Camera Movement Sensor with Object Detection

Low-Cost Solutions for Home Security

Prepared by:

Taqi Enari Syed

ENRMOH001

Department of Electrical Engineering
University of Cape Town

Prepared for:

Dr Francois Schonken

Department of Electrical Engineering
University of Cape Town

November 2021

Submitted to the Department of Electrical Engineering at the University of Cape Town in partial fulfilment of the academic requirements for a Bachelor of Science degree in Mechatronics Engineering.

Abstract

In this work we describe a novel Thesis Template, to be used by students in Electrical \& Engineering at the University of Cape Town. This section entails the abstract of the document.

Commented [T1]:

Commented [T2R1]:

Acknowledgements

Make relevant acknowledgements to people who have helped you complete or conduct this work, including sponsors or research funders.

Plagiarism declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the IEEE convention for citation and referencing. Each contribution to, and quotation in, this final year project report from the work(s) of other people, has been attributed and has been cited and referenced.
3. This final year project report is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as their own work or part thereof

Commented [RAV3]: You are welcome to use this document as a guideline and change the heading and font formatting, as well as naming of some of your chapters to more relevant names.

However you must keep the same basic flow of the document as well as NOT change the cover page or declaration which must be standard across the department.

Please complete your ethics form at the beginning of the project and check with your supervisor whether you need ethics clearance or not. If you answered NO to all questions on the first page you may delete all the pages following it.

To delete this comment box, right click on the comment box and click delete comment

Name: **Taqi Enari Syed**

Signature: 

Date: 01 November 2021

Table of Contents

Table of Nomenclature

In-text term	Term
GPU	Graphics Processing Unit
CPU	Computer Processing Unit
ML	Machine Learning
TL	Transfer Learning
DL	Deep Learning
AI	Artificial Intelligence
VM	Virtual Machine
CV	Computer Vision
BS	Background Subtraction
MID	MinMax Interframe Difference
1-G	One Gaussian
PDF	Probability Density Function

1. Introduction

1.1 Context

Crime is an issue that has long plagued humanity since the inception of society and human conscience. Despite the evolution of humans in the domain of morality and how members of society interact with one another, crime, unlike some diseases has not been eradicated yet. To remove crime at its roots is an issue that far goes beyond the realm of engineering, much less what this project entails. Thus, this project rather aims to provide a solution to counteract one of the more common forms of crime in home invasion and general trespassing of private property through means of fortifying monitoring systems. This perspective is especially pertinent given the fact that this project is being conducted in South Africa. 2020 saw reports of 1.2 million incidences of home invasions. This impacted 891'000 households which accounts for 5.3% of all households in South Africa [1]. This statistic is staggering and highlights the issues faced by people in this country.

With regards to the industry of motion detectors and object detection cameras, these devices are notorious for their inconsistent performance when deployed in field. Often, hypersensitivity seems to be an issue for these devices which gets alleviated by lowering the device's sensitivity. Of course, this leads to misdetections and false negatives which in the context provided above, could mean not alarming the homeowners to a potential trespasser and thus be a detriment to the user's safety. These security devices also aren't guaranteed to have been designed with the best technology as often corporations are more concerned with preserving a profit margin rather than providing the absolute best for their consumers. This outlook on the product alongside the challenges in implementing motion detection/object detection technology leads to the aforementioned issues.

1.2 Motivation

Engineering is frequently thought of as being a discipline in which one applies scientific knowledge to solve a real-world problem. This reduction, however, negates the humanity of the engineer. Engineering is more so about solving problems faced by mankind for the betterment of humanity and this perspective is one that must be preserved for humans to progress and improve the world. Having seen the harrowing statistics regarding property crime in South Africa, one can easily see the demand for safeguarding measures to provide security for citizens. This project's success could potentially benefit people in ways that are unquantifiable in a monetary sense. From an engineering lens, this would mean bringing a technological solution to counteract the impact these crimes have on civilians. The goal of this project is to provide a low-cost solution for home monitoring camera systems so that homeowners have a reliable tool that will provide security to their property.

1.3 Problem statement

Homeowners cannot be expected to constantly attend to their home monitoring systems and thus security firms are often hired to oversee security. This of course, can be problematic due to the element of human error involved in the process which can compromise the security of the user. Thus, there needs to be some application of AI that can alert the owners to movement within their homes so that they may act as soon as possible. Alongside this comes an even more challenging task of identifying the disturbance as although motion had been detected, it is still important to determine whether the disturbance is one that the owners should be concerned about or not. This is where the problem of object detection must be addressed.

