OBJECT TRACKING SYSTEM USING THE C++ DLIB LIBRARY AND THE YOLOV3 NEURAL NETWORK

By: Tafadzwa Brian Motsi 157900
Supervisors: Mr B. Matsebula
Prof B. Akinnuwesi

A project submitted to the Department of Computer Science as part of requirements for the award of a degree in Bachelor of Science

August 2020

University Of Eswatini

Faculty of Science and Engineering

Table of Contents

LIST OF FIGURES

LIST OF ABBREVIATIONS

CV	Computer Vision
HOG	Histogram of Oriented Gradients
CCTV	Closed-Circuit Television Camera
API	Application Program Interface
OBTS	Object Tracking System
EPTC	Eswatini Post Telecommunications Company
EEC	Eswatini Electricity Company
SVM	Support Vector Machine
GMM	Gaussian Mixture Modelling
EM	Expectation Maximization
D-CNN	Darknet Convolutional Neural Network
YOLOV3	You Only Look Once Version 3
R-CNN	Region-based Convolutional Neural Network
SSD	Single Shot MultiBox Detector
IBM	International Business Machines
UML	Unified Modeling Language
MMOD	Max-Margin Object Detection
PIR	Passive Infrared Sensors

CERTIFICATE

This is to certify that Tafadzwa .B. Motsi a student in the Department of Computer Science in the class of 2020 has successfully completed a research work titled Object Tracking System Using The C++ DLib Library and The YOLOV3 Algorithm. This was under the supervision of Mr B. Matsebula and Professor B. Akinnuwesi in partial fulfillment of the award of a degree in Bachelor of Science(B.Sc) at the University of Eswatini in August 2020.

Tafadzwa B. Motsi	Date
Mr B. Matsebula	Date
Prof B. Akinnuwesi	Date

DEDICATION

I dedicate this research work to my beloved parents Nelford J. Motsi and Thandiwe Chagwinya who have been anchors of my life throughout the years. Their hard work in supporting me leaves an ever enduring mark of love in my entire career. I also dedicate it to my three young brothers and my little sister, whom by looking up to me as the eldest brother have given me courage in this pursuit. Infinite thanks

ACKNOWLEDGEMENTS

I would like to extend my profound gratitude to all persons who in one way or the other have dedicated their efforts and time in supporting this work to become a reality.

Above all to Almighty God, for strength, wisdom, ideas, well-being and hope. For all Your mercy and love, Lord, my humble prayers of thanks.

To my dearest parents, for their unmatched support in making this study a reality. Without their funding and prayers throughout my journey, this project would have started and ended as imaginations. I cannot thank them more than enough.

To Mr. B Matsebula, for his supervision and guidance on this study. Despite his busy schedule, he has found quality time to give me direction and assistance during this endeavor. I can not thank him enough for his continual patience and understanding. I also thank him for his empowering and encouraging words. Working with him was an honor. I can only wish him well and many blessings.

To Prof. B Akinnuwesi, for his co-guidance in the physical absence of Mr. Matsebula. He has taken quality time to read through my work and has dedicated his experience and his work to guide my entire study. I would like to sincerely thank him for going out of his way in arranging our meetings, taking life threatening risks during the period of COVID-19 so that he could give all the necessary supervision. For his humility, counseling, passion in assisting, understanding, and patience regardless of his title, I thank him. I was honored to have been supervised by him. Finite words of thanks would not do justice in expressing my gratitude. I

can only wish him blessings and abundance of life.

To the Department of Computer Science, for a spirit of understanding. I would like to extend my sincere appreciation for the subsequent extensions of the submissions' deadlines and the provision of enough time to work on this study. Many thanks.

ABSTRACT

Object tracking has been one of the dominating subject of consideration across technological studies. With a significant increase in efficiency of computing devices over the years, CV has been used to implement diverse systems that analyze images and videos for security solutions. In conjunction with dedicated machine learning algorithms, CV becomes inevitable to every organization regardless of specialty.

Studies that have been carried out parade several attempts on using CV with modern technologies for object tracking. In particular, neural networks and library bindings have enabled development of object tracking systems that can be optimized to suit various business and personal applications.

In this study, CV is collaborated with the C++ DLib library and the Darken YOLOV3 neural network in implementing a home or office security solution based on face recognition and object tracking respectively. In seeking efficient algorithms suitable for image and video analysis, algorithms such as the HOG, GMM and a few more that compete with the YOLOV3 network are considered, and rigorous comparisons are made for befitting results.

CHAPTER ONE

1 INTRODUCTION

1.1 Background Of Study

Computer Vision (CV) is a field of computer science that focuses on developing techniques that enable computers to "see", analyze and understand content of digital images such as photographs and videos (Brownlee, 2019). According to Singh (2019), CV as a technology has been around for about 50 years but, it is recently that it has attracted so much attention of major and many organizations. It has been realized that this technology is applicable in various fields and areas where digital data processing is crucial. Amazon, Google, and Facebook are some of the major tech companies that have been shown to solely depend on computer vision in their daily production (Saputra et al., 2019). In the same manner, CV has now been adopted in food , medical, sports and entertainment industries. Bhargava and Bansal (2018) have written on how CV is being used in different countries to evaluate the quality of fruits and vegetables. This technology has also been considered crucial in lung cancer screening using Computed Tomography (CT) scans (Ritchie et al., 2016). Thomas et al. (2017) have also researched and shown on how sports and entertainment industries are relying on CV in their daily marketing and production activities.

Additionally, CV has been one of the most rising technology to achieve business heights. According to a market research, the computer vision market is valued at US$11.94 Billion and is likely to reach to US$17.38 Billion by 2023, growing at a Compound Annual Growth Rate (CAGR) of 7.80% from 2018

to 2023 (Some, 2019). The technology is worth engaging into due to critical problems it is used to solve. Apart from the ones aforementioned, CV has been highly adopted in security systems that are based on object detection, face recognition, and motion tracking.

It follows that the criticality of security in many countries, organizations, businesses, and households has increased the demand for effective security systems in all these areas. Problems such as theft, robbery, fraud and misplacement of valuable items have been considered threatening generally in all the prior mentioned areas and most researchers have studied and written on showing how CV technology is and can be used in implementing effective security systems (Mehboob et al., 2019).

Whilst CV was primarily designed to enable computers to "see" and analyze content of digital images, since more than a decade ago, developers and engineers have started to realize that modern machine learning and deep learning models and algorithms can be conveniently incorporated within the Application Program Interfaces (APIs) and libraries of CV to design and build even more effective systems (Nowozin et al., 2011; Shao et al., 2014; Zhang and Wang, 2019). With this ability, CV has been taken to new heights in building effective object detection, face recognition, and motion monitoring systems for various applications based on pictorial data analysis. In this study, attention is given on examining different deep learning algorithms and models developed for CV in building a security system based on object detection and image and/or video analysis.

In the same vein, developers and engineers have also discovered that generic algorithms can be designed, built and be able to be of use in various applications. For example, the histogram of oriented gradients (HOG) algorithm,

which is a feature descriptor used in computer vision and image processing used in object detection, has been developed to be used generically in any application based on object detection. Among a lot of researchers who have used HOG in different applications, Korkmaz and Binol (2018) have written on classification of molecular structure images and size reduction methods for early stomach cancer detection, Žemgulys et al. (2018) have designed a system to recognize referee's actions in a basketball match, Wei et al. (2019) have worked on a multi-vehicle detection algorithm, and NASSIH et al. (2019) have published their study on efficient face recognition and classification. Henceforth there are many of such algorithms, models and APIs provided for developers and researchers.

Moreover, deep learning models used in developing CV related systems are becoming comparatively cheaper (in terms of computational resources) than before, hence increasing development of efficiency enabled systems. Many Graphical Processing Unit (GPU) production companies are collaborating with machine and deep learning engineers to provide high performing processing power, some have even considered designing totally new hardware resources specifically for deep learning models (Madiajagan and Raj, 2019). The same fact applies to data collection and access, data are rapidly expanding than it was in a decade ago. Realizing that data is the key component in machine and deep learning as data create a platform for all the learning and predicting processes in building models and/or algorithms is crucial. Methods that are now used for data collection and analysis have remarkably improved. Thus, there are great communities of researchers and developers that are working on these methods and seek to present efficient and convenient ways of accessing data.

The prior asserted facts show that developers now have an opportunity to build more intelligent and reliable security systems however, there is still a wide gap between intelligence, reliability and most of the security systems currently offered in the Kingdom Of Eswatini. The prominent systems offered are Closed-Circuit Television Camera (CCTV) based. The systems use surveillance cameras to record scenes, and save them on hard disks with their respective dates and time. Administrators or system users have to literally navigate on the systems and look through computer screens in search of a particular scene. This implies that the user has to have an idea of the date and time of scene looked for otherwise they will have to look through all the records. Thus, the systems are not intelligent enough to record the specifics and details of the objects identified.

Furthermore, the systems can only be manageable provided that the user is right behind the monitor screen looking at the scenes, they can not manage the system when they are away or far from the vicinity of the system. In case of tragedies such as fire or damage of the system's hard disks, all the data are lost. Most of these systems do not provide cloud based backup systems due to large sized files they continuously record that are difficult to upload to cloud storage systems efficiently and in real time.

Hence, in this study, CV is considered in the development of a security system based on object detection and pictorial data analysis using deep learning algorithms. Aforementioned challenges of the current security systems are to be of serious consideration, realizing that efficient methods (algorithms and models) have become affordable and accessible.

1.2 Statement Of The Problem

Everyone can be unmindful of placing items where it is not always easy to remember. In many cases one would forget(or would not be able) to apply enough security and proper attention to their valuables which may end up go missing and, its quite frustrating if the item(s) are not to be locatable in times of need. Worst cases are, when intruders(thieves or robbers) find access to one's valuables and get hold of them for permanent loss. This would be sad but, it is actually a true matter and security issue remains one of the most critical subject that gravely needs attention.

In the Kingdom of Eswatini, early in the year 2019, on the 6th of April, Makhubu (2019) reported a criminal event where thugs sneaked into a supermarket, threatened the shop assistance's life using a knife, took away money amounting to E75000, and three cellphones estimated to worth E35000. The shop was reported to have CCTV(closed-circuit television) camera security system which recorded the whole scene but, clearly the system was not as efficient as to offer enhanced security because the thugs successfully robbed the shop and drove away without any immediate action that was able to be taken. Had the system been able to notify the security forces immediately, just maybe, less or no loss and damage could have been incurred.

It usually is rare to experience at most a season without theft and robbery-related cases. Now, these are the only ones the media would have got hold to, and one can imagine those that go unreported, which are usually the majority. On 08 December 2018 Andile (2018) reported of an armed chicken thief that killed a 53-year-old-soldier who is supposedly the owner of the chickens and the homestead. The thief was later on pursued by the police and eventually got shot to death but, the loss was already made, one can imagine

the difficulties the family and relatives of the late soldier had to undergo. It is just gnawing, and this is all because in such cases, the victims are found not aware on anything happening around them, and have no means to quick initiative steps that can improve chances of not incurring such devastating losses.

These terrible events clearly show how valid and serious the problem aforementioned is, and as it is true that there are no quite efficient and much affordable systems that have been built or adopted to be of use in different, critical places of concern(homes, offices, and other necessary places). The CCTV camera systems that are currently being used in most places are quite expensive, not the majority can afford them. Besides being quite costly, they perform common functions such as recording video scenes, saving them on disk and allow users to search and playback recorded scenes. They are not programmed to perform intelligent operations such as recognizing familiar objects or people, process live streamed videos or captured images and provide necessary, immediate notification(s) or act accordingly in cases of detecting unusual actions or movements. This is true for at least most of the systems currently being used in the Kingdom of Eswatini.

CCTV camera systems usually quite useful in case of misplacing item(s), the user can find a recorded scene of when last the item(s) was seen, from there, they will an idea of where to look. However, the process can be so tiresome especially when the user(s) have forgotten the date or time of when last they got hold of the item. One would have to go through several recorded scenes which clearly is inefficient. The systems are not programmed to be smart to process the scene, identity objects therein, store some information(location and time of identification) about them, and be able to retrieve such informa-

tion whenever queried. This again calls attention for such easy looking but very relevant problem(s) and this has motivated and inspired this research project.

1.3 Aim/Purpose Of Study

The aim of this study is to develop and implement an Object Tracking System (OBTS) using the C++ Dlib library and yolov3 algorithm that will help to track and hence secure user's valuable commodities and items

1.4 Objectives Of The Study

The objectives of this study are as follows:

- To establish an object tracking based security system that curbs the aforementioned problem in the Kingdom Of Eswatini.
- To study different security systems that are currently offered with the view to the core implementation details; determine their strength and weaknesses and capitalize on their strength and work on their weaknesses.
- To study different models, algorithms, and API(s) used for object tracking with the view to determine the ideal technique to adopt.
- To design and develop the proposed object tracking system using C++ Dlib library and yolov3 algorithm
- To evaluate the performance of the proposed system.

1.5 Significance Of The Study

Considering the nature of the problem that is intended to be solved, it has resulted in the breaking down of the intended solution in about four fundamental achievements targeted during this study.

The first and the ultimate target is to curb home robbery, theft related cases and minimize valuables' misplacement by providing an affordable, easy to manage, and worth it system to users in the Kingdom Of Eswatini. These cases are common and worth addressed in the Kingdom. Hence, showing that based on this fact, the study is relevant and worth exploring.

Secondly, the study targets to address the limitations of currently available security systems such as CCTV camera systems earlier mentioned. Not only do they have limitations but, most of them are quite costly to install and manage, and developing an affordable system that addresses some to most of these downsides, seeks to show how significant this study is.

Thirdly, as mentioned in the prior section, it is an objective to explore, assess, examine, and determine best algorithms, models, APIs, and other tools. It is therefore significant to engage into this study because real-time systems such as OBTS require outstanding tools.

