DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

Cat flap control system using a Raspberry Pi

Third (Four if applicable) year project report is submitted in part fulfilment of the requirements of the degree of Bachelor (Master if applicable) of Engineering.

Table of Contents

Abstract

This report provides the hardware and software development of a cat flap control system controlled by the Raspberry Pi. The system is designed to prevent the cat with “unwanted guest” entering the cat’s owner house by using an object detector to recognise the “unwanted guest” and some electrical components. Raspberry Pi is a small embedded computer that can replace a bulky personal computer in detecting the presence of objects of interest and capable of controlling electrical components through its general purpose input output (GPIO) pins. Details of how the hardware and software developed at each stage will be given in the report, as well as theories behind the object detector will be discussed.

Chapter 1: Introduction

Section 1.1: An Introduction

Cats are one of the most common household pets. These wonderful being can bring to human beings the happiness and joy that we so long desire that other things can't give to us. What pets can differ from human beings is that from cats we can obtain the feeling of being received, attachment and affection like no other beings can give us as we spent more and more time with them.

As we care for them a lot like caring for our own family, we want to make sure we're making our cat living happily with the freedom they deserve. This freedom, like human beings, means that cats should be free to wander anywhere they want whenever they want to just like how human beings roam about. It is also eventually hard to lock a cat inside the house as cat themselves are a symbol of freedom and it is in their nature to explore, that is why we introduce cat flaps to them.

However, we do not want other people's cats entering your house. The solution to this problem is to create a smarter cat flap that can identify only our cat. But then, there is still a need for an even smarter cat flap because cats are playful and sneaky beings and like to sometimes bring things or "unwanted guest”, such as, a mouse inside the house that we don't want to. To prevent "unwanted guest" entering the house, we can use image processing software to identify if the cat is bringing in any "unwanted guest" and prevents the cat from entering the house.

Section 1.2: Background

A cat flap is a door designed for cats that allows them to enter or exit the house whenever the cat or the owner wants the cat to. A cat flap has a 2-way closing panel(entering and leaving), while a more expensive model offers 4-way locking(in only, out only, in and out, and locked)[1].

Presently, there are many types of cat flap available commercially. Ranging from cheap ones to ones that can be a bit costly. All features that every cat flap has the ability to allow cats to exit the house whenever they want or you want. Cheap cat flaps allow cats to enter the house whenever they want but can't lock the flap's door.

Figure 1.1: Illustration of a cat entering a cat flap[2]

If the owner is living in a neighbourhood of cats' owner then the owner might want to consider a cat flap that has higher security features such as a lock and cat identifier that scans the cat's infra-red, RFID tags or injected microchip[1]. The identifier reads the identification from your cat when it walks inward the cat flap. The cat flap then verifies if the cat is your's or not and thereby allows or denies access to the house for the entering cat.

Conducting a background research on similar cat flap development projects, there are a couple of interesting projects that can be seen online. Several projects that were found were quite similar to each other, some examples of the projects are: "A cat detector using Raspberry Pi to take a photo of a cat when it enters the house and send an email to the owner[3]" and "a cat flap that post a tweet when that enters the house[4]". It can be seen that these two projects are quite similar to each other but doesn't better security to the cat flap.

There is a project that is very similar to the one in this report. It is called "The Flo Control Project[5]". This project has the same goal to this report's project but uses a different method of recognising the mouse from the cat's mouth. "The Flo Control Project" uses a silhouette from the shadow of the cat face to perform image processing, which seems quite impractical to be just analyse “uninvited guest” from just a silhouette, in addition, to the fact that the project is well over 10 years old.

Section 1.3: Aim of the Project

The aim of this project is to take inspiration from "The Flo Control Project" mentioned in the previous section and developed the idea further to make it much more practical and easy to setup with the aid of today's technology. In order to achieve this task, we can replace a large desktop computer used by “The Flo Control Project” with a small embedded computer such as a *Raspberry Pi* to analyse if the cat is bringing in any “uninvited guest” into the house and control the cat flap to allow or deny access for the cat to enter the house automatically by a developed software.

Referring to Figure 1.2 on the next page, the figure illustrates an overview of the design of the cat flap control system that is aimed to be constructed. The process of the system will be implemented as a three-step process. First, when the cat is walking towards the cat flap, the camera take pictures of the cat and is send to analyse inside the written software to analyse. Secondly, the image processing software will find if there are any “uninvited guest” apart from the cat in the picture. Lastly, according to the result from the previous step, the software will then tell the lock to open or close so that the cat can or cannot enter the house.

Note that the figure does not represent the final system, but is shown to illustrate the idea and aim of how the whole system is going to be implemented.

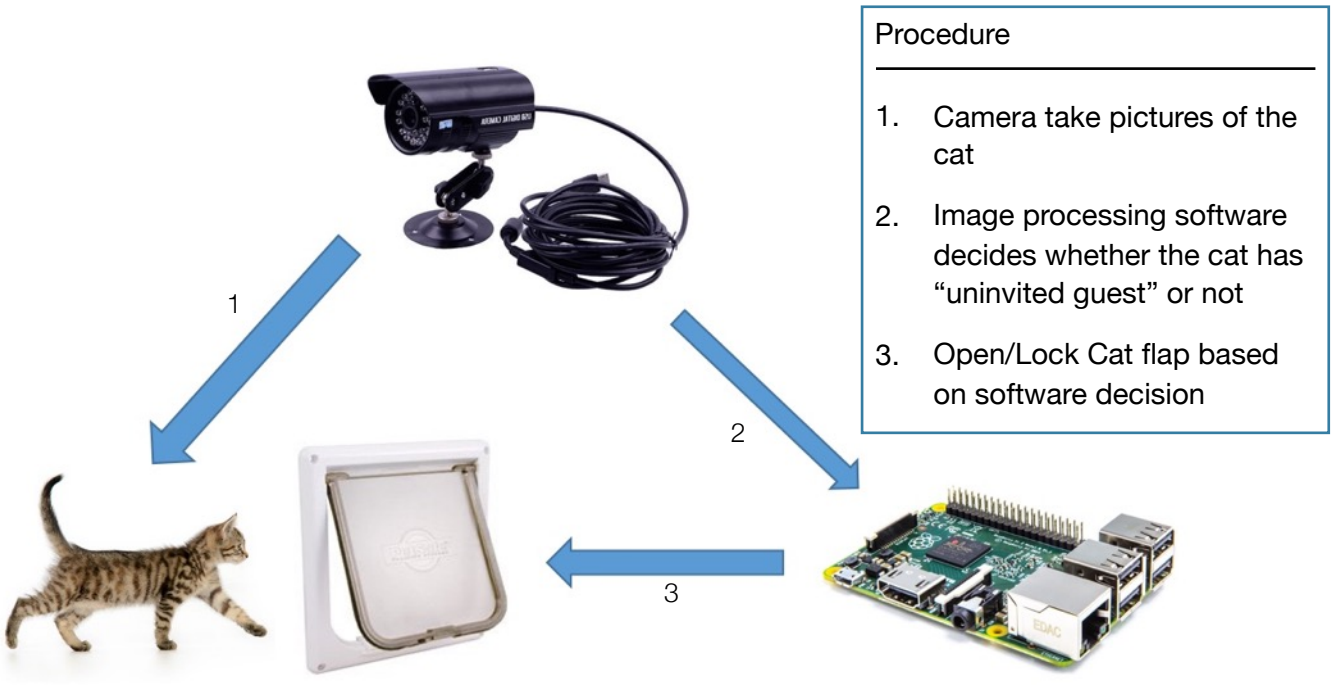

Figure 1.2: Diagram representation of the cat flap control system

Section 1.4: Objective of the Project

Looking further from the aim of the project, the objective of this project can be categorised into four major categories in chronological order:

Research

- Become familiar with Python with enough knowledge to complete project
- Become familiar with Command Line Interface and be able to use it efficiently
- Become familiar with image processing techniques
- Become familiar with open source image processing software(OpenCV and dlib)
- Understand and become familiar with Raspberry Pi and its General-purpose Input/Output (GPIO) pins

Hardware Design and Development

- Select a method of identifying owner's cat
- Select a suitable camera for the Raspberry Pi
- Design wooden plank to fit everything in placed together
- Design a suitable locking mechanism that will be controlled by Raspberry Pi
- Design a single circuit to implement the whole system efficiently including a lock mechanism, a motion detector, lighting and camera
- Design the circuit to prevent system losing power

Software Development

- Create several test programs to test electrical components(PIR sensor, LEDs, solenoid) being controlled by Raspberry Pi's GPIO pins
- Train datasets to recognise cat and mouse
- Develop image processing software to analyse image of the cat taken by the camera using Python programming Language
- Transfer software from Ubuntu to Raspberry Pi's Operating System, Raspbian
- Combine test program with image processing software to create a single program that analyse image and control all the electrical components

System Evaluation

- Evaluate image processing software without integrating it to the whole system
- Evaluate and test system's software and hardware as a whole
- Make sure the system has more than 75% detection accuracy rate when it is used to detect objects through live video streaming

Chapter 2: Literature Overview

Section 2.1: Image basics

Section 2.1.1: How an image is made

An image is basically a visual representation of an object. An object can be anything that human beings see with the eyes, such as, a person, a cat, a bottle or a mountain. An image is formed by building blocks of pixels. Depending on the quality of the image, the amount of pixels found in an image can ranges from low to very high. Therefore, if a very high quality image is needed, then the amount of pixels will be very high, but, if a low quality image is needed then the amount of pixels will be low too.

Usually, a pixel is considered to be the colour and intensity of light that appears on a given place inside the image. Most pixels are represented in two ways:

1. Grayscale

   In a grayscale image, each individual pixel ranges from 0 to 255. Where "0" makes the image black and "255" makes the image white. From figure 2.1, it can be seen that darker colours are closer to 0 and lighter colours are closer to 255.

Figure 2.1: Grayscale gradient demonstrating pixel value from black (0) to white (255). [6]

2. Colour

Colour pixels are normally represented in RGB colour space. There are three values for RGB colour space, [Red, Green, Blue] and is represented by an integer that ranges from 0 to 255 which indicates how much of the colour there is i.e. the intensity of each colour. Combining the three values together will obtain a colour.

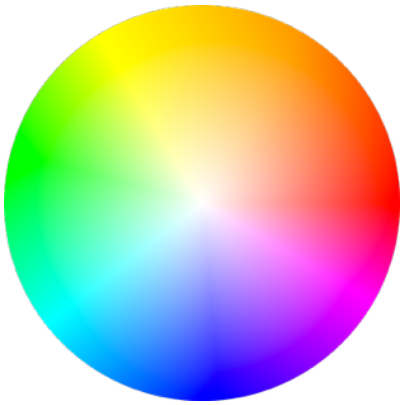

Figure 2.2: RGB Colour Wheel [7]

Some examples of standard colours in RGB([Red, Green, Blue]) spaces:

Black: [0, 0, 0]	White: [255, 255, 255]
Red: [255, 0, 0]	Blue: [0, 0, 255]
Green: [0, 255, 0]	Yellow: [255, 255, 0]
Pink: [255, 0, 255]	Purple: [128, 0, 128]

Although, other colour spaces are available, it will not be mentioned in the scope of this report

Section 2.1.2: Pixels

Pixels in an image can be considered as a small square inside a grid called an image. Lots and lots of pixels make up a single image, if an image is greatly enlarged, individual pixels can be seen rendered as small squares where each providing its own intensity[8].

The number of pixels in an image is called resolution. An image resolution can indicate how much detail is in an image. Referring to figure 2.3 in the next page, it can be seen that the higher the resolution, the more detail an image will have. Additionally, with higher resolution, it will be more difficult to see individual pixels, to see them, one would have to enlarge the image greatly.

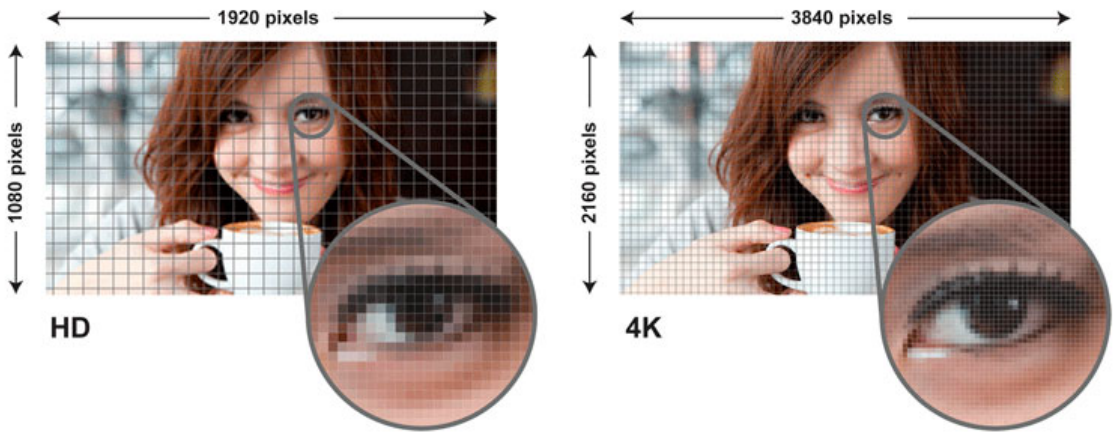

Figure 2.3: Image comparison between different resolutions to illustrate the point that higher resolution leads to more detail in an image[9]

An image with a width of *"W"* and a height of *"H"* will have a resolution of "*W x H" pixels.* Multiplying the width and height will give the total number of pixels in that image.

Some of the most common resolution used are:

- 800 x 600 pixels = 480,000 pixels
- 1280 x 720 pixels = 921,600 pixels
- 1920 x 1080 pixels = 2,073,600 pixels
- 3840 x 2160 pixels = 8,294,400 pixels

Pixels in an image can be accessed and modified using an open source image processing software called OpenCV[10]. OpenCV represents images as NumPy arrays, what this means is that pixels in an image are represented as the coordinate system starting from (0,0) because Python programming language is zero-indexed. Therefore, we can access a pixel value by just supplying the x and y coordinates of the pixel that we want to modify.

It is also important to point out that OpenCV stores RGB channels in the reverse order: BGR[Blue, Green, Red]. The reason is that in OpenCV each of the colour components are linked together in sub-columns to form the entire image which means it would be more efficient to reverse the order.

For more information on OpenCV, please refer to Chapter 4: Software Design and Appendix.

Section 2.2: Object Detector

Section 2.2.1: What is an object detector?

Object detection is a process of applying computer vision and machine learning algorithm to scan, detect and locate real-world object(s) from an image and labelling it from a set of given category[11]. The term object is quite a bit abstract, as objects can be anything that you can interact with in the real world. For instance, an object can be a person, a cat, a mouse, a ball, a glass of wine and the list goes on.

Object detection is widely used in everyday lives. Some of the most common applications of object detection are security, surveillance, vehicle parking system, and face detection. Although, these are the common applications, it can be used in a wide variety of ways as long as you want to detect objects.

There are several models that can be used to detect objects including[11]:

- Feature-based object detection
- Viola-Jones object detection [12] used for face detection
- Histograms of Oriented Gradients(HOG) features with Support Vector Machine(SVM) [13]
- Image segmentation and blob analysis

Object detection is one of the fundamental challenge of computer vision as objects in the real-world can vary greatly in appearance[14]. For example, a cup of coffee can have different appearances or non-rigid deformations, but as human beings, we can still recognise that this is a cup of coffee, but for the object detector it might be something entirely.

Figure 2.4: Varying shapes of a cup of coffee [15]

A robust object detector should be able to discriminate the presence of objects correctly even though the shape of the object is different and at difficult illumination.

Section 2.2.2: Image Pyramid and Sliding Window

A required step in creating a fully functional object detector is to scan an image at various locations and scales to search for the desired object no matter how big or small the object is.

This stage can be separated into two points: ***Image Pyramid*** and ***Sliding Window***.

Image Pyramid

An image pyramid (figure 2.5) is a multiple scale representation of an image. Using an image pyramid will help in finding objects in the image at different scales of an image. This is an important step of creating an object detector because in the real-world, objects comes in many sizes and can be at various distance from the viewer[16]. Therefore, analysing an image at a single scale may cause the detector to miss the object when the scale is not the same.

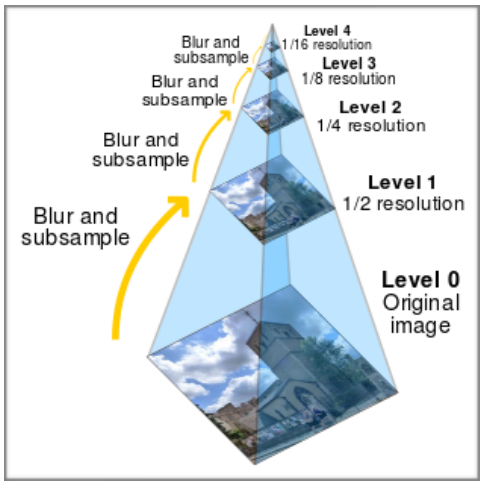

Figure 2.5: Construction of image pyramid by halving the image's size at each level[17]

At the bottom of the pyramid lies the original image in its original size, then as the level subsequently increases the image is resized by half, subsampled and blurred using Gaussian blurring until a minimum size has been reached.

Sliding Window

Sliding window is a technique of moving a rectangular region of fixed weight and height that slides across an image[18]. The rectangular region moves from the upper left corner of the image to the right and goes down to the bottom right. For each rectangular region, the object detector will extract features inside the rectangular region to see if there is our object of interest.