1.4 Objectives

The overall objective of this entire project is to provide ordinary citizens with a low-cost way of automating the monitoring of their homes using some motion detector which includes classification of object functionality. This objective may seem ambiguous however, it may be broken down into 2 sub-objectives as follows:

1. To find a suitable algorithm through which motion detection can be implemented.
2. To find a suitable framework upon which to host the object detection algorithm.

1.5 Scope and limitations

The real-time nature of the motion detector in this project requires significant computational resources and thus to implement this on a micro-electronics system is not feasible. This forms a limitation on portability and size of the system as the camera will now require a desktop PC in order to host the runtime of this project. The usage of Machine Learning in use cases such as the one outlined in this project often require a plethora of resources. Large, annotated datasets are directly proportional to the efficacy of these kinds of projects [2]. Once hurdling past the challenge of ethically sourcing an appropriate dataset for one's uses, the next challenge is procuring the necessary hardware to run the computation on. This is often a task given to the Graphics Processing Unit in computers [2]. Modern GPUs hold higher computational power than CPUs which makes them ideal for applications of Machine Learning [2] however in recent months, the computer market has seen an unprecedented rise in GPU prices due to a multitude of factors and thus the acquisition of a high-end GPU has been unsuccessful for the training aspect of this project. Due to the lack of computational power available, the size of the dataset was also forced to be smaller to allow training to finish within the timeframe of this project. Testing the system is another limitation. Due to the ethical issues that arise in using people in detection tasks, all the practical testing will be conducted on myself. The model that will be used for training will be a pre-trained model to make use of Transfer Learning and thus the design of creating a model from scratch is not part of the scope.

1.6 Plan of Development

The next chapter will discuss the literature relevant to this project. This will include investigation into previous attempts at solving problems analogous to the one presented within this project. Papers concerning Motion Detection algorithms, implementations of classifiers, comparisons of various AI frameworks, research of types of data to train on, tests on various models and choice of parameters are all themes to be discussed in the literature review chapter. Once all the literature has been unpacked and put into context, the next course of action will be the methodology and design section. This section deals with the design choices as well as the justification behind these design choices.

2. Literature Review

2.1 Overview

The usage of ML in detecting motion and classifying humans has been a progressive field as better and faster methods have risen to the surface. Detection of simple objects i.e., objects with a static build tend to get detected faster due to the AI recognizing the objects with more ease. The detection of humans, however, is a very challenging task. With all the methods presented by various researchers, they all seem to have their drawbacks when attempting to conduct human/person detection tasks [3]. The trend seems to be using ML methods to train networks that are either pre-trained (Transfer Learning) or creating a custom network. The necessary complexity of approach is due to the inherent complexity of a human being in that the form of a human differs. Humans are objects that have different colours (skin colour, clothes, shoes, caps etc) as well occupying various poses (standing, sitting, or moving). Thus, to train a network to identify a person is a difficult task and an appropriate approach to the problem must be taken. This project is multi-faceted and thus literature concerning each element of the project will be discussed. The two main functionalities required is motion detection as well as object detection. This means that the system should be able to correctly detect motion and thereafter be able to correctly classify the cause of motion.

2.2 Methods to Implement Motion Detection

The motion detection functionality is the first of the two required for this project. One of the issues with modern home security sensors are the inconsistent detections picked up the sensors and the aim of this project, as previously stated, is to improve the detection accuracy. This functionality can be implemented in various ways. The focus of this section is the concept of background subtraction and algorithms that deal with background subtraction. Benezeth et al. [4] discusses the various algorithms of background subtraction in great depth, however, for the sake of keeping the report concise, only the 3 fastest algorithms will be discussed. Background subtraction is often seen as a quick method to localize moving objects in a static frame. In the context of this project, that would mean detecting a moving object from the viewpoint of a static camera. Due to the lack of training involved in a CV task such as this, execution times are often fast even on computationally limited machines. Due to these factors, it makes sense to try incorporating CV techniques like background subtraction to implement the motion detection functionality of the project rather than the more computationally taxing ML methods.

The first method is a basic form of BS. It consists of modelling the background as a single image devoid of any moving objects. Should an object in the frame move, the pixel values of the object will differ to the pixel values of the background. A ‘movement’ threshold can be set and using any of the 4 distance equations seen in [4] provided below, movement can be detected by comparing the distance function to the threshold value.

$$d_0 = \left|I_{s,t} - B_{s,t}\right| \quad \textbf{[4, eq (3)]}$$

$$d_1 = |I^R_{s,t} - B^R_{s,t}| + |I^G_{s,t} - B^G_{s,t}| + |I^B_{s,t} - B^B_{s,t}| \quad \textbf{[4, eq (4)]}$$

$$d_2 = \left(I^R_{s,t} - B^R_{s,t}\right)^2 + \left(I^G_{s,t} - B^G_{s,t}\right)^2 + \left(I^B_{s,t} - B^B_{s,t}\right)^2 \quad \textbf{[4, eq (5)]}$$

$$d_\infty = \max\left\{|I^R_{s,t} - B^R_{s,t}|, |I^G_{s,t} - B^G_{s,t}|, |I^B_{s,t} - B^B_{s,t}|\right\} \quad \textbf{[4, eq (6)]}$$