Lastly, establishing a system that blends hardware, software, and external input-output devices and have them cooperate with each in real time is a special skill that if acquired in this study, it would expose the developer(s), giving a stepping stone to the developing and building of even much more critical and complex systems.

1.6 Limitations

There are about three major factors that would affect this study in a slight negative manner, thus, if not managed well, they might affect the intended results of this research project. These are; data accessibility and availability time, finances, and time.

Firstly, data accessibility and availability challenge could be a challenge. Some of the models to be used in this project would not function as intended, they might need re-training and/or further-training and they would need massive amounts of data that might not be freely available. In that case, the study would have to be carried out with the best models managed which would leave some loopholes in the system. Some of the data, even if they would be available, they might not be accessible due limited computing resources, and this would again hinder the study from producing best and intended results.

Secondly, there might again, be a challenge with the finances. This project is privately sponsored and some of the components (for the system to produce intended results) are to be ordered abroad. Depending on the availability of the finances, then only the required components would be able to be ordered and this implies that if anything goes as unplanned, the results of the project might be affected.

Lastly, this study involves a lot of researching, there are a number of new technical concepts and technologies to be involved. Most of the machine and/or deep learning models, algorithms, methods, and tools to be used in this study require a lot of attention and descent concentration. With the time available, if anything goes as unplanned, the study might not be com-

plete.

1.7 Scope Of Study

In as much as the idea of this study is open and can easily be conveyed to any environment and/or country, the project is targeted to the Kingdom Of Eswatini at least for the course of this particular session.

The system targets any environment of concern, thus, it could be a school, an office, a market place, or home. The system is to be designed to serve in any environment where digital camera devices would be able to capture scenes and/or photographic images and where some source of power would be available, solar energy or electrical energy would serve.

The study involves solving a social problem and there is no doubt, it is going to need engaging with the society, to hear from them on different experiences they have had and heard from other people. This is necessary during designing the system, to be aware of a number views. For example, the society would be aware of different criminals, their tactics and weaknesses. Getting informed about these views would help during development, specifically during training the models and therefore improve the solution. There are actually a lot of reasons to bring the society on board, and there is going to be a number of such engagements up to some descent level.

The major organizations that are going to be of great importance are the Eswatini Post Telecommunications Company(EPTC) and Eswatini Electricity Company(EEC). These are the most popular organizations as much as networking and electricity/power concepts are concerned, respectively. The

intended system is not going to be much subjected to these concepts but, they are needed in order for the intended results to be achievable and therefore, in case it becomes necessary, these are two targeted organizations to visit.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 Introduction

Object detection and tracking under computer vision, has been one of the critical phenomenon to be considered by most Artificial Intelligence(AI) developers and engineers for at least two decades from the time of documenting this study, and the subject has become unstoppable as endless efforts employed have made algorithms being built a lot better and much more useful (Zou et al., 2019), (NOVET, 2015). In this section, different papers and/or articles written on object detection and action recognition systems are considered, from the ones that more or less model based, to the ones that are more application based. All of these papers are to be considered with respect to security systems. In particular, their strengths are to be highly acknowledged and their downsides also recognized.

2.2 Yolov3 and It's Competitors on Object Tracking

Among a lot of researchers, Seemanthini and Manjunath (2018) have written on modeling human detection and tracking system. The implementation was to be done using cluster segmentation approach, which is a notion of extracting useful features in an image. The system takes an input video which will be divided into frames using frame generation, then out of each of these frames, segmentation is done and finally feature extraction. Classification was to be done using Support Vector Machine(SVM) algorithm, which is a

linear model for classification and regression problems (Pupale, 2018). Each object's activity is then detected based on the result by classification. The model is approximately 89.59% accurate and it can be used in various applications for as much as human detection and tracking is concerned. However, the model fails if the video frame has a group of objects (people), it cannot track events, thus the model might not be very suitable for critical systems where extreme accuracy is a requirement.

Additionally, Prasad et al. (2018) have modeled an object detecting system for visual surveillance using the Gaussian Mixture Modelling(GMM) algorithm. In this algorithm, each cluster of data is modeled according to a different Gaussian distribution. The approach used was very flexible in that there are soft assignments, meaning that each data point could be generated by any of the distributions with corresponding probability. Estimation of each point is then done using the Expectation Maximization(EM) algorithm which is used to predict values of latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables). The model uses the foreground image functionality of GMM to get the status of the object, thus if static or moving. However, the model is not trained to be generic that is, it can not detect any object such as a car, table and any other object that might be of interest to track. Otherwise, the team's efforts are highly remarkable for the solution to the addressed problem.

Among the most effective object detection related systems published in the year 2018, Shinde et al. (2018) have also contributed to the pool by modeling one of them. They used a Darknet Convolutional Neural Network(D-CNN) called You Only Look Once(YOLO). The approach is to detect, localize, and

recognize actions of interest of the object detected. From frame to frame obtained from a live stream of video data capture, frames are captured from a surveillance camera. The model was built using YOLO, a single CNN which is designed to predict multiple bounding boxes(concentrated areas on the frame) and class boundaries for those boxes. The algorithm achieves high accuracy yet it is said to "only look once" thus detecting most objects in just one pass, being extremely fast and able to handle real time scenarios (Scienc, 2018). The model attains about 88.372% of accuracy which is remarkable for action recognition. However, some of the actions are not being clearly recognized, for example, when a person opens the door, the system can not tell whether the person is entering or leaving the room of which this is a very important action to recognize in most critical places such offices, banks and homes. The model also uses multiple frames when recognizing an action, which is costly in terms of computer time and memory, whereas even a single frame could be used and yet achieve the same result.

Moreover, Kajabad and Ivanov (2019) have modeled a human detecting system that find attractive areas(areas most visited by people) using movement analysis and deep learning approach. It is a technical approach related to video analysis to detect and control human behavior in a public place. Among different objectives this team had, their system seeks to understand the overall human traffic in an area of concern and find out the interests of people in different parts of an area. The system takes input from surveillance cameras and use two methods, first to detect people in a closed space, and second finding dense areas in which people spend more time to visit. They have used the YOLO model which makes prediction with a single network evaluation, and incorporating Region-based Convolutional Neural Network(R-CNN) and Faster R-CNN, on the other to make multiple assessments for a single im-

age. This approach has made YOLO extremely fast, running in real-time. The team has also taken a step further to compare their model with other detecting models such as, Histogram of Oriented Gradients(HOG) which is a feature descriptor used in computer vision for the purpose of object detection and image classification (Kitayama and Kiya, 2019), Single Shot MultiBox Detector(SSD) a Faster R-CNN which uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects (Deguerre et al., 2019), and YOLO-tiny(a lesser version of YOLO v-series), and they concluded that theirs had highest performance at least up to end of their study. On the other hand, the system is model based, that is an application system would be need in order to evaluate this model.

It follows that this study is based on using the best and recent models in developing an applicable security system rather than building a model based system. To achieve this, two factors are to be considered. These are; area of concern or application and the performance rating of the algorithm(s) and/or model(s) of consideration. For example, areas such as an office or house can have varying lighting levels depending on the time of the day and if doors and windows are either opened or closed, and it is not obvious that any model would be able to meet the requirements. Hence, YOLOv3 has been selected to be of the highest priority so far. This is because it is the leading, most flexible , extremely fast and accurate detector as compared to other detectors (Redmon and Farhadi, 2018a). Subsequently in this section, a closer look at YOLOv3 will be considered.

Systems that use traditional classifiers and localizers to perform detection need to apply the model to an image at multiple locations and scales. This slows the system and possibly failure in darker areas and in real-time scenar-

ios. YOLOv3 uses a single neural to the full image and the network divides the image into small regions and predicts bounding boxes and probabilities for each region, thus looking at an image once and be able to draw appropriate conclusions. This is critical in real-time scenarios and where computational resources are a concern. On YOLOv3, Redmon and Farhadi (2018a) have also considered improving the training increase performance by including multi-scale predictions and a better backbone classifier than they had done in the previous versions of YOLO. This notion makes YOLOv3 1000x faster than Region-based Convolutional Neural Networks (R-CNNs) and 100x faster than Fast Region-based Convolutional Neural Networks (Fast R-CNNs) (Redmon and Farhadi, 2018a), and this is the fundamental principle of the security system that engaged this study. Speed, accuracy and computational units are a serious concern. Thus in this study we use YOLOv3 for object detection and analysis.

YOLOv3 security based projects that have been build in the past two years are considered. Haschek (2018) has demonstrated how he built an economized home security system using YOLOv3 running on Raspberry PI hardware. The system takes video streams from a series of Raspberry PI camera modules as input and process them by detecting objects and report to the user by sending them some notification of what has been detected. The system has scalability advantages because the user can use as many PI's as they may afford and there by increasing the processing power and performance of the system. This feature will be capitalized on in this research project. However, the system is not user friendly because the notifications are send in the console of the system, which means that the user is required to literally look into the console to get notifications. The system does not also consider that the user might not be in the vicinity of the system to look into the system

and yet they would need to be notified of the critical detection(s) made.

It follows that Ek (2017) has addressed the gap left by Haschek (2018) since he recognized that the user would be away and still have to be notified if there be any intruder detected. The system uses Unifi G3 cameras to record scenes whenever motion is detected and run a python script to process and send an image of the intruder identified to the user's mobile phone. It is clear that the communication carried out between the system and the user's mobile device is facilitated by some form of network, which in this case is a local area network such as WiFi. This is an import feature in a security system and it will be considered in this study. However, Ek (2017) does not consider tracking the object and process it's activities, this feature is important in providing enough and appropriate and accurate evidence for future purposes. The system does also not consider processing video scenes, draw meaningful conclusions and notify the user, store the results in the server or some database, or notify respective security forces. This is a serious downside of the system because the intruder could be detected and have the user notified by an image sent to them, but if the system could not continously track activities, then the intruder could do more harm without any immediate action from the security forces such as police and no enough evidence of the scene recorded. Hence tracking moving objects is one the main objectives of this study.

Moreover kkbankol ibm (2018) have developed a system that analyses real-time CCTV images and/or using CNNs. The team has done very well as they have considered most concepts involved in developing effective security systems and curbed most the limitations of the previously reviewed systems. The system runs a YOLOv3 neural network and uses deep learning algo-

rithms to detect objects and OpenCV to analyze the images. Useful results about the scenes are then uploaded to International Business Machines(IBM) cloud system in real time. The user would then access the information using a web application. The user can make variety of queries depending on what they seek and the database retrieves the corresponding data. It is clear that there is no direct or real-time communication between the system and the user, and this is a gap when implementing a security system. If there is no system-user real-time communication then real-time video analysis would not be of much use in case where there is an intruder. The team also did not consider that it is crucial that the user gets control over the system by accessing the remote server (system) even when they are away. This is because cloud computing systems could suffer traffic jams without notice or could suffer faults and therefore affecting the reliability of the security system. These gaps are critically considered in this study.

In conclusion of the section, this study considers YOLOv3 for implementation though other models such as the ones aforementioned will also be used during testing in order to draw solid conclusions and facts. The closely related security systems reviewed will have their strengths capitalized on, and on the outlined downsides; the research will be mainly centered on providing real-time accessibility of the system regardless of the location of the system-administrator or user.

2.3 Overview of Existing Object Tracking Algorithms

Considering that this study is mainly aimed at developing an object tracking system to implement security solutions to safeguard user's valuable property in cases of misplacement and intrusions, this section examines several

algorithms, and models that have been used in the implementation of the intended solution. From the examination, a justification as to why C++ Dlib library and the yolov3 algorithm are respectively used for image analysis and video processing will be asserted.

This study has involved documenting the use of Unified Modeling Language(UML), Flow Chart, and Use Case diagrams, Psuedo-Code, and all the algorithms that are used in the implementation of the intended solution to the aforementioned problem. However, the diagrams and psuedo-code have been documented separately from this paper. This paper documents the algorithms used across the study.

The implementation of the algorithms that are used in this study vary from module to module, thus, their documentation has been arranged as per their application in each particular module. Therefore, each of the following subsections will document the specific algorithms used in each corresponding module.

2.3.1 Face Classification

Keeping in mind that the study is entirely centered on image and/or video analysis. Firstly, image processing algorithms, particularly face detection, recognition and classification are considered.

Haar Cascades and Dlib C++ Library wrapper are the toolkits containing the machine learning algorithms that are used for face detection in this study.

2.3.2 Haar Cascades

Haar Cascade is a machine learning object detection algorithm used to identify objects in an image or video based on the concept of features(Viola et al.,

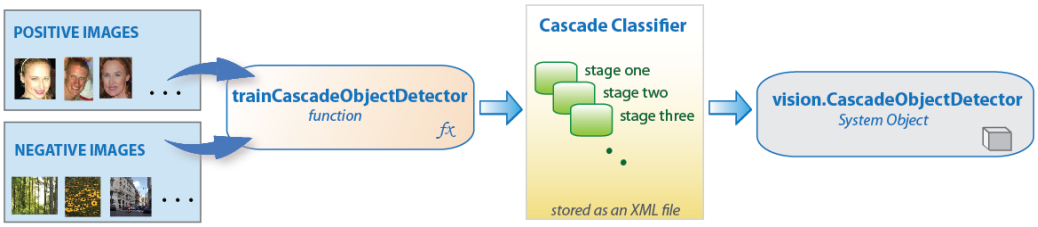

Figure 1: Cascade Classifier

2001). In this approach, a cascade function is trained from a lot of positive images(images with targeted objects) and negative images(images without targeted objects). After training, it can then used to detect objects off which in this study, it is used for face detection.

The algorithm has four sections:

1. Haar feature classification
2. Creating integral images
3. Adaboost training
4. Cascading classifiers

However, this study focuses on using the pre-trained cascading classifiers to detect faces rather than developing them. Therefore, only section four of the Haar Cascade algorithm is considered.

The cascade classifier consists of a collection of stages as shown in Figure 7, where each stage is made up of a group of weak learners(simple classifiers called decision stumps). Each stage will have been trained by a technique called boosting which provides the ability to train a highly accurate classifier by taking a weighted average of the decisions made by weak learners.