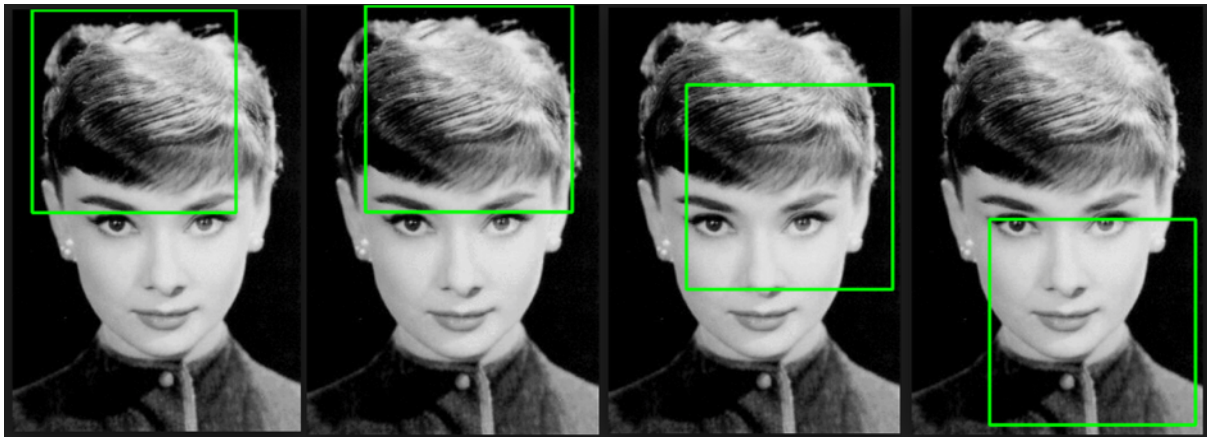

Figure 2.6: Example of sliding window technique[18]

Combining the sliding window technique with the image pyramid allows the object detector to recognise objects at varying locations and scales in an image.

Section 2.2.3: Histogram of Oriented Gradients (HOG)

One of the most commonly used model of the object detector is the Histogram of Oriented Gradients (HOG) combined with a Support Vector Machine as it has been proved to outperform existing edge and gradient based descriptor[13]. In this section, information on HOG will be given.

The HOG is a descriptor that is used in computer vision and machine learning to detect objects. It was improved from edge orientation histograms, SIFT descriptors and shape context but differs by computing on a dense overlapping grid to improve accuracy and can reduce false positive results[13].

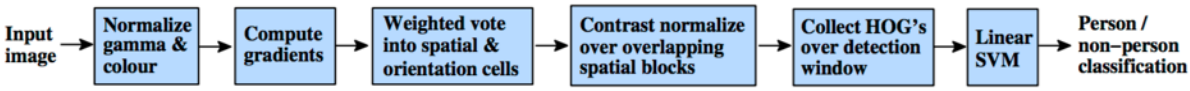

Figure 2.7: Overview of HOG process of extracting features [13]

The HOG descriptors is mainly used to describe the shape and appearance of objects in an image by "the distribution of local intensity gradients and edge directions [13]". Dalal and Triggs proposed in their paper that extracting features using HOG can be divided into 5 stages shown in figure 2.7 used to classify a person. The work from this paper can be applied to other objects as well. The result demonstrated from this paper shows that using HOG combine with Linear SVM produces a high accuracy object detector.

Section 2.2.4: Linear Support Vector Machine (Linear SVM)

In most object detection applications, HOG is usually used with Linear Support Vector Machine (Linear SVM) to detect objects [13],[14]. Referring to figure 2.7 we can see that the result of collecting all HOG over sliding window that forms the final feature vector[13] which is then used in the Linear SVM to become an object classifier.

A SVM is "a supervised learning model with learning algorithms that analyse data used for classification [19]". With this being said, the Linear SVM is in fact the detector that analyse data given from the HOG to find the object of interest in the image.

In a Linear SVM, there is one parameter that has to be carefully adjust or else it could make the object detector goes wrong. This parameter is called coefficient “C”. The parameter controls how strict the user wants the SVM to be. The larger the “C” value, the lesser the mistake it is allowed to make and fits the training data better[20]. Therefore, it can lead to a more accurate result of detecting an image. However, this parameter needs to be adjust carefully, if the parameter “C” is too high, it could lead to **over-fitting**. Over-fitting causes the model being unable to separate sample points.

A common practice when tuning the “C” parameter is to start from small values, for example, 0.1. Use this value to train the detector and then carefully increase the value of the parameter until achieving a satisfied result.

Section 2.2.5: Creating an object detector

In order to create a functional object detector, the following steps should be followed:

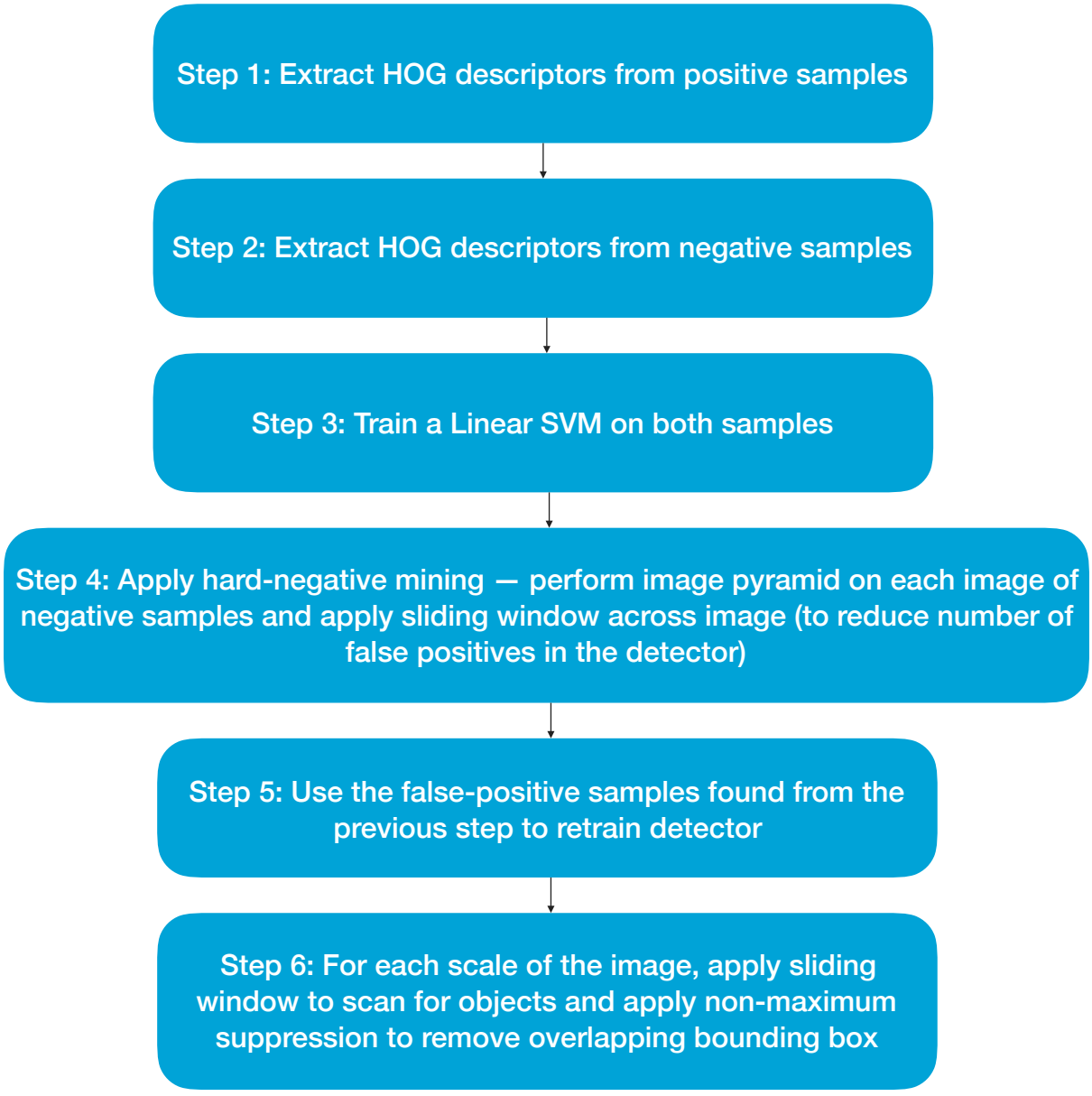

Figure 2.5: The process of constructing an object detector

However, instead of using the standard Linear SVM, a new method called Max-Margin Object Detector(MMOD) proposed by Davis E. King combined with HOG which is used for learning to detect objects in images [21] will be used for this project as it produces a much faster result when compared to the normal object detector outlined above. According to his paper [21] there are three changes made inside his object detector:

1. Instead of extracting features from both positive and negative samples, MMOD uses the training image to extract both the positive and negative samples from all other region in the image. This makes the MMOD method much faster when compared to others as it eliminates the need to provide negative samples.
2. Non-maxima suppression is applied during the actual training phase, which means that the samples would no longer need to undergo hard-negative mining.
3. Algorithms used in this method provide a much higher accuracy result than others state-of-the-art object detectors according to his paper.

Another reason for the adoption of this method is that looking at the time given to complete the project, having to carefully fine-tuned each parameters for each of the 6-step process would take too much time and would result in an unfinished project.

More details of the object detector used in this project will be provided in Chapter 4.

Section 2.3: Raspberry Pi

Raspberry Pi [23],[24] is a miniature sized computer that has a size of about a credit card that costs around £30. There are several models of the Raspberry Pi that has been released, the model used to complete the project is "Raspberry Pi 2 Model B". Raspberry Pi 2 features a quad core ARM CPU and a GPU that is powerful enough to watch high definition videos and play games on it.

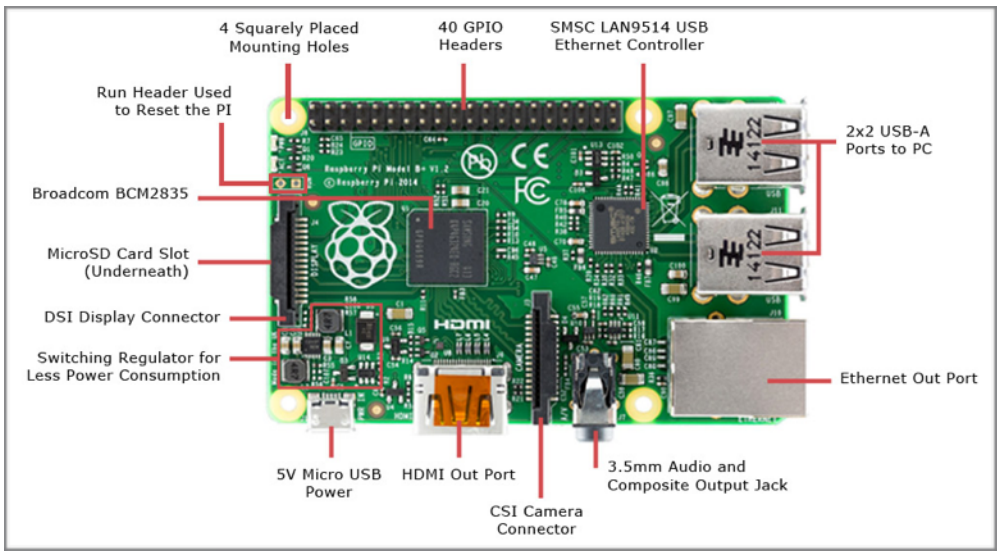

Figure 2.7: Raspberry Pi's Schematic[25]

The model features a built-in HDMI, four USB ports to connect peripherals, an Ethernet port, a camera interface to connect the Raspberry Pi's camera. Raspberry Pi uses a

Linux operating system called “Raspbian” which is the official operating system and the one used in this project, other operating systems are available. The operating system and user’s file are stored in a microSD card.

The Raspberry Pi 2 comes with 40 general purpose input output (GPIO) pins which allows the user to control electrical components such as a switch, an LED or a solenoid through the user’s written code.

With its functionality and low cost, Raspberry Pi is an ideal choice for the project because it has comes with a powerful processing unit that is able to perform object detection and has the ability to control electrical components using GPIO pins.

Section 2.4: Python programming language

The default programming language used by the Raspberry Pi is Python[24],[26]. Python is a high-level programming language that is widely used and easy to learn because the code and syntax of this programming language is very similar to the English language making it easier to understand when we compare it to other programming languages.

Due to its popularity, OpenCV which is written in C++ primarily also offer bindings for Python. Another reason for its popularity is that it is cross-platform: Windows, Mac, Linux and others.

This project will be associated with two platforms, Raspbian and Ubuntu. The reason is that the Raspberry Pi is quite slow when compared to a commercial used computer, so the object detector and all of the training dataset will be done on Ubuntu. After the object detector is obtained, it will be moved to the Raspberry Pi. Since both platform is capable of using Command Line Interface (CLI) to run Python codes, therefore, codes will be written on a text editor and tested using the operating system’s terminal.

To recreate this project, initial setup and several libraries installation is required on both operating systems. This project uses Python 2.7 instead of Python 3.0 as OpenCV doesn’t support Python 3.0 fully at the time of conducting the project. All installation can be done through the terminal of both operating systems, an installation guide will be provided in Appendix A and B.

Chapter 3: Hardware Design

This chapter will provide information on all the related hardware used to complete the project including the cat flap, circuit and the camera. As well as creating a practical setup for demonstration purposes.

Section 3.1: Choosing a suitable cat flap

In order to complete the project, the first thing to obtain is a cat flap. Currently, there are a wide variety of cat flaps to choose from that ranges from really cheap to ones that are a bit costly but provides more security to the user. A simple cat flap can provide access inside and outside of the house to cat. As the price increases with functionality, a 4-way lock mechanic is introduced up to a smart cat flap that can check your cat's identity by microchip or RFID.

(a) (b) (c)

Figure 3.1: (a) PetSafe StayWell 2-Way Pet Door, (b) PetSafe StayWell Deluxe 4-Way Locking Cat Flap, (c) Sureflap Microchip Cat Flap [27]

The initial choice of cat flap was to get the cheapest one available which was the one in figure 3.1a but then as the development of the project progresses and discussion with my supervisor, Dr. Steve Sharples, I wanted to create a full security system, which is not just detecting "unwanted guest" but also the ability to identify the owner's cat as well. Common methods of identifying are scanning RFID and microchip.

After much discussion, both of us agreed that having a method to identify the owner's cat would be a nice addition to the project. Fortunately, there is one cat flap that could scan both microchip and RFID, shown on figure 3.1c, it eliminates the time needed to construct an RFID scanner, moreover, microchips are impossible to purchase as these are injected directly into the cat. In addition, an RFID tag compatible with the cat flap was also purchased. It is more practical when the cat identifier when acts on a separate circuit from the designed circuit so that it doesn't interfere with each other and doesn't take the resource from the Raspberry Pi's CPU which was needed to perform object detection.

Section 3.2: Choosing a suitable camera

Another important requirement for this project is the camera. The Raspberry Pi was designed to be compatible with both its own camera[23], which can be connect through the CSI Camera connector mentioned in the last chapter and using a USB powered camera.

(a) (b)

Figure 3.2: (a) Raspberry Pi Camera Module, (b) USB Webcam [28]

The Raspberry Pi offers two camera modules at the normal camera module and a no infrared filter camera module. Both versions of the camera module comes with a clear 5 megapixels resolution with a fixed-focus lens and features a small compact size about a £2 coin. The camera is capable of taking videos at 1080p at 30 fps. The no infrared filter camera module is ideal for working under low light conditions, and produces a lower image quality

(a) (b)

Figure 3.3: Comparison of image taken from (a) normal Raspberry Pi camera module and (b) no infrared filter camera module [29]

in terms of colour contrast when compared to the normal camera module [29].

Conducting research on using a USB camera on the Raspberry Pi results in the fact that it is not as good as using the Raspberry Pi's camera module due to a major reason [30]. The Raspberry Pi's camera module utilises the Raspberry Pi GPU, which result in a low impact on its CPU. In contrast, using a USB webcam can make the CPU usage higher compared to Raspberry Pi's camera module.

Therefore, the normal Raspberry Pi camera module was chosen to achieve the task in this project as it doesn't have to work in low light conditions. Additionally, it would be better to give the CPU workload to the object detector rather than giving it to camera, so that the object detector can have all the CPU power it needs to detect objects in the hope that it will take less time to analyse the objects.

Section 3.3: Designing the practical setup

In order to place every components of the project together in a single place(electrical components, Raspberry Pi, cat flap and camera) a practical setup is quite necessary as well as for demonstration purposes too.

To imitate a real situation where cat flaps are normally installed, a wooden plank was immediately considered. Designs for the practical setup was created from a sketch with the assistance from my supervisor. The material used for this design was all made from wood, mainly to illustrate how the project could be setup so that everyone can recreate it easily. The sketch of the design can be seen in figure 3.4.