$I_{s,t}$ is the present frame at time *t* and pixel value *s*. $B_{s,t}$ denotes the background frame at time *t* and pixel value *s*. The R, G and B superscripts stand for Red, Green and Blue respectively. d_0 in [4, eq (3)] is the equation that is used in grey-scale images. All distance functions are compared to some threshold value to determine whether a pixel change has occurred which then defines motion. However, one can opt to only use one of the four distance functions.

Another method that is appropriate for this use-case is the MinMax Interframe Difference (MID) otherwise known as W^4 devised by Haritaoglu et al. [5]. Every background pixel is assigned a maximum value M_s, a minimum value m_s and a maximum value of the difference in consecutive frames D_s. The following governing equation can be seen in [4].

$$|M_s - l_{st}| < \tau d_\mu \text{ or } |m_s - l_{st}| < \tau d_\rho \quad \textbf{[4, eq (11)]}$$

τ is a defined value for the threshold and d_μ is the median of the largest interframe absolute difference over the entire image. This algorithm does not work on colour images and thus, is limited to grey-scale applications only. The d_μ is a significant addition because it requires a pixel in a noisy segment of the frame to change more than a pixel in a quieter section of the frame. This renders an augmented threshold value allowing MID to be more robust to when motion isn't happening in front of a static background.

The last method that will be covered is the One Gaussian algorithm. The one issue that the first BS algorithm suffers from is the non-uniform nature of the modelled background. That is, that although the still frame has been captured as the background, any changes in lighting conditions or any other element in the room may falsely trigger as motion. To counteract this inherent sensitivity, each pixel of the background frame is modelled as a Probability Density Function (PDF). A pixel with a low probability is likely to correspond to the pixel of a moving object. The robustness of this algorithm is further improved by Wren et al. in [6]. This is done by modelling every background pixel with distribution as seem below.

$$N(\mu_{s,t}, \Sigma_{s,t}) \quad \textbf{[4]}$$

Where $\mu_{s,t}$ and $\Sigma_{s,t}$ stand for the average background colour and pixel covariance matrix at pixel *s* and time *t*, respectively. The covariance matrix contains large values for areas of the background with noise and lower values for quieter areas of the frame. The distance function is computed as follows.

$$d_G = \frac{1}{2}\log\left[(2\pi)^3|\Sigma_{s.t}|\right] + \frac{1}{2}\left(1_{s.t} - \mu_{s.t}\right)\Sigma_{st}^{-1}\left(1_{s.t} - \mu_{s.t}\right)^7 \quad \textbf{[4, eq (7)]}$$

d_G is a logarithmic probability. The summation seen in [4, eq (7)] means that much like the MID algorithm, for a pixel to be categorized as 'in motion' it needs to be higher in value relative to the pixels situated in a noisier part of the background frame. This makes the 1-G algorithm robust to noise.

Comparing the 3 methods is simple enough as Benezeth et al. [4] were thorough with their testing. The testing included records of computational times on a Visual C++ compiler on a full dataset of videos. The average computation times relative to the basic BS algorithm were 1.32 and 1.47 for the 1-G and MID algorithms respectively [4].

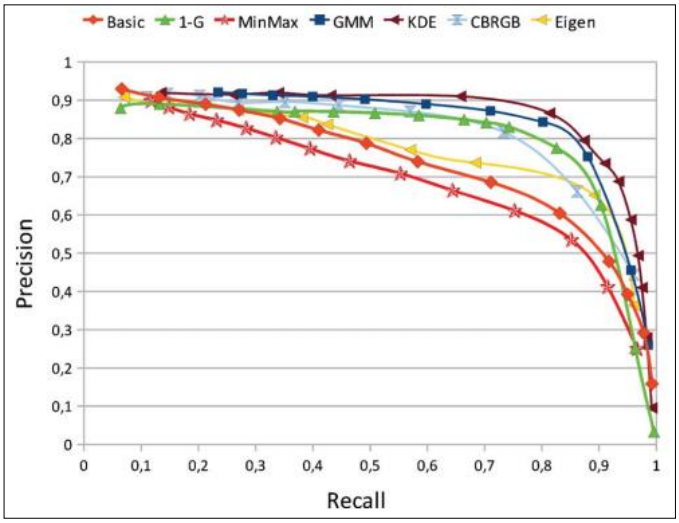

Figure 1. Precision vs Recall graphs for various algorithms. Sourced [4, Fig. 11]