Each stage labels the region defined by the current location of the sliding window(a rectangular region of fixed width and height that "slides" across an image) as either positive or negative(Rosebrock, 2015). If the region is positive, there is an indication that a face was found, and if negative, then no face was found. If the region is negative, the classification of this region is complete, and the detector slides the window to the next location. If the label is positive, the classifier passes the region to the next stage. Finally, the detector reports an object found at the current window location when the final stage classifies the region as positive(Berger, 2018).

Mathematical Model Of Haar Cascades For Face Detection:

For face detection, as mentioned in the prior section, the sliding window approach is used to test images over different locations and scales. Mathematically, the process starts by building the feature pyramid H from the input image which enables the detection of faces over multi-scales using filters of fixed size. The response of the i^{th} filter F_i in the l^{th} level of the feature pyramid is calculated by:

$$R_{i,l}(x, y) = F_{i.\varphi_a}\left(H(x, y)\right) \tag{1}$$

where φ_a is the Histogram Of Oriented Gradients(HOG) feature vector in a window of H with the top left corner at (x, y) and its size defined by the size of the filter. To find the deformable variations in the parts of different faces or other noises in the image, a generalized distance transform is used to find the optimum locations for the parts with respect to root location. The updated response of the i^{th} part filter in the location is given by:

$$D_{i,l}(x, y) = max_{dx,dy}\left[R_{i,l}(x + dx, y + dy) - d_{i.\varphi_d}(dx, dy)\right] \tag{2}$$

where $\varphi_d = (dx, dy, dx^2, dy^2)$ is the deformation feature and d_i is a four

dimensional vector specifying its coefficients. This transformation spreads high part scores to nearby locations according to a deformation cost.

Meanwhile, the Salford Predictive Modeler(SPM) is then used to select the model parts from a pool of parts that includes different sub-types of the main facial parts such as; eyes, nose and mouth, and common occluding objects such as; caps, sunglasses and hands that may hide these parts. The overall score of any window is then calculated by adding the root score on this window to the sum of the selected part scores using the following models:

$$S_t(x, y, l) = S_r(x, y, l) + \Sigma_{P_i \epsilon S} S_{P_i}(x, y, l) \tag{3}$$

$$S_r(x, y, l) = R_{0,l}(x, y) \tag{4}$$

$$S_{P_i}(x, y, l) = D_{i,l-\lambda}(2(x, y) + v_i) + b_i \tag{5}$$

Now, the root filter is at level l of the pyramid and the region filters are at level $l - \lambda$ which is twice the resolution of l, v_i is the anchor position for part i relative to the root position, b_i is a bias term to make the scores of different parts parts comparable for selection and using the same threshold. Hence the summation over the parts belonging to a set S containing the selected parts for this window will determine the score which is the detection of face(s)(EL-Barkouky, 2014).

2.3.3 Dlib C++ Library Wrapper

The Dlib C++ library wrapper consists of several machine learning, deep learning algorithms and convolutional neural network models that can be used in image analysis and classification. In particular, this study has capitalized on the cnn face detection model version one; a Dlib's algorithm that uses the a pre-trained network called max-margin object-detection human face detector, trained on ≈ 3 million images, to detect faces in an image.

The approach of this algorithm is to avoid sub-sampling (interpolating or smoothing the image to reduce aliasing) and instead optimizing all the sub-windows in an image (King, 2015). This implies optimizing computing resources (time and memory) and also improve any object detection method which is linear in the learned parameters, such as HOG models that can be utilized alongside with this model.

The algorithms has been broken down into stages (sub-algorithms), of which only two stages will be considered to describe the functionalities of the model as per its use in this study.

Stage 1: Object Detection

Keeping in mind that the objective is detect objects in an image, particularly faces; the following pseudo-code describes how King (2015) arrived at detecting the position(s) of the object(s) detected.

Input: image x window scoring function f

1. $D :=$ all rectangles $r \in R$ such that $f(x, r) > 0$
2. D such that $D_1 \geq D_2 \geq D_3 \geq ...$
3. $y^* := \{\}$
4. **for** $i = 1$ **to** $|D|$ **do**
5. **if** D_i does not overlap any rectangle in y^* **then**
6. $y^* := y^* \cup \{D_i\}$
7. **end if**
8. **end for**
9. **Return:** y^*, The detected object positions

Stage 2: Max-Margin Object Detection

An image x with valid labeling is considered for the actual object detection procedure and the only window scoring functions which are linear parameterized are considered. The sum of the window scores is then taken for a set of rectangles y. By going through this procedure, the interest is to find a parameter vector w that leads to fewest possible detection mistakes.

This is to say, given a randomly selected image and a label pair $(x_i, y_i) \in X \times Y$, the goal would be to have the score for correct labeling of x_i to be larger than the scores of all the incorrect labelings.

Thus, mathematically:

$$F(x_i, y_i) > max_{y \neq y_i} F(x_i, y)$$

would have to be satisfied. And the resulting score would be the one to be used for detection.

Mathematical Model For The Dlib C++ Library Wrapper

Let r denote denote a rectangular area of an image and R denote all the rectangular areas scanned by the object detection system. A valid labeling of an image as a subset of R is defined to incorporate the common non-maximum suppression practice such that each element of the labeling does not overlap with each.

The so called popular definition of "does not overlap" is used as follows: Rectangles r_1 and r_2 are said not to overlap if the ratio of their intersection area to the total area covered is less than 0.5. That is:

$$\frac{Area(r_1 \cap r_2)}{Area(r_1 \cup r_2)} < 0.5 \tag{6}$$

Using Y to denote the set of all valid labelings, then given an image x and a window scoring function f, the object detection procedure can be defined as:

$$y^* = argmax_{y \in Y} \sum_{r \in y} f(x, r) \tag{7}$$

Thus finding the set of sliding window positions largest scores but simultaneously not overlap. Considering the fact that the real world data is often noisy, not perfectly separable, or sometimes contains outliers, the procedure is extended into a soft-margin setting which results in arriving at the definition of the optimization for the Max-Margin Object Detection (MMOD).

$$min_{w,\xi} \quad \frac{1}{2}||w||^2 + \frac{C}{n}\sum_{i=1}^{n}\xi_i \tag{8}$$

$$s.t \quad F(x_i, y_i) \geq max_{y \in Y}\left[F(x_i, y) + \Delta(y, y_i)\right] - \xi_i, \quad \forall i$$

$$\xi_i \geq 0 \quad \forall i$$

Where C is the usual Support Vector Machine (SVM) parameter that controls the trade-off between trying to fit the data or obtain a large margin. The MMOD objective function defined in Equation(8) is said to be a convex upper bound on the average loss per training image:

$$\frac{C}{n}\sum_{i=1}^{n}\Delta(argmax_{y \in Y} F(x_i, y), y_i) \tag{9}$$

Meaning that if, ξ_i, from Equation(8) goes to zero, then the detector is guaranteed to produce the correct output from the corresponding training session. This type of max-margin approach is said to be efficient and very performant since popular and very successful papers such as Han et al. (2015), Altun et al. (2003), and Liu et al. (2009) have used it in their writings.

Given the MMOD in Equation(8), the Optimization problem can then be solved. Noting, that the MMOD in Equation(8) is equivalent to the following unconstrained problem:

$$min_w J(w) = \frac{1}{2}||w||^2 + R_{emp}(w) \tag{10}$$

where $R_{emp}(w)$ is $\frac{C}{n}\sum_{i=1}^{n}[F(x_i, y) + \Delta(y, y_i) - F(x_i, y_i)]$

Following a series of stages of which this study does not go into detail, it can be shown that the sliding window classification can be implemented efficiently using a set of integral images, and scanning the sliding window over every location in an image pyramid, down-sampling each layer by 4/5. To decide if two detections overlap for the purposes of non-max suppression and determining if the truth box box has been hit, thus an object has been detected, Equation(6) is used.

The network that comes with the Dlib wrapper delivers $\approx 99.38\%$ accuracy on detecting and predicting faces in an image which makes efficient for face recognition.

Secondly, faces classification is considered to follow detection. The comparison made between the Haar Cascades method and the Dlib c++ wrapper method on face detection showed that Dlib wrapper outperforms the Haar Cascades method and thus, the study carries on to use the Dlib wrapper for face classification and recognition.

The central idea on classifying faces lies on the proposed method of finding face landmarks which are referred to as the 68 specific points of a face (Arsenovic et al., 2017). These points mark areas such as the top of the chin, the outside edge of the eye, the nose, and so on. Figure 2 below shows the drawing of an imaginary face that depicts these landmarks.

Given these landmarks will eliminate complicated 3d image warping that would introduce distortions in the image by using simple transformations such as rotate, scaling and shear to straighten the image for classification. These transformations are necessary if the image (face) is turned because using the landmarks, the eyes and the mouth can be centered roughly in the same position, and this would make classification more accurate.

The next step of the classification process would be to find the face encodings (data about the image) of the image that is under classification. These encodings can be generated by a specific trained neural network. However, training such a neural network is computationally expensive and time consuming, hence in this study, a pre-trained one by *OpenFace* is utilized. These encodings have been discovered to be ≈128 float numbers for any general face, and for each face there are generally referred to as 128 measurements or an embedding (Schroff et al., 2015).

Therefore in this study, all that is done to get the 128 measurements of any face is to run an OpenFace neural network. The image in Figure 3 shows how an embedding of an input face image would look like, and there is no doubt that the measurements are inform of an $\approx 32 \times 4$ matrix, hence making it very intuitive during classification, that matrices (measurements) of two different face images of the same person should be nearly same numbers and two different face images of two different persons should significantly differ.

Moreover, the face encodings of users of the system are stored in database as known faces, and any incoming or newly detected face image will have its encodings generated and get compared with each and every known ones, implying that if any match occurs then the person is known otherwise un-

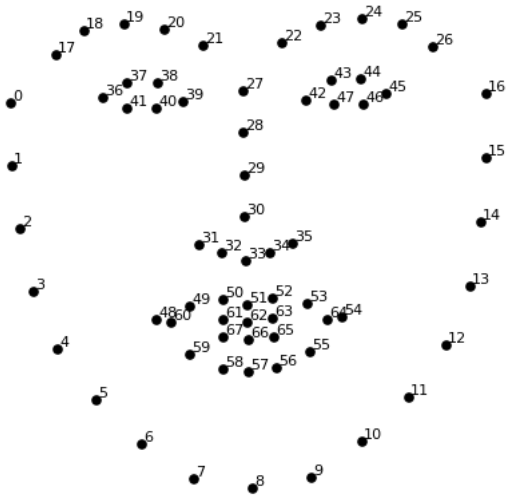

Figure 2: 68 Face Points

known. If the person is known then a necessary action can be performed depending on the application, it could be; tag the image, proceed, and so on. If the person is unknown a necessary action can be performed as well, it could be notifying the administrator of the system, restrict access, and so on. In this manner, face classification and recognition are successfully performed in this study.

2.4 Video Processing

The second phase of this study considers video analysis and/or processing. This involves object detection and tracking in live streamed videos, implying that given a video scene, the system is designed to observe objects therein, and draw necessary conclusions about the observed objects. Therefore, the following subsections discuss object detection and object tracking respectively.

Prior to object detection, the system performs motion detection on its entry

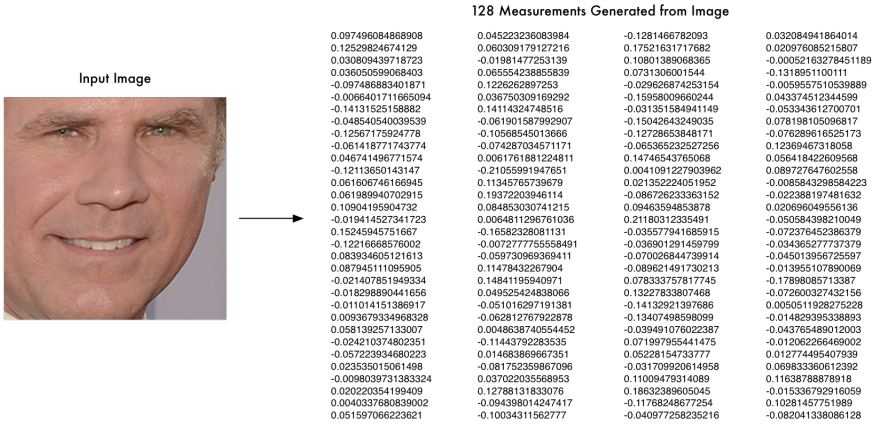

Figure 3: 128 face measurements

point. The idea is that there should be a cause for object detection and the assumption is that, the cause should be a moving object such as a person, or an animal. Thus, given a cause, then and only then should be need for detecting the objects.

Passive Infrared Sensors(PIR) which are electronic devices(sensors) that measure infrared(IR) light radiating from objects in their field or area are used in this study for motion detection purposes. With their ability to detect moving objects, they will be able to alert the system and thus initialize video taping and therefore object detection and/or tracking.

Considering that one of the main objectives of this study is comparing different algorithms' performances, the following five algorithms would be implemented and tested on the same input data:

1. Meanshift Algorithm,

2. Camshift Algorithm,

3. Lucas Kanade Optical Flow Algorithm,

4. Gunner Farneback Algorithm,

5. You Only Look Once Version 3(Yolov3) Algorithm.

2.4.1 Meanshift Algorithm

Intuitively, Meanshift Algorithm works by considering a set of points(such as a pixel distribution like histogram back-projection), then given a small window say a circle, and it has to be moved to the area of maximum pixel density(or maximum number of points) as shown in Figure 4 below.

Initially, the window is shown in blue with the name "C1" with center "C1_o", and finding the centroid/cluster inside the window, it can be seen that the point "C1_r"(marked in small blue circle) is the real cluster point of the window.

The window is moved so that "C1_r" becomes the center of the new window. Iterating these steps, will result into a window with the maximum pixel distribution(maximum number of points) marked with green circle, named "C2". This process is described on a static image(frame), thus depicting object detection.

