May 2016

Security Threats from UAS and Its Mitigation Through Detection and Tracking

Sai Ram Ganti
University of Nevada, Las Vegas, ganti@unlv.nevada.edu

SECURITY THREATS FROM UAS AND ITS MITIGATION THROUGH DETECTION AND TRACKING

By

Sai Ram Ganti

Bachelor of Technology, Computer Science
Jawaharlal Nehru Technological University, India
2014

A thesis submitted in partial fulfillment
of the requirements for the

Master of Science in Computer Science

Department of Computer Science
Howard R. Hughes College of Engineering
The Graduate College

University of Nevada, Las Vegas
May 2016

UNLV | GRADUATE COLLEGE

Thesis Approval

The Graduate College
The University of Nevada, Las Vegas

April 22, 2016

This thesis prepared by

Sai Ram Ganti

entitled

Security Threats from UAS and Its Mitigation through Detection and Tracking

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science
Department of Computer Science

Yoohwan Kim, Ph.D.
Examination Committee Chair

Kathryn Hausbeck Korgan, Ph.D.
Graduate College Interim Dean

Ajoy Datta, Ph.D.
Examination Committee Member

Jisoo Yang, Ph.D.
Examination Committee Member

Venkatesan Muthukumar, Ph.D.
Graduate College Faculty Representative

ABSTRACT

SECURITY THREATS FROM UAS AND ITS MITIGATION THROUGH DETECTION AND TRACKING

By

Sai Ram Ganti

Dr. Yoohwan Kim, Examination Committee Chair

Associate Professor, Department of Computer Science

University of Nevada, Las Vegas

Unmanned Aerial Systems (UAS) are being used commonly for surveillance, providing valuable video data and reducing risk for humans wherever applicable. The cost of small UAS can range from as low as $30 to high as $5000, which makes it affordable by everyone. Most of the UAS are equipped with a camera which results in activities like disruption of privacy or capturing sensitive data. This research is aimed at developing a system which can detect and identify a drone and apply some counter measures to stop its functions or make it go away. The air traffic will increase significantly in next 20 years according to FAA (Federal Aviation Administration) and is doubling every year which increases the happening of all risks due to them. This system will detect and identify a drone through various methods combined into a single algorithm which includes image processing and audio detection. This research also proposes a new method where two algorithms run in sequence namely motion detection and SURF (Speed up Robust features). This system as a whole will warn a person when a drone is nearby and apply all the above mentioned techniques to keep the privacy intact and will also help keep the malicious or harmful drones away from the restricted/residential areas in the coming years.

ACKNOWLEDGEMENTS

I would like to thank Dr. Yoohwan Kim, my research advisor for all the support and guidance he has offered me during the course of my graduate studies at University of Nevada, Las Vegas. His encouragement and valuable suggestions have helped me immensely in moving in the right direction for this thesis. He has helped me by providing all required resources for useful research with good hands on experience.

I would also like to thank Dr. Ajoy K. Datta, Dr. Jisoo Yang and Dr. Venkatesan Muthukumar for serving my committee and reviewing my thesis. The research work I did behind this thesis has been a challenging experience to me and was accomplished through help of many people. I would like to heart fully thank my parents Ganti Aruna and Ganti Sarma who have given me the motivation to learn, opportunities to grow. My uncle Vishwanatham Peri was the backbone of my education in terms of finances and his support shall never be forgotten. I would also like to thank my brother Uday Vadlamani for guiding me through entire process patiently and special thanks to my sister Shalene Vadlamani and Dr. Ajoy K. Datta who helped me the most in getting an assistantship at Department of Computer Science.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Outline

Small UAS including drones are having many advantages but also hold many cases of privacy disruption and other illegal activities that pose threat. These drones can be purchased by anybody and do not need any permits as of 2015 and FAA soon passed rules on the permits and altitude limitations later. As mentioned the drone traffic is doubling every year which exponentially increases the rate of risks and hazards caused by it. This paper proposes a design for detecting identifying UAS or drones within a limited distance and applies a counter measure. [1] Drones flying nearby can be disturbing in many ways which include capturing your images without permission or transmitting a live video. In the coming years it will be essential to have a technology or a system which can monitor [2], identify and repel the drones from these areas. The existing designs solve the problem to some extent and are very costly. This research is to develop a system which is affordable and applicable in most of the situations. It will include functions like motion detection with static background, motion tracking mechanism which will draw a line along the path of the drone and audio detection which identifies the presence of the drone nearby using frequency analysis. This design also includes the Speed up Robust Features (SURF) algorithm to identify the drone type and its brand, which will make the equipment apply right counter measure. The main motto of this research is to:

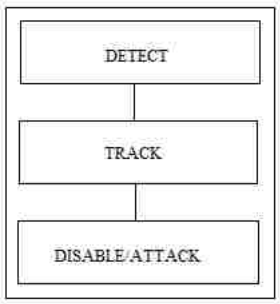

Figure 1: Major steps towards drone control

1.2 Methods of Detection

The static background motion detection will include a fixed camera and single board computer which will process the frames captured by the camera [3]. The system needs to be small and portable so that it can be fixed anywhere. The pan and tilt mechanism is responsible for keeping track of the flying drone within the frame. This tracking mechanism can be used to mount any counter measure equipment on to it like a vortex gun or a high energy RF (Radio Frequency) gun. There are many cases of drones flying nearby and trying to capture images, which makes this research important. Drones can be detected using various methods as mentioned below:

a) Audio Detection

b) Video Detection

c) Thermal Detection

d) RADAR Detection

e) Radio Frequency Detection

Each of these methods has their own limitations and advantages. Detection of drones is a challenging task as there are many objects in the air like birds, commercial airplanes and other objects. This tends to increase the rate of false positive detection. The detection rate is limited for every method, but accuracy varies. Each of these methods is described below.

A. Audio Detection

This technique uses microphones in multiple directions and records the nearby sound. Most of the microphones can pick up data from 25ft – 30ft maximum. The sound waves are then processed and average frequency is analyzed [4]. Drones are equipped with brushless direct current motors which

generate a hissing sound of high frequency around 40Khz. This sound is unique for most of the drones. Using digital signal processing the specific frequency can be identified and presence of a drone nearby can be detected. This method works in quieter environments and will fail in urban areas or noisy environments.

B. Video Detection

Video detection is a useful tool, but with some limitations. [5] Cameras can see out to about 350 feet with economic resolution, but have a very difficult time distinguishing birds from drones. Basically, anything flying in the air is a "drone" as far as a camera is concerned. Even by utilizing computer algorithms that look at flight patterns, the prevailing thought is that a bird will fly a more random pattern than a drone would. Unfortunately, as we have discovered, this notion fails in a place where birds glide. An excellent example of this is seagulls. They will ride wind currents and stay at a steady level, and this fools video systems. Motion detection combined with SURF (Speed up Robust Features) algorithm proper detection of drones can be done while successfully ignoring other flying objects. Motion detection can be useful for drawing the path along the motion of the drone.

C. Thermal Detection

Thermal detection works much better on drones that have a propulsion engine, mostly fixed wing drones. It works up to a distance of 350ft. [6] This detection fails when drones contain more amount of plastic or radiate less heat, like that of the plastic quad-copters which generate almost no heat. In such cases this mechanism is more likely to consider a bird as a drone as it carries more heat. Propulsion engines like the turbo-fan or the turbo-jet engines generate hot gases from the exhaust which makes it easy to detect. This mechanism can be used as an add-on with other detection mechanism but will not work as standalone. Implementation costs are very high and rate of detection is very low and limited by distance.

D. RADAR Detection

RADAR is useful in detecting large aircrafts but fails when the size is as small as a drone or a quad-copter. Radar has a hard time picking up these small, plastic, electric-powered drones because that is not what they were created to do. If this equipment is modified to detect small objects too then the amount of noise will increase exponentially as birds and all other flying objects will come under the threshold. This drawback limits the usage of RADAR for identifying small drones.

E. Radio Frequency Detection

This is one of the effective ways for long range detection of drones. Range of 1400ft can be achieved for successful detection. It is difficult to design a drone which can escape RF detection. UAVs with fixed wing propulsion engines can fly at very high altitude and they are an exception for this technique. The detection success rate depends on the power of the receiver.

CHAPTER 2

THREATS FROM UAS

Quad-copters and drones have no physical boundaries and can fly to anywhere until unless there is a physical barrier. [7] These flying drones possess severe threat as many of the octocopters are capable of lifting weights as much as 20lbs. There are capable of carrying weapons and trigger them at the controllers wish. This can be disastrous because as the drone traffic increases it becomes highly difficult and unpredictable to identify a drone that is potentially dangerous to life. Military issues arise due to disruption of privacy; these drones can carry powerful cameras which can gather high amount of detail. Drones are also a big threat for the commercial airliners. Single drone entering the engine of an airplane can cause engine failure which can be fatal.

The increase in the drone count every year will also increase the rate of above mentioned attacks. There are very few solutions and methods to prevent the drones from doing the illegal and dangerous activities. The existing solutions do not solve the problem completely, but are capable of detecting a drone to some extent that too in a specific environment only. Companies like Blighter, Drone shield, De-drone have designs which are very expensive and difficult to implement for residential areas. For small scale they are not at all feasible to implement. The Table below compares the features of existing anti-drone solutions in detail. The threats caused by drones are categorized briefly into 5 categories.

2.1 Types of Threats

A. Private Property surveillance

Drones equipped with high quality cameras are available at lower price and are accessible to everyone. This is a major issue as any drone with a camera can easy breakthrough the geo-boundaries along with video capture. This will need the civilians to use additional devices to secure their private places. It poses a great threat to the military where their strategies may be exposed to the opponent. The area owners

may declare their land as no drone zone. This will be effective only if GPS based central data surveillance is implemented and all drones must comply with it. This also includes capturing images or streaming videos from an undesired location such as hotels or high storied buildings.

B. Attack on Commercial Airlines

This type of attack can be disastrous and can be responsible for lives of many. Drones can reach up to altitude of 1600ft and a distance of 16000ft can be achieved using a drone like DJI Phantom-3. [8] A fixed wing drone can reach heights as much as 15,000ft. This is a potential risk for commercial airliners which just take off or are about to land. Smaller drones are capable of hovering at a height of 1000ft near the airports and can cause severe engine damage by entering into the propulsion system of an airplane.

C. Transportation of illegal goods

Depending on the propeller count, a drone's thrust is determined. An octocopter is capable of lifting 20lbs. This is really convenient to transport smaller goods or packages in a short distance without being noticed by anybody. This can also include dropping bombs or carrying dangerous goods at low altitudes.

D. Attacks on people

As mentioned a drone can lift up to 20lbs, any weapon can be mounted underneath and can be triggered at the controllers wish. This can be lethal in urban areas where people roam around on streets and a drone with a weapon is just unavoidable and can kill people instantly. These types of attacks are estimated to happen in the coming decade. FAA is already developing policies and regulations for all the drones that will be in the skies in the coming years. Due to different protocols, managing the rules becomes difficult.

E. Swarm Attacks

Military is one domain which has both advantages and disadvantages with this issue. Swarm attacks will help the military to save lives of many militants who actually involve in firing. Drones in a pattern can attack an area more effectively without loss of life. [9] Swarm attacks can help in speeding the process by several times. Multiple drones can coordinate together to perform this task.

Apart from these issues drones will have several other disadvantages because it is a device that an individual can control remotely and perform many actions using a small controller. People are more likely to get attacked by a drone in the future as the attacker can be secure few miles away and can attack easily.

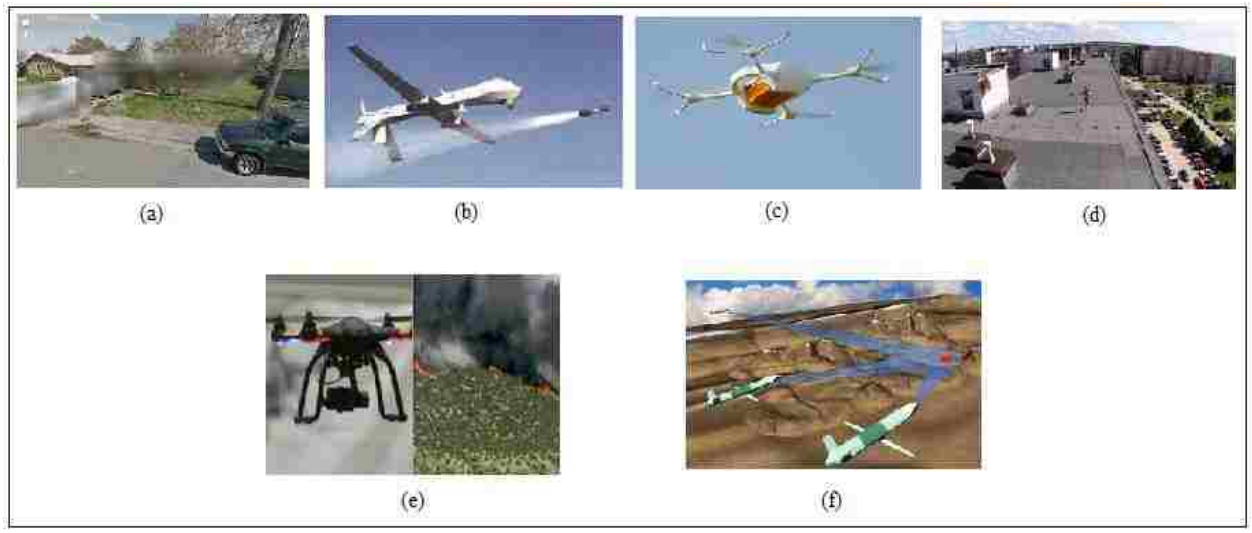

Figure 2: Example images for each of the above mentioned attacks [10]

CHAPTER 3

CURRENT TECHNOLOGIES

3.1 Comparison

There are very few solutions which are available in the market and lack in accuracy. Table 1 shows the existing industrial solutions to detect and identify drones and apply some counter measures to it.

Feature	Drone Shield	Orelia	Dedrone
Audio detection	YES	YES	YES
Radio Frequency Detection	NO	NO	NO
GPS Detection	NO	NO	NO
Video Detection	NO	NO	YES
Thermal Detection	NO	NO	NO
Radar Detection	NO	NO	NO
Detect Flying Drones	YES	YES	YES
Detect Ground based Drones	NO	NO	NO
Detection Strength	Weak (1-layer)	Weak (1-Layer)	Weak
Detection distance	40m Audio	40m Audio	100m Video

Table 1: Comparison of Anti-Drone Solutions by various vendors [11][12][13]

Three of the leading companies are compared in the above table namely Drone shield, Orelia and De-Drone. Apart from the basic detection methods discussed above, there are other factors which make the equipment reliable. Those are the range of detection and what all it can detect. It is also important to identify the loopholes and the false detections with each of the above design.

In video detection with motion tracking, most of the moving or flying objects will be considered as drones with respect to the camera. Advanced detection techniques with machine learning are required where the flying patterns and specific shapes of the drones are trained as data to a classifier.

Figure 3: Rapere, Interceptor, Battelle Drone Defender

In the above figure more current solutions are shown. The Rapere [14] is a drone that contains entangling net attached to the bottom part of the drone. Using camera and motion tracking it reaches on top of the target drone and then releases the cable so that the propellers of the target drone get entangled and the drop the drone down. This mechanism is effective but if the target drone specifications like air speed is high, then it is really difficult to reach on top of it. Interceptor is another design which does the same thing but in an efficient way. The net keeps hanging down and all it has to do is go past the target drone. This design also has limitations with respect to target drone speed and accuracy.

The third one "Battelle Drone Defender" [15] is currently the best available design. It has been designed for the military and is not yet approved by the FCC. Even after it gets approved, it won't be available for the common people as this involves high power radio guns which are illegal. Drone defender houses two directional antennas one for [16] jamming the drones GPS module and other for jamming the drones control signal [17]. This combination is so effective that it can bring down a drone in few secs and also has very long range. The major disadvantage is it is a high power device and is illegal to use in residential areas. The price is also not economic.

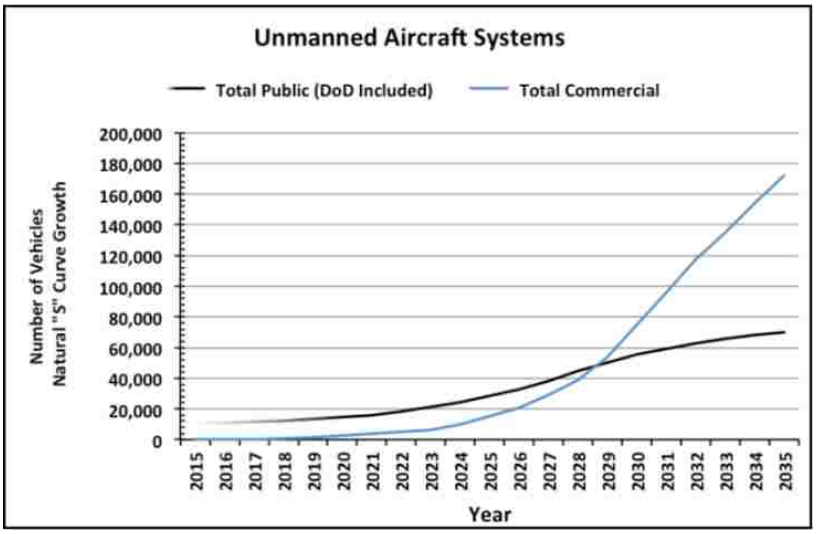

Figure 4: Drone traffic and public air traffic [18]

It is estimated by the Department of Transportation that around 250,000 drones will be in the US Airspace by 2035. This number includes military drones for Air Force, Navy and Army, Public Agencies like Federal, State and Local. [19] Hence it is important for the development of every technology to make sure that the increase in drones will not be catastrophic.