Precision is defined as the ratio between true positives and all positives. Recall is a measure of the model correctly identifying true positives. Analysing the Precision vs Recall graphs for the various algorithms, it can easily be seen that the MID, despite having a relative computational time 32% longer [4] than the basic BS algorithm, underperforms relative to the basic BS algorithm at all recall values. The 1-G model outperforms both the MID and basic BS algorithms at nearly all recall values whilst having a computational time 47% higher than the basic algorithm [4]. This performance gain is something to consider when deciding on which algorithm to incorporate for the motion detection. It is conclusive that for the computational resources required, the MID model is the only one that doesn't seem to be a feasible option given its precision at various recalls. The basic model is a far lighter algorithm in a computational sense and due to the lack of colour treatment within the algorithm, makes it ideal for all types of lighting conditions [4]. It is pertinent to note that the motion detection part of this project is not responsible for object detection and thus one may see why a fast algorithm like the basic algorithm may be suitable for a task that does not require accuracy (like the 1-G algorithm), rather it requires a fast computation time for real-time implementation.

2.3 Methods to Implement Object Detection

Once the motion detection functionality of the system has been achieved, the next step is to investigate the best way of implementing the object detection functionality of the project. Object detection is a task that has become very popular with the rise of machine learning in recent years. The norm used to be to use human resources to identify objects within videos and images however, the need for automation of this task gave way to a multitude of methods of allowing a model to predict what it was being shown based on prior training. ML based methods of tackling these object detection tasks are methods that remains unrefined. In order to procure a usable accuracy, researchers often have to finetune the features they wish to detect as well as finding a large enough dataset to train on [7]. Deep Learning (DL) is a sub-field of ML that focuses on creating artificial network mirroring the structure of human neural networks. The network consists of layers with neurons contained within them through which a signal, modulated by a weight value, is passed through the layer [8]. CV is not just limited to algorithms such as background subtraction, as previously seen. It can also be used as a means to implement object detection. All 3 methods have their own advantages and disadvantages. DL is a subset of ML with improved adaptability on new sets of data [8] as well as being better suited for object detection as it outperforms ML algorithms in these sorts of tasks [9]. Thus, the focus of this section will be to study the merits of a DL approach compared to the merits of a traditional Computer Vision approach. The advantages of going through a DL approach rather than a traditional CV approach can be seen as follows.

Firstly, the improvement in computational devices and image sensors in recent years has allowed implementations of DL to be far more accurate than traditional computer vision techniques especially in the realm of image classification. This inherent supremacy over traditional CV techniques is due to the fact that the networks in a DL model are trained rather than programmed [8]. This quality also allows researchers to implement transfer learning where a model trained on one dataset can then be ‘re-trained’ on another dataset.

Secondly, traditional CV techniques often have algorithms that are specific to the use-case in which the algorithm is being deployed and thus this flexibility of application is unique to DL models. O’Mahony et al. [8]demonstrates these qualities by explaining how an object detection algorithm would work using traditional CV techniques. Traditional CV techniques make use of a function known as ‘feature extraction’ to recognize objects. A researcher defines the most defining features of the object they wish to have the model identify. The model then looks for these sets of features in the other data samples and samples where a significant number of features is detected, the model classifies the image as having the object the researcher wanted to detect [8]. The difficulty in this approach lies in defining what features to choose and this step is further complicated when researchers must classify more than one class of object.

Deep Learning is a more ‘black-boxed’ approach whereby images with annotations of what class if contained in the image, is inputted into the DL model. The model, through its neural layers, finds the defining features and patterns within the images to then predict what class the object in the image belongs to[8]. Based on the accuracy of this prediction relative to the provided annotations in the training data, the model then optimizes its network to be able to predict the correct classification with improved accuracy [9]. This eliminates the feature extraction stage of traditional CV methods and hence saves the researchers time as they do not have to trial the choice of features when devising their algorithms.

A DL approach isn’t always ideal however, as there can be times where the problem is simple enough that a traditional CV approach would solve the problem with the same accuracy but in less lines of code[8]. DL networks also require significantly more data for training than traditional CV techniques [2]. This can be especially difficult as the data must be annotated to categorize the class otherwise it is useless to a DL model. This can mean that researchers may not only have to rigorously sample a sufficient quantity of data but also

may have to manually annotate the images themselves which can be not only time consuming but introduce an unnecessary element of human error within the design process.

Another downfall of a DL model is that its decision-making process is hidden behind a complex neural network. Despite the model performing well in terms of accuracy metrics, it can't always be relied on as researchers may not be aware of how the model reaches its conclusion [2]. Furthermore, a DL model lacks 'common sense'. Where it excels at finding patterns within images and autonomously picking up class-defining features, it is unable to rationalize the data that has been classified [2]. Zohuri and Moghaddam in [2] emphasise this quality by providing the analogy of a DL model being able to correctly classify a sofa after being trained on a dataset of sofa images yet not being able to tell whether an object is suitable for sitting on.