The histogram back-projected image and the initial target location are passed. When the object moves, the movement is reflected in histogram back-projected image and as a result, meanshift algorithm moves the window to the new location with maximum pixel density thus tracking the object.

Iswanto et al. (2019), and Maklin (2018) asserted some upsides of the meanshift algorithm, that when using it, there is no need of categories beforehand

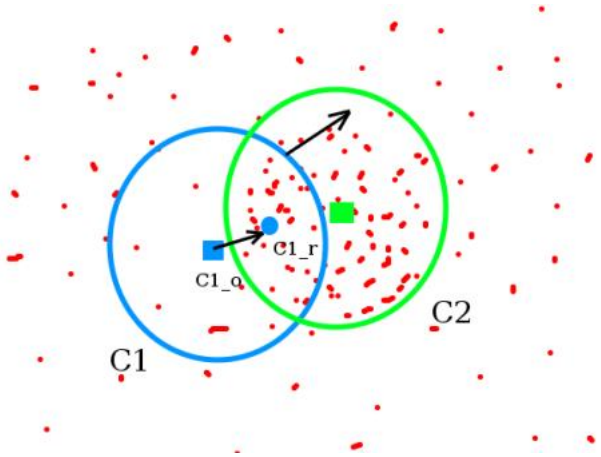

Figure 4: Meanshift Algorithm

and thus it can used efficiently in applications that require clustering such as search engines, academic rankings and medicine. However it is computationally expensive, and is $\approx O(n^2)$. Moreover the window size is fixed, it does not adapt the the size and the rotation of the target.

2.4.2 Camshift Algorithm

Camshift Algorithm was published by Bradski (1998) as a way of improving the meanshift algorithm, thus it is called Continuously Adaptive Meanshift(CAMshift). Meanshift is applied first, then the size of the window is updated to become:

$$s = 2 \times \sqrt{\frac{M_{00}}{256}} \tag{11}$$

It applies the meanshift with new scaled search window and previous window location again and the process is continued until required accuracy is met. Apart from the meanshift, it returns a rotated rectangle(the result), and the box parameters(used to be passed as search window in next iteration).

As a primary objective, Camshift is intended to perform efficient head and face tracking in a perceptual user interface, tracking the X, Y, and Area of the flesh color probability distribution representing a face. It is moreover designed to be an efficient and light-weight tracking algorithm to replace meanshift in tracking targets in simple cases.

However, it fails to track objects in more complex situation. It is also said to consider the Hue(H) component making is vulnerable to the interference of background color.The same is also true in the case of the target obscured or half shade that the algorithm easily lose track of the targets (Emami and Fathy, 2011).

2.4.3 Lucas Kanade Optical Flow Algorithm

Optical flow can be defined as the apparent motion of objects between two consecutive frames caused by the movement of the object or camera. The Optical Flow Algorithms such as one by Lucas Kanade, seek to solve the optical flow equation:

$$I_x V_x + I_y V_y = -I_t \tag{12}$$

of the image intensity given by $I(x, y, t)$, where the intensity is a function of x, y and time t. I_x and I_y denote the intensity change in x and in y respectively. V_x and V_y are the velocities in x and y respectively and are the unknowns to be computed.

Looking into the optical flow equation, it can be observed that there are two unknowns in one equation. The Lucas Kanade Algorithm assumes that the movement between successive frames is reasonably small and uniform within

a the window being considered. Then the equation is said to be modeled by system of:

$$\begin{bmatrix} I_x(p_1)\dot{I}_y(p_1) \\ I_x(p_2)\dot{I}_y(p_2) \\ I_x(p_3)\dot{I}_y(p_3) \\ \cdots \end{bmatrix} \times \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ -I_t(p_3) \end{bmatrix}$$

which clearly indicates that more equations than the unknowns. To solve the system of equations, the assumption of uniform velocity is used to reduce the size of V_x and V_y to two variables for a small window. Thus, all that would be needed is the window to look into (hrishioa, 2015). In this manner, the Lucas Kanade can be used for corner detection, object detection and tracking. Ahmine et al. (2019) give the details of how the Algorithm achieve the intended results.

Ahmine et al. (2019) also show that dense image alignment, which is a technique of warping one image (or sometimes two images) so that the features in the two images line up perfectly, is one of the strength of the algorithm. It is also less sense to the image noise that point-wise algorithms such as two prior discussed.

However, the assumption that all intensity changes can be explained by intensity gradients causes the method to breakdown if the gradients are random(image with random points) or when gradients are totally negligible, thus there is no structure, as in flat surfaces (Rojas).

2.4.4 Gunnar Farneback Algorithm

The Farneback method is defined to be a two-frame motion estimation algorithm that uses polynomial expansion, where a neighborhood of each image pixel is approximated by a polynomial. The algorithm only considers quadratic polynomials:

$$f(x) \sim x^T A x + b^T x c_1 \tag{13}$$

which give the local signal model represented in a local coordinate system (Husseini, 2017). Where the vector x is a 1×2 vector containing the running variables x and y.

A_1 is a symmetric 1×2 matrix of unknowns, which get the data about the even part of the signal, b_1 represents the odd part of the signal and a 2×1 vector of unknowns, and c_1 is an unknown scalar.

Writing the equation in terms of tensors of tensors gives the following system where the unknown coefficients are defined by the r_i:

$$\mathbf{f(x)} \sim [x, y] \begin{bmatrix} r_4 & \frac{r_6}{2} \\ \frac{r_6}{2} & r_5 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + [r1 \quad r3] \begin{bmatrix} x \\ y \end{bmatrix} + r_1$$

Computing the neighborhood polynomials on two subsequent images gives the displacement d in case of ideal translation. Figure 5 shows the neighborhood polynomials $f_1(x)$ and $f_2(x)$, the images taken at time t and $t + dt$ respectively.

Thus, if f_1 and f_2 could be measured, $d = [d_1, d_2]$ may be computed, which is the 2D displacement and hence the flow of the image pixel (Husseini, 2017). Equations $(14-16)$ show how the polynomial coefficients of the second frame are connected to the ones from the first frame and to the displacement d.

$$f_2(x) = f_1(x-d) = (x-d)^T A_1 (x-d) + b_1^T (x-d) + c_1 \tag{14}$$

$$= x^T A_1 x + (b_1 - 2A_1 d)^T x + d^T A_1 d - b_1^T d + c_1 \tag{15}$$

$$= x^T A_2 x + b_2^T x + c_2 \tag{16}$$

Assuming that brightness is constant in two-sequence image, the coefficients in the two polynomials can be equated as follows:

$$A_2 = A_1 \tag{17}$$

$$b_2 = b_1 - 2A_1 d \tag{18}$$

$$c_2 = d^T A_1 d - b_1^T d + c_1 \tag{19}$$

Solving for d from equation(18) gives:

$$d = -\frac{1}{2} A_1^{-1} (b_2 - b_1) \tag{20}$$

Considering that $-\frac{1}{2}(b_2 - b_1)$ can be replaced by Δb (Farnebäck, 2003), equation(20) becomes:

$$\Delta b = -\frac{1}{2}(b_2(x) - b_1(x)) \tag{21}$$

$$A(x) = \frac{A_1(x) + A_2(x)}{2} \tag{22}$$

Thus, equation(20) becomes:

$$A(x)d(x) = \Delta b(x) \tag{23}$$

Equation(23) could now be solved point-wise, however the results tend to be too noisy. It is then assumed that all the pixels in a small window say, I, behave the same. Thus, if A_1, b_1, and c_1 could be measured from f_1 for each pixel in I and A_2, b_2, and c_2 from f_2, d could be computed.

Thus, the goal is to find $d(x)$ that satisfies equation(23) as well as possible over a neighborhood I of x, or more formally minimizing:

$$\sum_{\Delta x \in I} w(\Delta x)||A(x + \Delta x)d(x) - \Delta b(x + \Delta x)||^2 \tag{24}$$

where w is the weight function for the points in the neighborhood. Upon solving for $d(x)$, the result becomes:

$$d(x) = \left(\sum wA^TA\right)^{-1} \sum wA^T\Delta b \tag{25}$$

Applying the pyramidal Lucas-Kanade method using the computed displacement $d(x)$, two-frame motion estimation can be carried out, thus motion and objects detection (Husseini, 2017). Hence the methods such as the Lucas-Kanade-Tomasi (KLT) could then be used for tracking objects (Yilmaz et al., 2006).

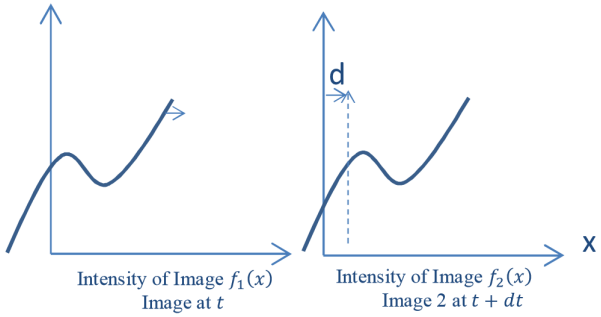

Figure 5: Neighborhood polynomials on two subsequent images, d is the displacement.

2.4.5 You Only Look Once Version 3(Yolov3) Algorithm

Lastly, yolov3 algorithm and/or neural network developed by *darknet* is considered. Essentially, yolov3 is a classifier neural network that is trained to detect object in images or video files and/or streams in real time.

The algorithm implements the following four stages:

1. Bounding Box Prediction,
2. Class Prediction,
3. Prediction Across Scales,
4. Feature Extraction.

Bounding Box Prediction

Considering the fact object detection is a technique of isolating distinct classes of related pixel values which can be then be categories as objects,

bounding box prediction is then described as a process of finding a rectangular structure superimposed over an image including all important features of a particular object residing in it. Redmon and Farhadi (2018b) have incremented on the yolo9000(a previous version) so that the system could predict boxes using dimension clusters as anchor boxes.

The network then predicts 4 coordinates for each bounding box, t_x, t_y, t_w, and t_h. It is then said that if the cell is offset from the top corner of the image by (c_x, c_y) and the bounding box prior has the width and height p_w, p_h, then the predictions corresponds to the following equations:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

As shown in the Figure 6 below:

It is also crucial to note that bounding box prediction is done both during training and testing. During training, the squared error loss is summed up to find the ground truth(the accuracy of the training set's classification).The team then define the ground truth for some coordinate prediction as $\hat{t}_*$, and the gradient, ground truth value (computed from the ground truth box) minus the prediction as $\hat{t}_* - t_*$, which can be computed using the above equations when used during training or production.

Based on the ground truth, yolov3 then predicts the score for each bounding box using the logistic regression(a predictive analysis algorithm). Then depending on the value of the score, the decision is made. The score is said to

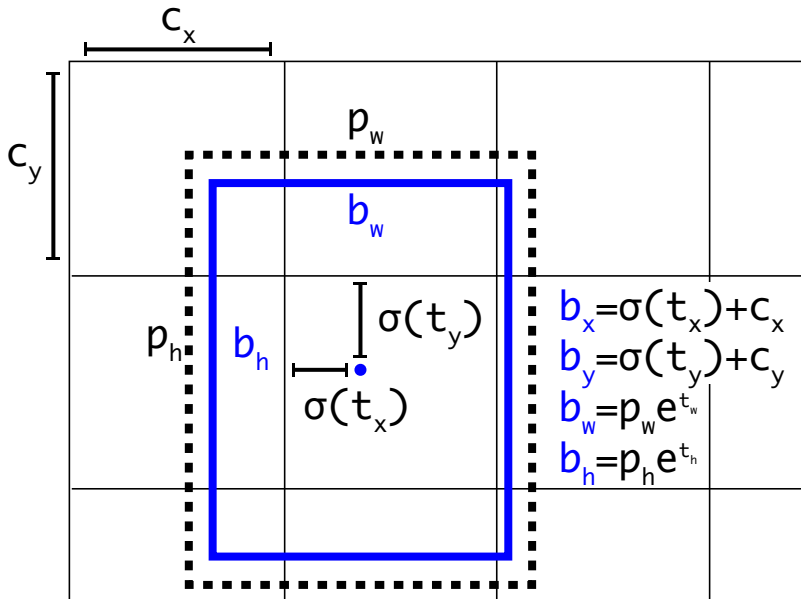

Figure 6: Bounding boxes with dimension priors and location prediction

be equal to 1 if bounding box prior overlaps a ground truth object by more than any other bounding box prior. Otherwise if the box prior is not the best but does overlap a ground truth object by more than some threshold, the prediction is ignored.

Class Prediction

Bounding Box Prediction as described earlier, it establishes score prediction for each bounding box. Class Prediction on the other hand relates to the prediction of classes of objects each bounding box has, using multi-label classification(categorizing instances into precisely one of more than two classes) (Tsoumakas and Katakis, 2007).

Using independent logistic classifiers(classifiers based on supervised learning), Redmon and Farhadi (2018b) have performed class prediction for the

yolov3 network, and binary cross-entropy loss(measuring how far away from the truth value(0 or 1) the prediction is for each of the classes and then averages these class-wise errors to obtain the final loss) is considered.

This formulation is said to help when more complex datasets are involved. These datasets could have overlapping labels such as *Women* or *Person*. Using classifiers such as *softmax* would impose the assumption that each box has exactly one class which is often not the case. Hence for performance requirements, yolov3 is implemented using logistic classifiers and binary cross-entropy loss functions which not only considered such complexities but also guarantees remarkable performances.

Prediction Across Scales

Redmon and Farhadi (2018b) affirm that yolov3 predicts boxes at three different scales and extracts features from those scales using the concept of feature pyramid networks, which are basic components used in recognition systems for detecting objects at different scales (Lin et al., 2017).

From the base feature extractor(the step to be discussed next), several convolutional layers were added of which the last of these predicts a 3-d tensor(matrix to perform operations on the image) that encodes the bounding box, the object and its class.