CHAPTER 4

PROPOSED DETECTION AND TRACKING TECHNOLOGY

A low cost system which has reliable detection rate which includes video detection, audio detection and a provision to mount a RF gun [20] if necessary will make it easy to use it for residential and small scale purposes. This design consists of a camera which captures frames at the resolution of 640x480 at a frame rate of 30Fps. Single board computer is used to process those frames by performing image processing. Motion detection is done using the absolute difference between two consecutive frames. This difference is then enhanced and tracked continuously over the loop. This tracking mechanism uses a set of pan and tilt servo mechanism, camera module and a microcontroller, Arduino.

Figure 5: Prototype for drone detection and Tracking

This design is very robust, simple and cost effective for residential areas specifically. The tracking mechanism may be mounted with a RF gun or a vortex gun which will deflect the drone when it hovers within the frame. This system also uses a microphone which is analyzed by a digital signal processor (DSP). The analyzed frequency undergoes a template matching technique where it is compared with

sounds with frequency around ~40KHz. Drones usually produce sound at this frequency when the propellers are spinning at high RPM.

Every camera frame is individually processed using robust functions of OpenCV [21], which is an open source library. Every frame undergoes a set of functions which reduce the output more specific towards motion detection. The microcontroller is connected over serial port to the main system and the camera is also connected to the system. An application is written in C++ which handles all the image processing functions which are linked to the OpenCV library. The application handles the serial port of the microcontroller and sends the tracking data of the object in motion. The coordinates in the frame of 640x480 are sent to the Arduino (Micro-Controller) each set (x,y) at a time [22]. It is a two tuple data of X coordinate and Y coordinate.

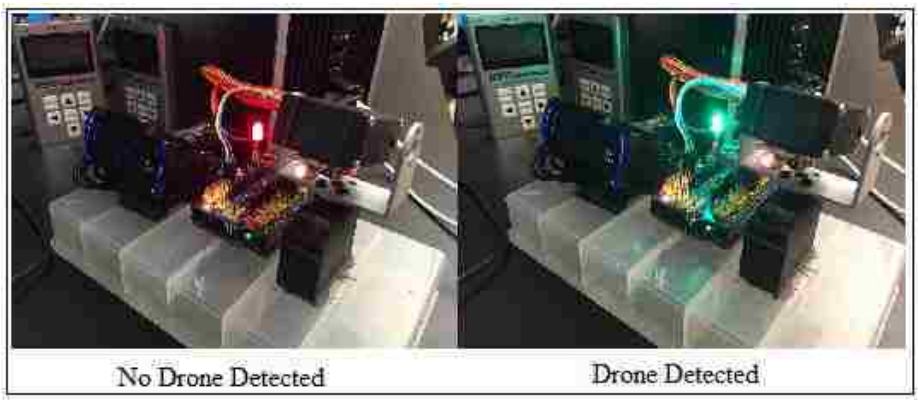

Figure 6: Status Indicator showing

There are two status indicators which indicate whether the presence of the drone. One is a light indicator which shows up green when a drone is hovering within the frame and red when it is not. The second status indicator is a sound alarm, which sounds in the presence of a drone. This system will continuously monitor for any motion detection which is supposed to happen with a static background, most probably sky and other non-moving objects. The status indicators may be disabled at the users wish.

4.1 Motion and Object Detection Mechanism

The camera frames are captured one by one and then processed by the OpenCV functions to detect the target. There are different detection mechanisms. Depending on the requirements one of them or a combination can be selected.

A. Motion Detection with Frame difference and Threshold.
B. Color based tracking
C. Speed up Robust Features (SURF)
D. Optical Flow

These four algorithms can be used to detect an object or its motion precisely. Each of the algorithm and its role in the current research are described below.

4.1.1 Motion Detection with Frame Difference and Threshold

This algorithm can at most identify the center point of the object that is in motion. It is assumed that the object is completely isolated within the frame with no contacts that extend out from the frame (Ex: A Quadcopter hovering in the mid-air). The flowchart for this algorithm is presented below.

This algorithm gives precise detection of movement and it can also detect multiple objects in motion at a time. Implementation of the algorithm is shown below, where the movement of the hand is recognized and the approximate center coordinate of the hand is indicated by a green circle [23]. The Frame-1 shows the subtracted image between Frame-2 and Frame-1. The threshold image shows the output after applying sequences of erosion and dilation. Now the binary image is sharp and contains the largest contour which is nothing but the object in motion.

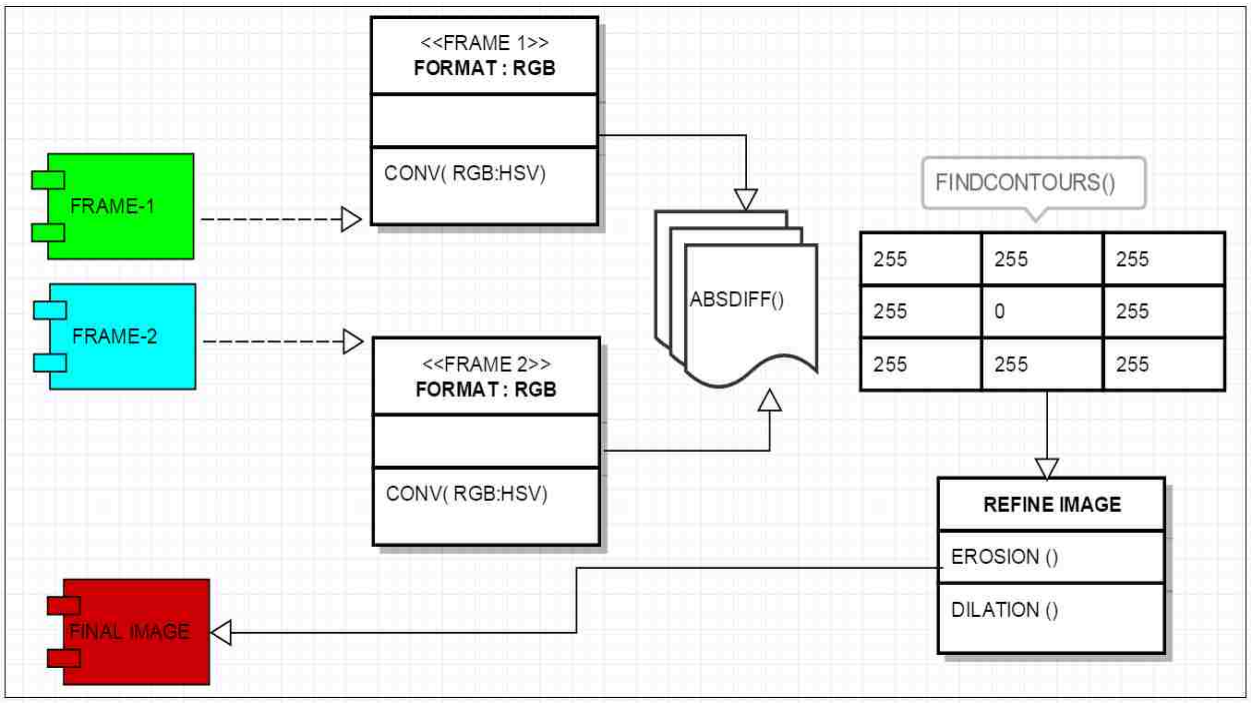

Figure 7: Algorithm for Motion Detection using Frame Difference Method

This algorithm will consider any motion as a drone. To avoid false detections, the SURF algorithm can run in parallel with this algorithm. This will allow the output to be filtered by motion detection first, and then followed by SURF. This will enable identifying a drone only if it has the form factor and shape of a drone or a quadcopter. This algorithm as a standalone without SURF will also work great at outdoors considering sky as most of the background for every frame. Birds and commercial aircrafts again come into exception. There are two ways this algorithm can be implemented:

- Case-1 where every alternative frame is updated as background for next consecutive frame. This will reduce the rate of false detection and consider only recent changes.
- Case-2 where a static background stays constant from the beginning and anything new in the picture is considered as motion detected. This will have poor performance when compared to above case and have high rate of false detections.

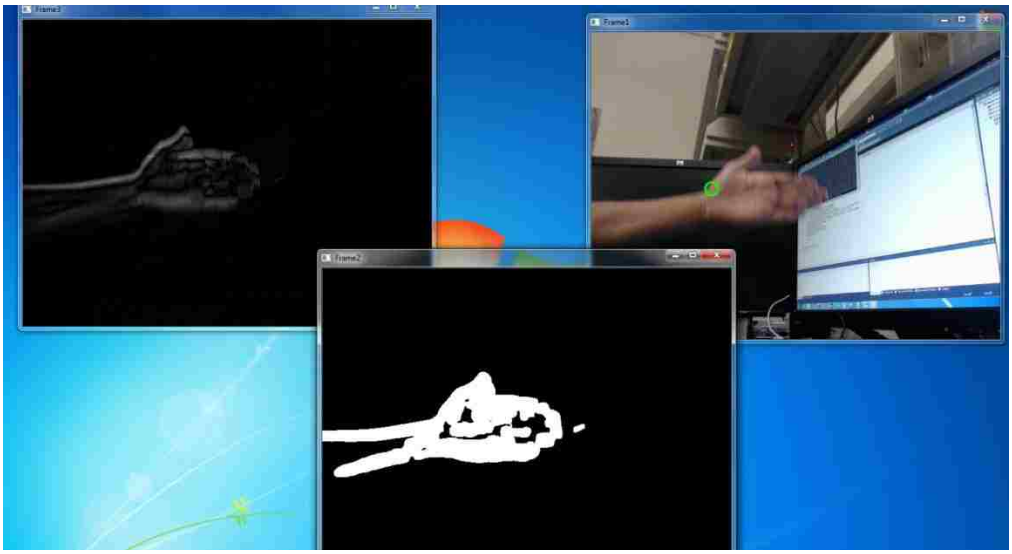

Figure 8: Sequences of the Motion Detection Algorithm

Below is the implementation of the algorithm where a drone is flown past the detecting equipment/camera. The above process of back ground subtraction and finding the largest contour is applied to each frame in real time. Below images illustrates different scenarios where the drone is identified during its flight. A simple mathematical notation for this algorithm would be to take the background image and the frames captured at time t, denoted by I(t) and compare with the background image denoted by B. Using arithmetic calculations the objects can be segmented by using image subtraction method for every pixel in I(t). The pixel value of P[I(t)] is subtracted from corresponding pixels of background image denoted as P[B] [24].

$$P[F(t)] = P[I(t)] - P[B]$$

Let the background image be some frame at time t. The difference image will show changes in pixel values only at some areas. This method needs the background to be static. To filter the image further we threshold it.

$$|P[F(t)] - P[F(t+1)]| > \text{Threshold}$$

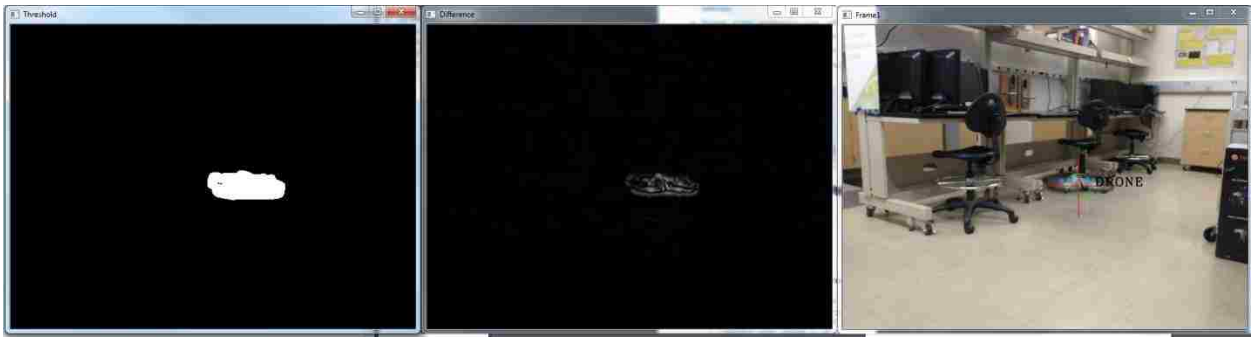

Figure 9: Optimum level of motion sensitivity

Figure 10: Increased sensitivity level is prone to false motion

Figure 11: High sensitivity will lead to false detection

Background is estimated to be the previous frame, and then the background subtraction equation becomes:

$$B(x, y, t) = I(x, y, t-1)$$

$$|I(x, y, y) - |(x, y, t-1)| > Th$$

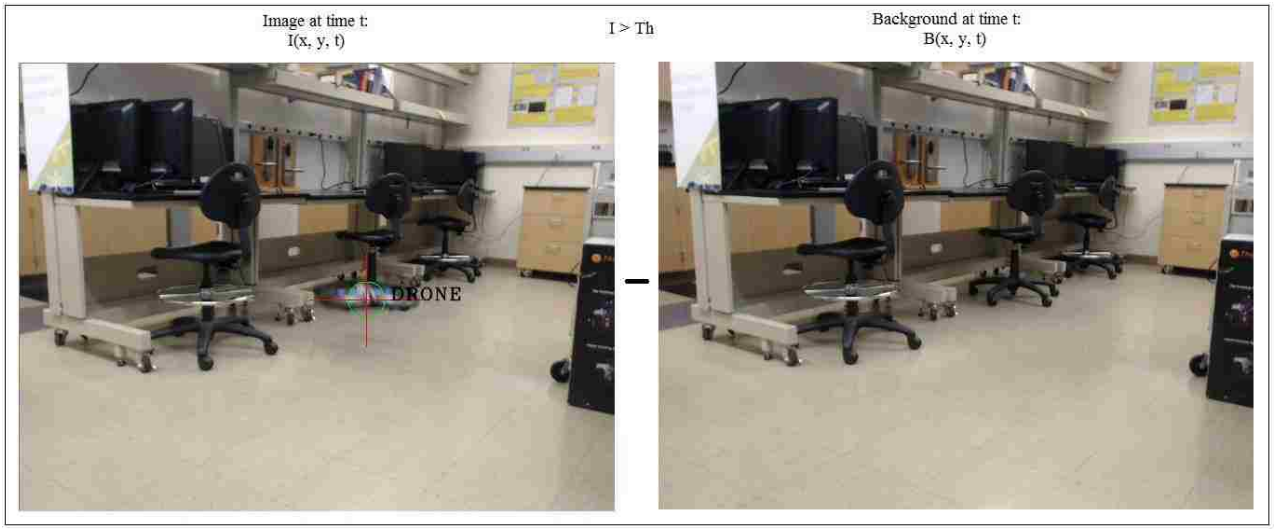

Figure 12: Background at time t and next consequent frame at time t

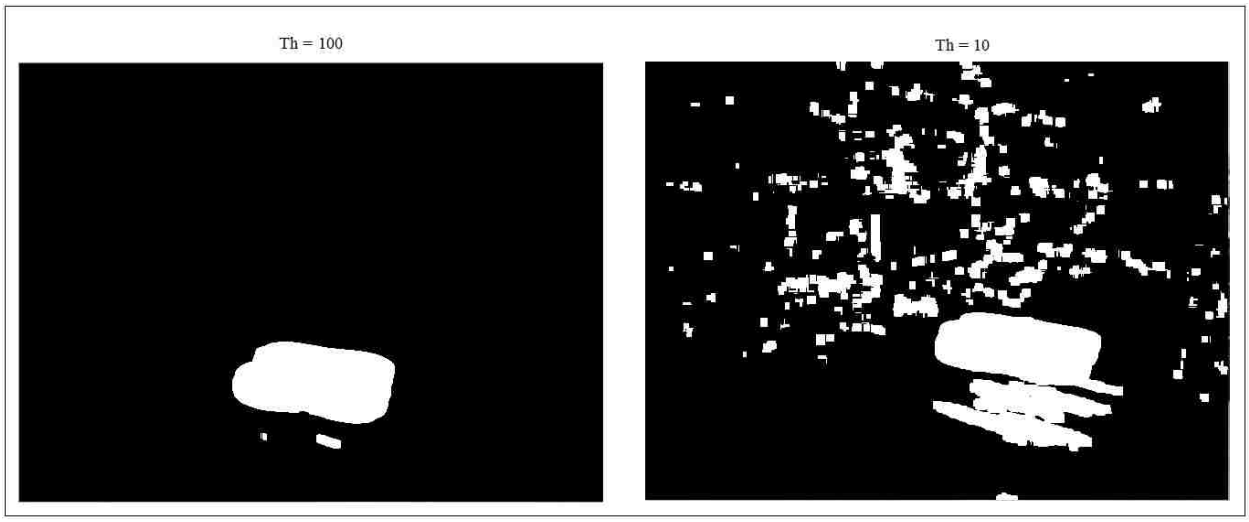

Figure 13: Threshold values control how much motion has to be identified

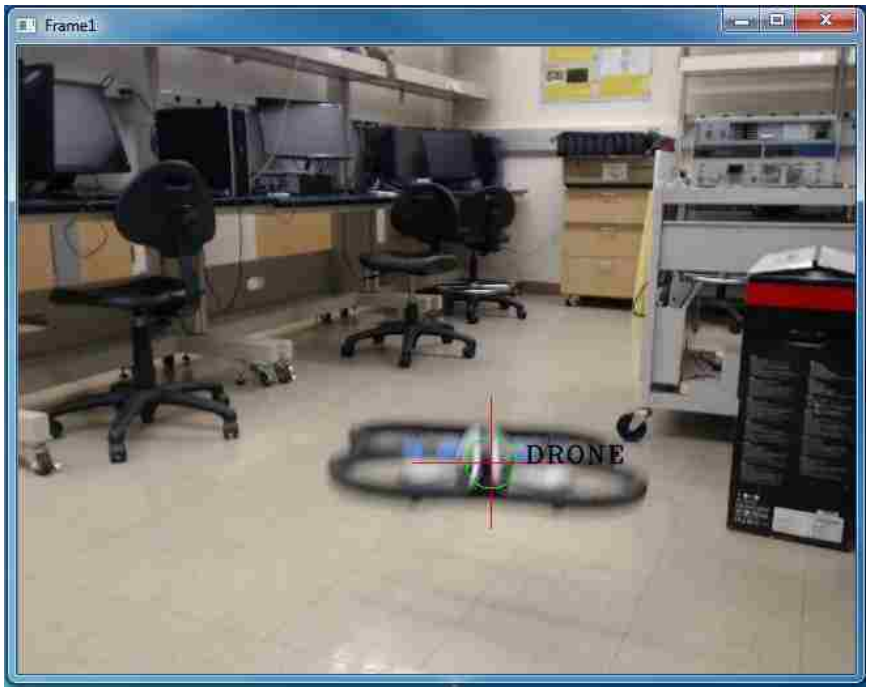

Figure 14: Drone identified during its flight

4.1.2 Color Based Tracking

This is faster when compared to the former algorithm in detecting objects. This algorithm can reach speeds up to 40 Frames per second. The major drawback of this algorithm would be it depends on the color of the object to detect it. [25] Drones of specific color can be distinguished better than other flying objects like birds. The flow of algorithm is described in the below image. This algorithm needs only one frame at a time to process hence the speed will be double of the motion detection algorithm. If the color intensity or shade varies during the capture then this algorithm may fail to identify even if the color is close to what the threshold value was.