Having discussed the literature above, one can easily see where the difficulties of using the two approaches may lie in this project. Whilst the lack of 'common sense' is a concerning feature of a DL model, the functionality of predicting whether an object is a threat to the user is beyond the scope of this project. Rather the focus is more on correctly classifying the object in the frame and thus given the advantages that DL has in the image classification domain, it can be seen why it is the more sensible approach for this project.

2.4 Analysis of Pre-Trained Models

As mentioned previously, the design of a model from scratch is beyond the scope of this project. A pre-trained model is a model trained by someone else for similar tasks that one can use to further train for their own purposes. In a TensorFlow (a widely used DL framework) context, a pre-trained model is a model that comes with a checkpoint file. Checkpoints are a type of file containing all model parameters and are generated at discrete stages of training thus allowing researchers to make use of transfer learning. TensorFlow offers researchers a well-renowned GitHub repository of pre-trained models known as the TensorFlow Model Zoo [10]. This repository contains models that have been trained on the Common Objects in Context (COCO) 2017 dataset. Using pre-trained models is a cornerstone of all modern deep learning applications [11]. The focus of this section is investigating the best pre-trained model for the purpose of this project. However, TensorFlow is not the only framework that offers a wide array of pre-trained networks. Bochkovskiy et al. [12] hosts a famous GitHub repository of pre-trained YOLO networks that can be trained on the Darknet framework curated by Joseph Redmon [13]. These 2 frameworks offer researchers much required flexibility in choosing frameworks upon which to develop models on.

Single Shot Detectors (SSD) are object detection models that only require an image and an annotative file containing data about a bounding box that bounds the class contained within the image. The SSD itself doesn't make up the entire model, it is simply at the output layer of another base neural network. An image is fed to the 'backbone' network from which the SSD extracts feature maps from to then detect objects [14]. SSDs work by generating multiple bounding boxes of various aspect ratios centred on each pixel of the extracted feature map. The SSD model makes use of a feed-forward neural structure [3] which means that data only moves in the direction of the output nodes in the network. There is no feedback loop in such networks.

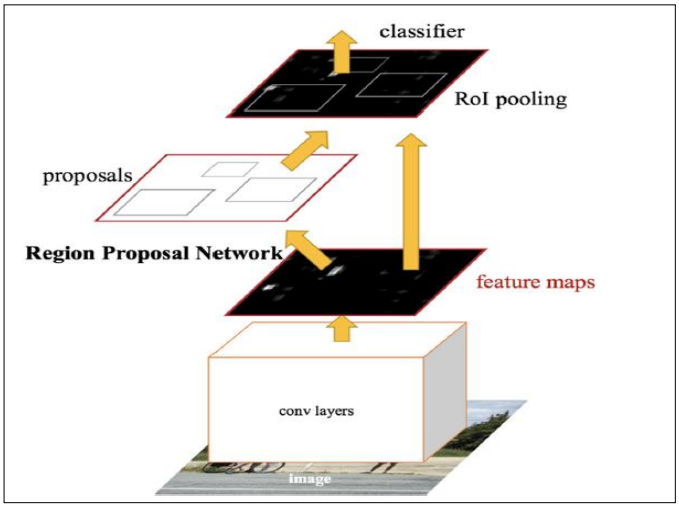

Figure 2. Architecture of R-CNN model. Sourced from [3, Fig. 13]

This is akin to the structure seen in Recurrent Neural Networks (RNN) [3] which is an older architecture however SSD makes an improvement to these architectures by removing the Region Proposal Network (as seen in Figure 2) and adding in default boxes and additional feature layers [3].

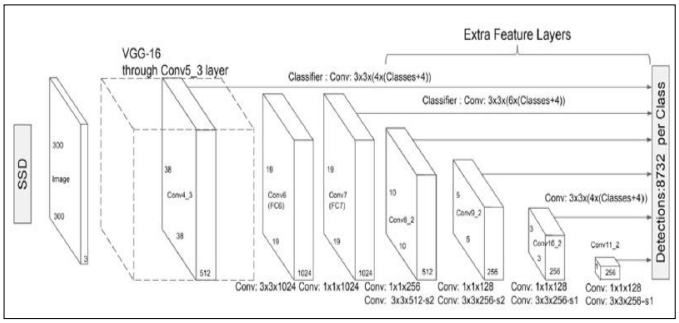

Figure 3. An SSD model with a VGG-16 backbone. Sourced from [3, Fig. 14]

The SSD model generally makes use of a VGG-16 backbone network however instead of making use of the VGG-16's fully connected layers, the SSD replaces those with its own feature layers allowing for fast feature detection as the input size decreases as it passes through each layer of the SSD model. A further improvement can be made by replacing the 'vulnerable' VGG-16 backbone [3] with a ResNet backbone [15]. Lu et al. [15] conducted training on the ResNet model using images from Google's OpenImages dataset and the following improvements in accuracy of detections were seen.