Following, is considering the feature map from two previous layers and up-sample it two times and take a feature map from earlier in the network and merge it with the up-sampled features using concatenation. This method allows getting more meaningful semantic information from the up-sampled features and finer-grained information from the earlier feature map and this

is critical in the detection process.

Afterwards, the team then added a few more convolutional layers to process this combined feature map, and eventually predict a similar tensor that is now twice the size. The same design is repeated one more time to predict boxes for the final scale. Implying that, predictions for the 3rd scale benefit from all the prior computations as well as fine-grained features from early on in the network.

With the completion of this stage, feature extraction is ready to be established and thus the process of detection and tracking would be viable.

Feature Extraction

Lastly, Redmon and Farhadi (2018b) implement feature extraction which is the process of describing the shape information of an image so that the task of classification becomes convenient. Hence they deployed, a hybrid network with 53 convolutional layers comes, which is the latest version of yolo; yolov3.

CHAPTER THREE

3 METHODOLOGY

The previous sections have discussed how security concerns modern age. Some of the available techniques and algorithms used in attempting to curb the aforementioned problem have also been reviewed.

In this section, the design and implementation details of the Object Tracking System (OBTS) are considered. The system's architecture, followed by the adopted algorithms, the UML diagrams, techniques to evaluate the system's performance, and the data gathering tools used for testing are discussed in that order.

Firstly, the architecture of the intended system and its labeled components is given in Figure 7. The architecture shows the components of the system labeled with letters A-F. However, the labeling does not imply the order of functionality, but the main parts of the system.

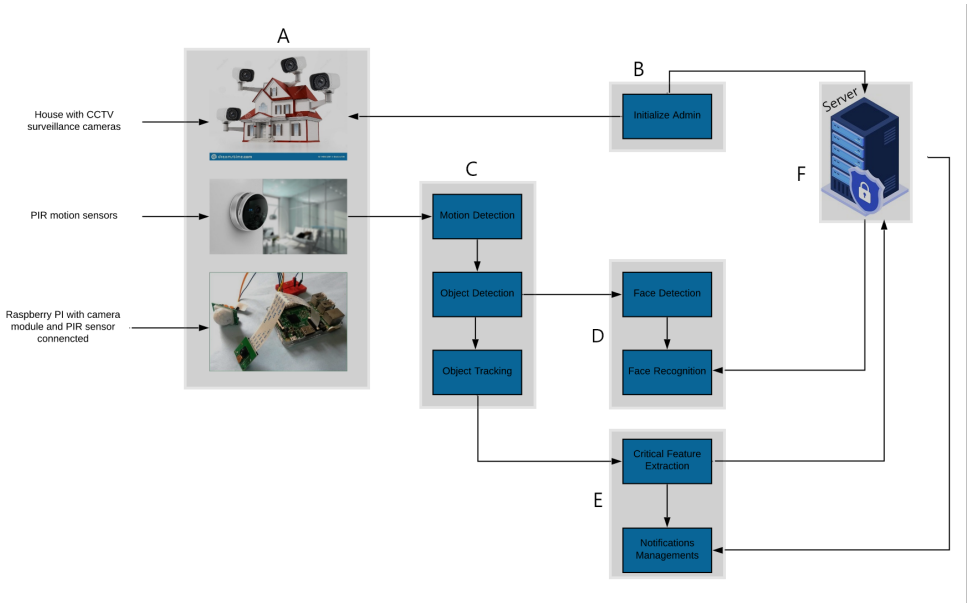

Figure 7: Architecture of the Object Tracking System

Initially, component A indicates the area of application, and label B shows the system's users and administrator initialization. The assumption is that in any set-up, home or office there is a number of people to be regarded to as familiar or known by the system, so that when performing face recognition, the system would be familiar with such faces. However, there should also be administrator(s) of the system. In this case, one administrator is considered. The admin would manage the system by logging in, update the software of the system, change different settings, receive critical notifications from the system, make major decisions to be performed by the system, and so on. All this information is stored in the server or database labelled F.

The cameras shown indicate that the place or area should have surveillance cameras installed. These will be capturing images and video scenes that will be used for image and video analysis. Shown also are the installed Passive Infrared (PIR) sensors which will be used to detect motion in the vicinity of

the system's installation area or place, and a single-board computer (Raspberry PI) with a camera module and a PIR sensor installed to the computer which will take care of motion detection, object detection and analysis in the interiors of the application area.

It then follows that there are two major concepts to consider on component A. First is that, the area should have been set up with surveillance cameras, and motion detectors. On motion detection by the PIR sensors, the system (waiting and listening for a detected signal) on the server (labeled F) fires up the cameras, and streaming is initialized. Second is the image and video capturing by the surveillance cameras for image and video analysis respectively. Here the assumption is that, it is a human whose motion should be detected and hence implying the criticality of recording the scene thereof. In simple terms, the system should guard against human beings only. Hence on motion detected, cameras begin to capture the scene.

Next, component C begin to describe the most inner-functions of the system. Here, motion detection prior mentioned is followed by object detection and/or object tracking. These processes are run on a computer hardware depicted by a Raspberry PI in A. On motion detection, the system captures several images and analyze them simultaneously. On detecting an object using the yolov3 detection algorithm, particularly a person in any of the analyzed images, the system takes note of the location (such as gate, door, passage, and so on) of the person detected and begin face classification process. If the system cannot get facial features in cases such as the person facing away the camera(s), object tracking begins.

Component D depicts the running of the face detection and recognition processes as a result of object detection in C. These two processes are carried

out by the C++ Dlib wrapper discussed in the previous section. On face detection as aforementioned, the algorithm tries to find facial features and if found, recognition process is carried out, and if the face is unknown, an alert notification to the admin is generated and sent as an SMS text and/or an email message, hence initializing scene capturing for tracking. Otherwise if the system recognizes the face, then the person is authenticated and object tracking is initialized as well. It is also crucial to note that the system can detect and analyze multiple faces all at once, and if any of the faces is recognized then all other ones are authenticated, assuming that the other people are visitors (friends or relatives).

Following, would be component E which describes two crucial concepts of the system. The feature extraction process shown on this label is determined by the object tracking process in C. Here, several activities that a human being can perform are recognized. These include; walking, running, lifting something, moving towards something, cooking, drinking, and so on. Among these activities, some such as moving towards something, lifting something, door opening would be regarded as critical, and they will be sent to the server labeled F along with their time of recognition. The criticality of the activity determines whether a notification should be generated or not. In case where the person (assumed intruder) is taking something out or away, a theft issue or notification is generated as per the time of the activity, hence sending it to the admin for decision making.

To conclude the depicted flow of functionality described by the system architecture above; there is the initialization of the system by the user(s) where the system is provided by the face images of all the users to be labeled as known, and by the admin in order for detection and tracking to kick start.

The system then uses motion sensors (PIR sensors) to detect human bodies, assuming that they are the only objects to be guarded against the user(s) valuable items. It then follows that when motion is detected, it is either the object(s) will be static or moving and these events are respectively handled by image processing and video analysis, modules to be facilitated by the yolov3 algorithm which resides on the server labeled F. On image analysis, it is either face(s) are detected and recognition is carried out, or no face(s) are detected and the system assumes that maybe the intruder(s) is/are hiding their face(s). In this case, a critical notification is generated and sent to the admin immediately. On video analysis, objects are tracked of their locations with respect to time and distance apart from each detected object, activities are uploaded to the server and notifications are generated and sent to the admin with respect to their criticality.

3.1 Adopted Algorithms

Image analysis and object tracking are the two fundamental aspects concerning this study, and the C++ Dlib library and the yolov3 deep learning algorithm are the adopted algorithms used in the implementation of the intended solution. This section considers these two algorithms and discusses how they intersect to accomplish the results.

The Dlib library with its pre-trained network called the max-margin object-detection human face detector as reflected in the early sections, is used to detect faces in an image. This implies that the input image should have had a person or people in it and this shows the first interaction between the yolov3 used for object detection and tracking, and the Dlib library (algorithm) used for face detection and recognition.

Firstly, when motion is detected by the PIR sensors as depicted in the system's architecture above, the yolov3 is loaded from the server and run to detect objects in any captured image, and when the object(s) is a person or are people then the Dlib algorithm is invoked for face detection and recognition. Keeping in mind that the so called known people (faces) should have been uploaded to the server during the initialization of the system as mentioned earlier, makes the detection and recognition possible. The Dlib algorithm goes through each of the detected faces, finding matches with the known faces, thus recognizing them if corresponding matches are found, else the unrecognized faces would be labeled as unknown, hence on failing to find even one match, the system begins to record the scene assuming that the detected person or people is/are intruder(s), henceforth the yolov3 would then begin tracking.

Second is object tracking by the yolov3 algorithm as mentioned prior. The working of the algorithm in this study can be easily conceptualized by considering a video feed whether live or pre-recorded as a series of pictures flipped at a particular rate. Technically, these are frames of images being run at a specified time(frame rate). With this idea, the algorithm is given a video as an input, analyses each frame of the video as an image by detecting objects, find their relative positions and distances between each detected object. This implies that if an object is detected in frame 1 with all the other information about it; position and distances from the other detected objects, and gets detected again in frame 2 but, on a different position or different distances from other detected objects, then the information derived from these two different frames about the same object is essentially its tracking.

Moreover, yolov3 algorithm allows finding the time instance of each detected

object in each individual frame. This is crucial in this study because when tracking objects such as a person or valuable items such a car, it necessary that the system records the location of such objects with respect to time. This would help when recognizing activities that are labeled as critical that the system would record them with their respective time instances.

Furthermore, it is also important to note the algorithm is said to have been tested in environments with massive objects and have been able to detect and track over nine thousand objects in real time with up to ninety eight percent accuracy rate (Redmon and Farhadi, 2018b). This gives this study a sense of confidence since the expected environments of usage, ones such as homes, offices, schools and banks could hardly have up to a thousand objects to track at the same time.

Finally on adopted algorithms of this study, it is necessary to mention that the discussed algorithms are the major ones used in the implementation, however there are other minor helper algorithms that have not been discussed in this section because they have not been implemented in the solution but, they have been used in comparison with the ones implemented which is one of the objectives of this study.

3.2 UML Diagrams

This study has incorporated three types of Unified Modeling Language (UML) diagrams to help describe the structure of the system from its physical and logical point of view. Following are three diagrams; the flow diagram, the sequence diagram, and the class diagram.

3.2.1 Flow Diagram

Below in Figure 8 is the flow diagram which presents the movement of data, showing the necessary logical decisions and process involved throughout the system.

Considering the shapes depicted on the flow diagram, the rounded oval shapes represent the initial and end states of the system, diamond shapes represent the logical decisions, rectangular shapes represent different processes, parallelogram shapes represent the passage of data from one process to another, cylinder shape represent the data store, and the crossed rectangle represents the dataset (yolov3) to be used for object detection and tracking by the system.

Initially, the system gets initialized by getting the user(s) face images (known images), and so does the system's administrator's and their face encodings (data about the face images) are stored in the database as shown by the diagram and the system then goes back to the initial state.

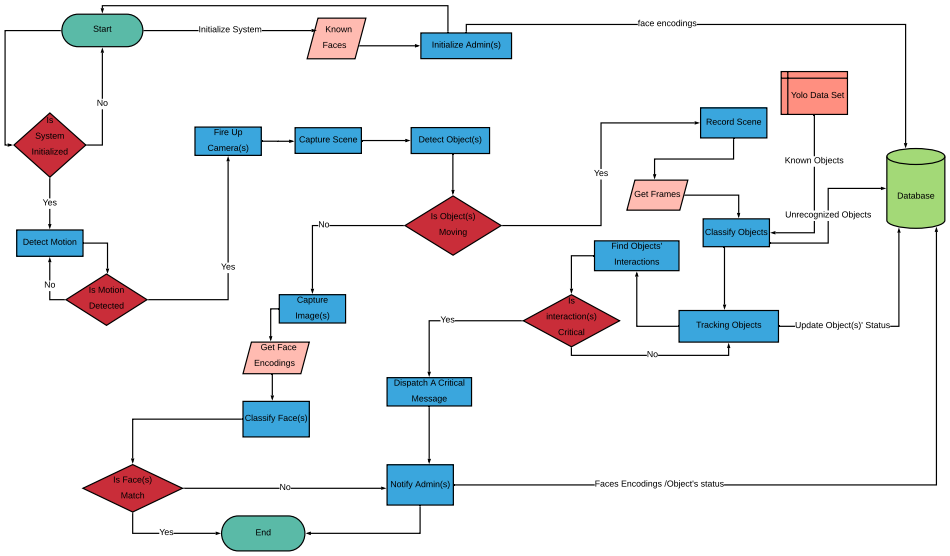

Figure 8: Flow diagram of the Object Tracking System

From the starting state, the system checks whether initialization has been properly done or not. If initialized, then the motion detection process is fired up where the system listens to any human body radiation using the PIR sensors as discussed earlier, else the system goes back to the starting state.

On motion detection, cameras are fired up and the scene gets recorded. Yolov3 is loaded to detect whether the objects are mobile or not. If the object(s) are static then then images are captured for face detection and recognition, else the scene is recorded for object classification and tracking.

In the capture images process, the system tries to get face encodings of the objects detected (assuming the objects are human beings), and classify them as either known or unknown, matching them with the ones stored in the database. If all the faces are labeled as unknown or no face image even gets

detected, then the notify admin process is invoked to generate an instant and appropriate notification, and the system will then go into the finish state. In case of recognizing at least one of the face(s), the system assumes that the recognized user(s) are in control of the situation regardless of the other unknown classifications and so the come to the end state.

On the other hand, the record scene process gets the input video stream for object detection and classification. From the video, each frame is analyzed, objects being detected and tracked. Here, the yolov3 dataset is loaded and used in classification, tracking, finding objects' interactions and detecting critical interaction such as the ones prior discussed. If any critical activity gets detected, the system then dispatch a message with urgency to the admin, thus the system instantly notifies the admin of such an event, and eventually come to the end state.