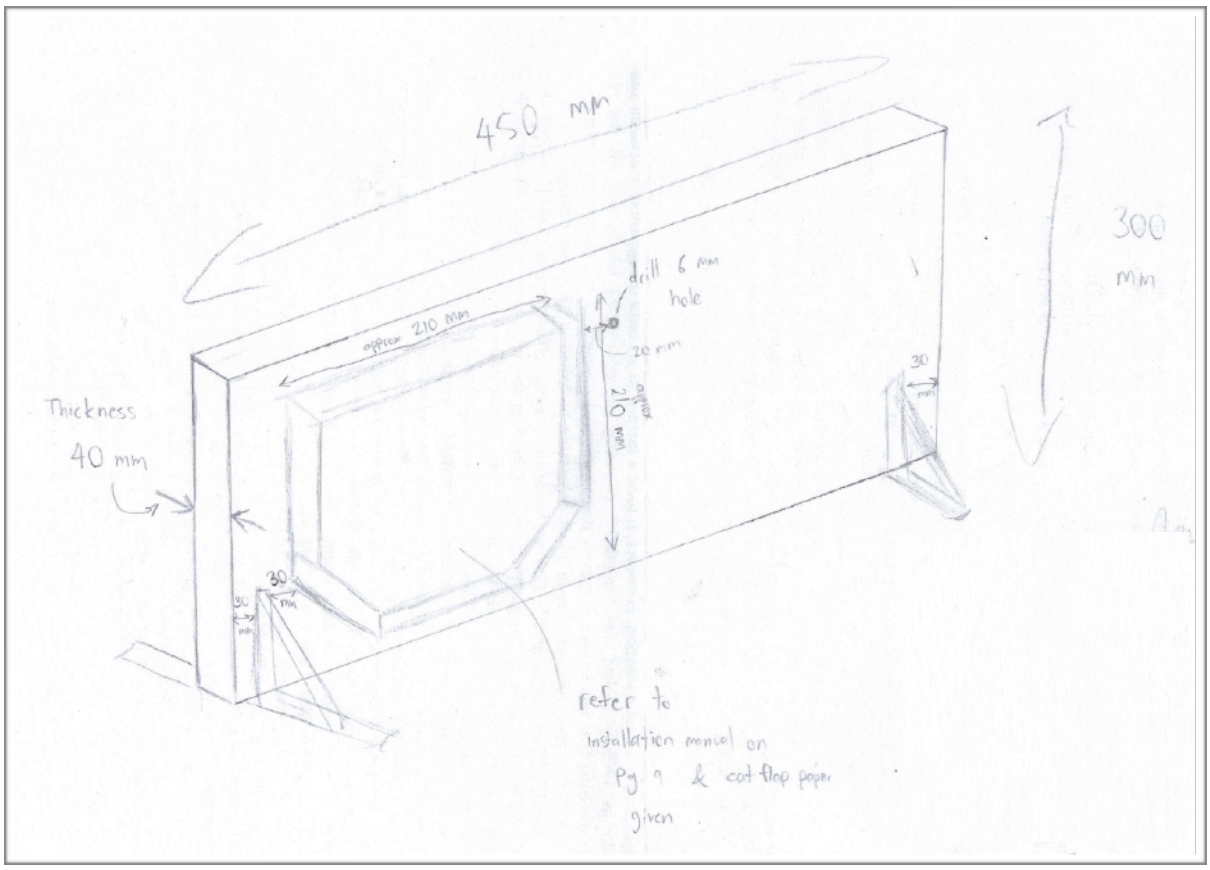

Figure 3.4: A sketch of the project's practical setup

The design was created to provide the right amount of space to place every hardware components on the plank of wood. Some additional holes were drill on the right of the cat flap to provide space for the wire to go through. With the aid of the university's technicians, James Hazzledine and Malcolm Dugdale from Architecture & Build Environment, the design was built into a real model.

The final practical setup can be seen in figure 3.5 and 3.6 below. Figure 3.5 illustrates the indoor-facing side and figure 3.6 illustrates the outdoor-facing side.

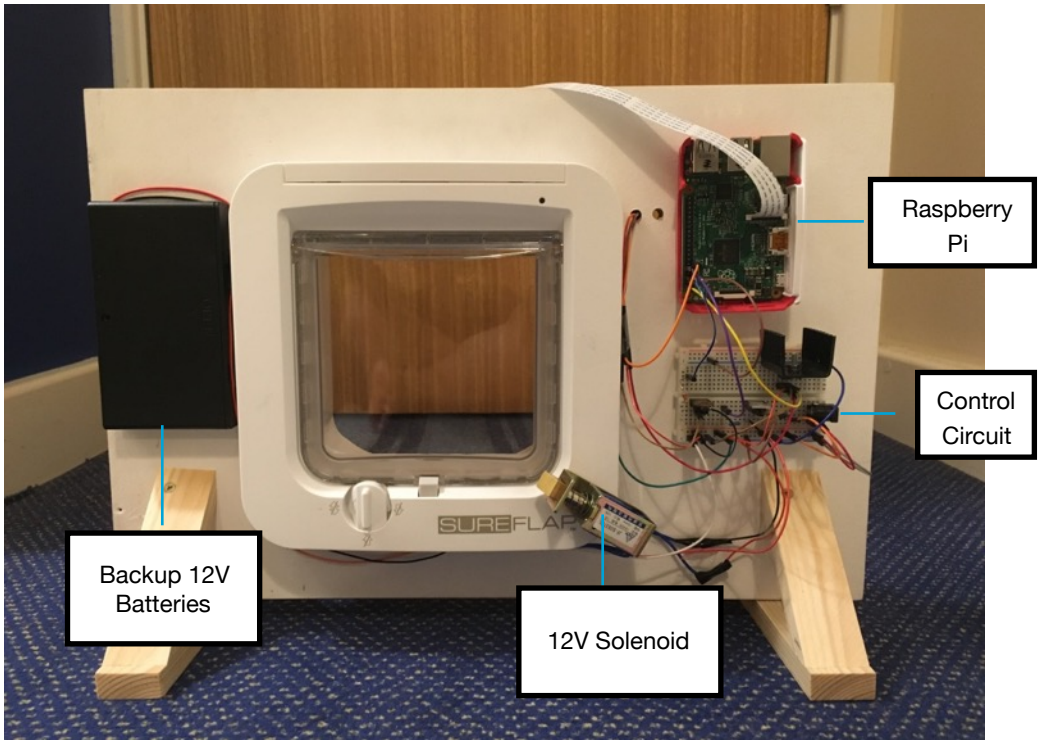

Figure 3.5: Indoor-facing side of the cat flap

From figure 3.5, all important electrical components are placed indoor to prevent damage coming from leaving it outdoor, while components left outside should still be cared for.

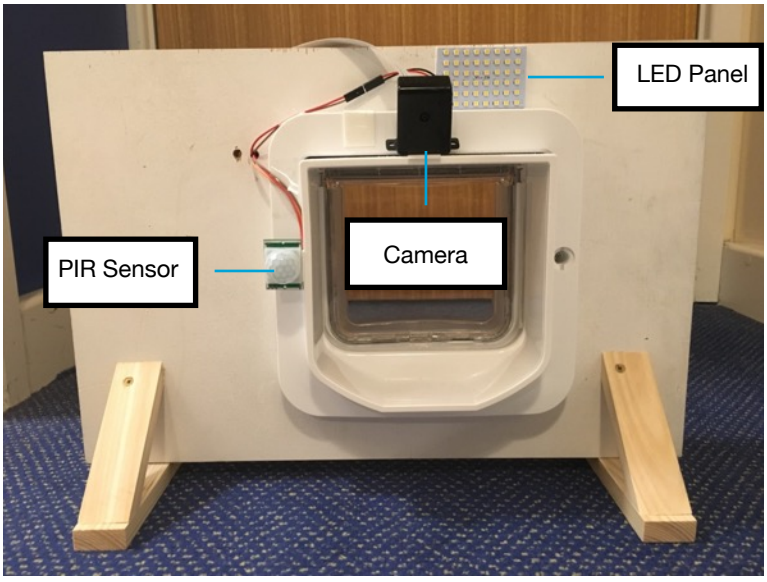

Figure 3.6: Outdoor-facing side of the cat flap

Section 3.4: Designing the circuit

One of the major components of the project is the circuit designed to control the cat flap. Not only does it controls the lock on the cat flap when told so by the software, it also controls when the software should start analysing the object that passes the camera. The circuit was designed to serve four purposes:

1. Activate the software to detect objects when the PIR sensor detects motion
2. To prevent low brightness situations and prevent a blind camera during night time, an LED Panel will always light up when the software has started
3. Unlock the solenoid when there is just only the cat entering the house
4. Switch to the backup batteries power source when there is power outage

The final design of the circuit is shown in figure 3.9 on the next page which is divided into four sections that served each of the four purposes above. The function of each electrical components will be shown in Table 3.1 on page 24.

The camera was not included in the schematic shown in figure 3.9 as the figure only show electrical components that are connected through the Raspberry Pi's GPIO pins. Please refer to Appendix C for more information on using the Raspberry Pi's camera module.

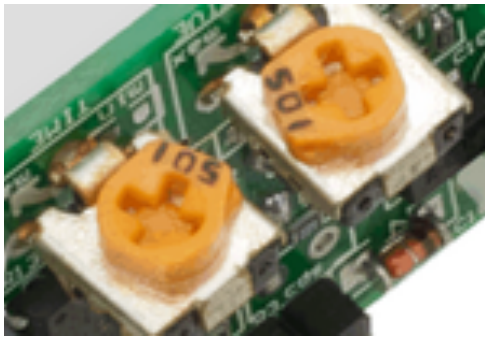

Figure 3.7: Adjustable sensitivity and delay of the PIR sensor with potentiometer [31]

In order to prevent the camera from finding the objects all the time, a Passive Infrared Sensor (PIR sensor) was added to the circuit. Using the PIR sensor allows the Raspberry Pi to reduce the workload on the CPU from the object detector software when nothing passes the camera.

The potentiometer, figure 3.7, is adjusted to have the maximum sensitivity and lowest delay time so that it can detect motion from quite a distant and instantly. The PIR sensor is able to detect motion within a maximum distant of 2.5 metre.

Another key problem for this project is that it has the risk of locking the cat out of the house when the power is out. To solve the problem, an additional 12 V battery-powered source was added to the circuit by connecting both positive terminal together. When the voltages at both terminal are equal to each other, there is no current flow, even though both of the power sources are turn on. Therefore, when one power source is out, the other power source can instantaneously replace it. A solution to the problem is needed due to the fact that the solenoid's position when the power is off is the "lock" position, in other words, the cat can't come in because of the solenoid.

The original power source for the Raspberry Pi is its 5 V microUSB power plug. At the current state the project now have 2 power plugs and a battery-powered source. This makes the project less practical as there are too many power sources used in a single designed system, the decision to merge the two power plugs into one is necessary. Looking at the GPIO pins on figure 3.8, we can pass 5 V to GPIO pin number 2 to supply power to the Raspberry Pi instead of the 5 V microUSB plug.

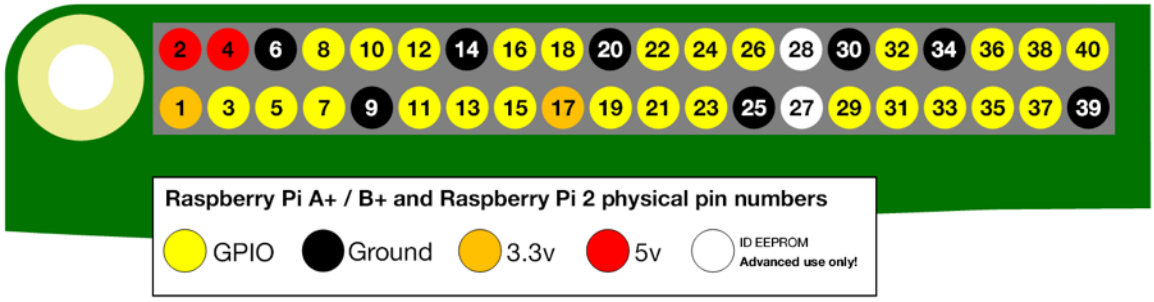

Figure 3.8: Raspberry Pi's GPIO [32]

This is achieved by using a 5 V voltage regulator 7805, it takes in a 12 V input voltage and gives a 5 V output voltage to the Raspberry Pi and the PIR sensor.

The battery-powered source is formed by using a battery pack with eight 1.5 V of AA alkaline batteries connected in series to give 12 V. When batteries are connected in series, it gives more voltage while the current remains the same. The Raspberry Pi uses about 420 mA when running at all 4 cores [33]. An alkaline battery is capable of providing approximately 3 hours of service when there is a current draw at around 500 mA [34]. This should give enough time for the main power source to come back and provide power for the whole system.

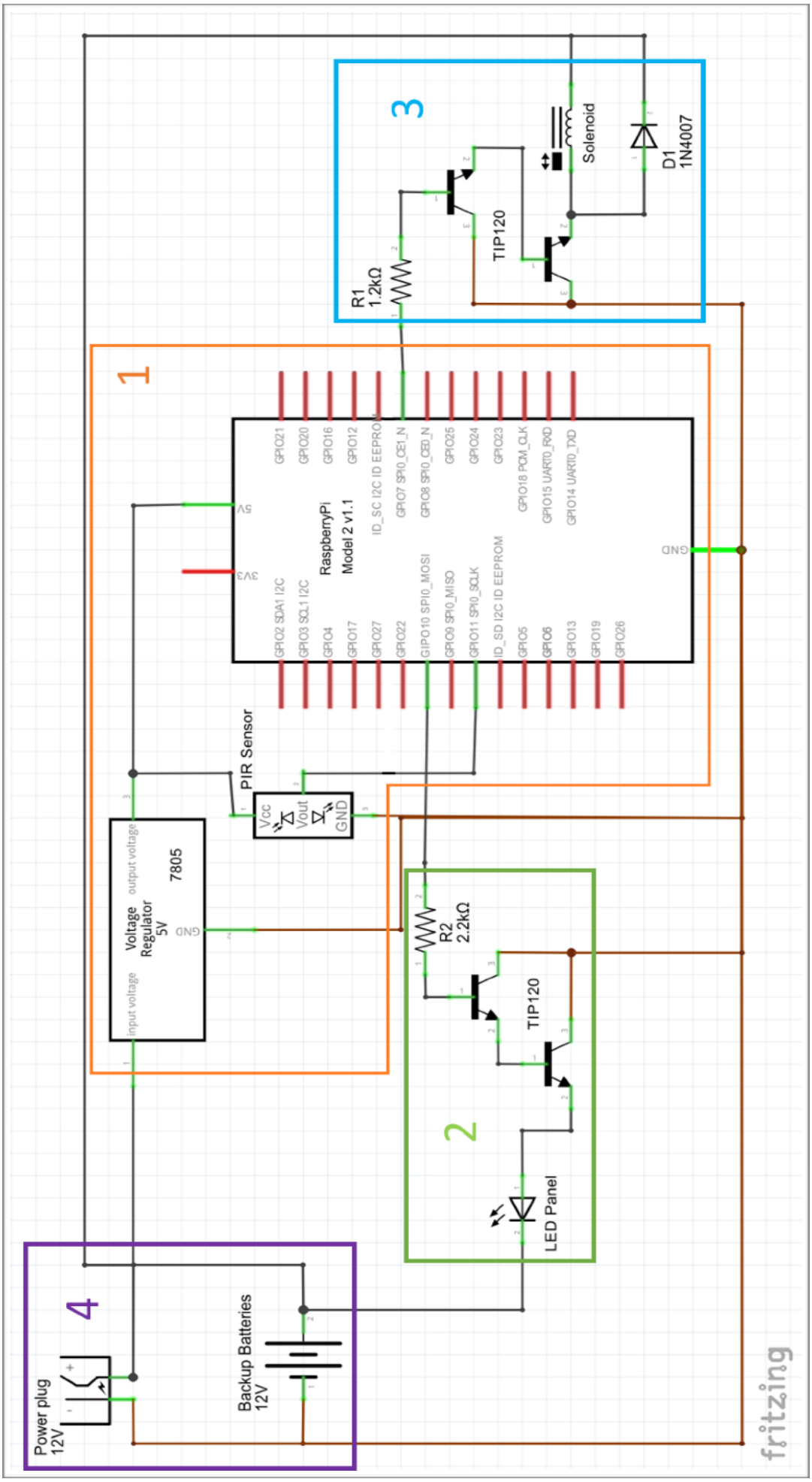

Figure 3.9: Schematic of the control circuit

Components	Functionality
Raspberry Pi	Wait for PIR sensor to detector motion then lights up the LED panel, start object detector software and controls the solenoid accordingly
5V Voltage Regulator	Reduce the voltage coming from the two 12V power source to just 5V to power the Raspberry Pi and PIR sensor
12V Power Plug	Provide power to the whole circuit, including: Raspberry Pi, Solenoid, PIR Sensor, and LED Panel
Backup 8 * 1.5V series of Alkaline Batteries	A series of eight 1.5 V batteries to form 12 V used to provide backup power to the circuit when there is a power outage
Passive Infrared Sensor (PIR sensor) Minimum detection range set to: 2.50 m	Used to detect motion and activate the object detector software so it reduces the Raspberry Pi's CPU workload when nothing passes the camera
12V Solenoid (Push-pull type)	Used to lock the cat flap when mouse is detected. Initial position of the solenoid when there is no power supplied to it is lock
LED Panel	Used to provide light for the camera to solve night time and low light problems, LED Panel is turned on when the PIR sensor detects movement
TIP120 Darlington Transistors [35]	Powerful electronic switches used to control solenoid and LED Panel. One TIP120 for each of the solenoid and LED Panel
Diode	Eliminate transient voltage that comes from the solenoid when there is sudden power loss
Resistors	Use as base resistors for both the solenoid and LED Panel to limit the current going into the base of both transistors so that the transistor doesn't blow up

Table 3.1: Functionality of each components used in the circuit

Chapter 4: Software Design

This chapter will provide all information related to the software development of this project including how electrical components are controlled via the Raspberry Pi, the different between baseline-HOG and MMOD-HOG and which one is used to develop the project, how to obtain appropriate dataset and how object detectors can be trained and tested.

Section 4.1: Controlling electrical components using the Raspberry Pi

The Raspberry Pi has the ability to control electrical components through the GPIO pins via a written code. In order to use the GPIO pins normally for this project, the Raspberry Pi's GPIO library must be installed on to the Raspberry Pi. A walkthrough guide will be given in Appendix B. Each electrical component can be tested individually without interfering each other which also make things easier to test too.