Category	SSD	SSD-ResNet
Knife	0.445	0.696
Revolver	0.736	0.868
Long whip	0.757	0.810
Rifle	0.418	0.691
Bullet	0.531	0.662
Missile	0.477	0.695
Bow	0.486	0.647

Figure 3. Results showing accuracy improve after replacing VGG-16 backbone. Sourced from [15, Table. 2]

It can clearly be seen how much of an improvement to SSD models was made just by replacing the VGG-16 backbone. Averaging the SSD VGG-16 results renders an average accuracy of 55.0% whilst averaging the results of the SSD-ResNet renders 72.4%. Of course, these results are on a limited set of data inputs however, Lu et al. [15] quantifies the exact improvement as a 17.40% improvement but also notes that this improvement comes at the cost of computational time. This observation is concerning given that the deployment conditions of this project will be in real-time.

Another popular object detection model is the You Only Look Once (YOLO) algorithm. YOLO also makes use of an SSD however the algorithm works very differently. The input image is first split into a grid of equal width and height. Thereafter the convolutional layers generate bounding boxes of various aspect ratios in each of the grid sections. The convolutional layers then predict the probability of each of the boxes containing the target class and based on the distance from the box to the ground truth bounding box, refine the output bounding box [3]. Various improvements were made on this architecture through the YOLOv2, YOLOv3, and YOLOv4 editions. The focus will be on YOLOv4, a recently developed architecture that has the darknet-53 network as its backbone followed by a Path Aggregation Network (PANet) and then 3 YOLO heads at the output layer [16].

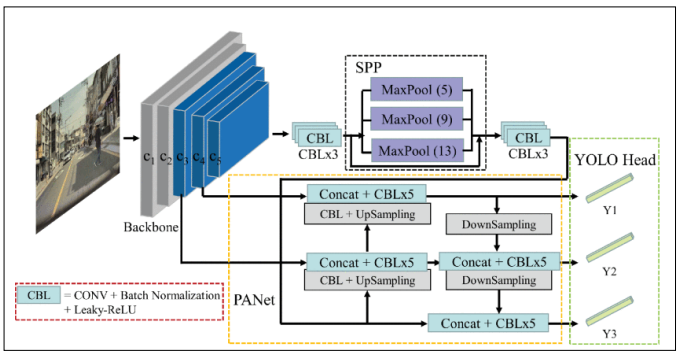

Figure 4. YOLOv4 architecture. Sourced from [16, Fig. 2]

YOLOv4 made improvements through the use of a new mosaic data augmentation method as well as adding in a new loss function called CIOU. The new loss function allows YOLOv4 to converge towards higher probability bounding box overlaps by optimizing weights in the neural network to get to these bounding box solutions more frequently and consistently. With this increase in accuracy, YOLOv4 requires more computational resources than its predecessors. YOLOv4 also struggles with extracting smaller features from

complex objects [17]. Despite these shortcomings, YOLOv4 still stands out for its performance to computational time which makes it ideal for real-time application.

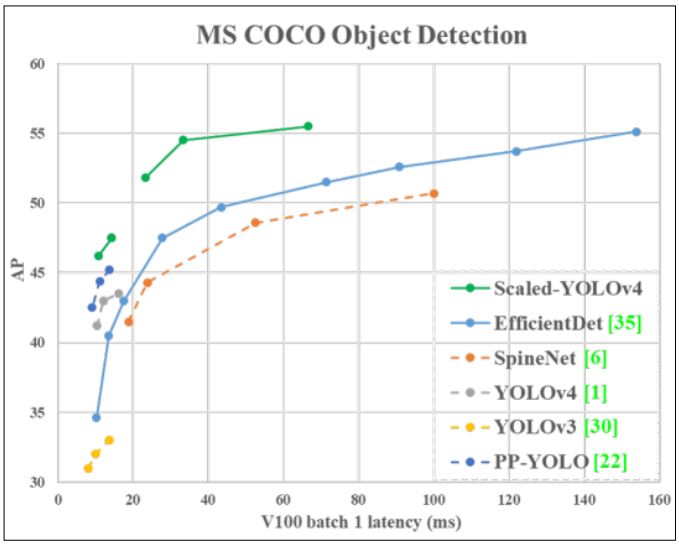

Figure 5. Average Precision plotted against batch latency. Sourced from [18, Fig. 1]

