Journal of Theoretical and Applied Information Technology
30th September 2021. Vol.99. No 18
© 2021 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A HAAR CASCADE CLASSIFIER BASED DEEP-DATASET FACE RECOGNITION ALGORITHM FOR LOCATING MISSING PERSONS

ANTHONY U ADOGHE[1], ETINOSA NOMA-OSAGHAE[2], KENNEDY OKOKPUJIE[3], UDUEBHOLO EMMANUEL EDOSE[4]

[1,2,3]Department of Electrical and Information Engineering, Covenant University, Ota, Nigeria

[1]anthony.adoghe@covenantuniversity.edu.ng, [2]etinosa.noma-osaghae@covenantuniversity.edu.ng, [3]kennedy.okokpujie@covenantuniversity.edu.ng,

ABSTRACT

A countless number of persons, including children, teenagers, adults, and mentally challenged people, go missing every day. Some are victims of kidnap and human trafficking, while others got missing in unfamiliar places. Effectively identifying people has always been a fascinating subject, both in industry and research. The majority of proposed solutions have not considered the possibility of using cameras in public places for detecting the faces of missing persons in real-time. Therefore, this paper presents implementing a Haar Cascade Classifier Based Deep-Dataset Face Recognition Algorithm on cameras to locate missing persons in public and notify law enforcement of missing persons found. This research study employs the in-depth learning approach using Open Computer Vision to automate searching for missing persons using public cameras, thereby improving security, safety, and reducing the time taken to find missing persons. The implemented system is the solution to a closed-set problem where the proposed algorithm assumes a deep dataset gallery of the trained face image of missing persons. The real-time implementation of the trained face recognition algorithm gave an average experimental accuracy of 72.9%.

Keywords— *Face Recognition, Missing People, Haar Cascade Classifier, OpenCV, Deep Dataset, Rasberry Pi*

1. INTRODUCTION

People these days, most especially women and children are highly vulnerable to kidnap. A lot of persons around the world are reported missing daily. Traditional methods of finding missing persons using the media may cause more harm than good. This may be because when the case of missing persons results from kidnapping, the kidnappers continue to evade detection as soon as it is known that a search has been launched for the victim. Families may also not feel comfortable about being advertised in the media due to a missing person incidence.

The human face has one of the unique characteristics of human recognition. In recognising missing persons and victims of human trafficking in public areas, technologies like the fingerprint and iris scanners will not be helpful, because unlike facial recognition, it requires a user to be physically present for scanning either by the placement of the finger on a hand-rest for hand geometry detection or by standing in a fixed position in front of a camera for iris identification. The human face can be captured passively by a real-time surveillance face recognition system, and this is beneficial for locating missing persons. Passive Closed Circuit Television (CCTV) cameras are employed by the legal bodies, prosecutors, secret agents and regulatory bodies to find criminals, and it has been found to be time-consuming, inefficient and tedious. The advent of Artificial Intelligence (AI) has made it possible to give cameras some form of intelligence for real-time surveillance applications. Real-time face recognition technology using deep learning can be used to detect faces in public and recognise people at once quickly. To avoid likely problems caused by mass advertisement on media, face recognition surveillance cameras can be installed at specific locations outdoors, to recognise and find missing people through a live video feed. The principle behind a typical deep learning face recognition system is that, after the algorithm has been trained, a face image is fed as input into the algorithm, first for the feature extractor to extract characteristic features which will then be compared with a stored gallery of face images for face matching. Face identification and verification are carried out during face matching. Face identification checks whether a pair of face images belong to the same person, and the verification checks whether the identified face matches with the one in the database. Face identification is treated as an open-set problem in the real world and not always closed because, for the closed-set, the algorithms assume there is a stored gallery of face images in the database for verification. This research study considers the closed-set problem case with faces of the missing persons stored in a database. This research study employs Computer Vision, because it assists with efficient detection and recognition of faces, especially in public areas with the crowd and high human traffic. For data collection, the choice of datasets depends on the neural network being trained. For instance, a deep network will perform well with deep datasets, while a shallow network will perform well with wide datasets. The deep datasets contain more changes in pose, and expressions, while the wide dataset has larger variations based on a large number of unique identities. In pre-processing, detection, alignment and face cropping is applied to the training dataset before training to enhance the performance of the system. Face alignment is essential because it identifies the geometric structure and orientation of human face images digitally. This is essential for face recognition. The aim of face alignment is to transform the face pose to a canonical view by locating key facial points in the image. Passive CCTV cameras installed in commercial buildings and in public places cannot detect and recognise faces. Using real-time video surveillance for face recognition, the faces of missing persons can be identified and detected effectively in a continuous sequence. The purpose of using deep learning in face recognition is to find a solution to the real-time face recognition challenge. Real-time face recognition using deep learning can tackle this problem by detecting faces in public areas and then recognising these faces by matching the extracted and selected features to the images stored in the database. After this is done, a security alert is sent to the security team with the exact time, date and location of the missing person. With the advent of AI, deep learning algorithms can detect and recognise faces in real-time. The purpose of this research study is to develop a face recognition algorithm that is capable of identifying or recognising the faces of missing persons from live video feeds from CCTVs. The objectives are to implement a real-time video face recognition algorithm. Create a database gallery for missing persons to be used by the law enforcement agency. Implement the algorithm created using deep learning frameworks, OpenCV libraries, and the Haar Cascade Classifier. Develop a prototype capable of giving law enforcement agents the time and location of discovered persons.