Each controlled electronic component must be correctly circuited to avoid harming the Raspberry Pi, then it can be tested on the Raspberry Pi using simple code commands. A pseudocode demonstrating how each component can be tested is shown below in figure 4.1.

```
Import necessary libraries

Set PIR sensor pin to input
Set Solenoid pin to output

Repeat forever

        If value of PIR sensor equals
1
        Set the output GPIO pin to 1

        If value of PIR sensor equals
0
        Set the output GPIO pin to 0
```

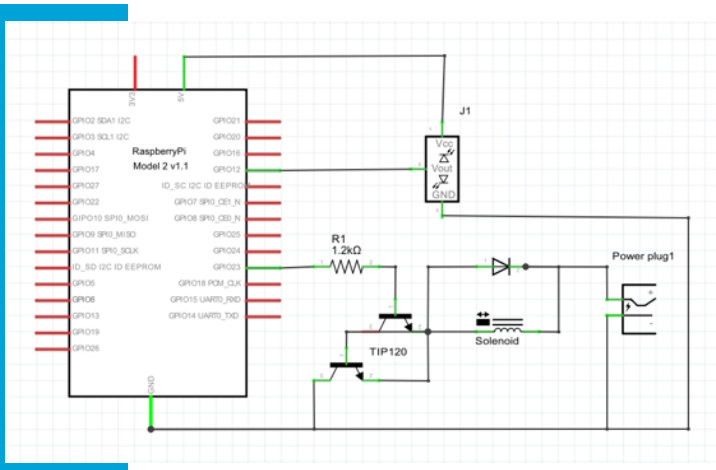

Figure 4.1: Example of controlling solenoid with PIR sensor using a written code

The figure illustrates that electrical components can be combine to perform a task that is controllable by the Raspberry Pi through a written code. Other components such as the LED or the PIR sensor undergo individual test with simple written codes then combine together into the circuit shown on Section 3.4 of Chapter 3, to work together as a single system.

Section 4.2: What is OpenCV

When talking about computer vision and machine learning software library, OpenCV is one of the most mentioned open source library that everyone related to this field knows the library. OpenCV is abbreviated from Open Source Computer Vision, is most renowned for its real-time computer vision and machine learning as it provides more than 2500 optimised state-of-the-art algorithms [10]. Algorithms that are provided can be used to perform many computer vision tasks such as detecting and recognising faces and objects, movement tracking and extract 3D models of objects. OpenCV is used extensively by many famous companies such as Google, Yahoo, Microsoft, IBM and many others more.

Due to its wide range of programming languages available for the library including C ++, C, Python and Java, OpenCV became popular to many users who wanted to perform real-time computer vision

As mentioned from Chapter 2, Section 2.1.2, OpenCV allows us to manipulate pixels of an image. The library provides many basic functionality which made it easy for users to perform simple image manipulation such as rotating, translating, resizing, flipping, cropping and converting colour images to grayscale for faster processing speed. For this project, some of the basic image manipulation functions provided by OpenCV will be used.

Unfortunately, using OpenCV's functionalities to perform object detection proved to be less effective than using Davis' method of object detection. This is due to the fact that using OpenCV's method of detection, Haar cascade, results in many noticeable false-positive results when searching for objects [22] even though hard negative mining was applied to each of the sample images. Moreover, it is very slow to train and detect objects from a given image with OpenCV but a lot faster with Davis' method.

Initially, the project was created based on the object detection steps in Section 2.2.5 in Chapter 2. The time it takes to train the dataset takes roughly around 1 - 2 hours on a high spec computer using OpenCV's method tested with CALTECH-101 dataset [15]. CALTECH-101 dataset is a popular dataset that has 101 categories of objects and is commonly used by many researchers and developers as a benchmark for object detection. Testing the trained object detector can only be done after the training is complete, and usually it can't be known if a single parameter is adjusted correctly, to change a single parameter the whole object detector has to be retrained again through the same process that takes 1 - 2 hours.

From the reasons mentioned above, it can be seen that this method is very time consuming as it goes through many process before the final functioning object detector is obtained. In order to avoid this, Davis' method of object detection was then adopted for the project.

Section 4.3: What is dlib?

The alternative solution to OpenCV method of detecting object is Davis's object detection method called Max-Margin Object Detection (MMOD) [21]. This method is available as an open source computer vision library just like OpenCV called dlib. Dlib is a toolkit containing machine learning and image processing algorithms written in the C++ programming language but the Python library is also provided.

The transition to this method of object detection was necessary because the training process takes too much times and many parameters have to be adjusted for the OpenCV's method. Whereas using dlib's object detection method does not only reduce the training time from hours to minutes, it also provide a much higher accuracy result while also reduce the number of parameter needed to be adjust from ten parameters to only one and eliminates the need for negative dataset.

The MMOD is a method used for machine learning to detect objects in an image. MMOD is used to improve any object detection method such as HOG filter. Proven in Davis King's paper, using a single HOG filter learned using MMOD has the capability to outperform state-of-the-art object detector [21].

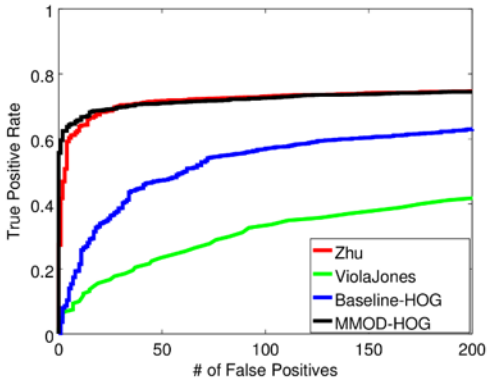

Figure 4.2: A comparison between other object detection methods with MMOD-HOG [21]

The three changes mentioned in Section 2.2.5, Chapter 2 is what makes the dlib's method of detection different from other baseline object detector as it reduce many steps taken to train an object detector when compared to others. It also benefits detecting objects using the Raspberry Pi as the Raspberry Pi is already slower than a normal personal computer. Using MMOD makes it quicker to detect objects while also produces impressive results.

Section 4.4: Obtaining the dataset

As mentioned from the previous sections, the object detector used in this project will be using the MMOD-HOG object detection method along with a few basic functions from OpenCV such as resizing, loading and displaying images and drawing a rectangle to display the object when it is detected.

In order to start training the object detector, the user need to obtain a positive dataset. Discussed in the Chapter 2 and 4, the user does not need to provide negative dataset to the object detector for this project. For practicality, dataset does not need to be very large, as this will slow down the training process drastically because the sliding window will have to be applied to each image of the dataset and will also lead to having too many layers of image pyramid. **A suitable resolution of a dataset should be less than 500 by 500 pixels**.

For this project, a cat puppet is used to represent a real cat and a yellow ball to represent "uninvited guest". The cat puppet and the ball will be trained in the object detector therefore, the positive dataset for this project will be both the cat puppet and the ball.

Figure 4.3: The cat puppet and yellow ball used to represent a real cat and "uninvited guest," respectively.

Appropriate samples can be obtained easily by just using images taken from the camera of a smartphone, but images have to be resized to a smaller resolution because of the indicated reason described above.

The resolution used for each of the samples of both the cat puppet and the yellow ball is 300 by 400 pixels. Both images are taken from different environments to produce varieties in the samples. Images of the yellow ball was combined with own samples and images of different kind of balls were taken from Google Images. Unfortunately, the cat puppet is quite unique with its own appearance, using other cat puppet images can affect the result of the object detector so images of the cat puppet were self-obtained.

The cat puppet samples contains "75" positive samples and yellow ball samples contains 80 positive samples. Each images were taken from various environment and different lighting conditions. Example of the image used as the dataset is shown in figure 4.4 and figure 4.5.

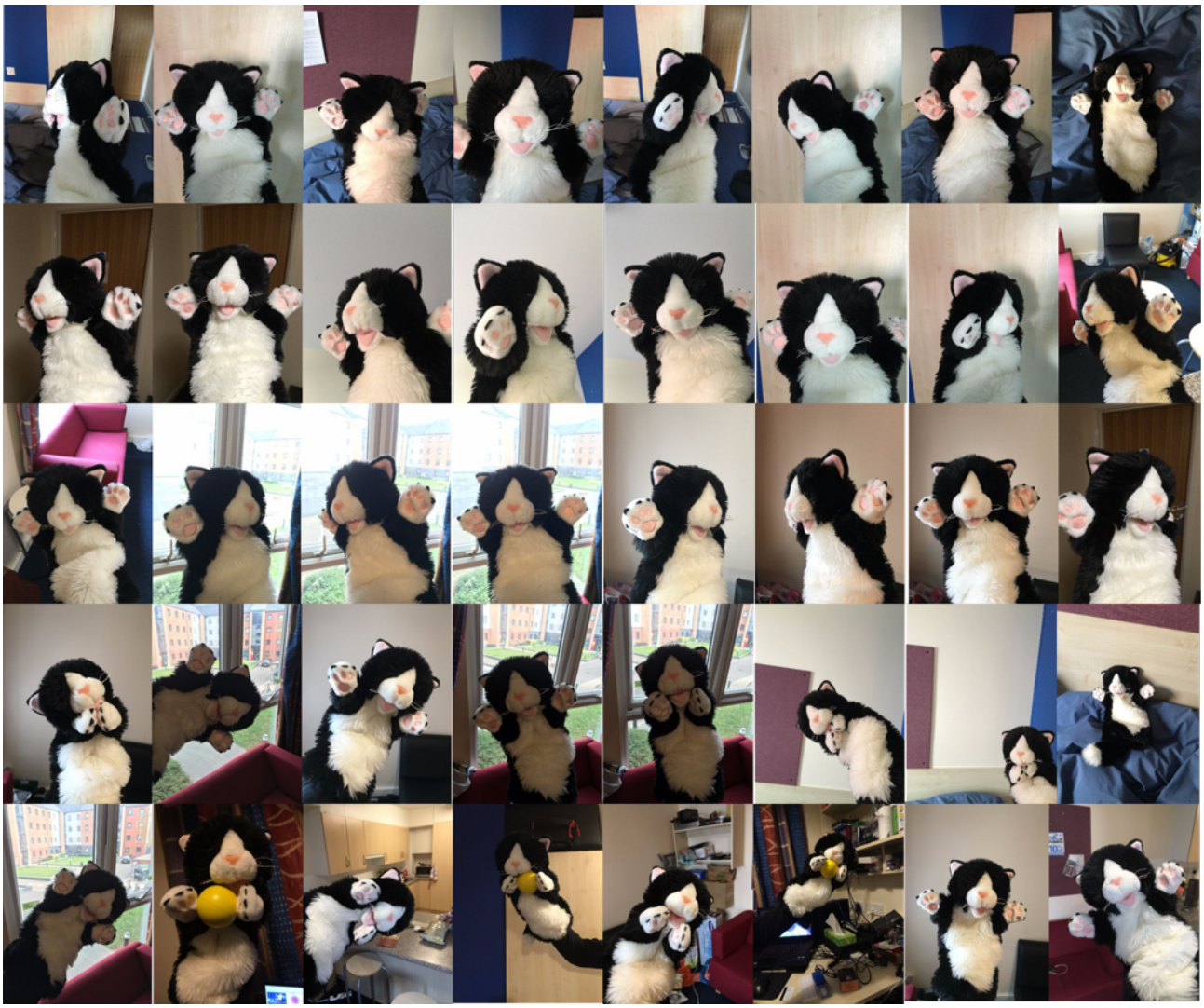

Figure 4.4: Examples of positive samples used to train the cat puppet detector

Figure 4.5: Examples of positive samples from random Google Images [36] — [65] and self-taken images used to train the ball detector

Section 4.5: Training the object detector

After obtaining the object detector, the dataset is then used to train the object detector. Before the object detector can be trained, there are two things that is required. One is the positive samples and each of the image's annotation which is called a bounding box. A bounding box is a set of coordinates, specifically at the top-left, bottom-left, top-right and bottom-right stored in a file that will be used as reference by the object detector to locate where the object of interest is located in an image.

The bounding box is obtained by labelling on **each** positive samples image where the object of interest is at. When using a publicly available datasets, these usually comes with bounding box, but when a personal dataset is used, bounding boxes are required to be obtain manually. In order to obtain bounding boxes for personal dataset, a tool provided by dlib called "imglab" can be used.

The imglab tool allows the user to create their own bounding boxes for each of the sample image. In order to use the imglab tool, the user has to install the tool first, installation instruction will be provided in Appendix A. Figure 4.6 illustrates an example of the imglab's GUI used to create bounding boxes for all images in the sample.

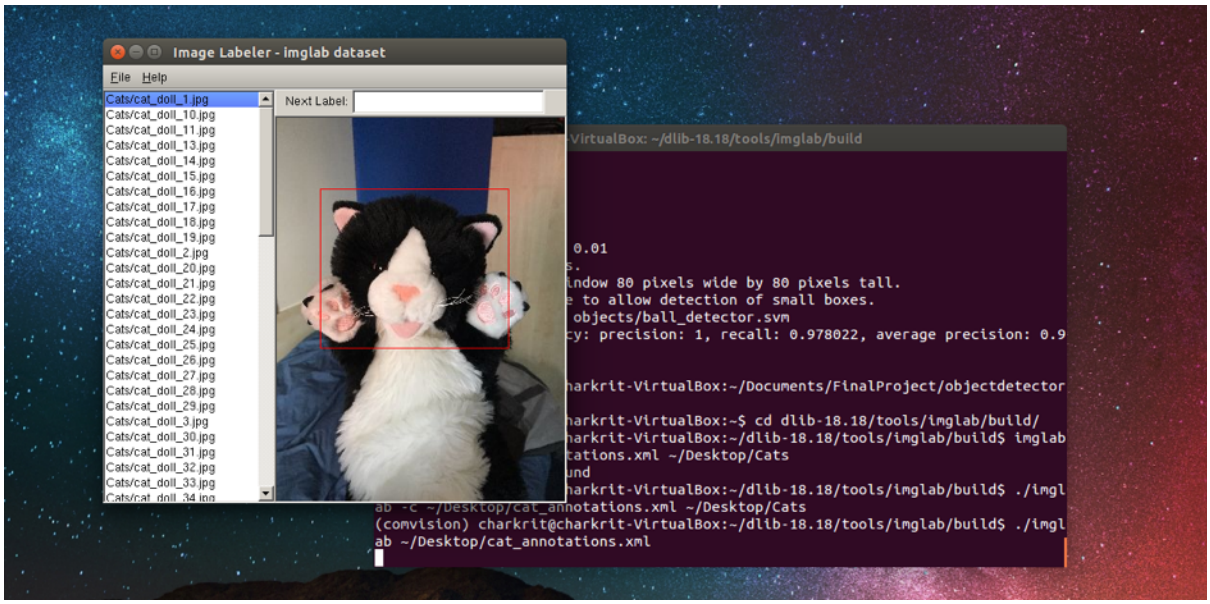

Figure 4.6: Example of imglab tool's user interface used to create bounding boxes for samples

After all labelling are completed, it will be saved into a file that contains all bounding boxes information for each image.

The tool is simple and easy enough for every developers. A user guide to using imglab tool will be provided in Appendix C.

One thing to be careful about is that great care must be taken when annotating bounding boxes because if images are not annotated correctly and properly, it would result in having unusable training datas for the object detector. It is also faster to train the object detector on a computer than using the Raspberry Pi as the CPU in computers are surely a lot faster than the Raspberry Pi's.

Once positive samples and bounding boxes are obtained. They would then be trained into the object detector by using dlib's training function [69]. When training, the C parameter mentioned in Section 2.2.4, Chapter 2, has to be adjusted carefully to avoid the object detector overfitting.

For detecting the yellow ball, the detector is trained with the parameter C of 100, sliding window size of 80 by 80 pixels and takes 351 iterations to complete and about minute and 40 seconds to train while using four CPU cores on a high-spec laptop.

For detecting the cat puppet, the detector is trained with the parameter C of 100, sliding window size of 81 by 79 pixels and takes 227 iterations to complete and about 35 seconds to train while using four CPU cores on a high-spec laptop.
Looking at figure 4.7 and figure 4.8, it can be seen that the shape of the HOG filter resembles the object that is used to train the object detector. This means the training process has completed successfully. The detector is then saved to a SVM file which is then used to detect objects.

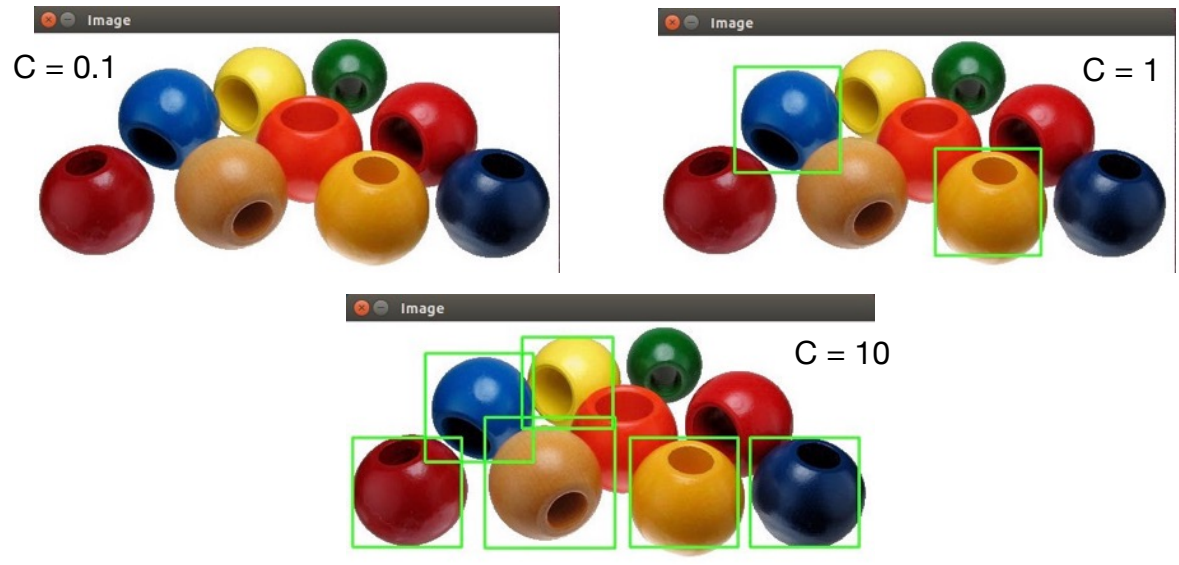

Figure 4.9: Result of changing C parameter [66]

Before training, the parameter "C" has to be adjusted so that it can differentiate the object from an image, but has to avoid the parameter being to high from overfitting. Figure 4.9 illustrates what happens when "C" is too low for the ball detector.