Wang et al. [18] plotted the above results shown in Figure 5. Various networks were trained on Microsoft's COCO dataset. As one may see, YOLOv4 maintains a competitive Average Precision (AP) value considering it holds such a low batch latency. More noticeably, YOLOv4's scaled version seems to outperform all the other networks with a slight penalty in batch latency. It can be seen that it achieves a peak AP value of approximately 55%. Model Scaling refers to the change of model parameters such as changing the number of convolutional layers in a network to optimise the performance of the network in some given tasks. Historically, this has proven to be beneficial as seen where researchers changed the number of kernels in the ResNet convolutional layer to create the Wide ResNet model (WRN) which proved to have a higher inference time than ResNet. Redmon, the creator of Darknet, alongside Farhadi in [19] saw how scaling the input image as it goes into YOLO's darknet53 backbone, affected the model's accuracy. In order to find whether model-scaling was done successfully, metrics such as training time, inference time and mAP values provide an insight to the scaled model's performance [12].

2.5 Datasets

Perhaps the most important part of the Deep Learning process is to feed the chosen network with the correct data. There are many ways of a researcher finding results that may not have been expected just due to the data their model was trained on. Person detection is the most complex part of this project. This is due to the fact that the DL model is expected to correctly identify a person within household video footage. Simply training a DL model on a dataset of human images is often not sufficient enough to acquire a reasonable accuracy. This is due to the fact that changes in illumination, colour of skin, clothes and other unique features of a human prevent DL models from finding consistent patterns to identify humans with especially in settings where people may be cluttered into groups [20].

Rodriguez et al. in [21] states that their choice of using the head of a human is due to the fact that in their deployment conditions (pedestrian counters) the other features of the human body such as their torso and legs are often blocked due to the cluttered nature of pedestrian footage.

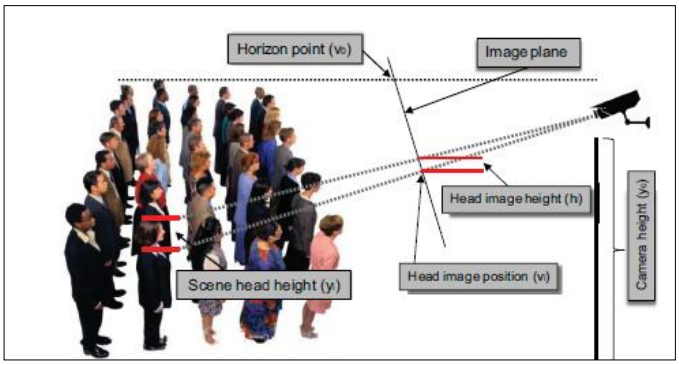

Figure 6. Estimate of camera geometry. Sourced from [21, Fig. 3]

As seen in Figure 6, one can easily see the perspective of Rodriguez et al. [21] as the camera can only see the full body image of the closest batch of humans. Saqib et al. in [20] went ahead and trained various models using annotated images of human heads and found the following results.

Detection frameworks	CNN architectures	mAP
DPM Face [16]		0.374
R-CNN [9]		0.671
Local model [24]		0.718
Local+Global+Pairwise model [24]		0.727
Faster R-CNN [18]	ZF [25]	0.765
	VGG16 [21]	0.791
	VGG_CNN_M_1024 [2]	0.791
YOLO v2 [17]	13-layered architecture	0.631
SSD MultiBox [13]	VGG16 [21]	0.788

Figure 7. Results of training on various networks. Sourced from [20, Table 3]

As seen in Figure 7, YOLOv2 trained on human heads managed to achieve an mAP value of 63.1%. This is not particularly impressive relative to the others' performance. However, the SSD on a VGG16 backbone achieved a mAP of 78.8%. Comparing this to the average accuracy it achieved in Figure 3, one can see a noticeable increase in accuracy when training on images of human heads. Considering that testing wasn't done on YOLOv4 with this dataset, one can reasonably extrapolate (given the results published by Bochkovskiy et al. [12]) that a YOLOv4 model would considerably outperform the YOLOv2.

2.6 Improvements

Having discussed relevant literature to the project, there are many areas where present work in this field may be improved. The following improvements should provide higher accuracy of motion detection as well as higher accuracy of object detection within the captured video frames. Improvement in this particular metric is especially important given the nature of this project being a home security real-time application.