A dynamic version of this is also available where in one can select the required colored object in the view finder. The HSV values for the objects color will be applied to the In-range function dynamically. In this method any required color can be identified and tracked without calibrating the values repeatedly.

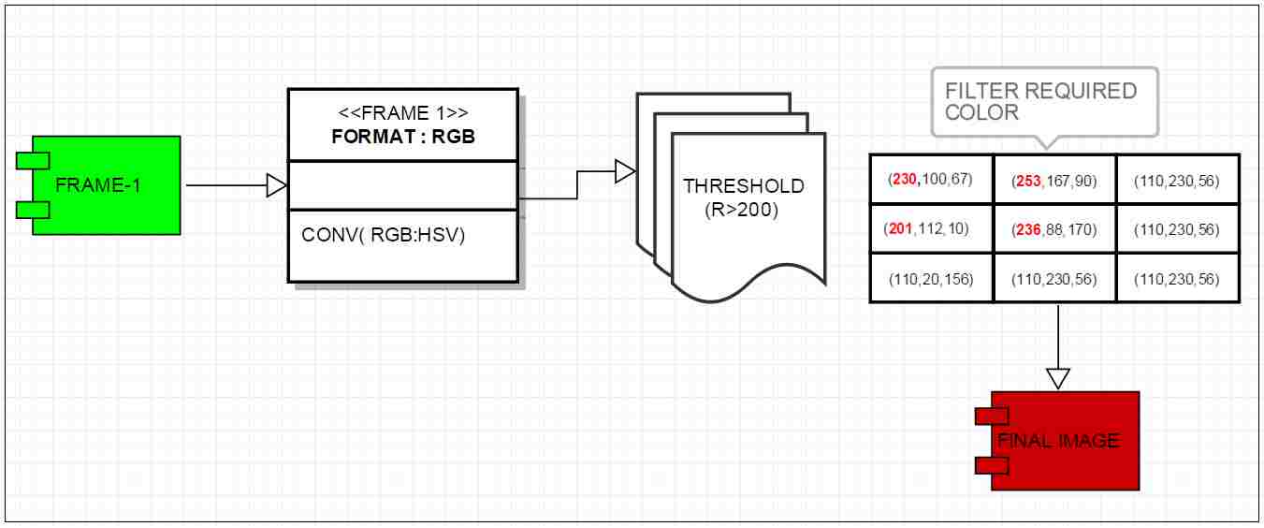

Figure 15: Algorithm for Color Based Tracking

The In-range function has two inputs in the form of HSV (Hue, Saturation and Value). The first set of input would be the maximum allowed HSV for a given color and second set is the minimum set of HSV for the same color to be identified. This is the function used to perform In-Range operation in C++ along with OpenCV, the values are to detect Red color.

InRange(hsv, Scalar(157, 72, 156), Scalar(180, 169, 255), binary);

After the In-Range function is applied [26], the pixel values which satisfy the condition are marked with "1" and rest with a "0". This gives a binary image which consists of the desired colored object in white. The implementation of the same is shown below.

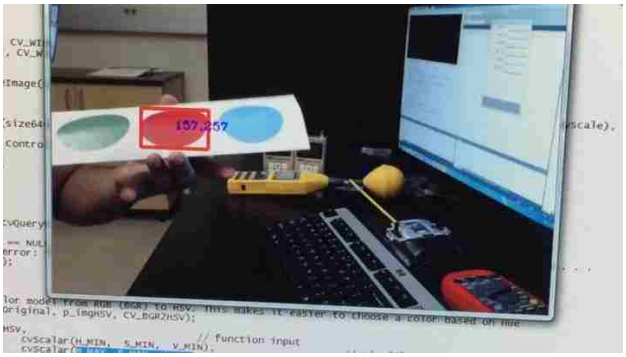

Figure 16: Red color detection using In-Range function

4.1.3 Speed Up Robust Features (SURF)

This is one of the advanced algorithms for object identification using feature extraction. [27] This is very robust and powerful as it does not depend on the color, scale and orientation of the object. This algorithm is scalable and can identify similar objects of any shape or color. This gives many advantages over the above mentioned algorithms. It takes a sample image of the object to be identified and then extracts the features into vectors with directions. Whenever a new frame is captured by the camera, this algorithm will search for the same features and then perform feature matching. The algorithm sequence is shown in the below figure.

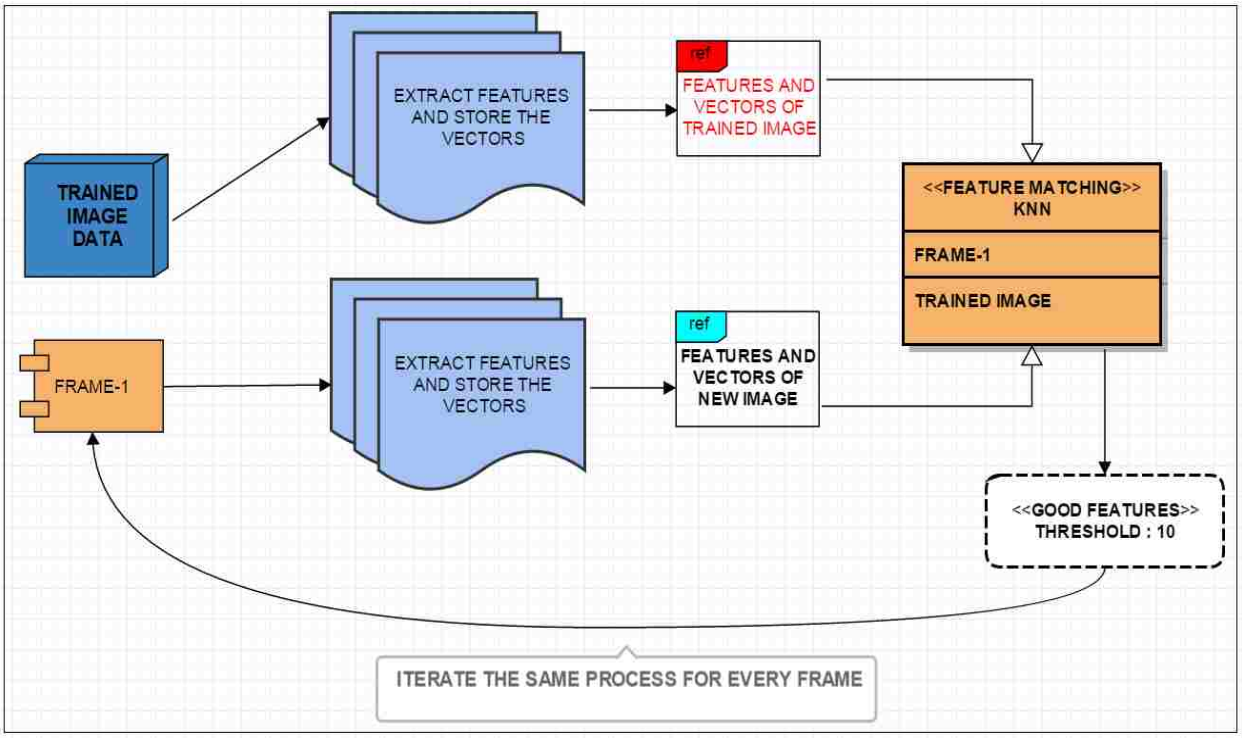

Figure 17: Algorithm for Speed up Robust Features

This algorithm is robust because orientation of the image doesn't matter. Even if the target image/object to be identified is inverted or upside down, it can still identify because the directions of the vectors and features match perfectly. Below image shows the method by which the feature vectors

are identified. The feature data is stored as matrices which contain the color distribution data in grayscale and the position data with respect to every other neighboring pixel. In the image below a 9x9 image is processed and the feature matrix contains only three sections of 1,-2 and 1. The positive values are for the white and negative are for black. The number indicates the amount of pixels occupied with white or black.

SURF uses a hessian based blob detector to find the features and interest points. The response is expressed by the determinant of the hessian matrix and is an expression of the changes in the pixel values of the image. Lxx(x, σ) in equation 2 is the convolution of the image with the second derivative of the Gaussian. The heart of the SURF detection is non-maximal-suppression of the determinants of the hessian matrices.

$$\mathcal{H}(\mathbf{x},\sigma) = \begin{bmatrix} L_{xx}(\mathbf{x},\sigma) & L_{xy}(\mathbf{x},\sigma) \\ L_{xy}(\mathbf{x},\sigma) & L_{yy}(\mathbf{x},\sigma) \end{bmatrix} \quad (1)$$

$$L_{xx}(\mathbf{x},\sigma) = I(\mathbf{x}) * \frac{\partial^2}{\partial x^2} g(\sigma) \quad (2)$$

$$L_{xy}(\mathbf{x},\sigma) = I(\mathbf{x}) * \frac{\partial^2}{\partial xy} g(\sigma) \quad (3)$$

Figure 18: Determinant of Hessian Matrix [28]

The convolutions are difficult and costly to calculate. They are approximated and speeded-up by using integral images and approximated kernels. This algorithm is rotation and scale in-variant, the descriptors are robust for images that are rotated and perfect with a tolerance of ±15◦ without performing any kind of rotation assignment.

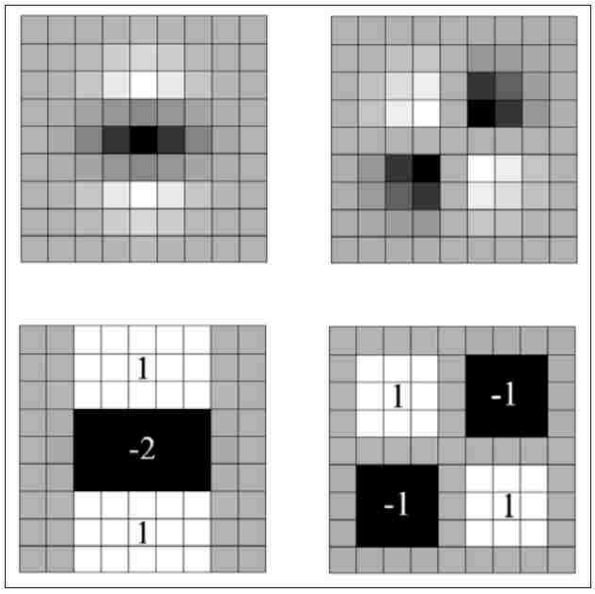

Figure 19: Gaussian Approximation by box filter in SURF [28]

The implementation of the SURF algorithm is shown below in a sequence of images which demonstrate that the algorithm is scale and rotation invariant.

Figure 20: Number of Key Points < Threshold (Object not found)

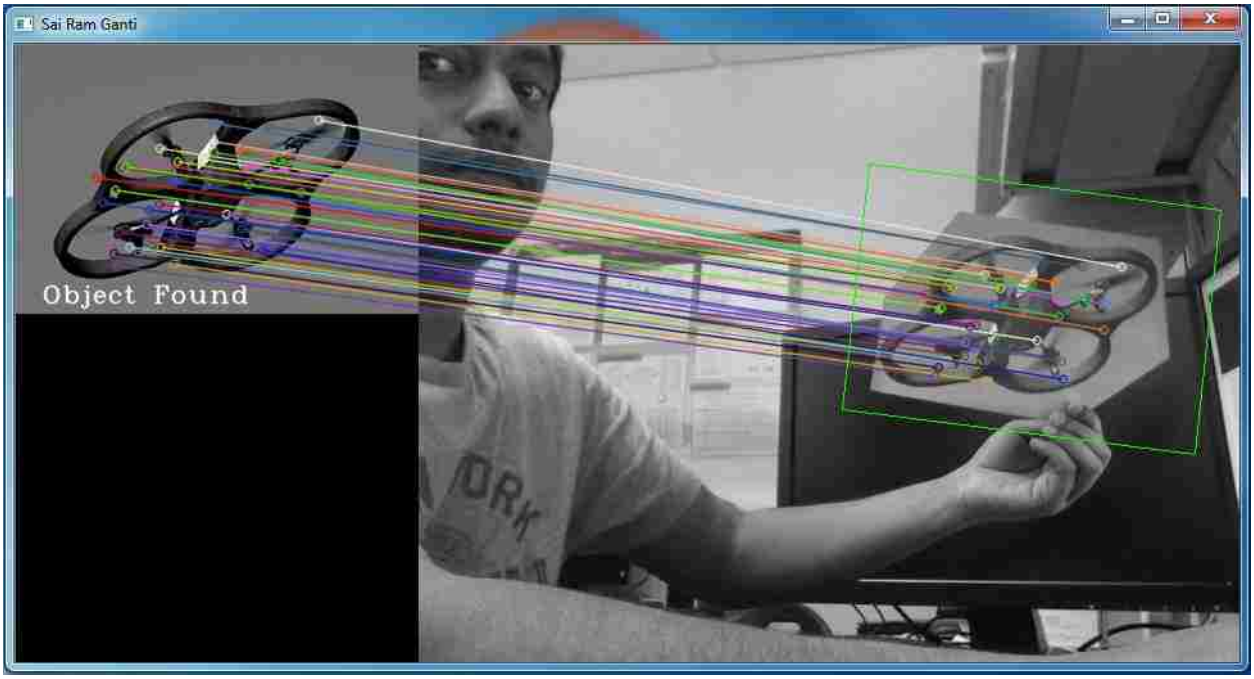

Figure 21: Number of Key Points > Threshold (Object found)

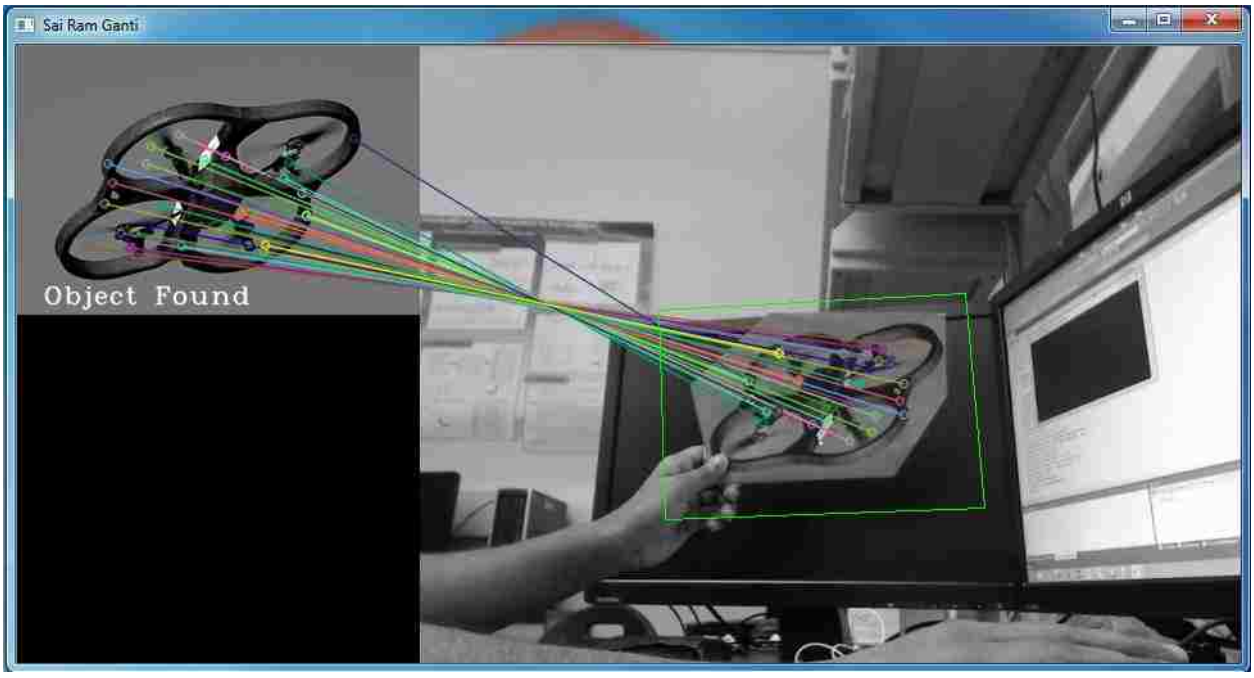

Figure 22: Object identified even if the image is upside down

This algorithm can be overlapped or run in parallel along with the first two algorithms in order to identify the object in motion first and then classify whether it is the required drone or something

irrelevant. This system uses multiple algorithms to determine whether the detected object is a drone or not. The former detection part is done using the motion detecting algorithm.