The object detector can be trained by using the following command line on the Ubuntu's terminal:

python train_detector.py --xml objects/catdoll_annotations.xml --detector objects/catdoll_detector.svm

The first command line argument takes in the annotation of the object which is an xml file. The second command line argument tells what the detector will be named as an its save location.

Section 4.6: Testing the Object Detector

Once training has completed, the object detector can be tested. Testing was done by

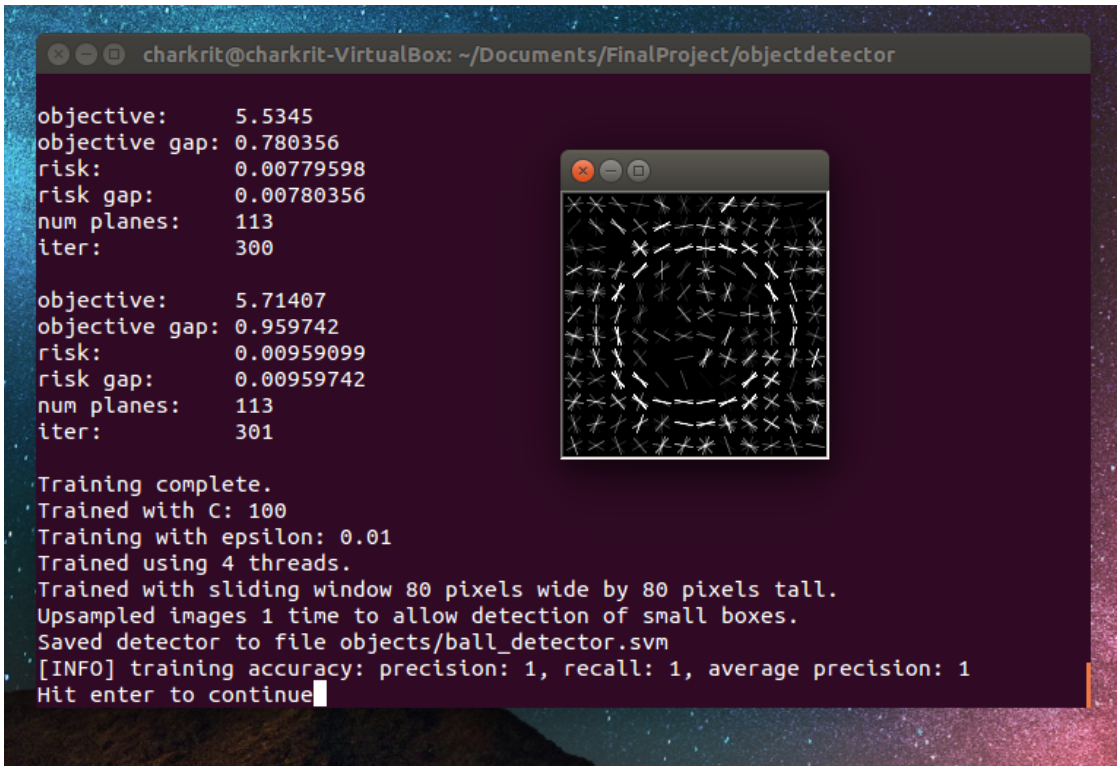

Figure 4.7: Result of HOG filter when training for ball detector is complete

taking some images used in the training process and new images to check if the object detector was still able to recognise and identify the correct location the object that it was trained on.

The object detector can be tested by running the following command line on the Ubuntu's terminal:

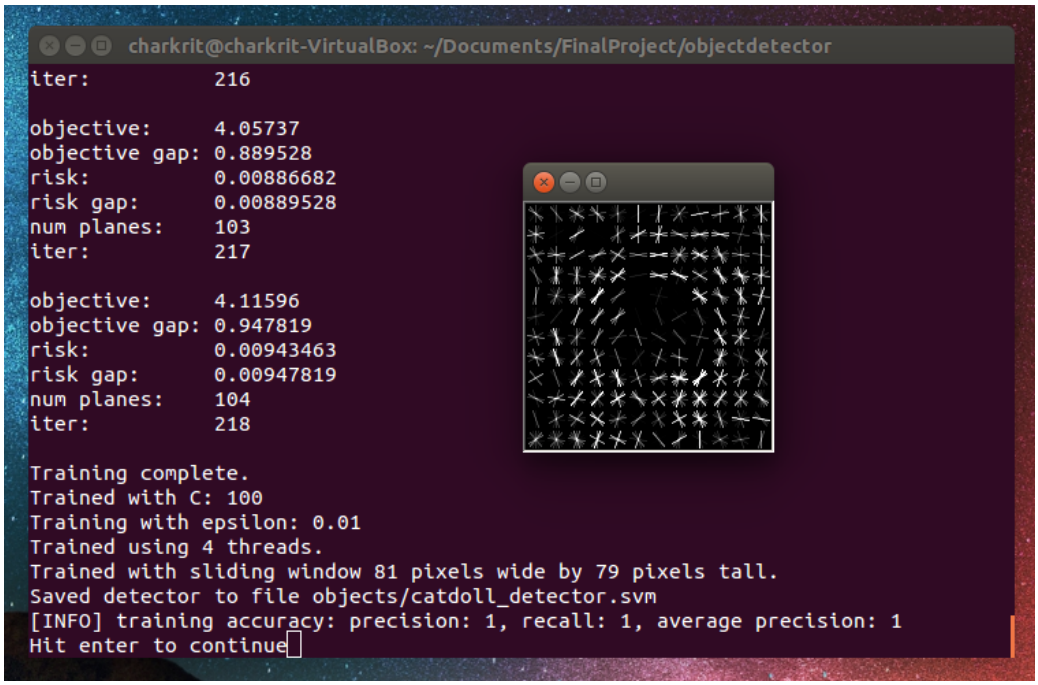

Figure 4.8: Result of HOG filter when training for cat puppet detector is complete

python test_detector.py --detector objects/catdoll_detector.svm --testing objects/catdoll_testing

The first command line argument takes in the object detector the user want to test, this can be changed by changing the file path next to it. The second command line takes in the path of the test images.

It can be seen that from figure 4.10 and 4.11, which shows some of the test results selected, both detectors were able to identify and locate all objects accurately in every supplied test images apart from objects that are already partially missing from the image. The results of both object detectors are very satisfactory.

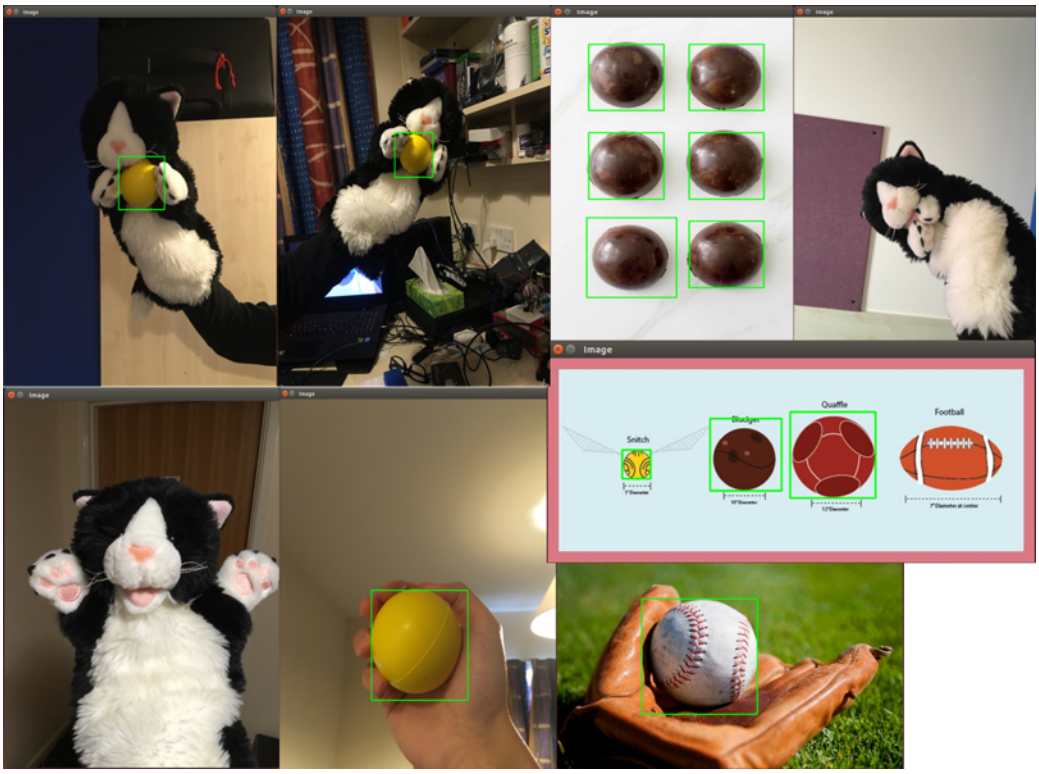

Figure 4.10: Ball detector test results [67], [68]

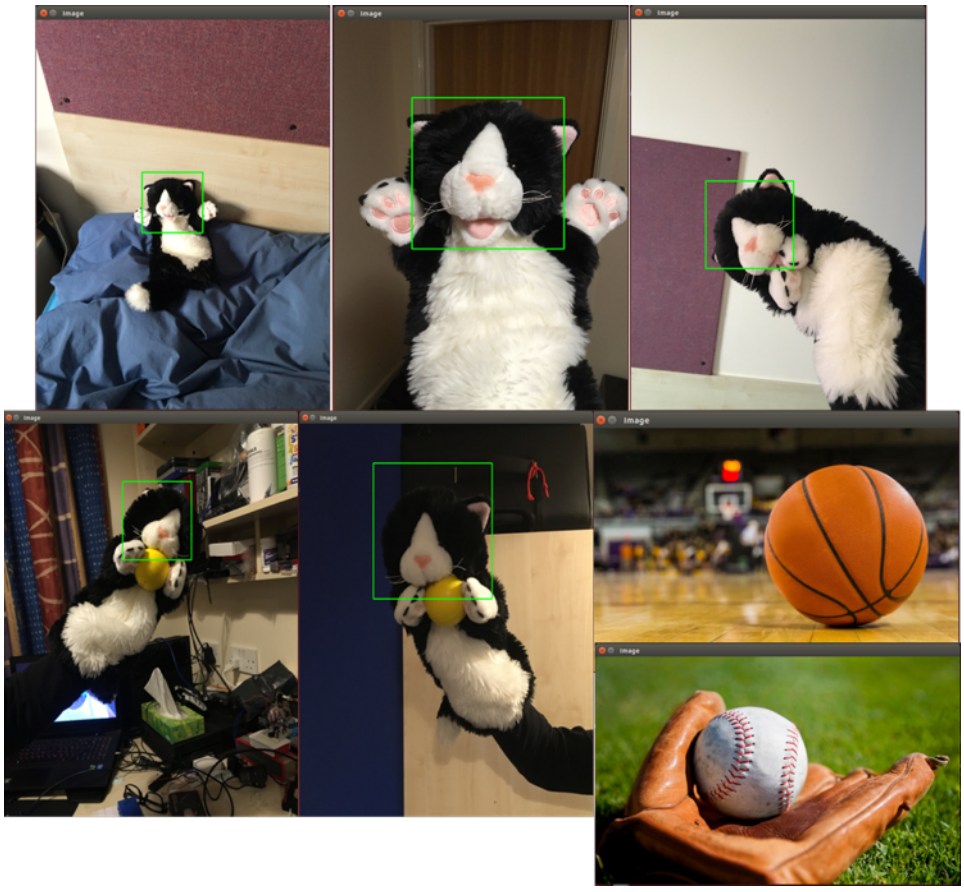

Figure 4.11: Cat detector test results [45], [55]

Chapter 5: System Evaluation

In this chapter, the development from hardware and software will be combine to create a single system. The mechanic of the system, the integration of object detectors with video streaming, increasing the frame rate of camera module and result of overall system and its problems will be discussed.

Section 5.1: System's mechanism

At the beginning of the project development, the system mechanics or the design of how the whole system is going to work out was planned. The planning was aimed to get an overview of how the whole system is going to work. Concerns for the project were addressed, including the software's flow, how the locking mechanics work and the ability to distinguish between the owner's cat and the stranger's cat.

The flowchart shown in figure 5.1 is divided into two sections, one section for the cat flap's controlled lock discussed in Chapter 3: Hardware Design and Raspberry Pi's controlled lock discussed in Chapter 4: Software Design. The figure is aimed to illustrate how independent locking mechanic works and how it can allow the system to be more practical as it doesn't have to rely on one another.

The system's mechanic is designed to work according to the following steps:

1. Detect motion from the PIR sensor
2. If motion is detected, turn on LED Panel to ensure enough light is shining the object for the object detector and start the object detection
3. The object detection now finds the object and report the result on to the screen which can be separated into three cases:
    1. Only cats was found in the captured image, unlock solenoid for around 10 seconds for the cat to enter
    2. Only "uninvited guest" was found in the captured image, solenoid remains locked
    3. Both cats and "uninvited guest" were found in the captured image, solenoid remains locked
4. In case of the user want the software to stop working, pressing "q" will stop the software, to reactivate, the command line used to launch the software will have to be reentered.

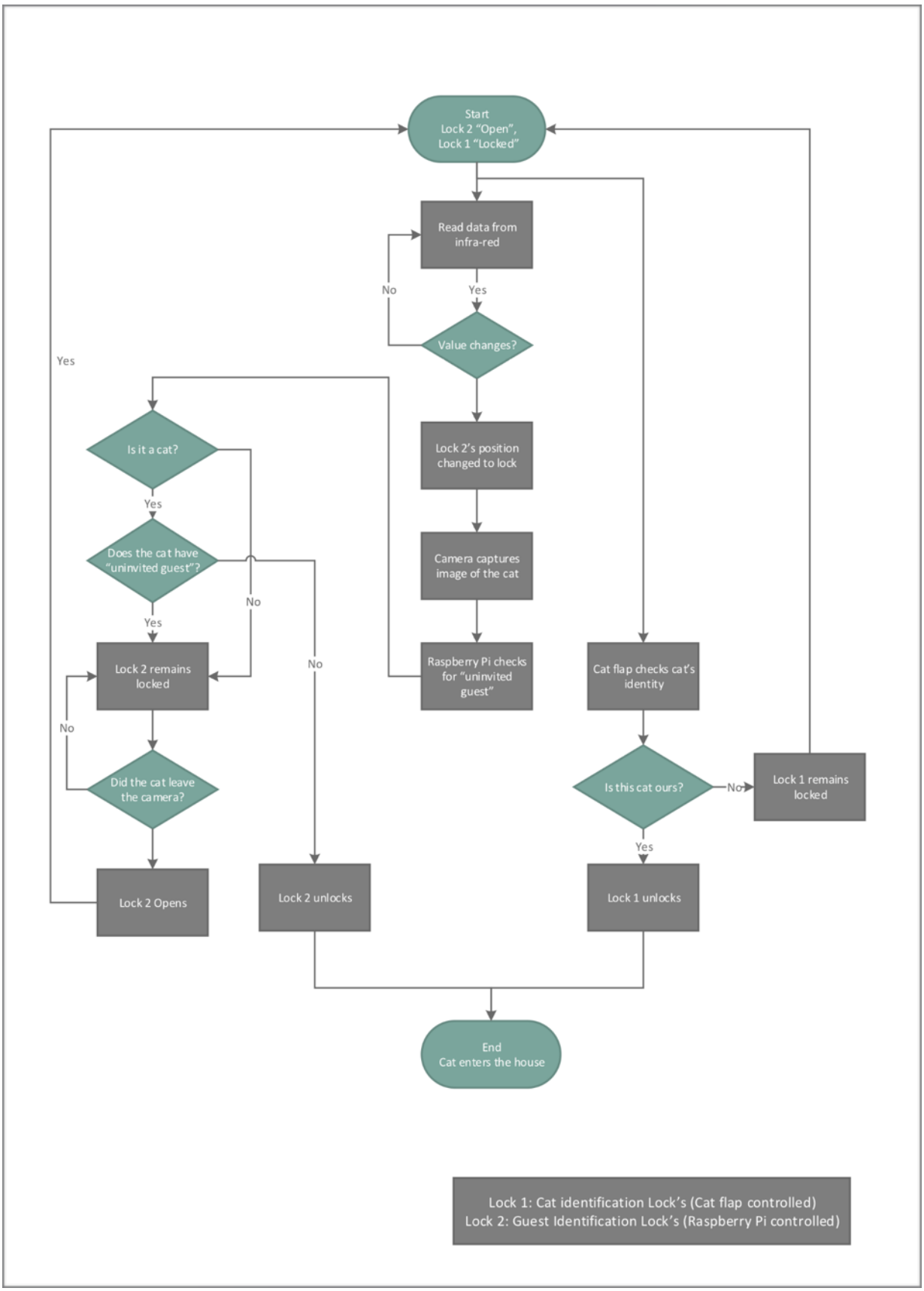

Figure 5.1: Flowchart of the system's mechanic

Section 5.2: Integration of object detector with system

Since the object detector was completed in another computer, it would have to be imported to the Raspberry Pi. Fortunately, this process is not difficult because the only thing that needs to be ensure when porting the software is that the version of Python and OpenCV on the Raspberry Pi matches with the version used on Ubuntu to train the object detector. The version needs to match because functions used in the software will not be able to work on a different version, meaning that the whole software would be useless.