The first improvement can be made to the motion detection algorithm. Benezeth et al. [4] discussed various background subtraction methods and all the algorithms discussed used a threshold value to decide whether a pixel belonged to the background or foreground. However, pixels can lighten up or darken depending on ambient lighting in the environment [4]. For this reason, I believe that modulating the threshold value with a variable dependant on time will effectively modulate the threshold at various times and thus provide a more consistent sensitivity throughout the day rather than heightened sensitivity at some lighting condition and then dampened sensitivity at another. This can be implemented through the use of Python's Astral library which is able to provide sunrise, noon, dusk and sunset times for various locations around the world.

With background subtraction methods, noise in the background causing misdetections and false positives is a detriment to performance. This can be alleviated by applying image filtering techniques to the image before it is passed through the background subtraction algorithm. The usage of a 1080p resolution IP camera in this project allows us to acquire a clear image but as the Basic background subtraction algorithm in [4] greyscales the background image and the present frame before finding the absolute difference, there are noise elements introduced into the process. These can be alleviated through the use of Gaussian-Blurs which is convolving the image with a kernel matrix of dimensions $n \times n$. This is the digital equivalent of passing the image through a low pass filter and thus eliminating high frequency elements of the image that can disturb the efficacy of the background subtraction algorithm [22].

For the Deep Learning object detection section, the use of Google's Colab VM allows researchers to conduct training on Google's machines and thus provides access to high performance GPUs. As discussed previously, stronger GPUs lead to faster training times and thus more training iterations can occur [2]. This also means that a more complex model can have the necessary resources it requires to detect objects in real-time.

Another improvement to the object detection algorithm will be the creation and usage of a custom dataset. Public datasets such as Google's OpenImages, Microsoft's COCO, and the PASCAL's VOC datasets provide millions of images of common objects and are annotated. However, to train a model on these datasets takes time due to the sheer amount of data contained within the datasets and can result in underfitting which would lead to detection accuracy diminishing [23]. To alleviate this issue, classes relevant to a home security use can be used to train the model. Any benefit from training a model on classes such as cars or aeroplanes cannot be seen as these objects do not appear in a home security video feed. Thus, by reducing the classes the model trains on, it can pick out features more efficiently and in a shorter space of time.

Due to the very particular use case of object detection in this project, the model scaling concept mentioned by Wang et al. [18] will allow the model to perform well for this project. The results seen in Figure 5 show a significant increase in accuracy with a relatively low computational penalty. Of course, this will further be optimized through the usage of a smaller customised dataset.

2.7 User and Functional Requirements

Given the objectives of the task, these can be translated into a set of User Requirements (UR). These would reflect the end-user's needs from this project. The User Requirements derived from the Objectives can be seen in Table 1.

Objective	UR	User Requirement
1	1	The system must detect when an intruder enters a room.
2	2	The system must be able to identify the intrusive body.

Table 1. User Requirements Table

Having acquired a set of User Requirements. The Functional Requirements to satisfy these User Requirements must be set.

UR	FR	Functional Requirement
1	1.1	The BS algorithm should be able to differentiate foreground elements and background elements.
	1.2	The algorithm should have a robust threshold variable that is dynamically able to distinguish background elements to foreign objects.
	1.3	The system should capture the image of the intrusion for further processing.
2	2.1	The system should identify the object in a short space of time.
	2.2	The system should be able to identify the object correctly.
	2.3	System should output the image with a bounding box around the intrusion.

Table 2. Functional Requirements Table

2.8 Acceptance Test Procedures

Having established the Functional Requirements to satisfy User Requirements, the necessary Acceptance Test Procedures (ATP) must also be set. The ATPs will ensure that all the user's needs have been accounted for and that the end product is ready for field deployment.

FR	AT	Acceptance Tests (AT)
1.1	1.1.1	Test to check whether the absolute pixel difference between background and foreground correlates to the actual difference in both images.
	1.1.2	Output pixel values for a background element and intrusive element to check that the system can quantitively differentiate between the two values.
1.2	1.2.1	
1.3		
2.1		
2.2		
2		The system must be able to identify the intrusive body.

3. Methodology and Design

3.1 Core Functionality

This project is divided into two functional subsystems. The first subsystem is the motion detector which establishes whether there is any occupation in the monitored environment. This segment makes use of background subtraction as well as various image processing techniques offered by Python's OpenCV library. Should the algorithm detect occupation within the environment it will pass the camera frame at the instance occupation was detected, to the object detection algorithm which will then determine what caused the disturbance.

Although the literature provided a guide as to how to implement these features

ADDENDUM 4: To be completed if you answered YES to Question 4

4.1 Is there any existing or potential conflict of interest between a research sponsor, academic supervisor, other researchers or participants?	YES	NO
4.2 Will information that reveals the identity of participants be supplied to a research sponsor, other than with the permission of the individuals?	YES	NO
4.3 Does the proposed research potentially conflict with the research of any other individual or group within the University?	YES	NO

If you have answered YES to any of these questions, please describe below how you plan to address these issues: