*/

int IsCTSEnabled(int comport_number)
{
  int status;

  status = ioctl(Cport[comport_number], TIOCMGET, &status);

  if(status&TIOCM_CTS) return(1);
  else return(0);
}


#else        /* windows */


HANDLE Cport[16];


char comports[16][10]={"\\\\.\\COM1",  "\\\\.\\COM2",  "\\\\.\\COM3",  "\\\\.\\COM4",
                       "\\\\.\\COM5",  "\\\\.\\COM6",  "\\\\.\\COM7",  "\\\\.\\COM8",
                       "\\\\.\\COM9",  "\\\\.\\COM10", "\\\\.\\COM11", "\\\\.\\COM12",
                       "\\\\.\\COM13", "\\\\.\\COM14", "\\\\.\\COM15", "\\\\.\\COM16"};

char baudr[64];
```

```
int OpenComport(int comport_number, int baudrate)
{
  if((comport_number>15)||(comport_number<0))
  {
    printf("illegal comport number\n");
    return(1);
  }

  switch(baudrate)
  {
    case     110 : strcpy(baudr, "baud=110 data=8 parity=N stop=1");
                   break;
    case     300 : strcpy(baudr, "baud=300 data=8 parity=N stop=1");
                   break;
    case     600 : strcpy(baudr, "baud=600 data=8 parity=N stop=1");
                   break;
    case    1200 : strcpy(baudr, "baud=1200 data=8 parity=N stop=1");
                   break;
    case    2400 : strcpy(baudr, "baud=2400 data=8 parity=N stop=1");
                   break;
    case    4800 : strcpy(baudr, "baud=4800 data=8 parity=N stop=1");
                   break;
    case    9600 : strcpy(baudr, "baud=9600 data=8 parity=N stop=1");
                   break;
    case   19200 : strcpy(baudr, "baud=19200 data=8 parity=N stop=1");
                   break;
    case   38400 : strcpy(baudr, "baud=38400 data=8 parity=N stop=1");
                   break;
    case   57600 : strcpy(baudr, "baud=57600 data=8 parity=N stop=1");
                   break;
    case  115200 : strcpy(baudr, "baud=115200 data=8 parity=N stop=1");
                   break;
    case  128000 : strcpy(baudr, "baud=128000 data=8 parity=N stop=1");
                   break;
    case  256000 : strcpy(baudr, "baud=256000 data=8 parity=N stop=1");
                   break;
    default      : printf("invalid baudrate\n");
```

```
            return(1);
            break;
}

Cport[comport_number] = CreateFileA(comports[comport_number],
              GENERIC_READ|GENERIC_WRITE,
              0,                    /* no share  */
              NULL,                 /* no security */
              OPEN_EXISTING,
              0,                    /* no threads */
              NULL);                /* no templates */

if(Cport[comport_number]==INVALID_HANDLE_VALUE)
{
 printf("unable to open comport\n");
 return(1);
}

DCB port_settings;
memset(&port_settings, 0, sizeof(port_settings));  /* clear the new struct  */
port_settings.DCBlength = sizeof(port_settings);

if(!BuildCommDCBA(baudr, &port_settings))
{
 printf("unable to set comport dcb settings\n");
 CloseHandle(Cport[comport_number]);
 return(1);
}

if(!SetCommState(Cport[comport_number], &port_settings))
{
 printf("unable to set comport cfg settings\n");
 CloseHandle(Cport[comport_number]);
 return(1);
}

COMMTIMEOUTS Cptimeouts;
```

```
  Cptimeouts.ReadIntervalTimeout        = MAXDWORD;
  Cptimeouts.ReadTotalTimeoutMultiplier  = 0;
  Cptimeouts.ReadTotalTimeoutConstant    = 0;
  Cptimeouts.WriteTotalTimeoutMultiplier = 0;
  Cptimeouts.WriteTotalTimeoutConstant   = 0;

  if(!SetCommTimeouts(Cport[comport_number], &Cptimeouts))
  {
    printf("unable to set comport time-out settings\n");
    CloseHandle(Cport[comport_number]);
    return(1);
  }

  return(0);
}


int PollComport(int comport_number, unsigned char *buf, int size)
{
  int n;

  if(size>4096)  size = 4096;