4.1.4 Optical Flow

This algorithm will allow us to track the area of our interest within the frame. Live feed of the video is given from the camera. Every frame is assigned with some feature points whose movement will be tracked with subsequent frames. [29] Optical flow is the 2D velocity field, describing apparent motion in the image, which results from independently moving objects in the scene or from observer motion. Below images show the implementation of optical flow. This algorithm has few use cases:

- Identifying objects from one frame in other frames
- Determine the speed and direction of movement of objects in motion
- Determine the structure of the environment.

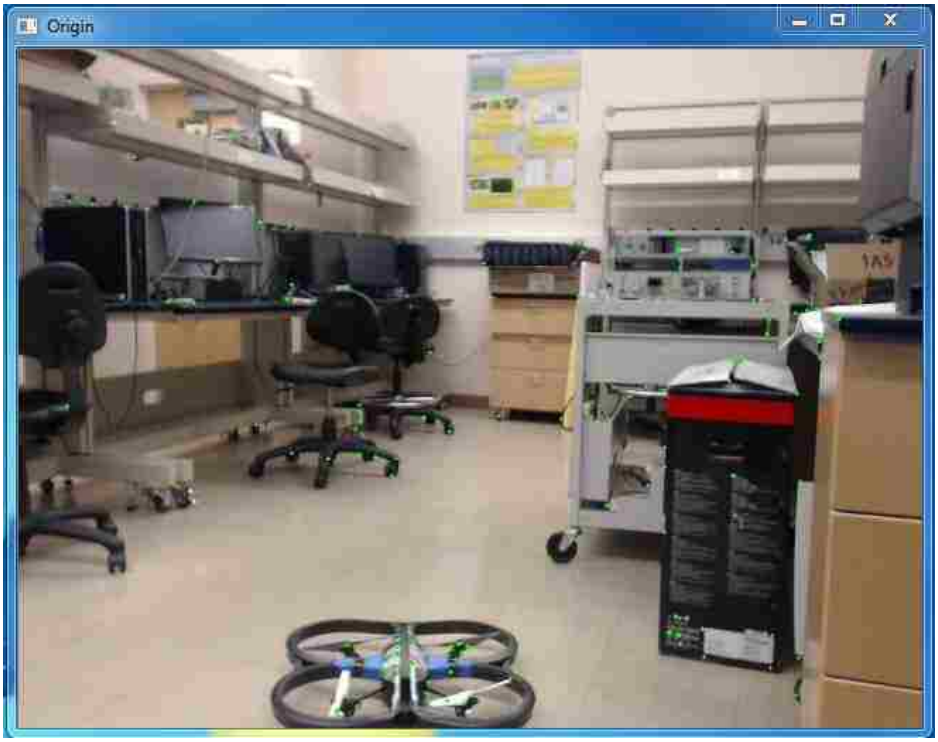

Figure 23: Vector points those are good to track in a given frame

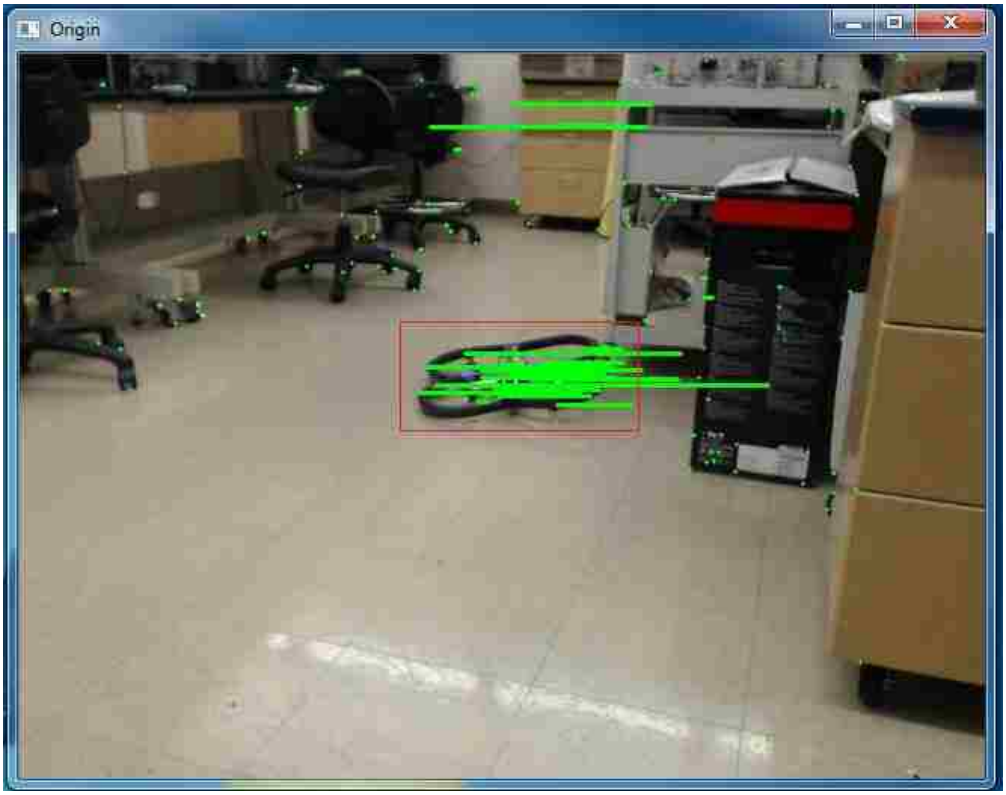

Figure 24: Direction of vectors change as the drone tries to fly

The length of the green lines signifies the drift of the good features to track. The longer green lines imply that the featured points located at that coordinates were moved comparatively faster and longer than other points. This gives us a motion estimate of entire drone with respect to other points. This gives crucial data like which part of the drone has more movement along with the direction [30], which will help in approximating drones movement. This also provides information like the speed of the moving drone and the distance by which the object drifts.

Optical flow alone is prone to false detections of any motion. A separate filtered algorithm was implemented for this where the area to be tracked is selected and the extracted features from that area are tracked on screen.

Figure 25: Remaining feature points remain stable, motion detected areas show change in length

4.2 Motion Tracking Mechanism

The tracking is done in parallel, [31] one happens at the application side where the trajectory of the object in motion is drawn along the object dynamically and the second one is the hardware which has a laser mounted on it. This laser is controlled by a set of pan and tilt servo motors which point it towards the center of the object in motion. This all happens dynamically and gets updated for every frame, 30 frames a sec approximately.

4.2.1 Trajectory Mapping

In all of the three algorithms mentioned above, [32] the center of the object in motion is known in terms of coordinates. This set of X and Y coordinates get updated for every frame along the object as it moves.

Every time the camera captures a new frame, new set of X and Y values are processed and stored in variables. Now for every X and Y the respective pixel can be highlighted by marking it with a specific color.

As this process iterates itself through every frame, the highlighted pixels form a trajectory followed by the object in motion. The image below shows the trajectory of the colored circle.

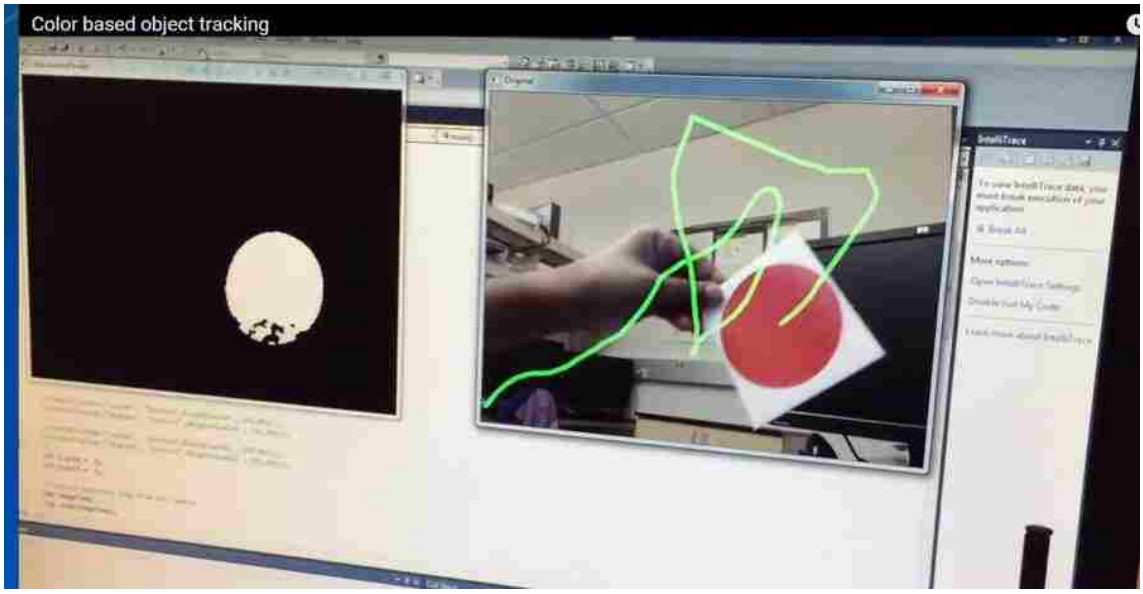

Figure 26: Trajectory of object in motion

4.2.2 Tracking Mechanism

This design has software and hardware modules which communicate with each other through serial port. [33] The software module consists of a C++ executable which binds the OpenCV libraries along the execution. This application first reads a frame from the camera feed and then reads a second frame immediately. Now each frame is converted into Hue, Saturation and Value format and then the second frame is subtracted from the first frame, technically absolute difference is taken and the left over after subtraction is represented in binary. The resulting image is a binary image where the white part

represents the detected motion or the areas where the pixel values were hugely changed depending on the threshold.

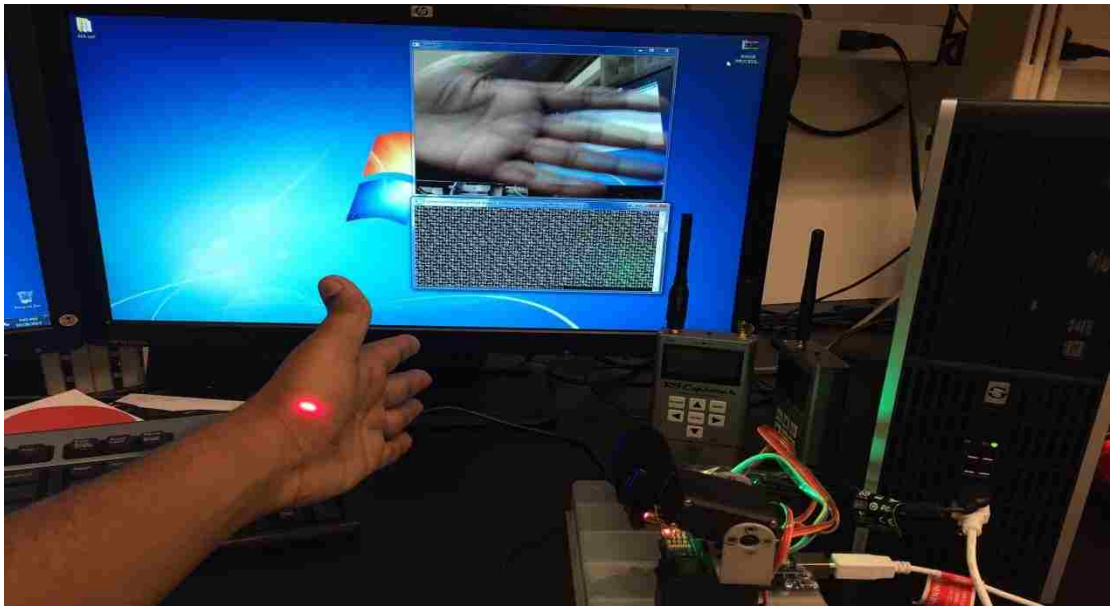

Figure 27: Servo mechanism with laser Tracking

Series of erosion and dilation operations are performed which result in filling the gaps and forming a bigger blob, and makes it easy to identify an object as a whole. The erosion and dilation take care of the noise which moves along with the main object like the shadows or an object smaller than the drone flying nearby. After the final threshold image is acquired, the center coordinates of the blob are taken and fed into variables X and Y which namely are the coordinates of the object in the frame.

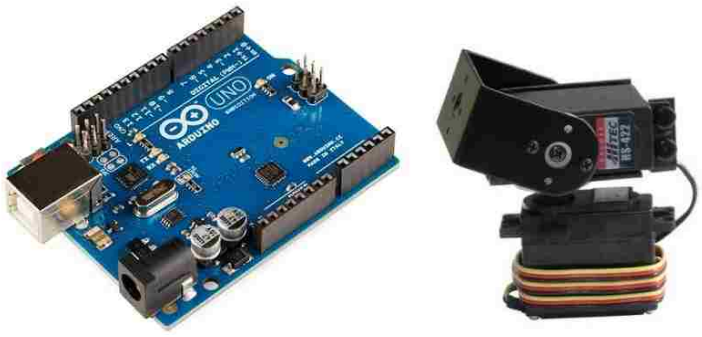

Figure 28: Arduino Uno and Pan-Tilt Servo mechanism [34] [35]

These coordinates are sent over the serial port from where the hardware takes control. A circle is drawn at the position where the detected object in motion is present. In the below image the movement of hand is tracked perfectly and a green color circle is seen on the approximate center. Second window on the bottom shows the values of X and Y being sent over the serial.

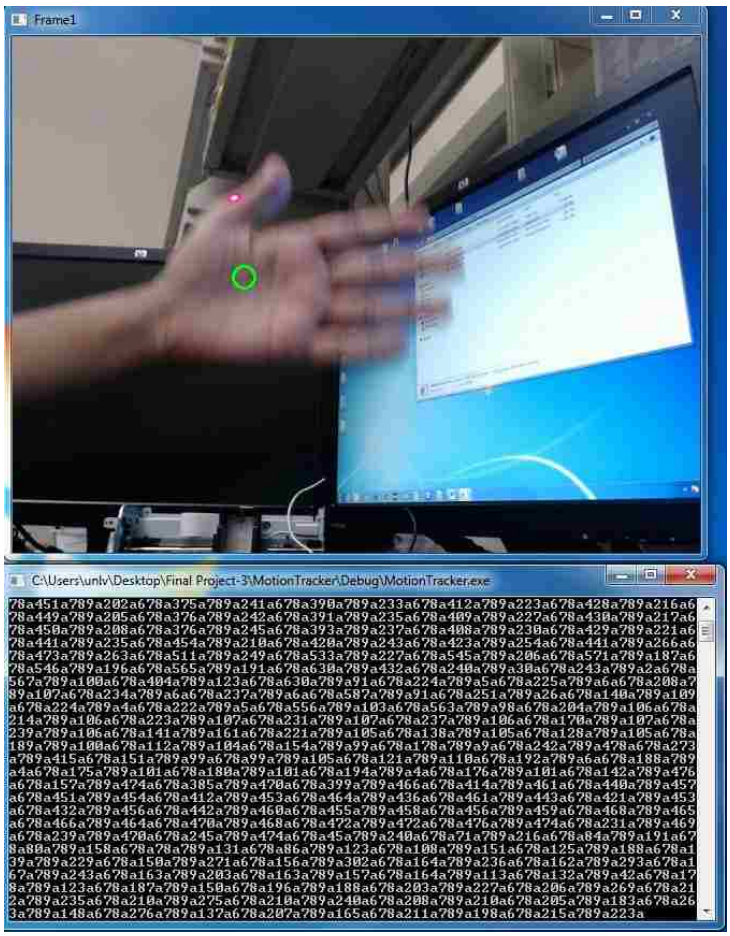

Figure 29: Serial Values sent to the microcontroller

The threshold image is the output of multiple erosion and dilation process [36], which will first remove smaller objects and then collate larger objects which are close to each other. The center of the detected white blob is transferred into X and Y, which immediately is sent over the serial [37]. The Arduino code reads the serial buffer continuously and places the variables by converting the string to integer. This

process happens very fast and an error check code is also implemented because over a period of time the serial buffer gets full and the X and Y values are sometimes swapped and the movement of the motor is affected.

This check code ensures that the Y value is written to the vertical axis motor and X value is written to the horizontal axis motor. This way the tracking mechanism is updated dynamically several 100's of times a second and it moves smoothly tracking the object in motion. Below is the image which demonstrates the working of whole equipment. The precision of the laser is dependent on the distance of the object. The servo has a precision of 1 degree. This spans up to 10ft-100ft at a greater distance. Hence using of higher gears will enable precision. A stepper motor with step count approximately 400 per rotation will enhance the precision even at longer distances. The tracking mechanism and the jitter stay in control. Applying filters will smooth out the jitters and jerky movement of the servo motor, because the camera data is pushed at a greater speed.