3.2.2 Sequence Diagram

Considering the diagram given in Figure 9, there shown are the major interactions between the system and the system's admin. The participation of each part (member) is shown by lines joining the participant and the respective process.

The sequence diagram shows that firstly, there should be initialization of the system. The admin provides the user(s) faces to the system in a process of logging or signing in, thus the entire initialization process includes getting the faces and admin's logging into the system monitor (system).

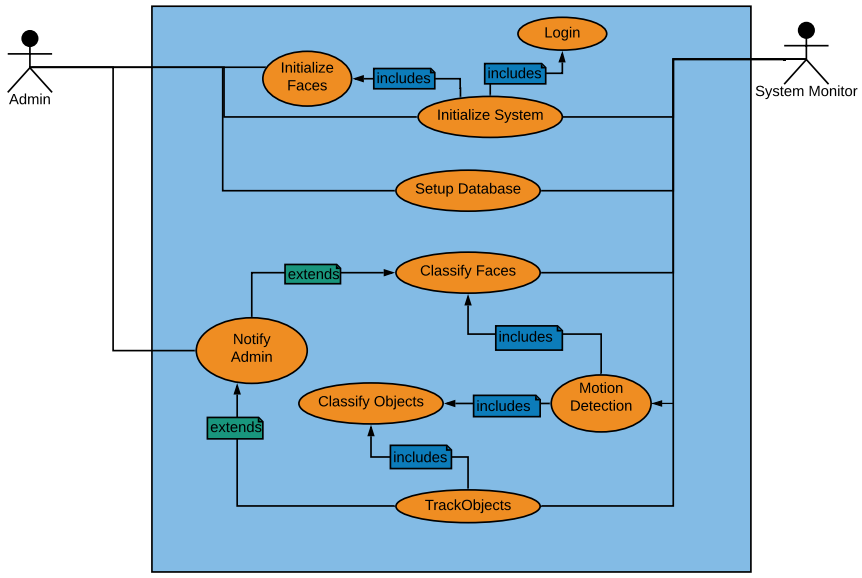

Figure 9: Sequence diagram of the Object Tracking System

The initialization of the system also includes setting up the database for user(s) information (faces, names, and other necessary details). This information is provided by the system's admin and the process is facilitated by the system's monitor, hence the two parties equally contributing to the setting up of the database process as shown in the diagram.

Next is the classify faces process, this is entirely managed by the system. This process involves detecting and recognizing faces in the captured images. From this process, it is worthy mentioning that for face detection and recognition to occur, the system should have performed the motion detection process and this is depicted by the includes relationship between the two processes in the diagram. The same implies with the notify admin process which as discussed earlier that detection of unknown faces leads to the system notifying the

admin, also can only occur when faces classification has been carried out, thus the diagram depicts an extends relationship between the classify faces and notify admin processes.

The classify objects process shown in the diagram which is facilitated by the yolov3 algorithm on a video input is entirely handled by the system and also depicts its includes relationships with both the motion detection and track objects processes. Thus showing total dependencies on these two processes as detailed in earlier sections.

Finally, the track objects process is also carried out by the system and is also included by the classify objects process. On detection of critical events, the notify admin process which again extends it would be invoked and by the participation of the admin, notifications are presented to him/her.

3.2.3 Class Diagram

Last to consider on the UML diagrams is the class diagram shown in the Figure 10. The diagram shows ten major class components and their fundamental relationships depicted by arrows, that hold in the implementation and the functionality of the system.

Following is the discussion of each of the classes, describing its major attribute and methods, and how each of the class component relates to others and how its implementation affects the functionality of the entire system.

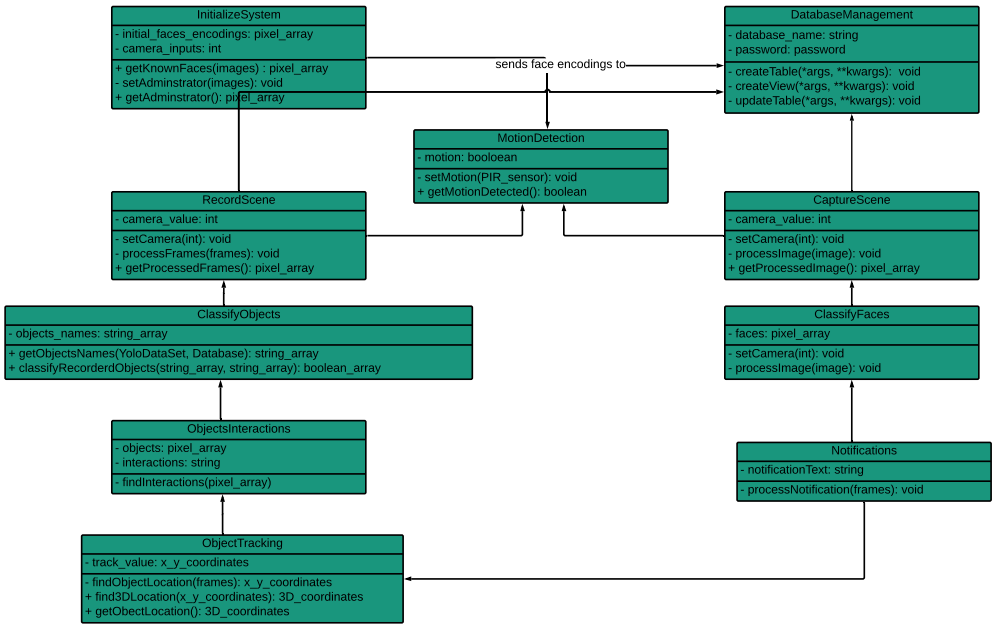

Figure 10: Class diagram of the Object Tracking System

Initialize system is the primary class component to be implemented as it holds as the entry point of the system. Here, the camera input which is an integer value determines the source of cameras, for example 0 would refer to the web cam, 1 would refer to raspberry pi camera modules, and so on. The initial faces encodings variable holds the data about the faces images (array of pixels) passed to the get known faces function. The set administrator function would initialize the system's admin upon receiving the corresponding faces encodings and the get administrator function returns the initialized admin. Upon setting up all the required information on this component, the face encodings are passed to the database management class component.

The database management class component sets up the database tables and the necessary views. It contains two fields which respectively hold the

database user's name and password for security reasons which in this case would be the systems admin's credentials. The create table and create view functions show the passed arguments and keyword arguments that would be required for any other extra information that will be needed in the creation of the database tables.

Following is the motion detection class component showing a boolean variable which holds either true when motion is detected or false when there is no motion detected. The set motion function which receives a PIR sensor signal takes care of assigning the variable and the get motion detected function returns the value of the motion variable which by default is set to false.

Next is the record scene class component which depends on the motion detection class component (depicted by the arrow). This means that on motion detection, the system initializes the cameras as determined by the camera value which is assigned by the set camera function, and it starts recording the scene. While recording the scene, the process frames function adopted from the classify objects class component gets the frames and begin to process them.

The classify objects component extended by the record scene class component shows an attribute of string array which holds all the names of the objects that can be detected by the yolov3 algorithms. These names are derived from the dataset by the get objects names function. The classify recorded objects function returns an array of boolean values based on the comparisons made between the detected objects' names and the dataset objects' names, passed as arguments.

Furthermore, there is the objects interactions module shown in the diagram.

This component is derived from the object tracking module. It sought to find interactions between tracked objects, this is the case when object gets in contact with each other, for example if a person touches or lifts up some other object.

Lastly on video analysis, is the object tracking class component. It has a track value attribute which holds a two dimensional point of each object which is calculated from each given frame. The find 3D location now uses the computed 2D coordinate to find the corresponding 3D coordinate, and this will be used to tell the real world location the object in any given scene.

On the other hand, when motion is detected, the objects might be static. In this case the system initializes the cameras to capture the scene (images) rather recording the video. Here, it is the images analysis concept that is implemented, and thus the diagram shows the capture scene module. The setting up of cameras is the same process as described in the record scene module. The difference is that on receiving an image, the process image function which is fully implemented in the classify faces module, detects faces and recognizes them. On finding unknown faces, it updates the database and pass the appropriate information to the notifications module.

Finally, the notifications class component shown by the diagram, extends both the classify faces and the object tracking modules. From these, it receives critical information that need special attention of the admin, and thus it would generate an appropriate notification and send it to the admin inform of either an email and/or an SMS text.

3.3 System's Performance Evaluation Technique

It is of importance to note that the aim of the system is to offer security to user(s) valuables. This is to be achieved by detecting objects, critical events and notifying system's administrator in real time. The implication therefore is that, the technique to be used to evaluate the system's performance will be around this aim.

Primarily, as stated in the objectives section, a number of algorithms that are available for image and video analysis have been employed and tested on the same input data to find the most efficient ones. A review of these algorithms made in earlier sections was based on the results they produced.

Secondly, reviews of other researchers' comments and results on these algorithms are carefully considered and are used in deciding which algorithms to use in the implementation of the system. Thus to say using the most recommended algorithms have shown greater performance of the system than that of the less recommended. This is true for video analysis algorithms, the yolov3 has been recommended by many researchers than is to RetinaNet, R-FCN among others, because of the trend that has shown how yolov3 is very much performant than most of its competitors (Redmon and Farhadi, 2018b).

Thirdly, different sections of the system are run on different hardware specifications (different GPU sizes) to evaluate performance in terms of hardware cost efficiency. Thus the system is tested on both minimal and maximum available hardware specifications, and so to find the minimal hardware specifications required.

Lastly, the system is tested on different internet connections to find down-

loading speeds when updating software required by the system, email sending to delivery rates to the system's performance on each connection.

3.4 Data Collection For System Testing

The system is application based and this has made the data collection process much convenient because not a lot of data is required to test the system for as much as its initial launch is concerned. The system works with pictorial data; images and videos.

Most of the data used for testing the system has been downloaded from the internet. These images and videos range from low to high resolution. Images include; face images of the same person in different places, being alone, among different people and with different objects. Videos related to office environment, home environment, and banking environment, all with different people have also been downloaded.

Finally, the system is fed with real time images and videos from raspberry pi camera modules rather than reading from disk. This is to resemble a real world environment where surveillance cameras are used as data input devices for the system to process.

CHAPTER FOUR

4 RESULTS

The implementation of the OBTS has addressed the following sections of this study:

4.1 Image Analysis

The system can detect objects such as human beings, cars, television, cell-phone, and so on in a given image. On detecting human faces the system labels the faces as either known or unknown. If known the associated name of the user is given. This is shown in Figure 11.

If the faces detected are all labeled as unknown as shown on the right side of Figure 11, the system will read the number of faces detected, the time of detection, and the location of the image, and update the database in real time as shown in Figure 12.

The system also notifies the administrator in real time by sending an email with the associated number of faces detected, the time of detection, and the captured image as an attachment.

4.2 Video Analysis

Given a video file, the system detects human beings, reading the number of people in each video frame, the range of time and date of detection to the