/* added the void pointer cast, otherwise gcc will complain about */
/* "warning: dereferencing type-punned pointer will break strict aliasing rules" */

  ReadFile(Cport[comport_number], buf, size, (LPDWORD)((void *)&n), NULL);

  return(n);
}


int SendByte(int comport_number, unsigned char byte)
{
  int n;

  WriteFile(Cport[comport_number], &byte, 1, (LPDWORD)((void *)&n), NULL);
```

```
 if(n<0)  return(1);

 return(0);
}


int SendBuf(int comport_number, unsigned char *buf, int size)
{
 int n;

 if(WriteFile(Cport[comport_number], buf, size, (LPDWORD)((void *)&n), NULL))
 {
  return(n);
 }

 return(-1);
}


void CloseComport(int comport_number)
{
 CloseHandle(Cport[comport_number]);
}


int IsCTSEnabled(int comport_number)
{
 int status;

 GetCommModemStatus(Cport[comport_number], (LPDWORD)((void *)&status));

 if(status&MS_CTS_ON) return(1);
 else return(0);
}


#endif
```

```
void cprintf(int comport_number, const char *text)  /* sends a string to serial port */
{
  while(*text != 0)   SendByte(comport_number, *(text++));
}

//Main, Let the tests begin


int main() {
        cout << "Server_Test" << endl; // prints Server_Test

        int  cport_nr=1,        /* /dev/ttyS0 (COM1 on windows) */
                bdrate=9600;       /* 9600 baud */


        int key;

        IplImage* img = cvLoadImage("instructions.jpg");
        //cvNamedWindow("Instructions",0);
        cvShowImage("Instructions", img);
        while(1)
        {

        key = cvWaitKey(10);

        switch (key)
        {
        case 27: //Exit program
                cout << "Escape le yona" << endl; // prints to check
                 cvDestroyWindow("Instructions");
                 cvReleaseImage(&img);

                break;

  //Pitch
  case 97 : //"a" pressed >> decrease Pitch
        cprintf(cport_nr,"a");
```

```
            cout << "Ke Tsona Bosso" << endl; // prints to check
            break;
case 100 : //"d" pressed>> increase Pitch
             cprintf(cport_nr,"d");
            break;

//Roll
case 122 : //"z" pressed >> decrease Roll
            cprintf(cport_nr,"z");
            break;
case 119 : //"w" pressed>> increase Roll
              cprintf(cport_nr,"w");
            break;

//Throttle
case 109 : //"m" pressed >> decrease Throttle
            cprintf(cport_nr,"m");
            break;
case 105 : //"i" pressed>> increase  Throttle
            cprintf(cport_nr,"i");
            break;

//Yaw
case 106 : //"j" pressed >> decrease Yaw
            cprintf(cport_nr,"j");
            break;
case 108 : //"l" pressed>> increase Yaw
            cprintf(cport_nr,"l");
            break;

case 49 : //"1" pressed>> arm the motors
            cprintf(cport_nr,"1");
            break;

case 48 : //"0" pressed>> disarm the motors
            cprintf(cport_nr,"0");
            break;
```

```
case 99 : //"c" pressed>> reset the control value
          cprintf(cport_nr,"c");
          break;

case 57 : //"9" pressed>> automatic take off
          cprintf(cport_nr,"9");
          break;

          } //End Switch

          } //End While frame not null

          return 0;

          }
```

Appendix H

This presents Human detection code using final detection model on Eclipse in Ubuntu12.04. The code is based on the facedect.c distributed with OpenCV.