CHAPTER 5

SOFTWARE AND HARDWARE ARCHITECTURE

5.1 Software Architecture

This is an integrated system where the actions performed by the hardware are determined by the software. The IDE's used to implement the design are:

- Visual Studio
- Arduino

Visual Studio is used to bind the libraries required for image processing with the main C++ code. The applications are developed with command line interface. Arduino IDE is used to program the microcontroller Atmega-328p on the Arduino Uno. This IDE has a serial monitor which makes it easy to debug the serial data that flows from Arduino to any other software.

Arduino micro-controllers can be programmed in its own IDE which use language syntax that is similar to C. The development board Arduino Uno has a 16.000 MHz clock crystal, below is the pin map of the chip. The internal memory specifications are shown in the table below.

Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB

Table 2: memory Specifications of Atmega-328P

There are various components which are used to alert and inform the user with data related to drone detection. The prototype designed contains a RGB LED and a buzzer. The LED color flashes red when no

drone is detected and changes to green when a drone is detected within the frame. Sound alert is also activated along with the green led.

Figure 30: Pin Map of Atmega-328P [38]

The other main component is the tracking equipment which is deployed using a set of pan and tilt servo motors. Each of the servo motor shafts can go from 0 degrees to 180 degree angle. Combining two degrees of freedom we can cover a hemisphere of 180 degrees in the tracking area. The servo motors are connected to the PWM (Pulse Width Modulation) pins on the Atmega-328P. Different PWM values on those pins will move and position the shaft at required angle. These values in real-time are sent by the image processing application in terms of the coordinates of the detected drone.

The angle of the servo motor shaft is dependent on the length of the pulse or the duty cycle [39]. As shown in the below figure the shaft is aligned at 0 degrees when a 1millisecond pulse every 20ms is provided. To align it to 90 degrees 1.25ms of pulse is required and 2ms for a full 180 degree angle.

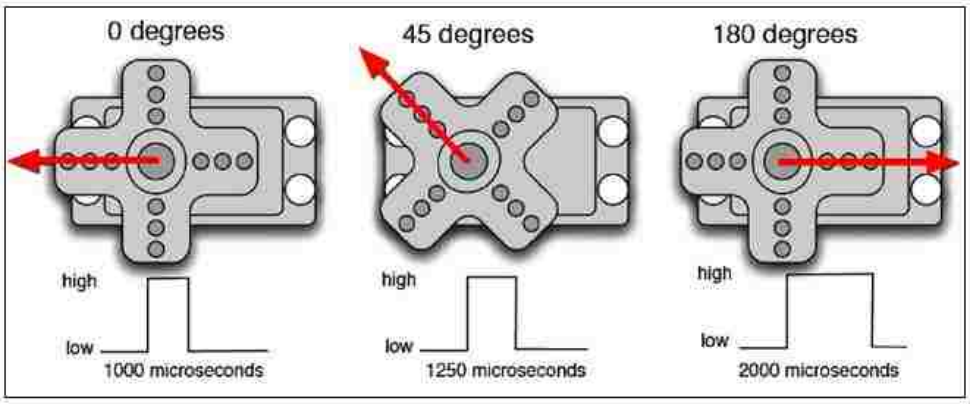

Figure 31: Pulse ON-Time and respective angle map [40]

This process happens dynamically and averaging the values helps in stabilizing the motors and avoids any unwanted jitter or noise due to sudden movements. Stepper motors can also be used to provide a 360 degree of movement and accurate tracking over long ranges.

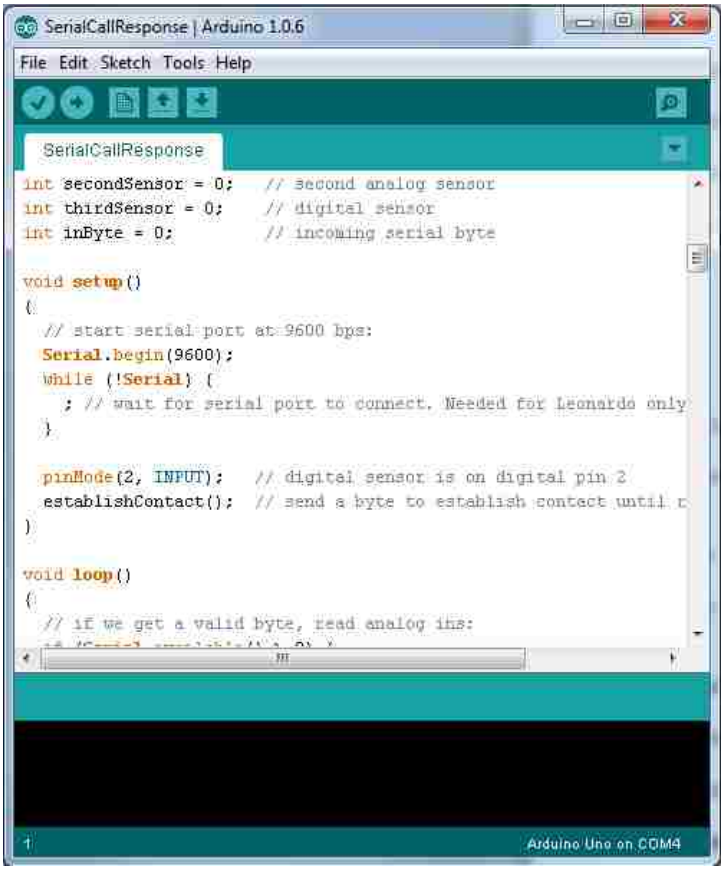

Figure 32: Arduino IDE with sample program

Arduino IDE supports lots of development boards designed by Arduino and other third party vendors. This IDE is used to burn the program to flash memory of the micro-controller. This software communicates through serial port at a standard baud rate of 9600 bits per second [41]. The program is divided into two sections:

- Void setup()
- Void loop()

All the code that rests in the void setup () function will only be executed once and it includes all variable declarations and the direction of registers for input output pins. For instance a sensor would be declared as an input and a motor or an LED will be declared as an output device. Any pins that require pull-ups are also declared in this section.

The void loop () section contains all the functions and lines of code which are executed for infinite number of times or limited number of times depending on the condition. This loop is used for iterative checking of conditions or sensor data. In our case we will be receiving data from the serial port and update those data values to the servo motors in order to keep the drone tracked all times.

5.1.1 OpenCV API

Open Source Computer Vision Library was started at Intel in 1999 by Gary Bradski for accelerating research in the field of computer vision. It is an open source C++ library for image processing which is now supported by Willow Garage. It contains many inbuilt functions mainly aimed at real time image processing. There are many modules in the OpenCv library; important ones are listed below [42]:

- **Core:** This is a very basic module of OpenCV. It consists of data structures like the Mat (matrix) data structure to store images in matrix form. This makes processing the images using their pixel values in matrix form.
- **Highgui:** This module is used for user interfaces, image and video codecs, image and video capturing, manipulating image windows, handling track bars and mouse events. This is also used for reading a video file or writing an image to file or reading frames from camera live feed.
- **Imgproc:** This module is used to convert the color formats of images like form RGB to HSV or Grayscale etc. This also includes image filtering using inRange functions and image transformations.
- **Video:** This is a video analysis module which includes object tracking algorithms, background subtraction algorithms.
- **Objdetect:** This module includes object detection and recognition algorithms for standard objects and shapes.

Artificial intelligence through neural networks is also supported by OpenCv where classifiers of our requirement can be developed specifically. These are helpful in image correction, noise reduction and intelligent detection of faces [43] or any required objects. Many algorithms have been developed for motion detection, color identification and feature extractions. Different versions of OpenCV have been released till date and all versions are compatible with different programming languages like JAVA PYTHON, C++, QT (For developing UI).

5.2 Hardware Architecture

The figure below shows all the components involved in tracking the drone. The communication between every component is also shown in detail. The servo motors are attached to the PWM pins and the sound alert, RGB LED are connected to the digital pins. RX and TX pins are used to receive the serial data over the serial port. The data is being sent by the CPP application along with OpenCV API.

The communication is unidirectional where the software application sends the drone positional data to the tracking equipment and also some bytes of data which signify the availability of the drone within the current frame. These extra bytes control the sound alert and the LED status indicator.

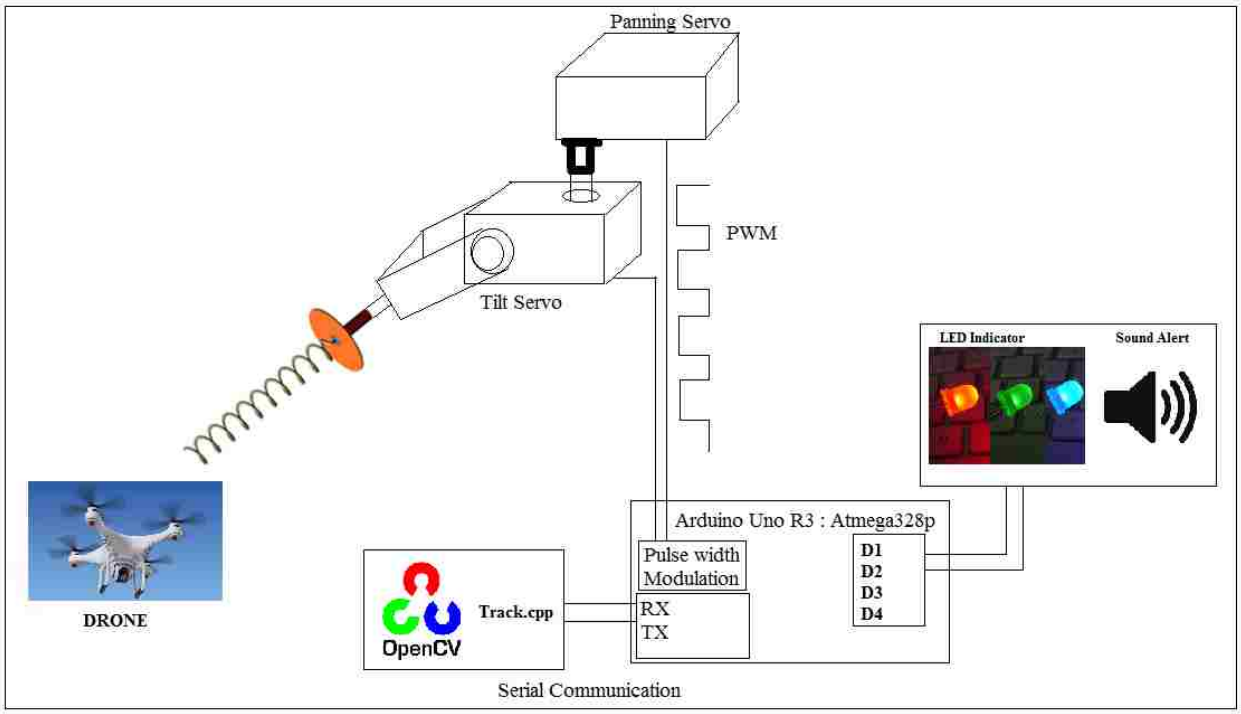

Figure 33: Hardware architecture and Pin Mapping

OpenCV library is linked to C++ file, both integrated together utilize the data from camera to extract required features and send control signals to motors. There are different modes through which sensors or data can communicate to Arduino.

- I2C (Inter-Integrated Circuit) [44]
- SPI (Serial Peripheral Interface) [45]

These two modes of communications are advanced enough for connecting multiple sensors with unique addresses. For primary motion detection an ultrasonic range finder is used which will activate the camera and further motion tracking mechanisms. This will save data overhead for the processing of unwanted frames and will also consume less power.

There are stackable modules or sensors like GPS, IMU (Accelerometer, Gyro and Magnetometer) which were used to develop a tracking antenna that keeps pointing towards a drone from the start of the flight. Below are the bus architectures for I2C and SPI which show the number of data pins that are required for minimal communication.

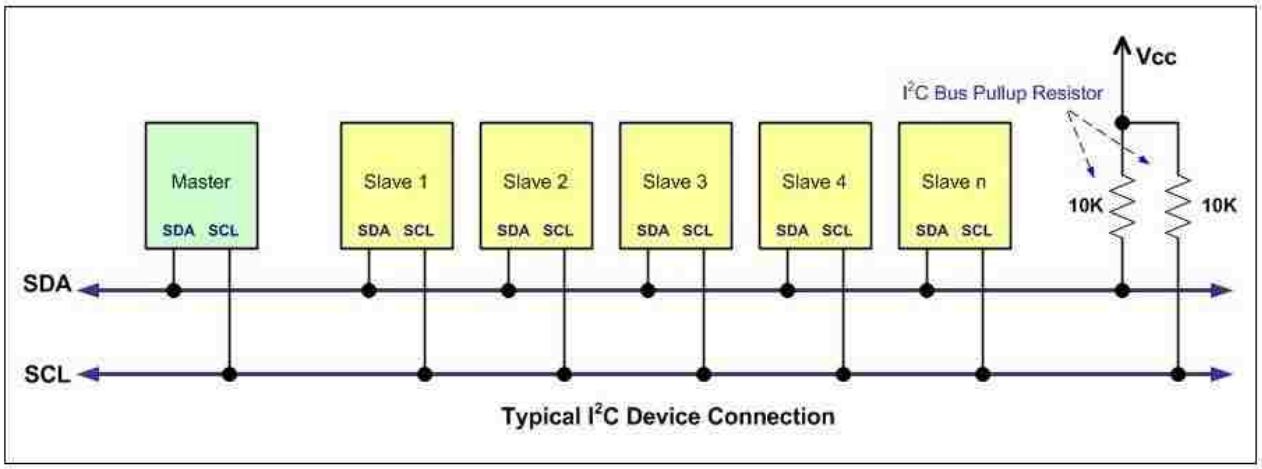

Figure 34: I2C Bus architecture [46]

In the above figure SDA denotes Serial Data and SCL denoted Serial Clock. This bus architecture and design was invented by Philips. This makes it easy to communicate with multiple sensors or modules by using only two digital pins on any controller. Every slave device on the bus has its own unique address

and the master will communicate using this. In our prototype the master role will be taken by the Arduino and GPS/IMU will act as slaves.

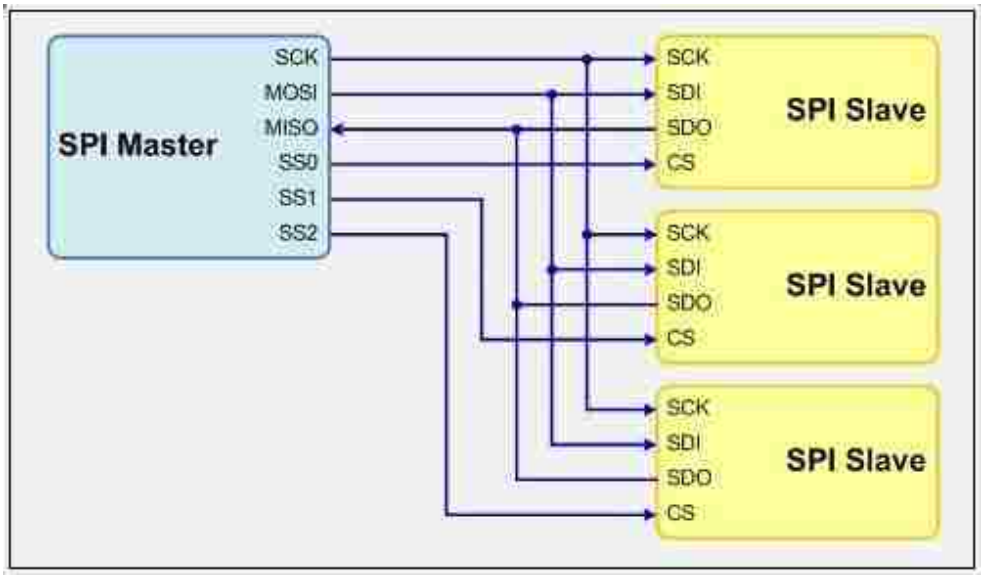

Figure 35: SPI Bus architecture [47]

SPI abbreviates as Serial Peripheral Interface, and it requires more number of pins when compared to I2C. The respective pins in the above figure are SCK (Serial Clock), SDI (Slave Data In), SDO (Slave Data Out), and CS (Chip Select). On the Master side we have MISO (Master In Slave Out), MOSI (Master Out Slave In), SS (0,1,2..n) is for Slave Select. [48] The major advantage of I2C over SPI is in the speed of communication. I2C can support up-to 1mbps where else SPI can support 25mbps. This difference is mainly due to the bus structure. IN I2C the communication is done through address, every slave whose address doesn't match is skipped and matched with other slave on the bus. This process consumes time and hence reducing the overall throughput. SPI talks directly to its slave modules by selecting their slave using Slave Select (SS). [49] This is a chip to chip direct connection and is really fast. For less data communication with lower number of slaves, I2C is preferred else SPI is used.

CHAPTER 6

IMPLEMENTATION OF DETECTION AND TRACKING A DRONE

The implementation is divided into two divisions where the control is handed off. The primary division is the software application which is responsible for identifying and detecting the drones. This software application is an executable that is generated in C++. OpenCV libraries are used for faster processing of frames and utilize the existing algorithms for better results. As mentioned in previous chapters there are four algorithms which would yield effective detection and tracking results. Motion detection using background subtraction is really effective with a static background and along a moving subject in the foreground. Color based tracking works efficiently when the target drone is of a specific color. SURF is the key algorithm in detecting the make and model of the drone so that exactly the required counter measure can be applied. Combination of two of these algorithms in parallel for every single frame will give perfect results.