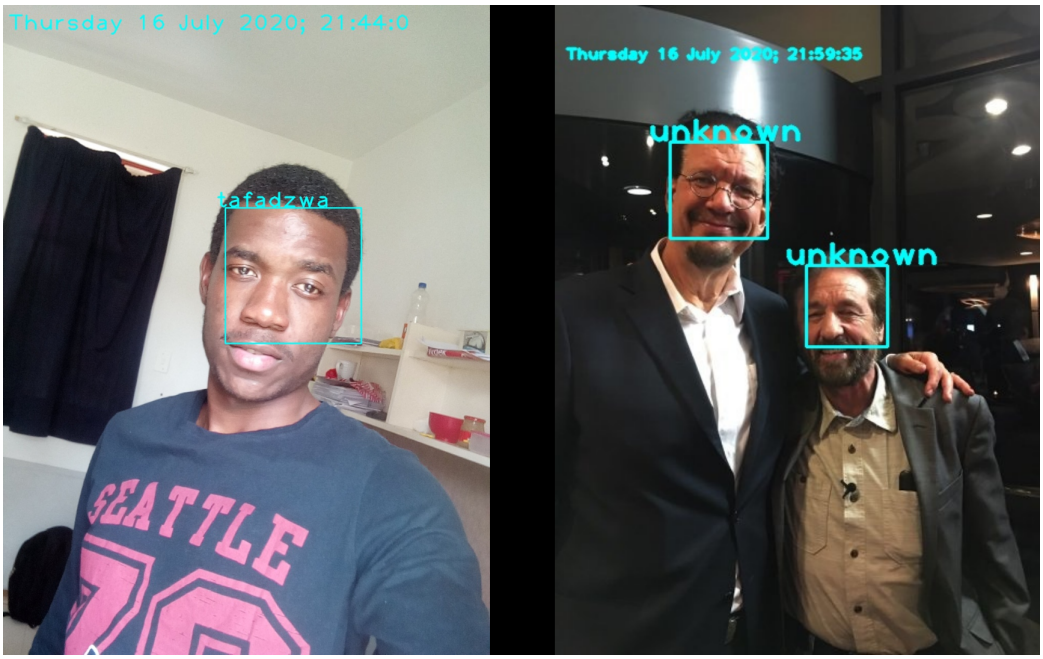

Figure 11: Face detection and recognition

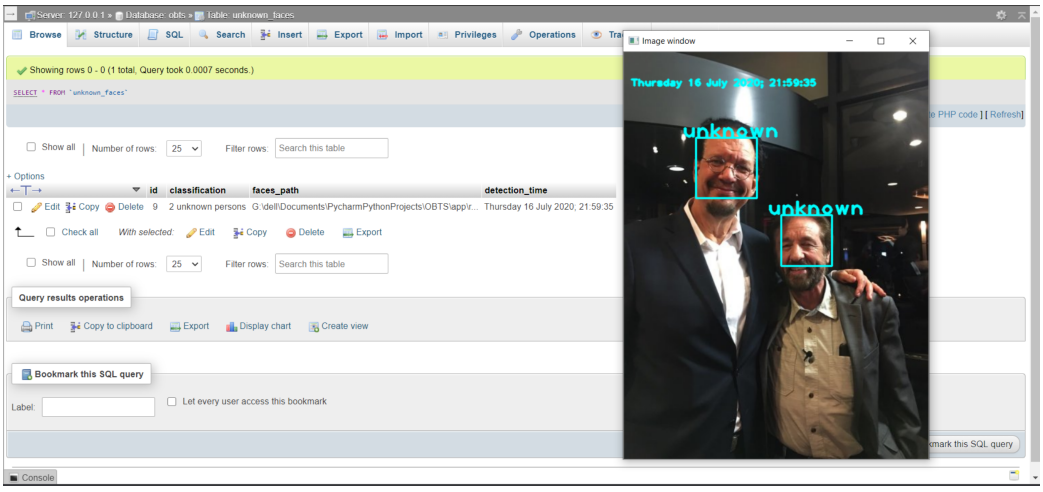

Figure 12: Updating database on face detection

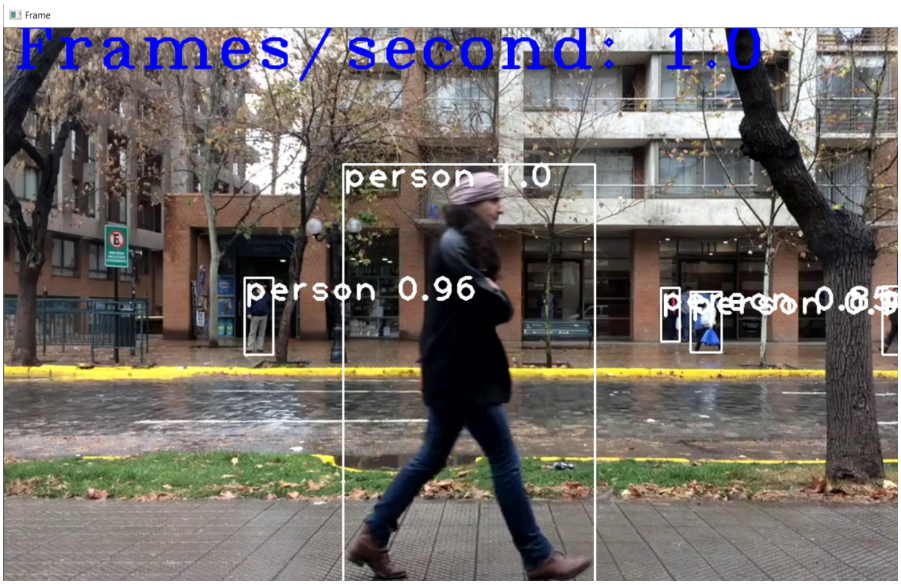

Figure 13: Human tracking

database. This is done in real time. Figure 13 shows a captured frame while the system tracks people.

The system also tracks other critical objects as highlighted in the previous section. Upon tracking them, it generates a report with respect to their detection date and time, update the database and inform the administrator in real time.

4.3 User Interfacing

An ElectronJS desktop application developed with HTML, CSS, JavaScript and Electron has been designed to help the administrator interact with the system. Windows to facilitate the system users' initialization, view image and video events and the functionality of each button has been implemented. Figures 14 and 15 show two main windows developed for user experience.

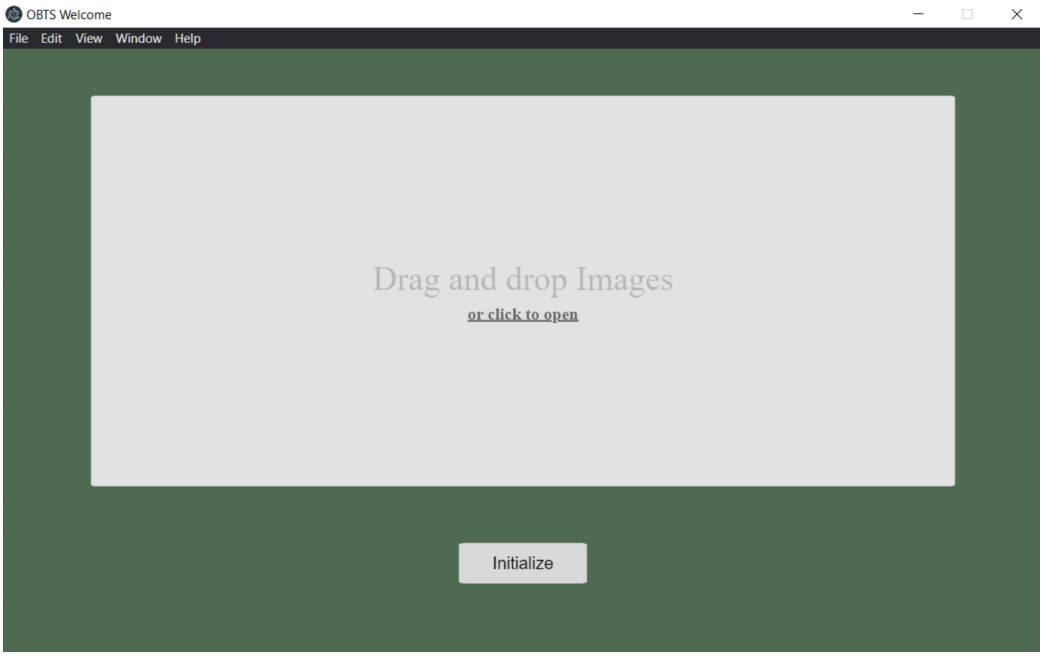

Figure 14: User initialization

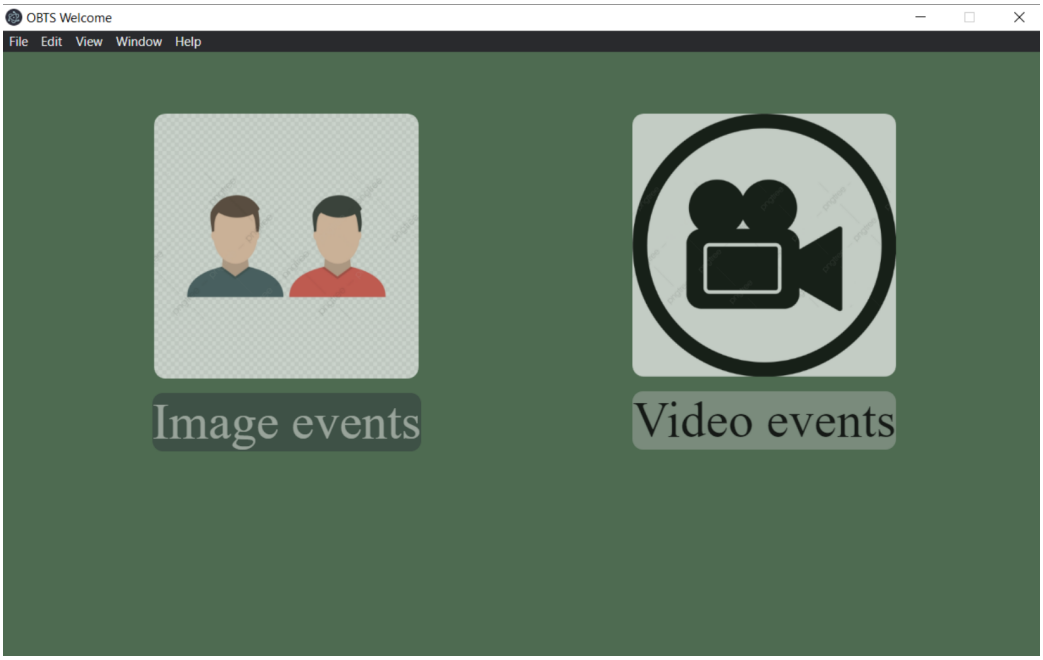

Figure 15: Images and video events

CHAPTER FIVE

5 CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

Developing security systems based on pictorial data is not a trivial process. The application areas are complex so much that object tracking is resource demanding. Variations in image quality depending on the amount of light in the images are difficult to handle using static libraries such as the C++ DLib used in this study.

The YOLOV3 neural network has been shown to be the best algorithm among it's competitors, the ones reflected in this study. On windows servers and on CPUs, YOLOV3 is a computationally resource demanding algorithm and it becomes very slow and not suitable for real time object tracking. YOLO-tiny is an optimized version of YOLO that can be optimized for such circumstances, but at a compromise on the system's performance.

OBTS is integrated with a mail system which is used for notifications, and mails are used to notify the system's admin about critical image and video events. The method is not efficient for real time applications as mails do not usually prompt user's immediate attention.

The hardware components such as cameras and motion sensors have not been implemented due to delays in shipping. The system can therefore not track objects based on their 3D view because it requires positioned cameras within defined ranges so that on detecting objects in such ranges, the respective 3D coordinates could be associated.

Face detection and recognition takes about 4 to 11 seconds on an Intel(R) Core(TM) i5-6500T CPU @ 2.50GHz with 12.0 GB (11.9 GB usable) RAM, 64 bit Operating System, x64-based processor, Windows 10 pro. This depends with the number of faces in the image. The system was programmed using the Python 3.7.5 language with the help of the openCV and numpy libraries for image and video analysis. The ElectronJS framework with the aid of the HTML, CSS, and JavaScript languages was used to implement the user interface.

5.2 Recommendations

The problem of light variations in the images could be addressed by using a neural network trained with images in environments with several light variations. This would require capturing images at different times of the day, in open and closed areas and train the system with those various images.

To improve the notifications system of OBTS, there is need to in-cooperate SMS and Phone APIs so that an SMS text or a mini voice call can be sent to the system's administrator, prompting their immediate attention. This can be done by accessing a license from network service providers such as EPTC.

Finally, proceeding in the implementation of the OBTS; there is need to consider developing customizable neural networks for both image and video analysis using libraries such as Tensorflow to be able to capitalize on performance and flexibility of the system on different servers and hardware specifications.

References

Ahmine, Y., Caron, G., Mouaddib, E. M., and Chouireb, F. (2019). Adaptive lucas-kanade tracking. *Image and Vision Computing*, 88:1 – 8.

Altun, Y., Tsochantaridis, I., and Hofmann, T. (2003). Hidden markov support vector machines. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 3–10.

Andile, N. (2018). Chicken thier shot, killed by the police. `https://bit.ly/2MS5lJo`. Police officers have shot and killed the suspect who murdered a soldier on Tuesday morning.

Arsenovic, M., Sladojevic, S., Anderla, A., and Stefanovic, D. (2017). Facetime—deep learning based face recognition attendance system. In *2017 IEEE 15th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 000053–000058. IEEE.

Berger, W. (2018). Deep learning haar cascade explained. `https://bit.ly/2G9Cw71`. Haar cascade.

Bhargava, A. and Bansal, A. (2018). Fruits and vegetables quality evaluation using computer vision: A review. *Journal of King Saud University - Computer and Information Sciences*.

Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface.

Brownlee, J. (July 5, 2019). A gentle introduction to computer vision. `https://bit.ly/32TlFPx`. Deep Learning for Computer Vision.

Deguerre, B., Chatelain, C., and Gasso, G. (2019). Fast object detection in compressed jpeg images.

Ek, C. (2017). Yolo object detection on unifi security camer. `https://bit.ly/2BFENEH`. unifi-person-detector.

EL-Barkouky, A. R. A. (November21, 2014). Mathematical modeling for partial object detection. `https://bit.ly/3avWYgu`. Object Detection.

Emami, E. and Fathy, M. (2011). Object tracking using improved camshift algorithm combined with motion segmentation. pages 1–4.

Žemgulys, J., Raudonis, V., Maskeliūnas, R., and Damaševičius, R. (2018). Recognition of basketball referee signals from videos using histogram of oriented gradients (hog) and support vector machine (svm). *Procedia Computer Science*, 130:953 – 960. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018) / The 8th International Conference on Sustainable Energy Information Technology (SEIT-2018) / Affiliated Workshops.

Farnebäck, G. (2003). Two-frame motion estimation based on polynomial expansion. In *Scandinavian conference on Image analysis*, pages 363–370. Springer.

Han, S., Zhang, M., Li, P., and Yao, J. (2015). Svm-hmm based human behavior recognition. In Zu, Q., Hu, B., Gu, N., and Seng, S., editors, *Human Centered Computing*, pages 93–103, Cham. Springer International Publishing.

Haschek, C. (2018). Raspberry pi + deep learning home security system. `https://bit.ly/342ga18`. deeplearning.

hrishioa (05 Oct, 2015). Implementing a lucas-kanade tracker from scratch. `https://bit.ly/3cbIJyi`. Understanding the basics of Optical Flow and XCode.

Husseini, S. (2017). A survey of optical flow techniques for object tracking.

Iswanto, I. A., Choa, T. W., and Li, B. (2019). Object tracking based on meanshift and particle-kalman filter algorithm with multi features. *Procedia Computer Science*, 157:521 – 529. The 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society.

Kajabad, E. N. and Ivanov, S. V. (2019). People detection and finding attractive areas by the use of movement detection analysis and deep learning approach. *Procedia Computer Science*, 156:327 – 337. 8th International Young Scientists Conference on Computational Science, YSC2019, 24-28 June 2019, Heraklion, Greece.

King, D. E. (2015). Max-margin object detection.

Kitayama, M. and Kiya, H. (2019). Hog feature extraction from encrypted images for privacy-preserving machine learning.

kkbankol ibm (2018). Analyze real-time cctv images with convolutional neural networks. `https://bit.ly/2olVOks`. ibm-cloud.

Korkmaz, S. A. and Binol, H. (2018). Classification of molecular structure images by using ann, rf, lbp, hog, and size reduction methods for early stomach cancer detection. *Journal of Molecular Structure*, 1156:255 – 263.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Liu, B., Wang, X., Lin, L., Tang, B., Dong, Q., and Wang, X. (2009). Prediction of protein binding sites in protein structures using hidden markov support vector machine. *BMC bioinformatics*, 10(1):381.