```
//===========================================================================

// Name        : HaarJournal2013.cpp

// Author      : Sibonelo Motepe

// Version     :

// Copyright   : Open Source

// Description : Human Detection using haar cascade classiffier

//===========================================================================


//Based on OpenCV Sample Application: facedetect.c

// Include header files

#include <iostream>

using namespace std;


#include <cv.h>

#include <highgui.h>

#include <cxcore.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <assert.h>

#include <math.h>

#include <float.h>
```

```
#include <limits.h>

#include <time.h>

#include <ctype.h>

#include <unistd.h>


// Structure to get video from camera or video

CvCapture* CamCapture = 0;


// memory

static CvMemStorage* storage = 0;


// Images to capture the frame from video or camera or from file

IplImage *PicFrame, *PicFrame_Copy = 0;


// Function prototype for detecting and drawing an object from an image

void human_detect( IplImage* img,CvHaarClassifierCascade* kascade,const char* framename);


// Create string that contains the cascade name

const char* cascade_name =

   "/home/scvii/OpenCV-2.4.0/data/haarcascades/fullbody21.xml";


// Main function, defines proogram entry point for the program.

int main( void )

{    //Mat image;
```

```
//IplImage *frame, *frame_copy = 0;

        const char* results1="ResultsFullBodyFinal..";

        cout << "Serial_Test2CPP_SCVII" << endl; // prints Serial_Test2CPP_SCVII


        // Create a new Haar classifier

        static CvHaarClassifierCascade* cascade= 0;


  // Load HaarClassifierCascade

  cascade = (CvHaarClassifierCascade*)cvLoad( cascade_name, 0, 0, 0 );


  // Check the cascade successful load

  if( !cascade )

  {

     fprintf( stderr, "ERROR: classifier cascade not loaded\n" );

     return -1;

  }


  // Allocate memory storage

   storage = cvCreateMemStorage(0);

   // Structure to get video from camera

   CamCapture = cvCaptureFromCAM(0);


  // If succesfully loaded, then:

  if( CamCapture )

  {
```

```
// Capture from the camera.
for(;;)
{
   // Capture the frame and load it in IplImage
   if( !cvGrabFrame( CamCapture ))
      break;
   PicFrame = cvRetrieveFrame( CamCapture );

   // If no frame, quit loop
   if( !PicFrame )
      break;

   // make the copy the same size as the frame
   if( !PicFrame_Copy )
      PicFrame_Copy= cvCreateImage( cvSize(PicFrame->width,PicFrame->height),
                        IPL_DEPTH_8U, PicFrame->nChannels );

   // Check the origin of image. If top left, copy the image frame to frame_copy.
   if( PicFrame->origin == IPL_ORIGIN_TL )
      cvCopy( PicFrame, PicFrame_Copy, 0 );
         // Else flip and copy the image
   else
      cvFlip( PicFrame, PicFrame_Copy, 0 );

   // Call the function to detect and draw the human body
```

```
        //pthread_mutex_lock(&mutex);
         human_detect( PicFrame_Copy,cascade,results1 );
         //pthread_mutex_unlock(&mutex);
         // Wait for a while before proceeding to the next frame
         if( cvWaitKey( 10 ) >= 0 )
            break;
      }
   }
 else
{           IplImage* image = cvLoadImage("PositiveAdd1.jpg");
                    human_detect( image, cascade,results1 );
                    cvWaitKey(0);
             }
}

// Function to detect and draw human bodies that are present in an image
void human_detect( IplImage* img,CvHaarClassifierCascade* kascade,const char* framename)
{
   int scale = 1;

   // Create new image based on input image
   IplImage* temp = cvCreateImage( cvSize(img->width/scale,img->height/scale), 8, 3 );

   // Create two points to represent human locations
   CvPoint pt1, pt2;
```

```
int i;


// Clear memory storage
cvClearMemStorage( storage );


// check if cascade is loaded, to find the people
if( kascade )
{
   // There can be more than one human in an image. growable sequence of people is created .
   // Detect objects and store them in the sequence
   CvSeq* human = cvHaarDetectObjects( img, kascade, storage,
                                       1.1, 2, CV_HAAR_DO_CANNY_PRUNING,
                                       cvSize(20, 20) );


   // Loop/repeat>> number of people found.
   for( i = 0; i < (human ? human->total : 0); i++ )
   {
     // Create rectangle for drawing the detection
      CvRect* r = (CvRect*)cvGetSeqElem( human, i );


      // Find the dimensions of the human feature.scale if necessary
      pt1.x = r->x*scale;
      pt2.x = (r->x+r->width)*scale;
      pt1.y = r->y*scale;
      pt2.y = (r->y+r->height)*scale;
```

```
            // Draw the rectangle in the input image
            cvRectangle( img, pt1, pt2, CV_RGB(255,0,0), 3, 8, 0 );
        }
    }

    // Show the image in a window
    cvShowImage( framename, img );

    // Release temp image
    cvReleaseImage( &temp );
}
```

Appendix I

Arduino code for controlling the UAV by passing the ground station instruction received via the On Board Processor, ODROID-X), to the APM.

```
int incomingByte = 0;   // for incoming serial data

int outPinPPM_P =2;     //Channel 2 Pitch

int outPinPPM_T =3;     //Channel 3 Throttle

int outPinPPM_Y =4;     //Channel 4 Yaw

int outPinPPM_R =5;     //Channel 1 Roll

int L_Count=0;     // Lift off counter

int ThrottleTime=192; //ch3

int YawTime=192; //ch4

int RollTime=192; //ch1

int PitchTime=192; //ch2


void setup () {

 Serial.begin(9600);

 pinMode(outPinPPM_T, OUTPUT);   // sets the digital pin as output

 pinMode(outPinPPM_Y, OUTPUT);   // sets the digital pin as output

 pinMode(outPinPPM_R, OUTPUT);   // sets the digital pin as output

 pinMode(outPinPPM_P, OUTPUT);   // sets the digital pin as output

//pinMode(7, INPUT);

}
```

```
void loop(){
if (Serial.available() > 0){ //Read the serial data
incomingByte = Serial.read();}
delay(100);


    switch(incomingByte)  {// assign values dependinding on the key board button pressed
       //Speed steps introduced to lower or increase the speed gradually


          //Pitch
        case 97 : //"a" pressed >> decrease Pitch
          PitchTime=PitchTime-10;
          if (PitchTime<=0){//limit the pitch time to positive values 0
          PitchTime=0;}
          break;
        case 100 : //"d" pressed>> max Pitch
          PitchTime=PitchTime+10;
          if (PitchTime>=255){//limit the pitch time to positive values less than 255
          PitchTime=255;}
          break;


        //Roll
        case 122 : //"z" pressed >> decrease Roll
          RollTime=RollTime-10;
```

```
  if (RollTime<=0){//limit the Roll time to positive values 0
  RollTime=0;}
  break;
case 119 : //"w" pressed>> max Roll
  RollTime=RollTime+10;
  if (RollTime>=255){//limit the pitch time to positive values less than 255
  RollTime=255;}
  break;

//Throttle
case 109 : //"m" pressed >> decrease Throttle
  ThrottleTime=ThrottleTime-10;
  if (ThrottleTime<=0){//limit the Roll time to positive values
  ThrottleTime=0;}
  //RollTime = map(RollTime, 1000, 2000, 0, 255);
  break;
case 105 : //"i" pressed>> max Throttle
  ThrottleTime=ThrottleTime+10;
  if (ThrottleTime>=255){//limit the pitch time to positive values less than 255
  ThrottleTime=255;}
  break;

//Yaw
```

```
        case 106 : //"j" pressed >> decrease Yaw
          YawTime=YawTime-10;
          if (YawTime<=0){//limit the pitch time to positive values
          YawTime=0;}
          break;
        case 108 : //"l" pressed>> max Yaw
          YawTime=YawTime+10;
          if (YawTime>=255){//limit the pitch time to positive values less than 255
          YawTime=255;}
          break;