6.1 Software Application for Detecting and Tracking

Much before developing an application in C++, the required version of OpenCV has to be linked with the program. First an empty project is created and then the library files and include files are mapped to respective directories. All the required additional dependencies are added in the linker section. After performing the steps to link the libraries to our project files, we can test our very basic code to open and stream video from the attached camera.

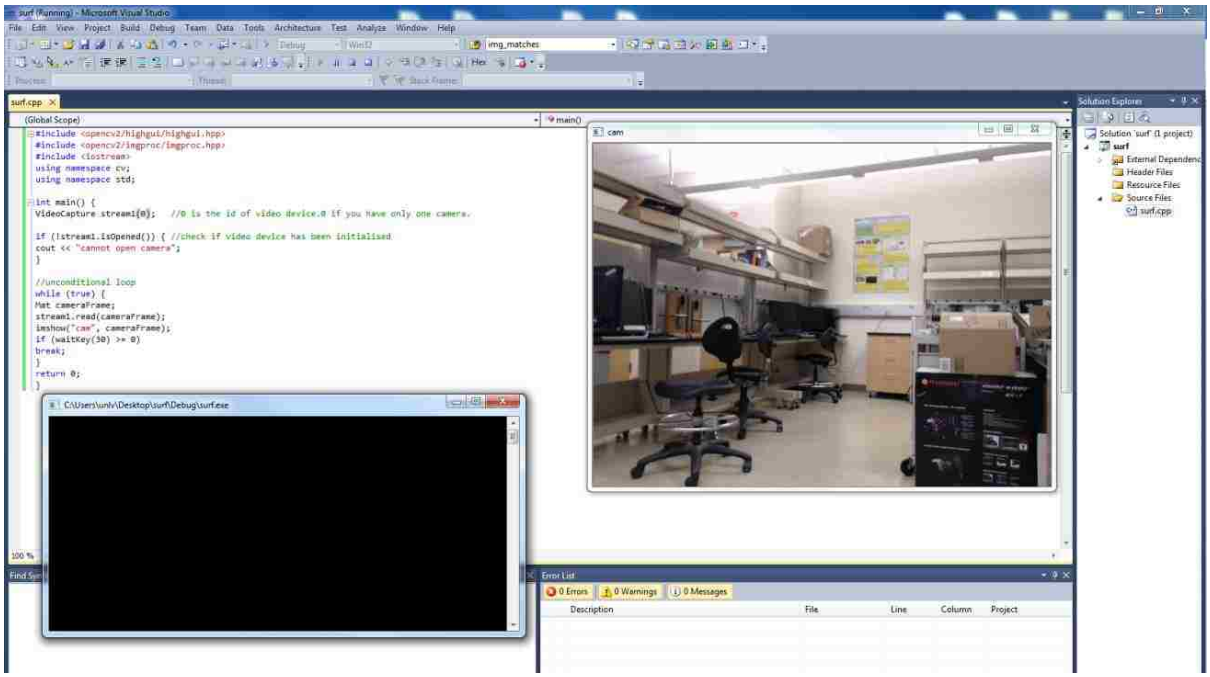

Figure 36: Streaming video from camera

There are many functions which are inbuilt in OpenCV which are used to perform activities like image format conversion, capturing video-stream, creating windows etc. The primary data structure used to hold images is “Mat”, where images are stored in the format of a matrix [50]. Making any changes to entire image becomes very easy and quick. It just is merely a matrix operation and the pixel values can be altered.

The main algorithms which we will be combining are SURF (Speed up Robust Features) and Motion Detection using absolute difference between two consequent frames. Motion detection will run on lesser load of data and activate the SURF algorithm when a drone or motion is detected. This will reduce the runtime overhead for the prototype. We will see the main functions and process of SURF and motion detection algorithms.

```
#include <opencv2/highgui/highgui.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <iostream>
using namespace cv;
using namespace std;

int main() {
VideoCapture stream1(1);   //0 is the id of video device. "0" if you have only one camera.

if (!stream1.isOpened()) { //check if video device has been initialised
cout << "cannot open camera";
}

//unconditional loop
while (true) {
Mat cameraFrame; // create a matrix to hold image data
stream1.read(cameraFrame);//read one frame at a time
imshow("cam", cameraFrame); //creates a window and displays the image
if (waitKey(30) >= 0)//exit the program if escape key is pressed.
break;
}
return 0;
}
```

Figure 37: Bare minimum code to stream video form a camera

6.1.1 Functions of SURF (Speed Up Robust Feature)

SURF algorithm has various methods which together help in extracting the features of an image. The main functions are located in the header file “features2d.hpp” [51]. This file has to be included in the include section. The extracted key-points and descriptors have to be stored in a special format of vectors and matrices. Hence we declare vector of Key-points and matrix of descriptors.

```
//vector of keypoints
vector< cv::KeyPoint > keypointsO; //keypoints for object
vector< cv::KeyPoint > keypointsS; //keypoints for scene

//Descriptor matrices
Mat descriptors_object, descriptors_scene;
```

Figure 38: Declaring Vectors key-points and Matrix Descriptors

Next step is to declare a SURF object which will extract the key-points from the given image of frame from the camera video. Then the descriptors are calculated and stored in memory. When we declare a SURF object we need to provide the minimum hessian value. The smaller this value is the more key-points the program would be able to find with the cost of performance. Depending on the specification of object to be found we fix this value.

```
2   SurfFeatureDetector surf(1500);
3   surf.detect(sceneMat,keypointsS);
4   surf.detect(objectMat,keypointsO);
```

Figure 39: Creating SURF Object and detecting key-points

The number "1500" passed as parameter to the function is the hessian value. Hessian matrix describes the second derivatives of a function, which stand for curvatures. The determinant of a hessian matrix is threshold here and the threshold value in the above example is set as "1500". We calculate determinant of each hessian matrix patch in the image and then threshold it to find robust feature points. If the min-threshold for hessian is increased then we get fewer amounts of feature points and upon decrement of the same value we get more feature points. The important property of a feature is repeatability or the tendency of re-detection of same feature within the image or other image. If the threshold value is set to very low then we will get many feature points but they will be weak features and have less repeatability. If the threshold is overdone then in this case we won't have enough features to describe the image.

The next step after detecting the key-points will be to calculate the descriptors. Now we have to perform the actual comparison or object detection. Much before that a matcher has to be selected and amongst the available "Flann Based Matcher" is fastest. Below are the steps for calculating descriptors and declaring a Flann based matcher.

```
2   SurfDescriptorExtractor extractor;
3   extractor.compute( sceneMat, keypointsS, descriptors_scene );
4   extractor.compute( objectMat, keypointsO, descriptors_object );
5   //Declaring flann based matcher
6   FlannBasedMatcher matcher;
```

Figure 40: Computing descriptors and declaring a Flann based matcher

Nearest Neighbor Matching is used which is in built in the OpenCV library. In the image below we have two entities namely blue squares and red triangles. Now if a new entity has to be added to one of the existing entity, this process is known as classification.

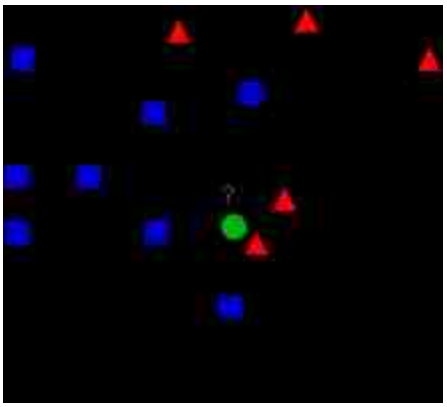

Figure 41: Example for KNN

The next step will be to check the nearest neighbor for a given new entity. From the image it is clear that red triangle is the closer one. If there are many blue entities around the green dot then it becomes difficult. So just checking the nearest is not sufficient, instead some n nearest entities have to be checked. In the image below let us take n=3, then we have two red and one blue in this case the new entity should be added to red triangle. If n=4 then in this case we have equal distant neighbors and it is a tie, hence usually n is an odd number. Below is the function set for KNN [52].

```
2   vector< vector< DMatch >  > matches;
3   matcher.knnMatch( descriptors_object, descriptors_scene, matches, 2 ); // find the 2 nearest neighbors
```

Figure 42: Function to find 2 nearest neighbors

Now after matching we need to remove or ignore the invalid or unwanted results. These false matches needed to be filtered out of all the available good matches. This is achieved by using nearest neighbor distance ratio.

```
3   vector< DMatch > good_matches;
4   good_matches.reserve(matches.size());
5
6   for (size_t i = 0; i < matches.size(); ++i)
7   {
8       if (matches[i].size() < 2)
9                   continue;
10
11      const DMatch &m1 = matches[i][0];
12      const DMatch &m2 = matches[i][1];
13
14      if(m1.distance <= nndrRatio * m2.distance)
15      good_matches.push_back(m1);
16  }
```

Figure 43: Filtering good matches out based on distance ratio

Now it is up to the user to set a threshold of good matches, to decide whether the desired object is found or further images are needed to identify. So once we have good matches and the features, we can get the coordinates of the object in the video frame, which will allow us to track the position approximately.

6.1.2 Motion Detection

This is the primary algorithm that runs on every frame of the video input. The basic concept is to identify the difference between two consecutive frames and amplify the difference between them. Essentially the amplified difference is the object in motion. This technique works for all objects that occupy good amount of pixels per frame in a video. This mechanism runs primarily on our implementation setup. Whenever valid motion is detected, the SURF algorithm takes command over the frames to process them and identify the drone. The algorithms are split into two in order to reduce the data overhead. SURF requires good amount of computing power even for frames with small resolution. A convenient frame rate of 15-20Fps is achieved with SURF at the resolution of 640x480. Any resolution less than this would be obviously faster and also yield less features and key-points which reduce the overall identification rate.

The process of this algorithm is explained in sequential way along with code snippets. After declaring required matrices and variables to hold images we have two functions in the application, one main function and other processing function which does the motion detection part. First step would be to initialize the serial communication between the micro-controller and C++ application. After the serial communication is initialized, matrices are declared to hold images for two consecutive frames from the video feed, difference image and threshold image [53]. Then an object is created for Video-Capture and the function to open the default camera is invoked.

```
77      comm.startDevice("COM5", 9600);
78      Mat frame1,frame2;
79      Mat grayImage1,grayImage2;
80      Mat differenceImage;
81      Mat thresholdImage;
82      VideoCapture capture;
83      capture.open(0);
```

Figure 44: Initial steps in the main function

The first line has two parameters of which the first one is “COM5”. This number indicates the ID assigned to the serial port to which the micro-controller is connected. The second parameter “9600” is the baud rate or the speed of communication in terms of bits per second. The last function is to open the camera which has ID as “0”. The default camera of a system is considered as ID “0”, any new add-on camera is assigned with incremented values of ID’s.

```
while(1)
{                                   //while starts
        capture.read(frame1);
        capture.read(frame2);
        cv::cvtColor(frame1,grayImage1,COLOR_BGR2GRAY);
        cv::cvtColor(frame2,grayImage2,COLOR_BGR2GRAY);
        cv::absdiff(grayImage1,grayImage2,differenceImage);
        cv::threshold(differenceImage,thresholdImage,SENSITIVITY_VALUE,255,THRESH_BINARY);
        cv::blur(thresholdImage,thresholdImage,cv::Size(BLUR_SIZE,BLUR_SIZE));
        cv::threshold(thresholdImage,thresholdImage,SENSITIVITY_VALUE,255,THRESH_BINARY);
        searchForMovement(thresholdImage,frame1);
        imshow("Frame1",frame1);
```

Figure 45: Converting two sequenced frames for further processing

Above image shows the sequential operations that are performed for motion detection. First two consequent frames are captured from the live camera. Each of the images is stored in two matrices frame1 and frame2. Comparing and performing operations on the images becomes easy and fast when they are available in grayscale format. Hence both the sequential frames are converted to grayscale. Now the absdiff() function is performed frame1 with respect to frame2. This gives the data of pixels whose values have changed in the time duration of two captures. The next step is to amplify the difference and highlight the pixels which change. Blur filter is applied and image is passed through a threshold. The final image is sent to the function "searchForMovement" as argument along with the first reference image i.e. frame1.

The search for movement function does further processing by finding out largest contours in the threshold image and then getting the coordinates of the contour which essentially are the coordinates of the moving object. The coordinates are sent through the serial port one after the other in sequence to Arduino.

6.2 HARDWARE IMPLEMENTATION

The software application transfers the control to hardware tracking equipment at the serial port. The hardware equipment consists of a micro-controller based development board (Arduino), pan and tilt servo motor mechanism and status indicators (LED & Sound Alert). The microcontroller used is Atmega328p and it is responsible for receiving the data over serial and control the motors based on that data.

Servo library for Arduino is used to generate required PWM pulses. Both the servos and the entire prototype are powered by a 5V 1A power supply. The camera is connected to the system through USB. The coordinates form the C++ application range from (0,0) to (640,480). Here the numbers 640 and 480 is the resolution of the frame captured by the camera.

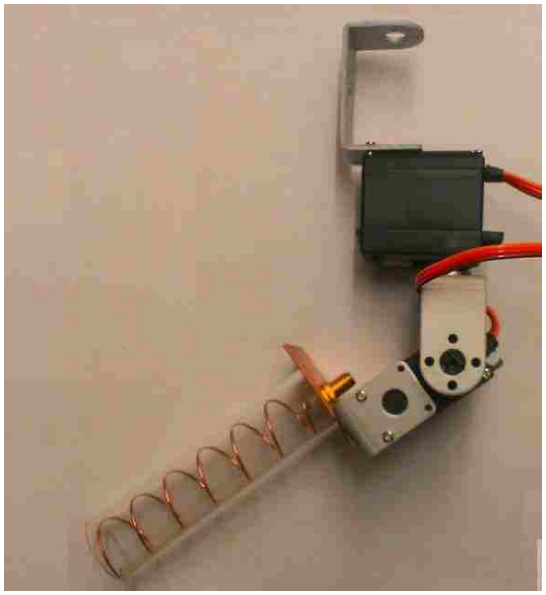

Figure 46: Pan and Tilt servo mechanism with a helical antenna to disable the drone

Each servo motor can go up to 180 degrees on its axis. Using this feature the coordinates of the object in motion are mapped to the angle of the servo motor after calibration.

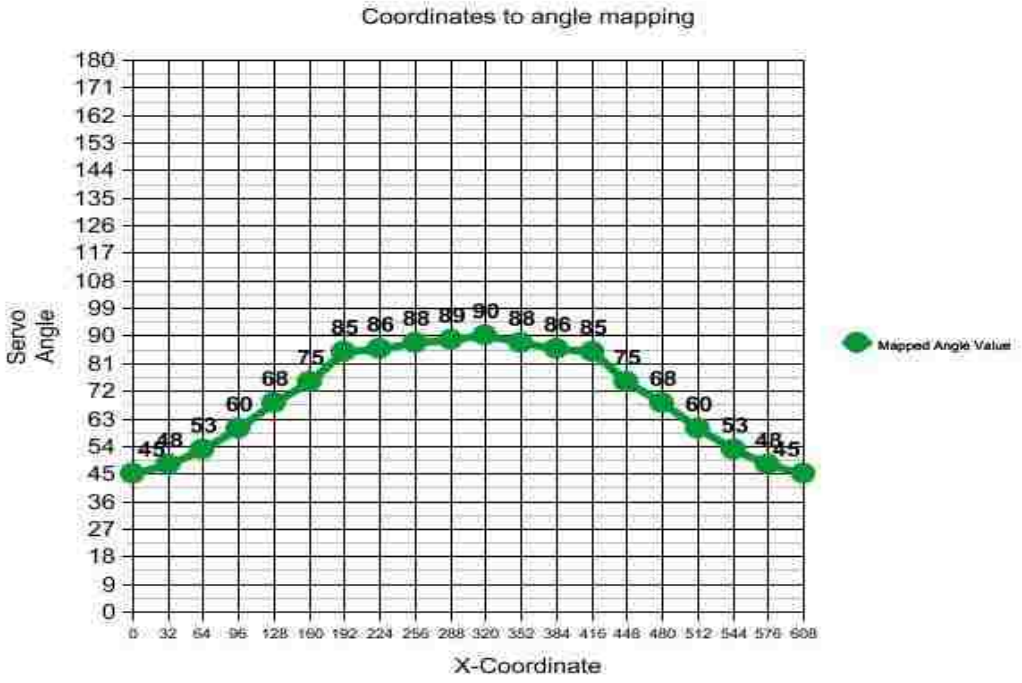

Figure 47: Graph representing servo angle and respective X coordinate

Depending on the camera used and the distance of the moving object the graph varies. When the moving object is at the center of the frame, then the horizontal servo points exactly at 90 degrees and as the object shifts towards left or right, the rate of decrease in the angle is different and is not constant. This is due to the curved lens in cameras which does not produce images that are linear in terms of distance and ratios. [54] Tilt shifting lens can produce images that are perfectly straight and are in linear ratio. Belo is the graph for Y-Axis.