Madiajagan, M. and Raj, S. S. (2019). Chapter 1 - parallel computing, graphics processing unit (gpu) and new hardware for deep learning in computational intelligence research. In Sangaiah, A. K., editor, *Deep Learning and Parallel Computing Environment for Bioengineering Systems*, pages 1 – 15. Academic Press.

Makhubu, B. (2019). Armed robbery video goes viral. `https://bit.ly/2orUzQQ`. A video footage circulating on social media has made an armed robbery episode that took place at Sandla on Thursday night the talk of the town.

Maklin, C. (Dec 31, 2018). Mean shift clustering algorithm. `https://bit.ly/2vYxOqQ`. Data Science.

Mehboob, F., Abbas, M., Rauf, A., Khan, S. A., and Jiang, R. (2019). Video surveillance-based intelligent traffic management in smart cities. In *Intelligent Video Surveillance*. IntechOpen.

NASSIH, B., AMINE, A., NGADI, M., and HMINA, N. (2019). Dct and hog feature sets combined with bpnn for efficient face classification. *Procedia Computer Science*, 148:116 – 125. THE SECOND INTERNATIONAL CONFERENCE ON INTELLIGENT COMPUTING IN DATA SCIENCES, ICDS2018.

NOVET, J. (2015). Facebook says its object detection tech has improved by 60year. `https://bit.ly/347BId0`. Facebook News Feed.

Nowozin, S., Lampert, C. H., et al. (2011). Structured learning and prediction in computer vision. *Foundations and Trends® in Computer Graphics and Vision*, 6(3–4):185–365.

Prasad, V. V., Rayi, C. S., Naveen, C., and satpute, V. R. (2018). Object's action detection using gmm algorithm for smart visual surveillance system. *Procedia Computer Science*, 133:276 – 283. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

Pupale, R. (2018). Support vector machines(svm) — an overview. `https://bit.ly/2NazvGA`. Towards Data Science.

Redmon, J. and Farhadi, A. (2018a). Yolov3: An incremental improvement. *arXiv*.

Redmon, J. and Farhadi, A. (2018b). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ritchie, A. J., Sanghera, C., Jacobs, C., Zhang, W., Mayo, J., Schmidt, H., Gingras, M., Pasian, S., Stewart, L., Tsai, S., Manos, D., Seely, J. M., Burrowes, P., Bhatia, R., Atkar-Khattra, S., van Ginneken, B., Tammemagi, M., Tsao, M. S., and Lam, S. (2016). Computer vision tool and technician as first reader of lung cancer screening ct scans. *Journal of Thoracic Oncology*, 11(5):709 – 717.

Rojas, R. Lucas-kanade in a nutshell. `https://bit.ly/32vAsAV`.

Rosebrock, A. (March 23, 2015). Sliding windows for object detection with python and opencv. `https://bit.ly/30I2WWQ`. Machine Learning.

Saputra, K. D., Rahmaastri, D. A., Setiawan, K., Suryani, D., and Purnama, Y. (2019). Mobile financial management application using google cloud vi-

sion api. *Procedia Computer Science*, 157:596 – 604. The 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019) : Enabling Collaboration to Escalate Impact of Research Results for Society.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Scienc, O. O. D. (2018). Overview of the yolo object detection algorithm. `https://bit.ly/2MPL2fM`. Our passion is bringing thousands of the best and brightest data scientists together under one roof for an incredible learning and networking experience.

Seemanthini, K. and Manjunath, S. (2018). Human detection and tracking using hog for action recognition. *Procedia Computer Science*, 132:1317 – 1326. International Conference on Computational Intelligence and Data Science.

Shao, L., Han, J., Kohli, P., and Zhang, Z. (2014). *Computer vision and machine learning with RGB-D sensors*, volume 20. Springer.

Shinde, S., Kothari, A., and Gupta, V. (2018). Yolo based human action recognition and localization. *Procedia Computer Science*, 133:831 – 838. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

Singh, R. (Jun 8, 2019). Computer vision—an introduction. `https://bit.ly/2NkfsWe`. Demystifying the meaning behind pixel.

Some, K. (February 2, 2019). The rise of computer vision technology. `https://bit.ly/368rZ7X`. COMPUTER VISION LATEST NEWS.

Thomas, G., Gade, R., Moeslund, T. B., Carr, P., and Hilton, A. (2017). Computer vision for sports: Current applications and research topics. *Computer Vision and Image Understanding*, 159:3 – 18. Computer Vision in Sports.

Tsoumakas, G. and Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining (IJDWM)*, 3(3):1–13.

Viola, P., Jones, M., et al. (2001). Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3.

Wei, Y., Tian, Q., Guo, J., Huang, W., and Cao, J. (2019). Multi-vehicle detection algorithm through combining harr and hog features. *Mathematics and Computers in Simulation*, 155:130 – 145. International Conference on Mathematical Modeling and Computational Methods in Science and Engineering.

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13–es.

Zhang, G. and Wang, Y. (2019). Chapter 3 - machine learning and computer vision-enabled traffic sensing data analysis and quality enhancement. In Wang, Y. and Zeng, Z., editors, *Data-Driven Solutions to Transportation Problems*, pages 51 – 79. Elsevier.

Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object detection in 20 years: A survey.

6 APPENDICES

A ENCODE FACES

```
import face_recognition
import imutils

# find face encodings of a face in one image
def one_face_encodings(image_path):
    try:
        image = face_recognition.load_image_file(image_path)
        _face_encodings = face_recognition.face_encodings(image)[0]
        return _face_encodings
    except FileNotFoundError as fnf_exception:
        print(fnf_exception)


# find face encodings on each face
def multiple_face_encodings(image_path):
    try:
        image = face_recognition.load_image_file(image_path)
        _face_encodings = face_recognition.face_encodings(image)
        return _face_encodings
    except FileNotFoundError as fnf_exception:
        print(fnf_exception)
```

B CLASSIFY FACES

```
from faces_processing.encode_faces_methods import *
from faces_processing.detect_faces import DetectFaces
import face_recognition
import cv2
import numpy as np
import imghdr
import os
import datetime
from paths.paths import *
import random

class ClassifyFaces:
    def __init__(self, known_faces_path, unknown_faces_path):
        self.known_faces_path = known_faces_path
        self.unknown_faces_path = unknown_faces_path

        self.multiple_face_encodings =
                multiple_face_encodings(self.unknown_faces_path)
        self.faces_locations =
                DetectFaces(self.unknown_faces_path).face_locations()


    # compare known face and unknown ones in one image
    def compare_known_unknown_faces(self):
```

```
match_faces_list = list()

for root, directories, files in
        os.walk(self.known_faces_path, topdown=True):
    for file in files:
        if file.endswith('.jpg') or file.endswith('.png'):
            path_1 = self.known_faces_path + ext + file

            for face_encoding in self.multiple_face_encodings:
                face_comp =
                face_recognition.compare_faces(
                        [one_face_encodings(path_1)],
                        face_encoding
                )

                if face_comp == [True] and len(
                        self.multiple_face_encodings
                                ) == 1:

                    match_faces_list.append(
                    (True, file.split('.')[0]))
                    self.label_faces(
                        self.unknown_faces_path,
                        match_faces_list,
                        detection_time
                    )
```

```
                    return match_faces_list

                elif face_comp == [True]:
                    match_faces_list.append((True,
                    file.split('.')[0])
                    )
                    self.multiple_face_encodings
                        .remove(face_encoding)

                else:
                    match_faces_list.append(
                    (False, 'Person unknown')
                   )

        return match_faces_list
```

C IMAGE PROCESSING

```
# find the class ids, confidences, and boxes of detection
    def class_ids_confidences_boxes(self):
        class_ids = list()
        confidences = list()
        boxes = list()

        for detections in self.outputs():
            for detection in detections:
```

```
                scores = detection[5:]
                classid = np.argmax(scores)
                confidence = scores[classid]

                if confidence > 0.5:  # object detected

                    width = self.read_image()[2]
                    height = self.read_image()[1]

                    centerx = int(detection[0] * width)
                    centery = int(detection[1] * height)

                    w = int(detection[2] * width)
                    h = int(detection[3] * height)

                    boxes.append([x, y, w, h])
                    confidences.append(float(confidence))
                    classids.append(classid)

        return [classids, confidences, boxes]
```

D HUMAN TRACKING

```
import cv2
from image_processing.image_processing import ImageProcessing as ImgP
from video_processing.human_detection import HumanDetection
import datetime
```

```
# human tracking
class HumanTracking:
    def __init__(
            self,
            yolov3_weights,
            yolov3_config,
            yolov3_coco_names,
            image_path,
            video_path,
            yolo_tiny_videos_model_path=None,
            video_output_path=None,
    ):

        # get processed image with detected objects
        self.processed_image = ImgP(
                yolov3_weights,
                yolov3_config,
                yolov3_coco_names,
                image_path
        )

        self.human_detection = HumanDetection(
                yolov3_weights,
                yolov3_config,
                yolov3_coco_names,
```

```
            video_path
        )

        # get the labels from yolov3 coco names
        self.labels = self.processed_image.labels

        # get the video path
        self.video_path = video_path

        # get the yolo tiny videos model path
        self.yolo_tiny_videos_model_path = yolo_tiny_videos_model_path

        # get the video output path
        self.video_output_path = video_output_path

    @staticmethod
    # draw rectangle around an object
    def draw_rectangle_around_object(
        image, xy_coordinate_1,
        xy_coordinate_2,
        rbg_color,
        thickness
        ):

        x_1, y_1 = xy_coordinate_1
        x_2, y_2 = xy_coordinate_2
```

```
        red, blue, green = rbg_color

        cv2.rectangle(
            img=image,
            pt1=(x_1, y_1),
            pt2=(x_2, y_2),
            color=(red, blue, green),
            thickness=thickness
        )

    # track human on a video
    def track_human(self):
        if 'person' in self.labels:
            self.human_detection
                .object_detection_objects_using_yolo()

            print(self.human_detection
                          .get_times_of_detection()
                         )

            print(self.human_detection
                          .get_numbers_of_people_detected()
                         )
```

E NOTIFICATIONS

```
import sys
import shutil
import os

sys.path.append("..")

from .authentication import *
import smtplib
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.image import MIMEImage



class EmailingHandler:
    def __init__(self, recipient, path_to_image):
        self.recipient = recipient
        self.path_to_image = path_to_image
        self.copied_image_name = None

    def copy_image_to_current_dir(self):
        image_name = str(self.path_to_image).split('/')[1]

        if image_name is not None:
            self.copied_image_name = image_name
```

```
            shutil.copyfile(self.path_to_image, str(os.getcwd()) + '/'
        else:
            pass

        return image_name

    def send_email(self, subject, body_content):
        msg = MIMEMultipart()

        # prepare an email
        msg['From'] = user_name
        msg['To'] = self.recipient
        msg['Subject'] = subject

        msg.attach(MIMEText(body_content, 'plain'))

        # attach an image
        image_name = self.copy_image_to_current_dir()

        if image_name is not None:
            image_data = open(str(image_name), 'rb').read()
            image = MIMEImage(
                image_data,
                name=os
                        .path
                        .basename(str(image_name)
                )
```

```
        )

        msg.attach(image)

        os.remove(str(image_name))

        # send the email
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            try:
                print('sending...')
                smtp.login(user_name, password)
                smtp.send_message(msg)
                print('email message sent successfully...')
            except Exception as e:
                print(e)

    else:
        print('could not attach image...')
        with smtplib.SMTP_SSL('smtp.gmail.com', 465) as smtp:
            try:
                print('sending...')
                smtp.login(user_name, password)
                smtp.send_message(msg)
                print('email message sent successfully...')
            except Exception as e:
                print(e)
```

F DATABASE OPERATIONS

```
import mysql.connector


# noinspection SqlNoDataSourceInspection

class DetectedFacesDatabaseOperations:
    def __init__(
        self,
        classification_value,
        path_to_faces,
        detection_time
        ):

        self.database = mysql.connector.connect(
            user='user_name',
            password='user_password',
            host='localhost',
            database='database_name'
        )

        self.path_to_faces = path_to_faces
        self.classification_value = classification_value
        self.detection_time = detection_time

    # check if the unknown faces table exists
```

```
def is_unkown_faces_table_existing(self):
    cursor = self.database.cursor()
    cursor.execute("SHOW TABLES LIKE 'unknown_faces'")

    if cursor.fetchone():
        return True
    else:
        return False

# create the unknown_faces table
def create_unknown_faces_table(self):
    self.database
            .cursor()
            .execute("CREATE TABLE unknown_faces(
                    id INT AUTO_INCREMENT PRIMARY KEY,
                    classification VARCHAR(50),
                    faces_path VARCHAR(255),
                    detection_time VARCHAR(50))"
            )

def insert_image_into_the_database(self):
    try:
        cursor = self.database.cursor()

        add_unknown_face =

            "INSERT INTO obts.unknown_faces (
```

```
                classification,
                faces_path,
                detection_time)

        VALUES (%s, %s, %s)"

    if len(self.get_unknown_faces_table()) != 0:
        paths_list = [
                row[2] for row in
                        self.get_unknown_faces_table()
        ]

        if self.path_to_faces not in paths_list:

            add_unknown_face_value =    (
                        self.classification_value,
                        self.path_to_faces,
                        self.detection_time
                )

            cursor.execute(
                add_unknown_face,
                add_unknown_face_value
            )

            self.database.commit()
```

```
                print("Case 1: Record added successfully...")

            else:
                add_unknown_face_value = (
                        self.classification_value,
                        self.path_to_faces,
                        self.detection_time
                )

                cursor.execute(
                        add_unknown_face,
                        add_unknown_face_value
                )

                self.database.commit()

                print("Case 2: Record added successfully...")

            cursor.close()

        except mysql.connector.errors.ProgrammingError:
            if not self.is_unkown_faces_table_existing():
                self.create_unknown_faces_table()

            self.insert_image_into_the_database()

    # get data from unknown_faces table
```

```
def get_unknown_faces_table(self):
    cursor = self.database.cursor()
    cursor.execute("SELECT * FROM unknown_faces")

    return cursor.fetchall()
```