       //Clear or Reset all >> mid value
      case 99 :   //Pressed "c"
          RollTime=128; //ch1
          PitchTime=128; //ch2
          ThrottleTime=128; //ch3
          YawTime=128; //ch4
          break;
//Arm The Motors
   case 49: //press "1" >>character one to arm the motors
   //This is done by pushing the throttle/yaw stick to the minimum throttle and max yaw
   //Where both values are maximum
    ThrottleTime=128; //ch3
```

```
      YawTime=255; //ch4


    break;
  //Disarm The Motors
    case 48: //press "0" >>character zero to disarm the motors
    //This is done by pushing the throttle/yaw stick to the bottom left corner
    //Where both values are minimum
      ThrottleTime=128; //ch3
      YawTime=128; //ch4
    break;


 case 57:// press "9" >>automatic lift off
   for (L_Count=0;L_Count++;L_Count=23)
   ThrottleTime=ThrottleTime+10;
   analogWrite(outPinPPM_T, ThrottleTime);
   delay(230); //delay for 230 milliseconds to have the UAV in air in about 5 seconds[5290ms]
 break;
}
analogWrite(outPinPPM_T, ThrottleTime);
analogWrite(outPinPPM_Y, YawTime);
analogWrite(outPinPPM_R, RollTime);
analogWrite(outPinPPM_P, PitchTime);
}
```

Appendix J

The results for the experiment discussed in are presented here. A "1" presents a positive detection and a "0" no detection, where there is a human present in the video frame. Table J 1 summarizes the detections and the average detection for each of the sides of the human body.

Table J 1: Final Model Detection Results

Front View					
Distance From UAV(m)	Experinment1	Experinment2	Experinment3	Experinment4	Average
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	1	1	1	1	1
Side View Left					
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	0	1	1	1	0.75
Side View Right					
1	1	1	1	1	1
2	1	1	1	1	1
3	1	1	1	1	1
4	1	1	1	1	1
5	0	1	0	0	0.25
Back View					
1	0	1	0	1	0.5
2	1	1	0	1	0.75
3	1	1	1	0	0.75
4	0	1	0	0	0.25
5	0	1	0	1	0.5

Appendix K

This section presents the experiment results for the OpenCV distributed Haar-Cascade full body model generic model, preliminary Haar-Cascade full body model and final Haar-Cascade full body model. The results were used to compare the generic model, preliminary model and final model.

Figure J 1: OpenCV Generic Model Experiments Results

OpenCV Distribution Generic Model											
Back View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	0	0	0	0	0	0	0	0	1	0	0.1
2	1	1	1	0	1	0	1	0	1	0	0.6
3	0	1	0	1	1	0	0	0	1	0	0.4
4	1	1	1	1	1	1	1	1	0	1	0.9
5	1	1	0	1	1	0	1	1	1	1	0.8
Back View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	0	0	1	0	0	1	1	0	1	0	0.4
2	0	1	1	1	1	1	0	1	0	1	0.7
3	2	1	1	1	1	1	1	1	1	1	1.1
4	1	1	1	1	1	0	1	1	1	1	0.9
5	1	1	1	1	0	0	1	1	1	1	0.8
Front View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	0	0	0	1	0	0	0	0	0	1	0.2
2	0	0	0	1	0	1	1	1	0	0	0.4
3	1	1	0	1	0	1	1	1	1	1	0.8
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	0	0.9
Front View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average

1	0	1	1	0	1	0	0	1	1	1	0.6
2	1	1	1	1	1	1	0	1	1	0	0.8
3	1	1	1	1	1	1	1	1	1	0	0.9
4	1	2	1	0	1	1	0	1	2	1	1
5	1	1	1	1	1	2	1	1	1	1	1.1
Left View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	0	0	0	0	0	0	0	0	0	0	0
2	1	1	0	0	1	1	0	1	1	1	0.7
3	1	0	1	0	0	1	1	1	0	0	0.5
4	1	1	1	1	1	0	1	1	1	1	0.9
5	0	0	1	0	1	0	0	0	0	0	0.2
Left View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	1	1	0	1	2	0	0	0	0	1	0.6
2	1	1	0	1	0	1	0	1	1	1	0.7
3	1	0	1	0	0	1	1	1	1	1	0.7
4	2	1	1	2	2	1	0	1	1	1	1.2
5	1	2	1	2	2	1	1	0	1	1	1.2
Right View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	0	0	1	0	0	0	0	1	1	0	0.3
2	0	1	0	1	1	0	1	0	1	0	0.5