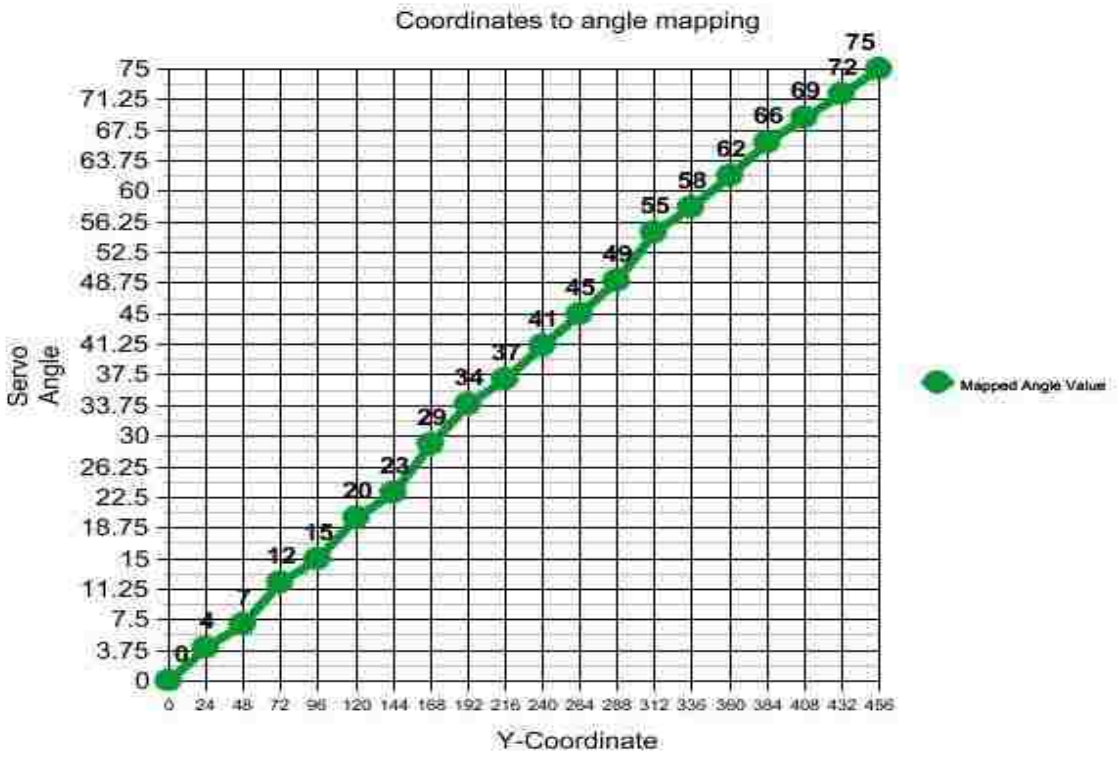

Figure 48: Graph representing servo angle and respective Y coordinate

The graph for Y-Axis is almost linear because the vertical servo is mounted parallel and calibrated to the starting of the Y-Axis. This gives us the maximum angle required to reach the top of the frame as 75 degrees. These graphs are calculated with respect to the image located at 5-10 meters away from the camera.

The Serial buffer is filled with values of X and Y coordinates sent by the C++ application. These values are then parsed and sorted into respective vertical and horizontal components. After the sorting process is done, mapping function is applied to the values and the output can be found on the graphs shown above. These angles are converted to pulses of length in few micro-seconds which control the shaft of the servo motor precisely.

There are two status indicators connected to Arduino on digital pins. One of the indicators is an RGB LED and other is a sound alert.

CHAPTER 7

PERFORMANCE ANALYSIS AND LIMITATIONS

7.1 Performance Analysis

Overall performance and limitations will be analyzed in this chapter. There are multiple parameters that affect the overall reliability of this design.

- Distance of the drone from the camera
- Speed of the drone
- Ambient light available
- Shutter speed of the camera
- Type of motor used and its precision

Each of these parameters is compared with rate of detection and the listed parameters themselves. Improvements in the hardware and processing section will overcome the limitations.

7.1.1 Distance of the drone from the camera

This is an important factor in terms of detection when the process solely relies on the images. The drone to be detected needs to occupy considerable amount of pixel values in order to be identified. The distance of drone to camera is in-directly proportional to the number of pixels occupied by the drone decreases. Below is a graph that shows the number of effective pixels [55] drone occupies vs distance in meters. A Parrot drone has been used to carry on all the experiments and the results are based on lab experiments and few real-time tests.

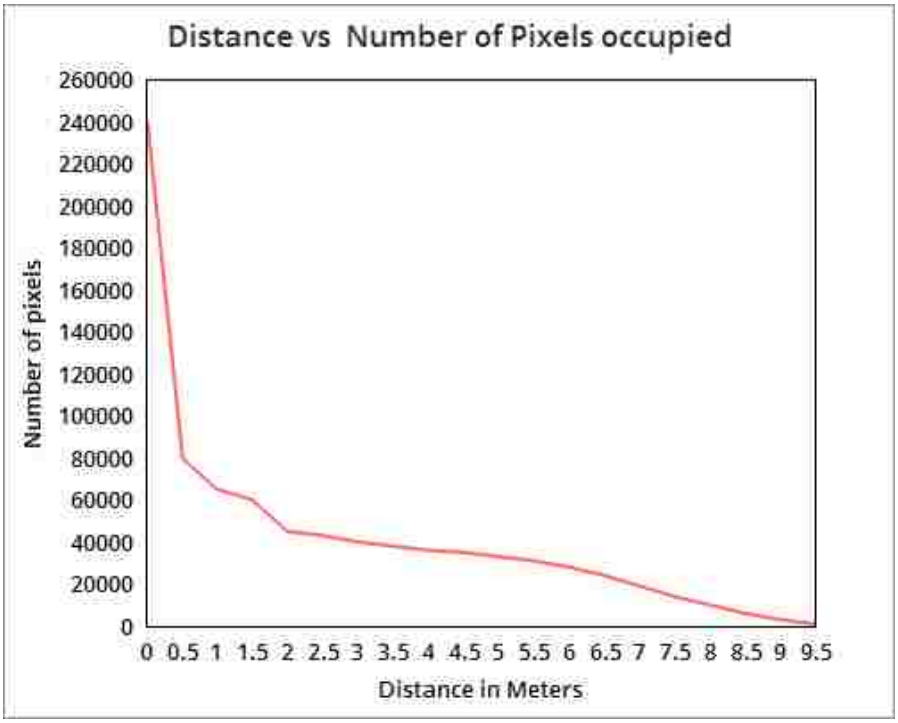

Figure 49: Distance vs Number of pixels graph

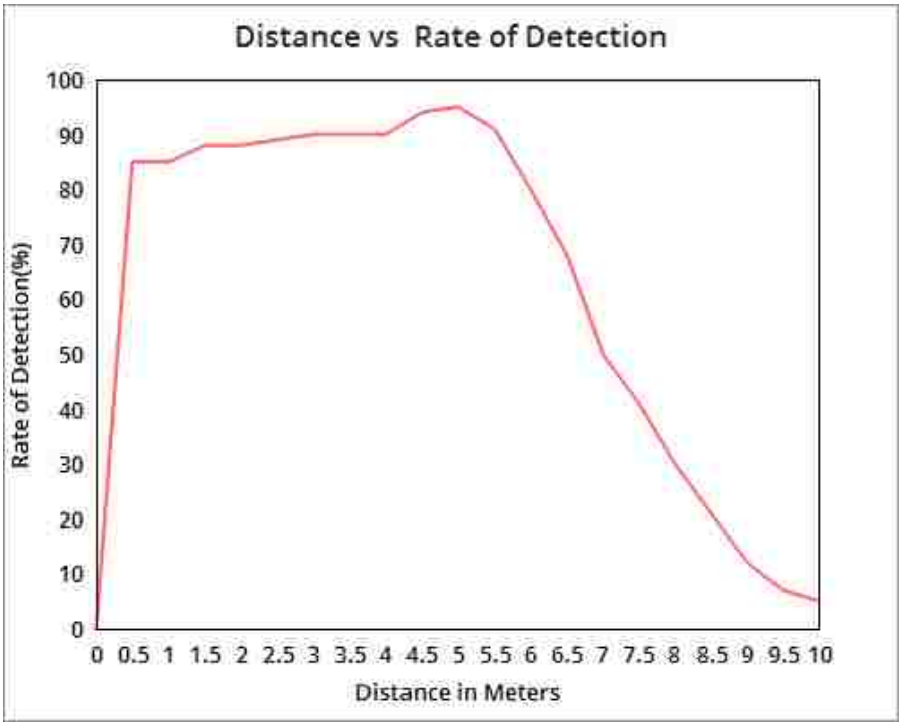

Figure 50: Distance vs Rate of Detection

This gives us the optimal distance at which the drone can be detected is from 0.5m to 5m. After this distance the size begins to shrink and anything before 0.5m will occupy most of the frame and make it difficult to identify.

7.1.2 Speed of the Drone

This is crucial and is linked with the frame rate of the camera. [56] If the camera is operating and processing the frames at 30-60fps, this is typically good to identify a drone flying across fast. With this statistics we can calculate the maximum speed at which drone can be identified with a given frame rate and the distance of drone from camera. A drone travelling at high speed at far away distance will appear moving slow in the frame and a drone moving at normal speeds but close to camera will appear moving fast across the frame. Below graphs show the performance of the process.

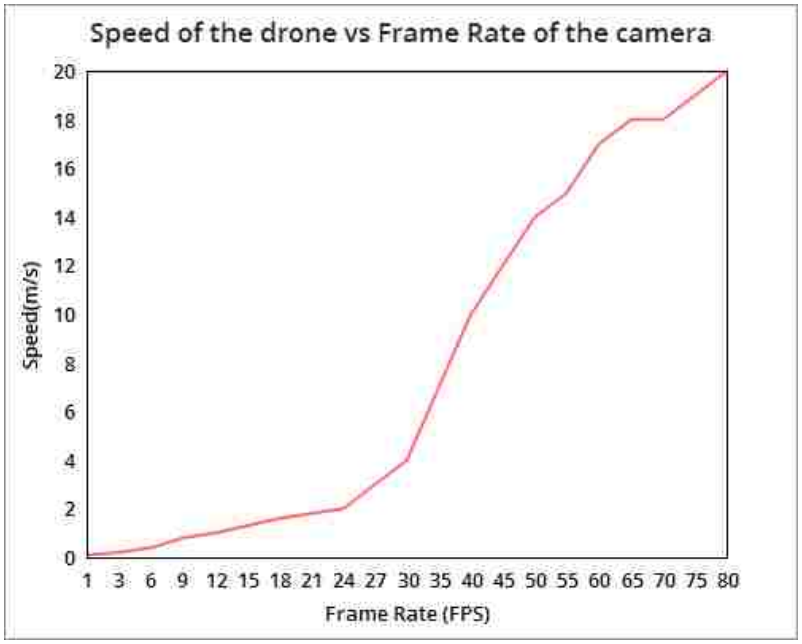

Figure 51: Speed of the drone vs Required Frame rate

Above Graph shows us the optimum required frame rate at which the frames are processed for motion detection vs the speed of the drone in meters per second. High end processors are used to capture a

video at 60fps and consume lots of memory. So effectively the real-time frame rate will be way lower than 60fps, somewhere around 24fps-30fps. This frame rate is good enough to process each frame for detection. So the maximum speed at which this prototype can detect some motion would be at 6m/s or 18km/h. This again is not a fixed value and is dependent on the distance of drone from the camera. If this distance is good enough as much as 5.5m - 8m (From Fig 55) then the speed of the drone can go up-to 70km/h.

7.1.3 Ambient Light available

This is another important factor when considering detection of drones in low light. If the drones have some extra lighting like indicator lights or visible LED's then it is detectable in pitch dark conditions too. In other case where the drone has no on-board lights, then it is not possible to detect it during night. This limitation can be overcome by using a night vision camera or a thermal camera with ultrasonic range finders. Below is a graph that shows rate of detection and timings during an entire day.

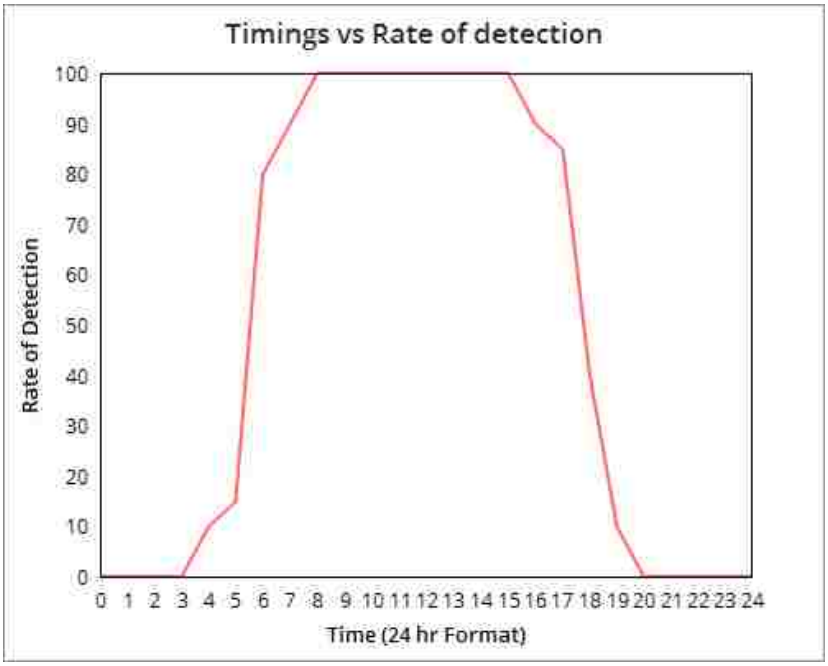

Figure 52: Time vs Rate of Detection (Assuming no external source of light)

7.1.4 Shutter Speed of the camera

This parameter is useful for SURF (Speed up Robust Features) algorithm. SURF is dependent on shutter speed and it efficiency is determined with distance along with shutter speed of a camera. In the below image we have 4 photographs taken at different shutter speeds. We can observe the amount of details available in each case.

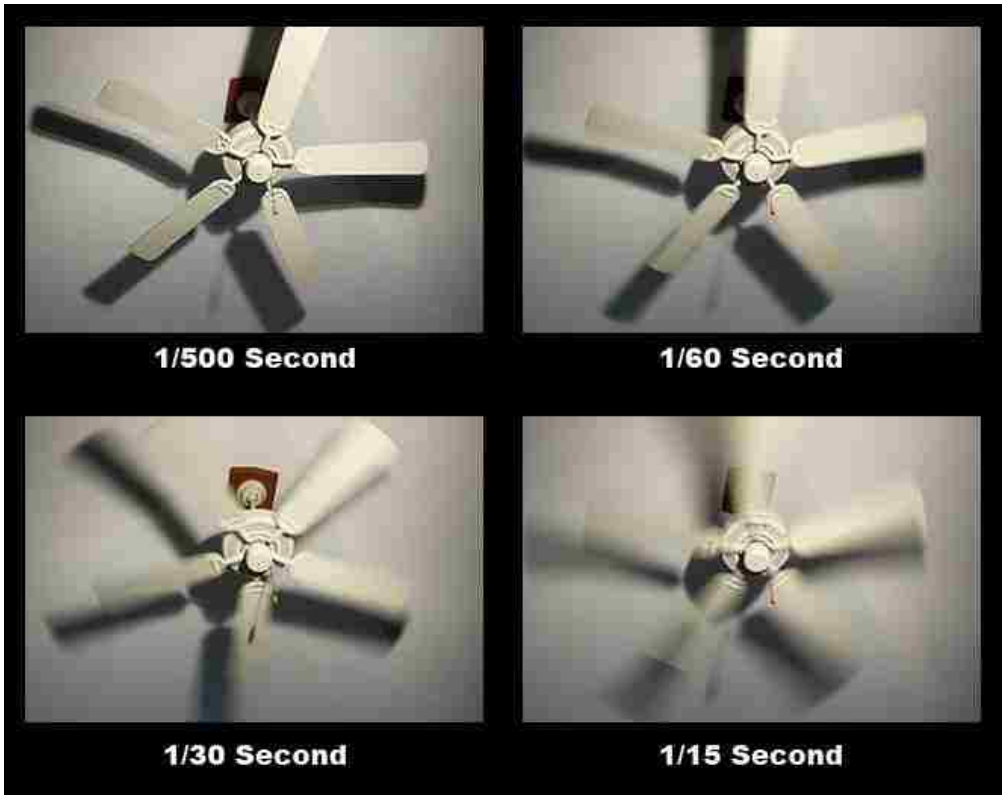

Figure 53: Shutter speed and details of image

From the above image we can infer that at lower shutter speed the object in motion will be blurred as the object (Drone in our case) makes large displacement within the capture time. Higher shutter speeds allow us to freeze the motion of the object and capture it perfectly without any distortion or deforming of image. In the above image $1/500^{th}$ of a second shutter produces crisp image of the rotating fan blades, which is optimum for identifying and extracting the features for SURF.

7.1.5 Type of Motor used to Track

There are many motors available to control the tracking head for the UAV. The high precision motors that are used for controlled motion are of two types:

7.1.5.1 Servo Motor

These motors have a precision of 1 degree. They are again classified into two categories of half-rotational servos and continuous rotation servos. In the prototype we have used two half-rotational servo motors. [57] These servos have shafts which can be positioned at anywhere from 0 degrees – 180 degrees with a precision of 1 degree minimum. They are controlled through pulse width modulation. Any resolution less than 1 degree cannot be achieved until unless gearing is used. Also the angle of view is limited to 180 degrees. Below image shows the drones which pass behind the area of coverage are un-detectable.