Developing the software on the Ubuntu computer was developed on a virtual machine since the libraries support on Windows was not entirely available, instructions on installing OpenCV on Windows have been long-obsolete [70]. An alternative solution is to use OSX to install OpenCV and other necessary libraries. Unfortunately, virtual machine does not have support camera functions directly, therefore scanning for objects with video streaming was impossible on the virtual machine. Implementing object detectors with video streaming was then completed on the Raspberry Pi. Additionally, webcams and Raspberry Pi's camera module does not used the same function to operate and activate therefore, time can be reduced by just integrating the object detector directly on the Raspberry Pi to solve the problem of having to use incompatible functions.

Changing from detecting objects in image to detecting objects from video streaming was a bit of a challenging task. The user has to activate the camera at the start of the software and tells the camera to capture the image of the cat and analyse the image with the object detector. Unfortunately, due to the Raspberry Pi's speed integrating the object detector with video streaming can cause the video streaming to have a very low frame per second. This problem can be improved, and will be detailed in the next section.

Integrating the software with the test code used to control electrical components was done successfully. The electrical components were controlled as an input and output by the Raspberry Pi. This was achieved by combining the test codes mentioned in the beginning of Chapter 4, with the object detectors that is already integrated with video streaming. As for the input, the PIR sensor, its value is read at the start of the software and the output LED is then changed to light up the environment when the value of PIR sensor has been changed by detected motion. The solenoid is controlled according to the three conditions mentioned in the previous section after the object detector has already finished its job.

The final system can be seen in figure 3.5 and figure 3.6 in Section 3.3 of Chapter 3.
At the moment, the system works according to expectation, except for the camera's frame rate.

Section 5.3: Increasing the camera's module frame per second

The Raspberry Pi is expected to be slow when it comes to image processing as the CPU and GPU power is not that great when compared to a large computer. When running video streaming without the camera module, the Raspberry Pi already gives quite a low frame rate of less than 20 frame per second (FPS). The normal human eyes perceives 24 FPS as normal movement, therefore any FPS lower than this would result in us seeing movement as still images. Therefore, a video that has about 20 FPS may allow us to notice it as videos instead.

Integrating the object detector with video streaming results in a poor 1-2 FPS. The poor performance is expected and can be improved using a few solutions.
The first solution is to set the software to detect objects at every 100th frame, or a different number is also applicable. Doing this helps reduce the workload of the CPU as it does not have to scan for objects in every single frame of the streaming video. The second solution is to use multi-threading to enhance the camera's FPS [71]. This means we use the other cores of the CPU to work with the camera instead of just one. The final solution is to limit the resolution of the video streaming frame, by limiting the resolution to a smaller size, we can increase the FPS of the video streaming as it will allow lesser pixels needed to be analyse by the object detector. The resolution of the video streaming was limit to a maximum width of 400 pixels.

Applying the three solutions mentioned above allows the video streaming to achieve a high FPS than it was before. When the object detector is not analysing objects, the video streaming operates at a normal speed of about 20 FPS, the object detector starts analysing objects when the 100th frame is reached and at that frame, the video streaming will freeze, and resumes when the object detector is done analysing the frame.

Quantitatively, running at an average of 20 FPS, when not detecting objects, is quite a success because it allows the user to see it as a video streaming now instead of still images. When the 100th frame is reached, the object detector is activated, this would mean that the object detector will start scanning for objects (reaching the 100th frame) approximately every 3 — 5 seconds. This is a number that can be changed accordingly to match the owner’s cat style, if the cat is fast, then the number can be dropped down or if the cat is slow, the number can just be increased. For the purpose of demonstration, it is kept 100th frame as the cat puppet is controlled by our hand.

Section 5.4: System evaluation

In this section, the results of the final testing where the hardware and software sections are combined into a single system will be evaluated. The final system is shown in figure 3.5 and 3.6 in Chapter 3. Looking at the hardware section of the system, the immediate switch of the two power source, battery-powered and power plug, were successful. The system wasn't forced turn off when changing the supplied power.

The software was then tested 200 times in total to test for three conditions: just cat, just mouse and both cat and mouse. The environment that was tested was in a well lit room (100 times) and in a dark room (100 times) to simulate night time. The results can be seen in the table and figures below:

Experimentation results in a well-lit environment (100 times)			
Detection matches:	**(Out of 40 times) Testing with "Just Cat"**	**(Out of 20 times) Testing with "Just Mouse"**	**(Out of 40 times) Testing with "Cat & Mouse)**
Just Cat	34	0	1
Just Mouse	2	20	8
Cat & Mouse	4	0	31
No detection	0	0	0
Correct Detection Percentage	85%	100%	77.5%, including "just mouse": 97.5%

Table 5.1: Experimentation results on the number of detections for each conditions in a well-lit environment

Experimentation results in a dark environment (100 times)			
Detection matches:	**(Out of 40 times) Testing with "Just Cat"**	**(Out of 20 times) Testing with "Just Mouse"**	**(Out of 40 times) Testing with "Cat & Mouse)**
Just Cat	31	0	3
Just Mouse	3	20	7
Cat & Mouse	6	0	32
No detection	0	0	0
Correct Detection Percentage	77.5%	100%	80%, including "just mouse": 97.5%

Table 5.2: Experimentation results on the number of detections for each conditions in a dark environment

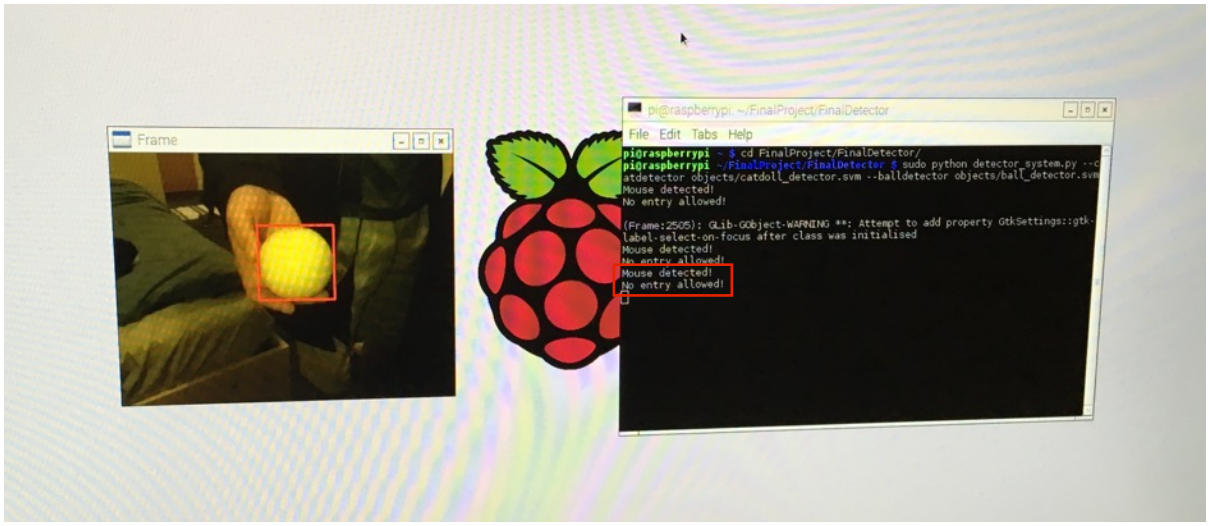

Figure 5.1: Object detection testing with just "uninvited guest" in the frame

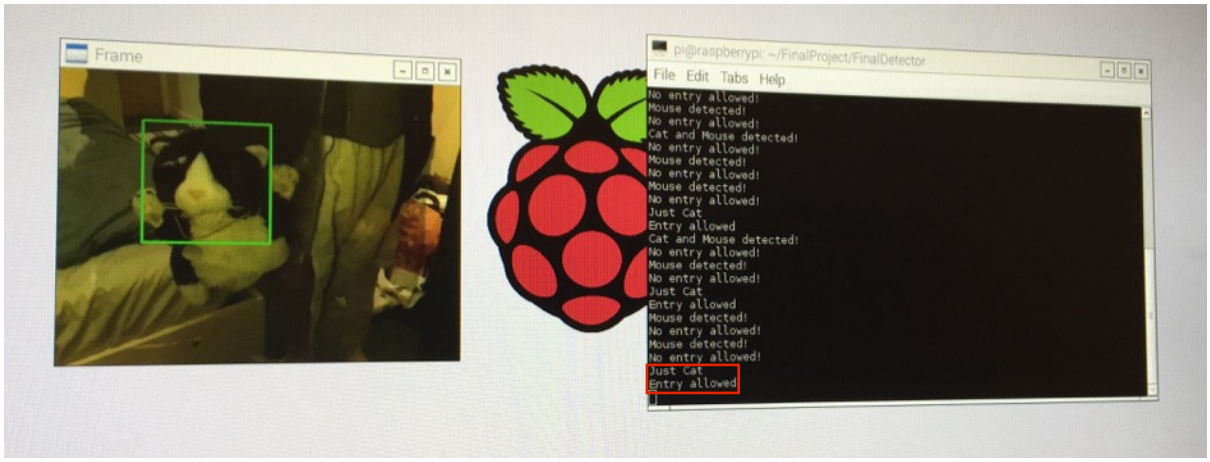

Figure 5.2: Object detection testing with just cat puppet in the frame

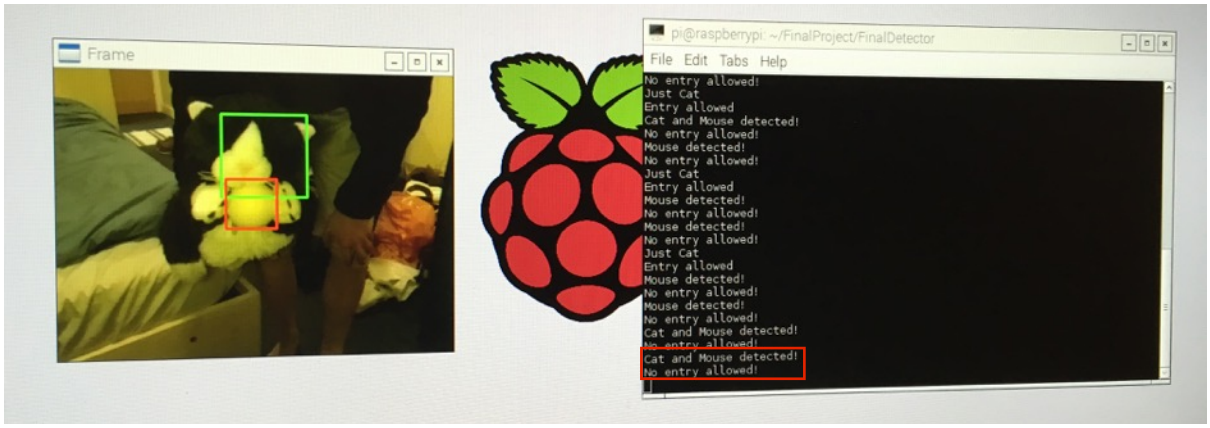

Figure 5.3: Object detection testing with both cat puppet and "uninvited guest"in the frame

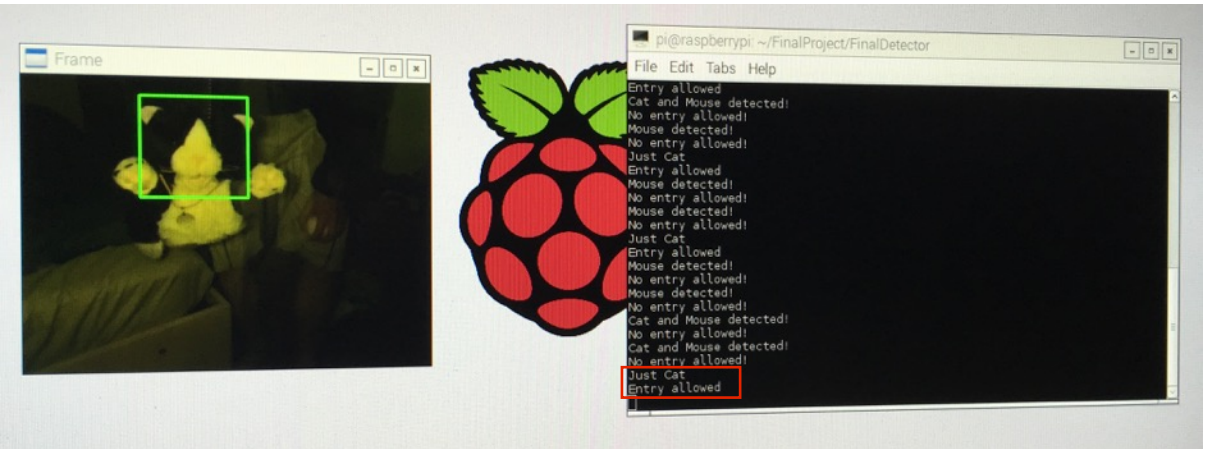

Figure 5.4: Object detector working in a dark environment

Results from the table shows that there is more than 77.5% correct detection rate on every case tested. This could be considered as a success and accurate detection, in most cases, there are up to 85% correct detection rate, the only time the performance drops is at dark environments when detecting the cat. In fact, in the “both Cat and mouse” case, the results can actually be combine with the result from “just mouse” case because the priority in this project is to be able to detect the mouse(ball) in the frame, we can see that the detection rate had increased to 97.5%.

The time it takes for the object detector to analyse the frame is about 6 — 8 seconds but can rarely take up to 10 seconds. Combining this with the time mentioned in the above section, 3 - 5 seconds. In total, it can take a maximum of 15 seconds to complete the whole process. Knowing both times, the user can adjust at which frame should the object detector start analysing the image to match the time it takes for the cat to arrive at the cat flap.

The software can be initiated by typing into the command line using the following command:

sudo python detector_system.py --catdetector objects/
catdoll_detector.svm --balldetector objects/ball_detector.svm

The command line takes in two arguments, the first argument is the cat puppet detector and second, the ball detector. The "sudo" command is "super user do" which is similar to "Run as administrator" on Windows, this is required to access the Raspberry Pi's GPIO libraries. The software is written to take in command line arguments to allow room for other developers to change the detector to detect something else if they want to in the future.

When the object detector detects any of the three cases, it prints the result and the status of the solenoid onto the terminal screen of the Raspberry Pi. From the results shown on both screens, it can be seen that the object detectors were successful in identifying both of the objects. In addition, the object detector is also able to work during night time thanks to the LED panel, figure 5.4 shows the system working at a dark environment.

Although, results looks satisfying, there are still problems with the system. The first problem is that it sometimes recognises the cat puppet's paws as a ball even though testing the ball detector on images doesn't show this result, this is rather understandable as the cat puppet's paws does look like a round shaped object. Due to this particular reason, the C parameter used when training the object detector to detect the ball was increased from 10 to 100 to improve its strictness in the hope that it might be able to tell the difference between the ball and the cat's paws.

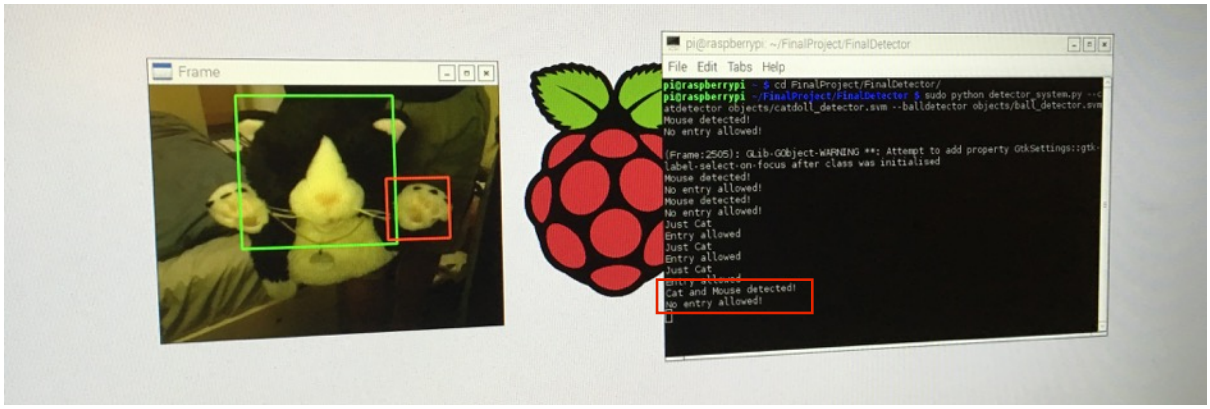

Figure 5.5: Incorrect detection of the ball object detector

The result from increasing the C parameter show signs of decrease in incorrect detection, it still shows up on rare occasions, but is still better than using the C parameter of 10.

Another problem is the hardware, using too many USB drives and connecting the internet cable to the Raspberry Pi at the same time can result in the overheat of the 7805 5V voltage regulator. This will cause the Raspberry Pi to be forced shutdown/reboot all the time, which can break the Raspberry Pi if left unsupervised. This should not be a problem when the system is operated normally as the user wouldn't need to be required to use any USB drives, internet connection is not required at all and the HDMI cable isn't needed to be plugged at all if the user doesn't want to look at the video streaming.