3	0	1	1	0	1	0	1	1	1	1	0.7
4	1	1	1	1	1	0	1	1	1	1	0.9
5	0	1	1	0	0	1	1	1	0	0	0.5
Right View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment8	Experinment9	Experinment10	Average
1	1	2	1	1	1	1	1	0	0	1	0.9
2	0	1	1	1	1	0	0	1	0	1	0.6
3	1	1	1	0	0	1	1	1	1	1	0.8
4	1	1	1	1	1	1	1	1	1	3	1.2
5	1	1	0	1	1	1	1	0	1	1	0.8

Figure J 2: Preliminary Model Experiments Results

Preliminary Model											
Back View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment9	Experinment10	Average
1	1	1	1	1	1	1	1	1	0	1	0.9
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Back View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment9	Experinment10	Average
1	2	1	2	2	3	4	3	2	2	4	2.5
2	8	2	7	5	2	3	3	5	5	2	4.2
3	6	3	4	4	3	8	8	8	6	4	5.4
4	6	7	7	4	3	7	4	5	6	6	5.5
5	5	4	4	4	5	4	7	6	6	4	4.9
Front View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment9	Experinment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Front View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment9	Experinment10	Average
1	5	1	2	2	1	2	3	3	4	4	2.7
2	6	8	8	3	3	5	7	6	4	6	5.6
3	9	7	10	7	10	9	7	8	6	8	8.1
4	4	7	5	7	5	6	6	7	5	8	6
5	5	6	5	5	6	6	4	5	4	4	5

Left View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experiment9	Experinment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Left View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experiment9	Experinment10	Average
1	3	2	1	3	3	4	3	4	2	2	2.7
2	7	6	2	4	6	3	4	4	6	5	4.7
3	5	3	2	7	3	2	5	2	5	5	3.9
4	7	7	5	6	4	7	7	4	7	5	5.9
5	4	7	4	4	5	8	4	3	4	4	4.7
Right View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experiment9	Experinment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Right View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experiment9	Experinment10	Average
1	2	3	2	4	2	3	2	8	6	4	3.6
2	5	5	6	4	7	6	7	5	5	6	5.6
3	7	7	8	10	8	5	7	7	6	8	7.3
4	6	4	5	3	4	7	7	7	7	7	5.7
5	6	5	4	5	5	5	6	6	6	7	5.5

Figure J 3: Final Model Experiments Results

Final Model											
Back View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experinment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	0	1	1	0	0.8
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	0	1	1	1	1	0.9
Back View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experinment10	Average
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0.1
3	0	0	0	0	0	1	0	0	0	0	0.1
4	0	0	0	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	0.1
Front View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experinment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	0	0	1	1	1	0.8
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Front View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experinment10	Average
1	0	0	0	0	1	0	0	0	0	0	0.1
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0

Left View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experiment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	0	1	1	1	1	1	0.9
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Left View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experiment10	Average
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0
Right View (Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experiment10	Average
1	1	1	1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	1	1	1
4	1	1	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1	1	1
Right View (False Detection)											
Distance From UAV(m)	Experinment 1	Experinment 2	Experinment 3	Experinment 4	Experinment 5	Experinment 6	Experinment 7	Experinment 8	Experinment 9	Experiment10	Average
1	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0	0.1
3	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0