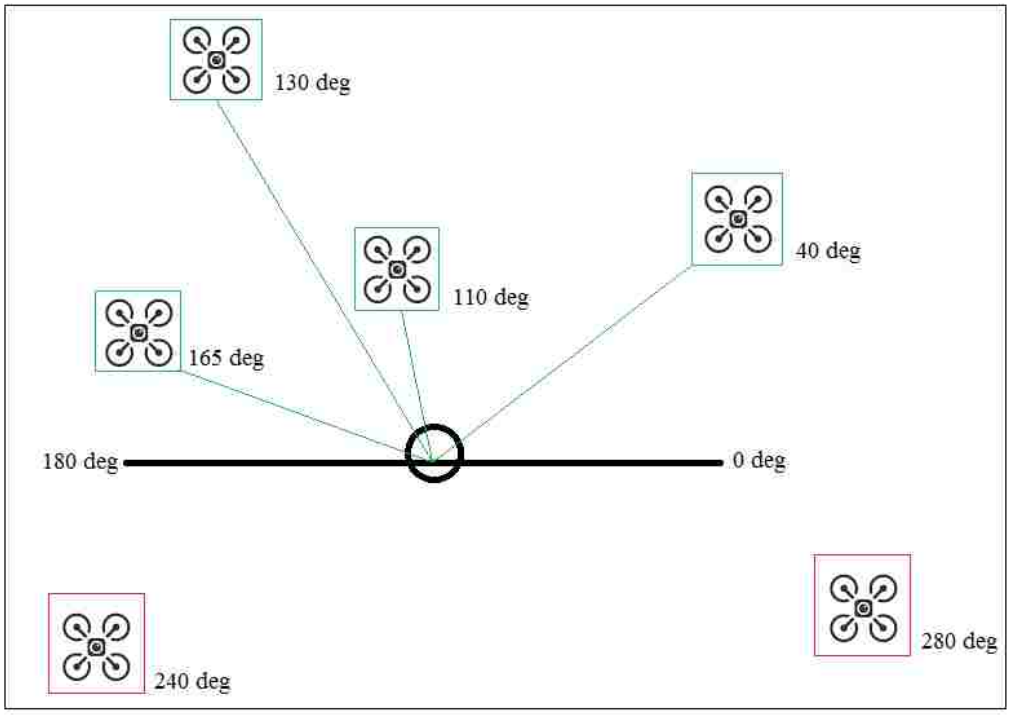

Figure 54: Detectable areas for servo based tracking

The precision of 1 degree is good for normal movements and simulation, but the rate of error increases as the distance of the drone from tracking equipment increases. Below image simplifies the statement.

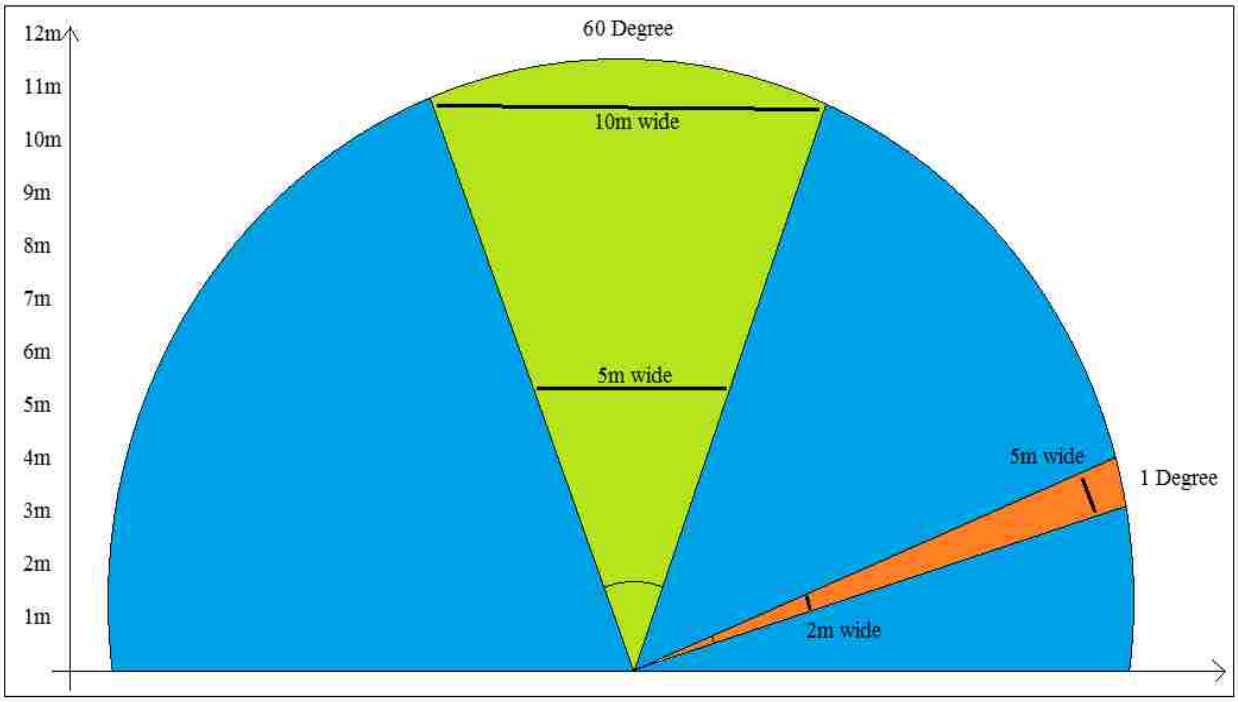

Figure 55: Decreasing precision with increase in distance

The above figure explains why a precision of 1 degree is not sufficient over long distances. Considering the green colored section, if the drone is at a distance less than 2m from the camera, then the servo motor can precisely aim at the target with few cm of tolerance. In the same green area if the drone is present at a distance of 12m, then the precision of target drops drastically. Hence at a larger distance deflection of 1 degree will cover a length of few hundreds of meters, and at a shorter distance only a meter approximately. [58] Usage of gears with very low ratio as 500:1 or 250:1 along with the shaft will increase the precision drastically with the existing servo motors.

7.1.5.2 Stepper Motor

This type of motor comes with different levels of precision. A stepper motor is similar to a servo, but it is continuous rotation motor. Its precision is not limited like that of a servo to 1 degree maximum. Stepper motors rotate their shafts in terms of small steps, and these motors come in different number of steps. The more steps we have, the higher precision can be achieved. Stepper motor along with low ratio gearing can enable super-high precision in tracking.

The only limit is the speed of the motors itself, as there is a minimum time at which a servo or a stepper motor can move from position X to position Y. This can't be changed as this is solely dependent on the electrical characteristics of that motor (Voltage and Current).

7.2 Limitations

This design has various limitations depending on the algorithms used and the above mentioned parameters. A simple motion detection algorithm like this would consider any moving object in the air as a drone. But assuming that the surveillance is done with sky as the static background, even though few buildings can occupy the lower part of a frame, at most the objects in motion could be birds and commercial aircrafts other than drones. This narrows down the segregation process to 3 entities namely birds, drones and commercial airlines. Commercial airlines fly at an altitude of 35,000ft and will not be in the threshold limit. Birds are only the entities which will generate false positives in the process.

The case of birds can also be eliminated by training the flight patterns of drones and ignore any irregular pattern of flying objects other than that of a drone.

CHAPTER 8

FUTURE WORK AND CONCLUSION

8.1 Conclusion

With the advent of new technology and radios getting cheaper, it will be really difficult to control drone traffic or malicious drones. The existing solutions can manage the issue to certain extent but are not reliable enough. The proposed method detects and tracks the trajectory of the drones, and a high resolution camera captures one image during the flight on the drone to perform SURF and identify the brand of the drone. This will help in applying right frequencies for jamming depending on the make and specifications of the drone, which we can get through SURF analysis.

8.2 Future Work

The future work of this design/prototype would be to train the application with variety of drone images and flight patterns so that the rate of false identification reduces. This design is to detect and keep track of the drones that pass nearby. Counter measures could be developed by any vendors and should be made mechanically compatible to mount of the tracking equipment. This is a minimalistic design for residential and commercial areas in future where the chance of malicious drones hovering around is high. Embedding artificial intelligence and machine learning to this design will save lots of power and computational overhead.

BIBLIOGRAPHY

[1] D.C. Lannicca, D.P. Young, S.K. Thadhani, G.A. Winter, *Security risk assessment process for UAS in NAS CNPC architecture*, Integrated Communications, Navigation and Surveillance Conference, 2013.

[2] M. Andreetto, M. Pacher, D. Fontanelli, D. Macii, *A cooperative monitoring technique using visually served drones*, Environmental, Energy and structural monitoring systems workshop, 2015.

[3] K. Boudjit, C. Larbes, *Detection and implementation autonomous target tracking with a quad-rotor AR.Drone,* Informatics in Control, Automation and Robotics, 2015 12th International Conference.

[4] J. Mezei, V. Fiaska, A. Molnar, *Drone Sound Detection*, Computational Intelligence and Informatics 2015 IEEE International Symposium.

[5] R. Bart, A. Vykovsk, *Any Object Tracking and following by a Flying Drone*, 2015 Fourteenth Mexican International Conference on Artificial Intelligence.

[6] www.dronedetector.com/portfolio/drone-detection-what-works-and-what-doesnt

[7] M. S. Faughnan, B. J. Hourican, G. C. MacDonald, M. Srivastava, J. P. A. Wright, Y. Y. Haimes, E. Andrijcic, Z. Guo, J. C. White, *Risk analysis of Unmanned Aerial Vehicle hijacking and methods of its detection*, Systems and Information Engineering Design Symposium (SIEDS), 2013 IEEE.

[8] P. T. Jardine, S. Givigi, A. Noureldin, *Incorporating feedback predictions for optimized UAV attack mission planning*, Control and Automation (MED), 2015 23th Mediterranean Conference.

[9] I. C. Price, G. B. Lamont, *GA Directed Self-Organized Search and Attack UAV Swarms*, Simulation Conference, 2006. WSC06.

[10] www.techtimes.com, Personal Drones prevented helicopters from putting out a California wildfire, by Lauren Keating, 2015.

[11] www.droneshield.com

[12] www.drone-detector.com

[13] www.dedrone.com

[14] Evan Ackerman, Rapere: *An Intercept Drone to Seek and Destroy Other Drones*, 2015 IEEE Spectrum.

[15] www.battelle.org/our-work/national-security/tactical-systems/battelle-dronedefender

[16] B. Kelley, G. De La Torre Parra, D. Akopian, *Cognitive interference avoidance in 4th generation GPS*, System of Systems Engineering Conference (SoSE), 2015.

[17] S.M. Giray, *Anatomy of unmanned aerial vehicle hijacking with signal spoofing*, Recent Advances in Space Technologies (RAST), 2013 6th International Conference.

[18] Department of Transportation Report Estimates 250,000 Drones in US Airspace by 2035, public-intelligence, 2013.

[19] M. Gharibi, R. Boutaba, S. Waslander, *Internet of Drones*, IEEE Access, 2016, Issue: 99.

[20] J. Yang, F. Sakai, Y. Aoki, M. Yorozu, Y. Okada, A. Endo, *Generation of a short pulse electron beam by using a photocathode RF gun and a picosecond pulse laser*, Lasers and Electro-Optics, 1999.

[21] www.opencv.org

[22] D. Gornea, D. Popescu, G. Stamatescu, R. Fratila, *Mono-camera robotic system for tracking moving objects*, Industrial Electronics and Applications (ICIEA), 2014 IEEE 9th Conference.

[23] A. Pal, G. Schaefer, M. E. Celebi, *Robust codebook-based video background subtraction*, Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference.

[24] www.wikipedia.org/wiki/Background_subtraction

[25] M. A. Memon, P. Angelov, H. Ahmed, *An Approach to Real-time Color-based Object Tracking, Evolving Fuzzy Systems*, 2006 International Symposium.

[26] A. Mirkazemi, S. E. Alavi, G. Akbarizadeh, *Fast image segmentation based on adaptive histogram threshold*, Artificial Intelligence and Signal Processing (AISP), 2015 International Symposium.

[27] Baofeng Zhang, Yingkui Jiao, Zhijun Ma, Yongchen Li, Junchao Zhu, *An efficient image matching method using Speed Up Robust Features*, Mechatronics and Automation (ICMA), 2014 IEEE International Conference.

[28] Jacob Toft Pedersen, *SURF: Feature detection and description*, Q4, 2011.

[29] Y. Sugiki, T. Yamaguchi, H. Harada, *Implementation of optical flow measurement system with an embedded processor*, Control, Automation and Systems (ICCAS), 2015 15th International Conference.

[30] T. Senst, J. Geistert, I. Keller, T. Sikora, *Robust local optical flow estimation using bilinear equations for sparse motion estimation*, Image Processing (ICIP), 2013 20th IEEE International Conference.

[31] U. Bhagat, B. Shirinzadeh, L. Clark, Yanding Qin, Yanling Tian, Dawei Zhang, *Experimental Investigation of Robust Motion Tracking Control for a 2-DOF Flexure-Based Mechanism*, IEEE/ASME Transactions on Mechatronics, 2014.

[32] Si Wu, Zhen Ren, *Video Stabilization by Multi-Trajectory Mapping and Smoothing*, Information, Communication and Signal Processing, 2005 Fifth International Conference.

[33] A. A. Morye, C. Ding, B. Song, A. Roy-Chowdhury, J. A. Farrell, *Optimized imaging and target tracking within a distributed camera network*, American Control Conference (ACC), 2011.

[34] www.arduino.cc

[35] www.active-robots.com/lynx-b-pan-and-tilt-kit-no-servos

[36] M. H. Siddiqi, I. Ahmad, S. B. Sulaiman, *Weed Recognition Based on Erosion and Dilation Segmentation Algorithm*, Education Technology and Computer, 2009 ICETC-09 International Conference.

[37] Yu Hong-li, Cao Xin-Yan, *Serial communication control of frequency conversion speed control for motor*, Computer Application and System Modeling (ICCASM), 2010 International Conference.

[38] Ryan Winters, Jameco Electronics, and Article: Build your own circuit.

[39] K. Shinohara, E. Sakasegawa, *A new PWM method with suppressed neutral point potential variation of a three level inverter for AC servo motor drive*, Power Electronics and Drive Systems, 1999 Proceedings of the IEEE 1999 International Conference.

[40] www.sweb.cityu.edu.hk

[41] L. J. Liu, J. Zhou, G. Y. Xiu, J. Gui, *Design of Point-to-Point Communication of Wireless Sensor Network*, Computational and Information Sciences (ICCIS), 2011 International Conference.

[42] www.opencv.org, OpenCV API Reference guide.

[43] Xianghua Fan, Fuyou Zhang, Haixia Wang, Xiao Lu, *The system of face detection based on OpenCV*, Control and Decision Conference (CCDC), 2012.

[44] http://www.i2c-bus.org/i2c-bus/

[45] Overview and use of the PICmicro Serial Peripheral Interface, Microchip.

[46] www.emicro.com I2C bus architecture.

[47] www.corelis.com, SPI Interface Bus Architecture.

[48] K. Mikhaylov, J. Tervonen, *Evaluation of Power Efficiency for Digital Serial Interfaces of Microcontrollers*, New Technologies, Mobility and Security (NTMS), 2012 5th International Conference.

[49] Y. Gui, *1-Wire search algorithm and its application*, Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference.

[50] C. Meng, X. Zhang, *Video Encryption Based on OpenCV*, Database Technology and Applications(DBTA), 2010.

[51] B. Besbes, A. Apatean, A. Rogozan, A. Bensrhair, *Combining SURF-based local and global features for road obstacle recognition in far infrared images*, Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference.

[52] R. Min, D. A. Stanley, Z. Yuan, A. Bonner, Z. Zhang, *A Deep Non-linear Feature Mapping for Large-Margin kNN Classification*, Data Mining, 2009 ICDM Ninth IEEE International Conference.

[53] L. Liu, L. Zhao, *Moving target detection algorithm combined background compensation with optical flow*, Guidance, Navigation and Control Conference (CGNCC), 2014.

[54] Mei-Ling Chen, Chia-Jung Chou, Min-Tzu Wu, Chen-Kuo Chiang, *Object-based tilt-shift photography*, Consumer Electronics - Taiwan (ICCE-TW), 2015 IEEE International Conference.

[55] E. S. Eid, *Study of limitations on pixel size of very high resolution image sensors*, Radio Science Conference, 2001.

[56] H. Toyoda, Y. Kobayashi, N. Mukohzaka, N. Yoshida, T. Hara, T. Ohno, *Frame-rate displacement measurement system utilizing an ultra-high-speed shutter camera and an optical correlator*, IEEE Transactions on Instrumentation and Measurement, 1995.

[57] Sho-Tsung Kao, Zong-Yu Yang, Ming-Tzu Ho, *Design and implementation of a color-based visual tracking control system*, Automatic Control Conference (CACS), 2013 CACS.

[58] Seok-Hee Han, Young-Hoon Kim, In-Joong Ha, *Iterative identification of state-dependent disturbance torque for high-precision velocity control of servo motors*, IEEE Transactions on Automatic Control, 1998.

CURRICULUM VITAE

Graduate College

University of Nevada, Las Vegas

Sai Ram Ganti

Degrees:

Bachelor of Technology in Computer Science, 2014

Jawaharlal Nehru technological University

Master of Science in Computer Science, 2016

University of Nevada, Las Vegas

Thesis Title: Security Threats from UAS and its mitigation through detection and tracking

Thesis examination Committee:

Chair Person, Dr. Yoohwan Kim, Ph.D.

Committee Member, Dr. Ajoy K. Datta, Ph.D.

Committee Member, Dr. Jisoo yang, Ph.D.

Graduate College Representative, Dr. Venkatesan Muthukumar, Ph.D.