The recommended amount of USB drives used is no more than 2.

Chapter 6: Future Work

At the moment, there are a lot of room for further development of this project. Given more time for the project, there are things that can be further develop to improve the system.

1. Upgrade to Raspberry Pi 3

Given that the Raspberry Pi 2 still does not provide enough speed for the detector to detect objects fast enough, at the time of this report's writing, May 10th 2016, there was not enough time to transfer everything to the newly announced Raspberry Pi 3 which provides a 50% increase in CPU speed [72], which should then be able to reduce the time taken for the object detector to detect objects.

Additionally, the FPS of the camera module could potentially be improved by the new CPU as well when using the camera with multi-threading.

2. Train object detectors with more positive samples

Normally, an object detector is trained with more than 300 positive samples, which would result in a much more accurate object detection. Both detectors in this project were only trained from 75 images, for cat puppet, and 80 images, for ball detector. It would really benefit the object detector if there were more positive samples to train with.

It might also solve the problem of the ball detector's incorrect detection of the cat's paw as the ball.

3. Train object detectors with real cat and uninvited guest

Since this project is not capable of testing the system with a real cat and mouse, the cat puppet and ball was used to represent the cat and mouse, respectively. In order to use this project in real situations, the object detectors have to be retrained using a real cat and uninvited guest such as a mouse or a bird.

It is also possible to add more types of uninvited guest to the project, but the software would have to take in a third command line argument.

Chapter 7: Conclusion

Section 7.1: Time Plan Reflection

Looking back at the time plan created which includes a list of objectives that were needed to complete the project, it can be seen that all of the objectives are complete within the time given to create the project.

Although, the time taken to complete the project were a bit slow on some objectives and some of the objectives were fast to complete. Comparing the time plan written in the interim report and the actual objectives that were made, it can be seen that there is not a lot of mentioning in the hardware section of the project. This is because the initial thought was that the hardware for this project would not be very difficult to create, where in actuality, it proves to be a bit of challenge. During the initial planning, there were unknown problems that appeared as the project progresses. For example, adding a backup power source or eliminating two power plugs to a single power plug that provides power to the whole circuit. These challenges take extra time to accomplished and was not foreseen at the initial planning phase.

Another difficulty that I faced was learning how to use command line and Python programming language. Having grown up in the era of graphical user interfaces, GUI, using command line seems a bit outdated for me. Python language was a new language that I have never used before. Fortunately, due to the nature of the language and having a background on C and C++ programming language, learning Python was not that hard because the syntax of Python is still very similar to C and C++. The command line however, was difficult to remember and took a lot of time to actually use it fluently, as it is not something I am familiar to. As I progress through the project, it seems that command line is not outdated at all and is an essential tool to work with that have its own benefits that a GUI can't accomplished.

All in all, I think that all the objectives set for this project was completed with great success and with improvements to the objectives as well.

Time Plan

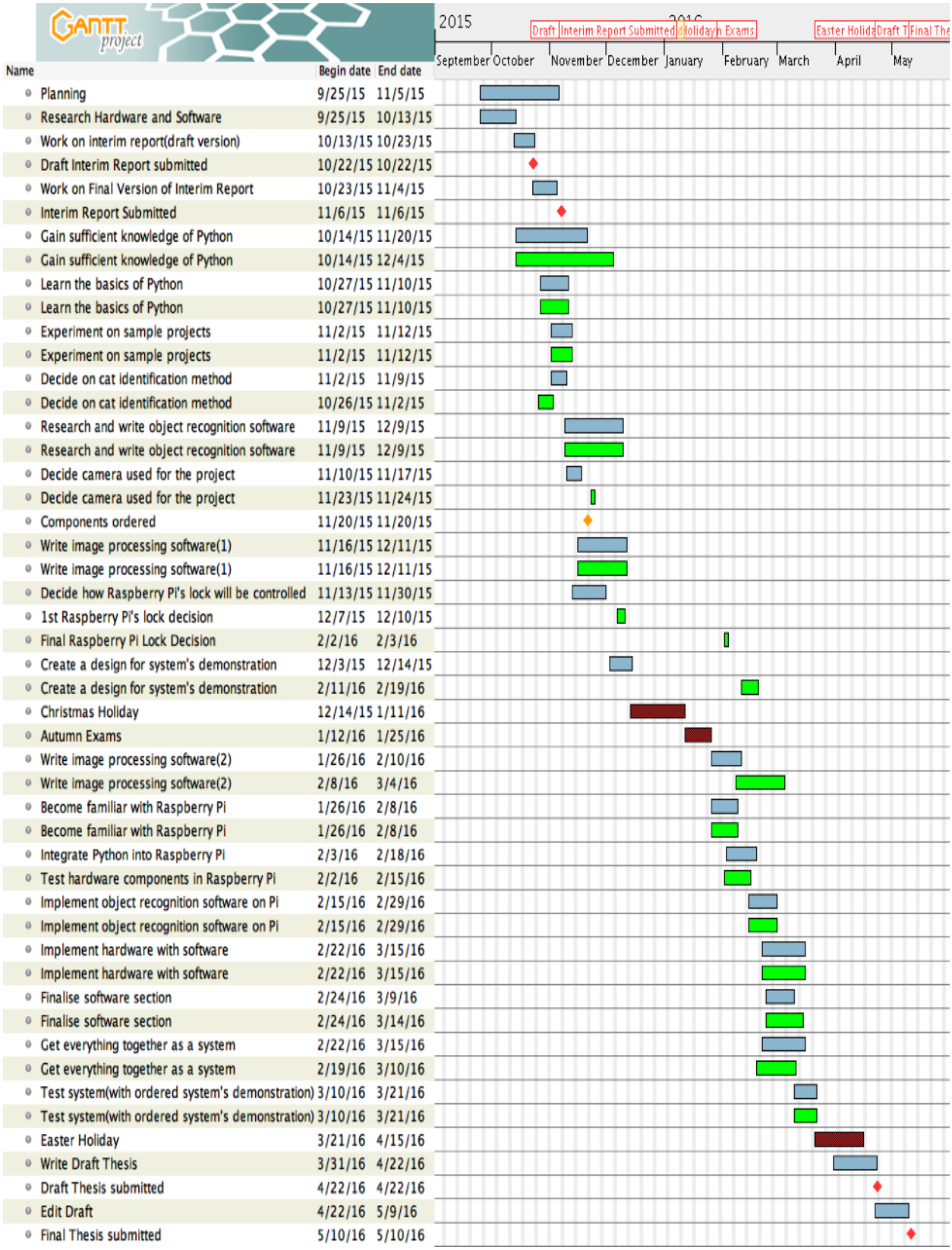

Figure 7.1: Side by side comparison of original time plan with the actual time taken to complete objectives

Section 7.2: Conclusion

The report shows the hardware and software development stages of the project are told and the system functionality was demonstrated on the Raspberry Pi. The project had followed the main proposed objective successfully, which was to "create a cat flap control system that can replace a bulky computer by using a Raspberry Pi to control the cat from bringing uninvited guest into the house".

The system demonstrated in this report shows that the project is practical and uses a minimum amount of space to fully implement a system that has the ability to detect foreign objects using an embedded computer like the Raspberry Pi. The report was completed in the hope that developers can pick up the project and continue to improve the project even further after reading the report. The software itself is easy to use and easy to develop even further as Python is a very popular and easy to understand programming language.

All proposed goals are completed with great satisfaction and within the time given to complete the project.

Appendix

Appendix A: Installing development environment on Ubuntu

In order to recreate the project, the development environment needs to be setup on both the computer and Raspberry Pi. Unfortunately, OpenCV doesn't have official support for some libraries on Windows, installation is then recommended on Ubuntu or OSX. For Windows system, Ubuntu can be installed using virtual machine software called VirtualBox. In Appendix A, the installation guide will provided for just Ubuntu which can referred to at [73] and [74] for OSX, but, we will be using OpenCV 2.4.11 instead of 3.0 to match the version used in Raspberry Pi. Additionally, some other libraries not mentioned in [73] and [74] will also be installed.

Step 1: Open the terminal and update apt-get package manager and install other necessary tools by typing in the following command line

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install build-essential cmake pkg-config
```

Step 2: Install necessary image packages

```
sudo apt-get install libjpeg8-dev libtiff4-dev libjasper-dev libpng12-dev
```

Step 3: Install GTK development library and other libraries required for the OpenCV to operate normally.

```
sudo apt-get install libgtk2.0-dev
sudo apt-get install libatlas-base-dev gfortran
```

Step 4: Install necessary video packages

```
sudo apt-get install libavcodec-dev libavformat-dev libswscale-dev libv4l-dev
```

Step 5: Install pip, a Python package manager

```
wget https://bootstrap.pypa.io/get-pip.py
sudo python get-pip.py
```

Step 6: Install virtualenv and virtualwrapper. The two packages allow users who want to create a separate Python environment for each project that the user is working on so that the project doesn't interfere each other(not required but useful)

```
sudo pip install virtualenv virtualenvwrapper
sudo rm. -rf ~/.cache/pip
```

To use the two packages ~/.bashrc file is required to be updated, this file is hidden under the ~(root) directory, add these line in the file and save:

```
# virtualenv and virtualenvwrapper
export WORKON_HOME=$HOME/.virtualenvs
source /usr/local/bin/virtualenvwrapper.sh
```

Back in the terminal, reload ./bashrc file by: source ~/.bashrc

Virtual environment can be created using: mkvirtualenv name
The "name" can be changed to what ever the user wants

Step 7: Install Python 2.7 development tools and other required Python packages

```
sudo apt-get install python2.7-dev
pip install scipy
pip install -U scikit-image
```

Step 8: Install NumPy, an OpenCV Python bindings used to represent image as multi-dimensional arrays

```
pip install numpy
```

Step 9: In the root directory, download OpenCV and install it

cd ~ (if not in root directory)
wget -O opencv-2.4-11.zip http://sourceforge.net/projects/opencvlibrary/files/opencv-unix/2.4.11/opencv-2.4.11.zip/download
unzip opencv-2.4.11.zip
cd opencv-2.4.11
mkdir build
cd build

```
cmake -D CMAKE_BUILD_TYPE=RELEASE -D CMAKE_INSTALL_PREFIX=/
usr/local -D BUILD_NEW_PYTHON_SUPPORT=ON -D
INSTALL_C_EXAMPLES=ON -D INSTALL_PYTHON_EXAMPLES=ON -D
BUILD_EXAMPLES=ON ..
make -j4
sudo make install
sudo ldconfig
```

Step 10: Sym-link OpenCV to virtual environment, note from step 6, the name of the virtual environment created is "name". ***This has to be changed to the user's name****.*

```
cd ~/.virtualenvs/name/lib/python2.7/site-packages/
ln -s /usr/local/lib/python2.7/site-packages/cv2.so cv2.so
```

Step 11: Install Boost Python, a C++ library required by dlib library

```
sudo apt-get install libboost-all-dev
```

Step 12: Install dlib library

Download dlib library from its website at: dlib.net
At the moment of writing this report, April 2016, dlib's version is 18.18
Then back in terminal, enter the following command line:

```
tar xvjf dlib-18.18.tar.bz2
cd dlib-18.18/python_examples/
mkdir build
cd build
cmake ../../tools/python
cmake --build . --config Release --target install
cd ..
```

Installation should not be finished, like OpenCV, dlib also has to be sym-link with the virtual environment by:

```
ls -l dlib.so
mv dlib.so ~/.virtualenvs/gurus/lib/python2.7/site-packages/
```

Step 13: To work on the virtual environment, the user must use, every time the user login to the Operating System —> workon name ("name" is the name of your virtual environment)

Appendix B: Installing development environment on Raspbian

Since Raspbian, Raspberry Pi's official operating system, is based on Linux just like Ubuntu, installing development uses the same steps mentioned in Appendix A. But, in **Step 1** of Appendix A there is a slight addition. The installation guide can be referred to at [75].

When installing the development environment, make sure that the steps that involves installing virtual environment are skipped. This is crucial because installing virtual environment makes the Raspberry Pi GPIO pins incompatible.

Add "**sudo rpi-update**" command before the command:

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

to install Raspberry Pi update.

The reason that OpenCV version 2.4.11 is used, is because Raspbian Wheezy, the OS used in this project does not support OpenCV 3.0.

In order to use the camera, it must be enabled by in the Raspberry Pi's LX Terminal by typing:

sudo raspi-config

Entering this commands then shows the configuration that users can adjust, the camera has to be enable by selecting "Enable camera" menu and a reboot is required to apply any changes made in the configuration menu.

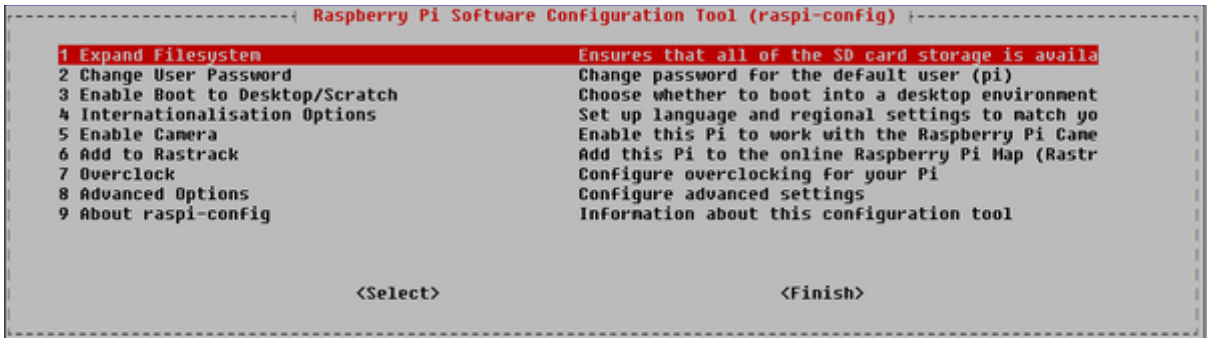

Figure A.1: Raspberry Pi's configuration menu

Appendix C: Using dlib's imglab tool to create bounding boxes

From Chapter 4, in order to train the object detector with the user's own positive samples, an image annotation is required. Fortunately, dlib provides us with a tool that is easy to use called imglab tool. This tool is included when we install dlib library in Appendix A, but in order to use the tool, it needs to be install as well. The installation and usage guide can also be found in the README.txt in the imglab folder inside dlib's folder. The steps to create bounding box for user's own samples are as follows:

Step 1: Install imglab tool

Go to the terminal, and enter the following command line:

```
cd dlib-18.18/tools/imglab
mkdir build
cd build
cmake ..
cmake --build . --config Release
```

Step 2: Running imglab tool

In order to run the imglab tool, two command line arguments need to supply into the terminal, ***when the directory is still in dlib18.18/tools/imglab/build***.

The first argument in the first command line is the output annotation file that has the bounding boxes of all the images that the user will manually draw. The second argument is where the positive samples is located.

```
./imglab -c ~/Desktop/catdoll_annotations.xml ~/Desktop/Cats
```

The second command line needed to launch imglab's GUI to start annotating is:

```
./imglab ~/Desktop/catdoll_annotations.xml
```

After entering the last command line argument, a GUI will appear on the screen. In this GUI, the user can start manually draw bounding box on each positive images. Figure A.2 shows an example of the GUI of imglab.

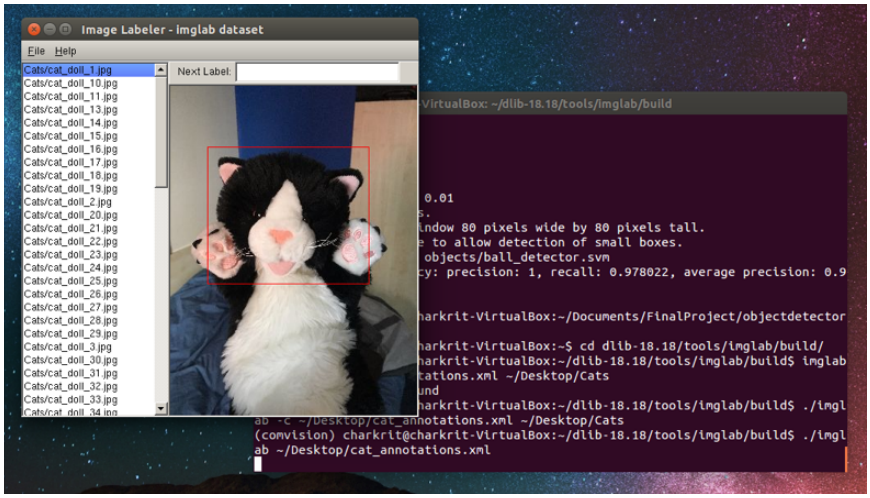

Figure A.2: Example of imglab tool's GUI

Note that the program can be relaunched and continue drawing whenever the user wants to by entering just the second command line argument. The instruction to use the imglab tool are as follows:

To draw bounding box on an image:

Press and hold the "left shift button" and "left mouse button" and move the mouse to draw

To delete a bounding box:

Double-clicking on the line of the box will change the colour of the line to blue, when the colour had changed press "Delete" button

To Save annotations when completed (Figure A.3):

Click on 'File' --> 'Save'

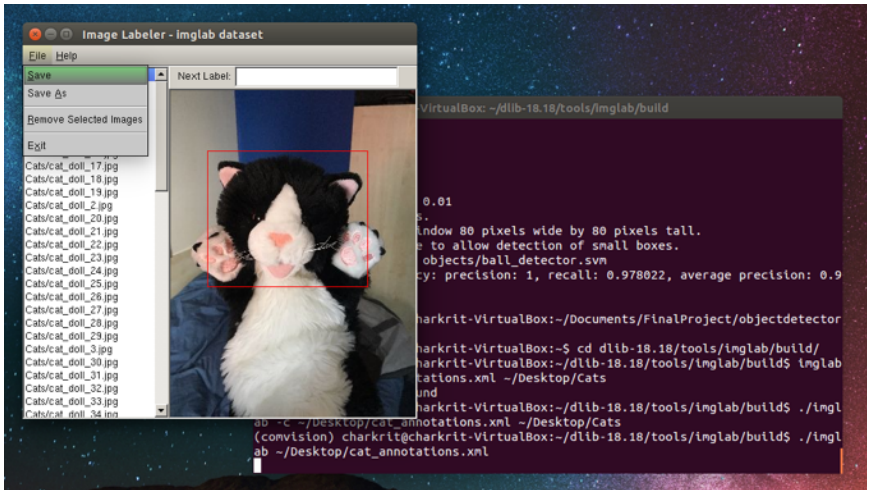

Figure A.3: Saving bounding box annotations

References

[1] : Catflaps.co.uk, (2015). *Microchip Magnetic and Manual Cat Flaps | CatFlaps.co.uk*. [online] Available at: http://www.catflaps.co.uk/Cat-Flaps [Accessed 23 Oct. 2015].

[2] : Nt.petplanet.co.uk, (2015). *Cat flap image*. [online] Available at: http://nt.petplanet.co.uk/images/product_images/extra_images/55537/20_Sureflap.jpg [Accessed 25 Oct. 2015].

[3] :Greenwood, T. (2015). *Creating a Raspberry Pi Cat Detector*. [online] Projectpi.org.uk. Available at: http://projectpi.org.uk/creating-a-raspberry-pi-cat-detector/ [Accessed 26 Oct. 2015].

[4] : [Wildfire Tech PR Blog, (2014). *From educational programming to cat flap spying: How the Raspberry Pi is looking out for pets | Wildfire Tech PR Blog*. [online] Available at: http://blog.wildfirepr.com/2014/01/programming-cat-flap-raspberry-pi/ [Accessed 26 Oct. 2015].

[5] "Flo Control Project", Quantumpicture.com. [Online]. Available: http://www.quantumpicture.com/Flo_Control/flo_control.htm. [Accessed: 02- Apr- 2016].

[6] "Monitor Grayscale Test Image", Drycreekphoto.com. [Online]. Available: https://www.drycreekphoto.com/Learn/Calibration/monitor_gradient.htm. [Accessed: 04- Apr- 2016].

[7] "RGB Colour Wheel", Adobe. [Online]. Available: https://color.adobe.com/build2.0.0-buildNo/resource/img/kuler/color_wheel_730.png. [Accessed: 04- Apr- 2016].

[8] J. Foley and A. Van Dam, Fundamentals of interactive computer graphics. Reading, Mass.: Addison-Wesley Pub. Co., 1982.

[9] Ubergizmo. [Online]. Available: http://cdn2.ubergizmo.com/wp-content/uploads/2014/06/1920-pixels-vs-3840-pixels.jpg. [Accessed: 04- Apr- 2016].

[10] "ABOUT | OpenCV", Opencv.org. [Online]. Available: http://opencv.org/about.html. [Accessed: 04- Apr- 2016].

[11] "Object Detection - MATLAB", Mathworks, 2016. [Online]. Available: http://uk.mathworks.com/discovery/object-detection.html?requestedDomain=www.mathworks.com. [Accessed: 05- Apr- 2016].

[12] P. Viola and M. Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision, vol. 57, no. 2, pp. 137-154, 2004.

[13] IEEE Computer Society Conference on Computer Vision and Pattern Recognition., N. Dalal and B. Triggs, Proceedings. Los Alamitos ; Tokyo: IEEE Computer Society, 2005, pp. 886 - 893.

[14] P. Felzenszwalb, R. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 32, no. 9, pp. 1627-1645, 2010.

[15] Li Fei-Fei, R. Fergus and P. Perona, "One-shot learning of object categories", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28, no. 4, pp. 594-611, 2006.

[16] C. Anderson, J. Bergen, E. Adelson, P. Burt and J. Ogden, Pyramid methods in image processing, 1st ed. Princeton: RCA Corporation, 2016, pp. 33 - 41.

[17] "Image Pyramid", Upload.wikimedia.org, 2015. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/4/43/Image_pyramid.svg. [Accessed: 07- Apr- 2016].

[18] A. Rosebrock, "Sliding Windows for Object Detection with Python and OpenCV", PyImageSearch, 2015. [Online]. Available: http://www.pyimagesearch.com/2015/03/23/sliding-windows-for-object-detection-with-python-and-opencv/. [Accessed: 08- Apr- 2016].

[19] C. Cortes and V. Vapnik, "Support-vector networks", Machine Learning, vol. 20, no. 3, pp. 273, 1995.

[20] E. Alpaydin, Introduction to machine learning. Cambridge, Mass.: MIT Press, 2004, p. 224.

[21] D. King, Max-margin Object Detection, 1st ed. 2015, pp. 1 - 8.

[22] D. King, "dlib C++ Library: Dlib 18.6 released: Make your own object detector!", Blog.dlib.net, 2014. [Online]. Available: http://blog.dlib.net/2014/02/dlib-186-released-make-your-own-object.html. [Accessed: 09- Apr- 2016].

[23] "Raspberry Pi - Teach, Learn, and Make with Raspberry Pi", Raspberry Pi. [Online]. Available: https://www.raspberrypi.org/. [Accessed: 10- Apr- 2016].

[24] K. Partner, Raspberry pi for beginners. London: Dennis Publ., 2013.

[25] "Raspberry Pi 2 Schematic", Jameco.com. [Online]. Available: http://www.jameco.com/Jameco/workshop/circuitnotes/raspberry_pi_circuit_note_fig2a.jpg. [Accessed: 10- Apr- 2016].

[26] "Welcome to Python.org", Python.org, 2015. [Online]. Available: https://www.python.org/. [Accessed: 10- Apr- 2016].

[27] "Cat Flap", Amazon.co.uk, 2016. [Online]. Available: http://www.amazon.co.uk/s/ref=nb_sb_noss_1?url=search-alias%3Daps&field-keywords=cat+flap. [Accessed: 11- Apr- 2016].

[28] "TeckNet® C016 USB HD 720P Webcam", Amazon.co.uk, 2016. [Online]. Available: http://www.amazon.co.uk/TeckNet%C2%AE-C016-Webcam-MegaPixel-Microphone/dp/B000Q3VECE/ref=sr_1_3?s=computers&ie=UTF8&qid=1459983266&sr=1-3&keywords=usb+webcam. [Accessed: 11- Apr- 2016].

[29] "Comparison of RasPiCam and Pi NoIR output in daylight", RasPi.TV, 2013. [Online]. Available: http://raspi.tv/2013/pinoir-whats-it-for-comparison-of-raspicam-and-pi-noir-output-in-daylight. [Accessed: 11- Apr- 2016].

[30] "USB cameras (webcams) on Raspberry Pi", Raspberrypi.org, 2013. [Online]. Available: https://www.raspberrypi.org/forums/viewtopic.php?t=55798. [Accessed: 11- Apr- 2016].

[31] "PIR potentiometer", Raspberrypi.org, 2016. [Online]. Available: https://www.raspberrypi.org/learning/parent-detector/worksheet/. [Accessed: 11- Apr- 2016].

[32]"GPIO: Models A+, B+, Raspberry Pi 2 B and Raspberry Pi 3 B - Raspberry Pi Documentation", Raspberrypi.org, 2016. [Online]. Available: https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/. [Accessed: 12- Apr- 2016].

[33] "Raspberry Pi2 – Power and Performance Measurement", RasPi.TV, 2015. [Online]. Available: http://raspi.tv/2015/raspberry-pi2-power-and-performance-measurement. [Accessed: 12- Apr- 2016].

[34] "Duracell MN1500 Datasheet", Duracell, 2016. [Online]. Available: http://professional.duracell.com/downloads//datasheets/product/Plus%20Power/Plus_AA_MN1500.pdf. [Accessed: 13- Apr- 2016].

[35] "TIP120 Datasheet", Farnell, 2005. [Online]. Available: http://www.farnell.com/datasheets/296713.pdf. [Accessed: 13- Apr- 2016].

[36] "Football 1, Random Google image". [Online]. Available: http://www.esportsbets.com/968/dota-2-csgo-scii-tournaments-esports-betting-markets/. [Accessed: 07- May- 2016].

[37] "Tennis Ball Water Splash, Random Google image", Wallpaperspal.com, 2014. [Online]. Available: https://wallpaperspal.com/tennis-ball-water-splash-background/. [Accessed: 07- May- 2016].

[38] "Volleyball 1, Random Google image", Whatsupdetroit.com. [Online]. Available: http://whatsupdetroit.com/in-the-d-summer-outdoor-sports/. [Accessed: 07- May- 2016].

[39] "Volleyball 2, Random Google image", Eurosport, 2015. [Online]. Available: http://tr.eurosport.com/voleybol/kadinlar-voleybol-ligi/2014-2015/antrenor-murat-yedidag-ile-idmanocagi-nin-yollari-ayrildi_sto4643138/story.shtml. [Accessed: 07- May- 2016].

[40] "Volleyball 3, Random Google image", Volleyball. [Online]. Available: http://courtvolleyball.weebly.com/equipment.html. [Accessed: 07- May- 2016].

[41] "Volleyball 4, Random Google image", Pix-hd. [Online]. Available: http://pix-hd.com/olympic+beach+volleyball+players. [Accessed: 07- May- 2016].

[42] "Volleyball 5, Random Google image", World of Volley. [Online]. Available: http://www.worldofvolley.com/News/Latest_news/Othercountries/67294/volleyball-revolution-new-rule--team-consists-of-7-players.html. [Accessed: 07- May- 2016].

[43] "Volleyball 6, Random Google image", Borongaja.com. [Online]. Available: http://www.borongaja.com/665245-japan-volleyball.html. [Accessed: 07- May- 2016].

[44] "Volleyball 7, Random Google image", VistaNews.ru. [Online]. Available: http://vistanews.ru/sport/43509. [Accessed: 07- May- 2016].

[45] "Baseball 1, Random Google image", Fullpermissionliving.blogspot.co.uk. [Online]. Available: http://fullpermissionliving.blogspot.co.uk/2014_03_01_archive.html. [Accessed: 07- May- 2016].

[46] "Baseball 2, Random Google image", Parachute.mapquest.com. [Online]. Available: https://parachute.mapquest.com/2016/03/24/mlb-opening-day-traditions-for-the-cincinnati-reds/. [Accessed: 07- May- 2016].

[47] "Baseball 3, Random Google image", Worldsports-c.com. [Online]. Available: http://worldsports-c.com/blog/page_602.html. [Accessed: 07- May- 2016].

[48] "Baseball 4, Random Google image", Pinterest. [Online]. Available: https://www.pinterest.com/rondakoch/baseball/. [Accessed: 07- May- 2016].

[49] "Basketball 1, Random Google image", Plus.google.com. [Online]. Available: https://plus.google.com/photos/photo/112614224930440945521/6271289366685818578. [Accessed: 07- May- 2016].

[50] "Basketball 2, Random Google image", Borongaja.com. [Online]. Available: http://www.borongaja.com/625213-famous-michael-jordan.html. [Accessed: 07- May- 2016].

[51] "Beach ball 1, Random Google image", Swimmingpool.com. [Online]. Available: http://www.swimmingpool.com/sites/default/files/beach-ball_0.jpg. [Accessed: 07- May- 2016].

[52] "Beach ball 2, Random Google image", Flickr - Photo Sharing!. [Online]. Available: https://www.flickr.com/photos/red11group/4912246740. [Accessed: 07- May- 2016].

[53] "Baseball 5, Random Google image", Plus.google.com. [Online]. Available: https://plus.google.com/104966692375076790805/posts/ABkE36scDBN. [Accessed: 07- May- 2016].

[54] D. Penkala, "Baseball 6, Random Google image", Kspiaseczno.jogger.pl. [Online]. Available: http://kspiaseczno.jogger.pl/2013/09/22/azja-wazniejsza-od-piaseczna/. [Accessed: 07- May- 2016].

[55] "Basketball 3, Random Google image", Ruv.is. [Online]. Available: http://www.ruv.is/sites/default/files/fr_20150825_021621.jpg. [Accessed: 07- May- 2016].

[56] "Tennis ball 1, Random Google image", Ycsports.com. [Online]. Available: http://www.ycsports.com/blog/wp-content/uploads/2015/01/cropped-tennis-stock-photo-1.jpg. [Accessed: 07- May- 2016].

[57] "Tennis ball 2, Random Google image", Orig04.deviantart.net. [Online]. Available: http://orig04.deviantart.net/1bc4/f/2008/195/6/1/61e1918320bcd244849afdf04984e228.jpg. [Accessed: 07- May- 2016].

[58] "Tennis ball 3, Random Google image", I.kinja-img.com. [Online]. Available: http://i.kinja-img.com/gawker-media/image/upload/s---N_TRdUI--/18k1nicpblbl9jpg.jpg. [Accessed: 07- May- 2016].

[59] "Football 2, Random Google image", S.yimg.com. [Online]. Available: https://s.yimg.com/fz/api/res/1.2/fQTGOgjAsawhDKa21uPVdQ--/YXBwaWQ9c3JjaGRkO2g9MTM1NDtxPTk1O3c9MjA0OA--/http://graphics8.nytimes.com/images/2013/07/17/sports/soccer/17iht-soccer17a/17iht-soccer17a-superJumbo.jpg. [Accessed: 07- May- 2016].

[60] "Football 3, Random Google image", S.yimg.com. [Online]. Available: https://s.yimg.com/uu/api/res/1.2/C6Nm6ADYq.g7OfzrH79IsQ--/aD0yMTI0O3c9MzA0NztzbT0xO2FwcGlkPXl0YWNoeW9u/http://media.zenfs.com/en_us/News/afp.com/par7292858.jpg. [Accessed: 07- May- 2016].

[61] "Basketball 3, Random Google image", G02.a.alicdn.com. [Online]. Available: http://g02.a.alicdn.com/kf/HTB1t0zDLXXXXXc2XXXXq6xXFXXXN/Official-Size-5-Outdoor-Indoor-Rubber-Pelota-Basketball-Ball-Training-For-Baby-Child-Children-Teenage-Gift.jpg. [Accessed: 07- May- 2016].

[62] "Tennis ball 4, Random Google image", Stockarch.com. [Online]. Available: http://stockarch.com/files/12/07/ball_and_racquet_0.jpg. [Accessed: 07- May- 2016].

[63] "Tennis ball 5, Random Google image", Elkvalleytimes.com. [Online]. Available: http://www.elkvalleytimes.com/wp-content/uploads/2013/03/Tennis-Racket-and-Balls.jpg. [Accessed: 07- May- 2016].

[64] "Tennis ball 6, Random Google image", Pixabay.com. [Online]. Available: https://pixabay.com/static/uploads/photo/2016/01/26/15/03/tennis-ball-1162640_640.jpg. [Accessed: 07- May- 2016].

[65] "Football 4, Random Google image", Sites.google.com, 2016. [Online]. Available: https://sites.google.com/site/yuliesportfolio/_/rsrc/1391030752381/config/soccer-ball.jpg.1391030752158.jpg. [Accessed: 07- May- 2016].

[66] "Bead balls image", Mainewoodconcepts.com. [Online]. Available: http://www.mainewoodconcepts.com/image_upload/Beads_Balls/Painted_Round_Wood_Beads___Large.jpg. [Accessed: 07- May- 2016].

[67] "Ball test, random Google image", Pinterest. [Online]. Available: https://www.pinterest.com/pin/486881409691803052/. [Accessed: 07- May- 2016].

[68] "Chocolate ball, random Google image", chocolate and connie, 2015. [Online]. Available: http://www.chocolateandconnie.com/melting-chocolate-orbs. [Accessed: 07- May- 2016].

[69] "Classes — dlib documentation", Dlib.net. [Online]. Available: http://dlib.net/python/index.html#dlib.train_simple_object_detector. [Accessed: 13- Apr- 2016].

[70] "OpenCV Windows Installation", OpenCV, 2016. [Online]. Available: http://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install.html#cpptutwindowsmakeown. [Accessed: 14- Apr- 2016].

[71] A. Rosebrock, "Increasing Raspberry Pi FPS with Python and OpenCV", PyImageSearch, 2015. [Online]. Available: http://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/. [Accessed: 15- Apr- 2016].

[72] A. Williams, "Raspberry Pi 3 vs Pi 2: What's the difference?", TrustedReviews, 2016. [Online]. Available: http://www.trustedreviews.com/opinions/raspberry-pi-3-vs-pi-2. [Accessed: 15- Apr- 2016].

[73] A. Rosebrock, "Install OpenCV 3 and Python 2.7+ on Ubuntu", PyImageSearch, 2015. [Online]. Available: http://www.pyimagesearch.com/2015/06/22/install-opencv-3-0-and-python-2-7-on-ubuntu/. [Accessed: 16- Apr- 2016].

[74] A. Rosebrock, "Install OpenCV 3.0 and Python 2.7+ on OSX", PyImageSearch, 2015. [Online]. Available: http://www.pyimagesearch.com/2015/06/15/install-opencv-3-0-and-python-2-7-on-osx/. [Accessed: 16- Apr- 2016].

[75] A. Rosebrock, "Install OpenCV and Python on your Raspberry Pi 2 and B+", PyImageSearch, 2015. [Online]. Available: http://www.pyimagesearch.com/2015/02/23/install-opencv-and-python-on-your-raspberry-pi-2-and-b/. [Accessed: 16- Apr- 2016].