In this study, face detection and recognition system using python along with OpenCV package is proposed. The proposed system contains three modules, which are detection, training, and recognition. Basically, the detection module detects the face which gets into the field of vision of the camera and saves face in the form of an image in JPG format. Then the training modules train the system using Haar cascade algorithm.

1.1 Face Recognition System Design

The proposed model algorithm was coded with python programming language and some of its libraries like OpenCV (Computer vision), Haar Cascade Classifier to train the network, using Jupyter notebook and Visual Studio Code. The algorithm was trained on sample face images. The database was created. After this was done, the algorithm was deployed to the hardware components consisting of the Raspberry Pi and the Pi camera node. The camera continually monitors the environment, detects a human face and then runs the frame processing locally in the Raspberry Pi by using Dlib library and OpenCV installed inside the Raspberry Pi.

1.2 Face Detection and Face Tracking

The pi camera connected to the raspberry pi was able to detect human faces and showed this by drawing a rectangular frame around the face. The algorithm was also able to track face movements within the confines of the camera coverage

1.3 Face Positioning and Alignment

The detected face is correctly aligned by facial position to determine facial feature points such as eyes, nose, mouth, within the confines of the face area detected, and tracked.

1.4 Face Feature Extraction

Once the captured face has been detected and tracked, the algorithm extracts key facial points, which are the features of the detected face. This will be very useful for recognising detected faces during testing.

1.5 Face Recognition and Matching

This phase involves building a database having the face images of missing persons to train the face recogniser. Once this is done, real faces are detected and recognised by the system.

1.6 Email Notification System

When faces representing missing persons are detected, recognised and matched with a trained face image in the database, it will automatically send an email notification to the email address entered into the program. The email will carry both a message alert and an attachment of the captured image of the person found. The notification system uses, "smtplib", a native python library to send emails and MIME protocols to send attachments. The camera continuously looks for a face, and once a face is detected, the frame will be analysed and compared with the trained gallery of face images. If a matching frame is found, it is sent as an email notification to law enforcement.

These are the major contributions of this research study to the existing body of knowledge in missing persons' identification research:

- A real-time video face recognition algorithm.
- The use of Haar Cascade Classifier for deep dataset classification, thus reducing the complexity of the developed algorithm.
- Addition of a pan and tilt function to the camera in real-time to track faces using two servos. The Pi camera coupled with a pan-tilt servo controller for face tracking, panning 180 degrees (-90 to +90) and tilting 90 degrees (-45 to +45) up and down. This allowed the Pi camera rotate in a balanced manner to track the faces of persons in a public space.
- The integration of an email notification system to alert law enforcement of the missing persons found, with evidence of photos captured and name identity.

In order to develop applications that require face recognition technology, the findings obtained from this research study can be used as a primary reference and guidance for other researchers. This research study can also be used as a short reading to obtain the basic methodology for implementing deep learning face recognition systems capable of recognising and finding missing persons in public areas from live video feeds.

Although the research study was carefully planned, the limitation of this research study was accuracy. The accuracy was affected by some unconstrained factors such as facial expressions and illumination. In tackling the illumination problem, stored faces were pre-processed and stored with different illumination intensity to increase the accuracy of recognition. The varied expression problem could not be solved satisfactorily as getting other facial expressions of missing people was not easy. This sometimes made a detected face having a match in the database not to be recognised. Also, faces undergo some form of ageing and may not always be frontal to the camera. These variations cause some form of inaccuracy in the system. The use of sunglasses also affected the performance of the system algorithm, including changes in makeup and appearance. Pose variation largely affects the system performance, because the difference between faces of the same person is large under varied poses. The issue of privacy was raised as a concern among users and citizens who though are aware of the security and tracking benefits, also contend with the idea of face recognition. Some citizens desire to remain anonymous all through their lives without having these active surveillance system detecting and tracking their faces, by law enforcement agents.

Law enforcements prefer biometric technologies like fingerprint scanners, because they operate with due permission from the individual, unlike the face recognition technology that operates clandestinely and identifies anonymous individuals. Many consider this to be an invasion of privacy. The face recognition technology may be misused by Law enforcement agents, who may readily have access to personal data of individuals, unlike the traditional name-checking and verification procedure. The challenge lies in these enforcement agents, creating a security mechanism that prevents misuse. Wearing of face masks posed an enormous challenge as persons will not be easily recognised with masks on. Facial side-views posed a significant challenge also in terms of accuracy. The system will have to be improved to converted faces from side to canonical views with the aid of multiple cameras. The remaining part of the research study is organised; thus; The second section is a review of the existing literature on the research study. The third section is the methodology used for implementing the proposed solution. Details about the workflow and simulations carried out are included in this section. The fourth section details the results and findings of this research study. Furthermore, these findings are discussed clearly. The research study is then concluded with suggestions for future work and the acknowledgement of valuable inputs to the research study.

2 LITERATURE REVIEW

In 2019, B.D. Balar and his team presented a paper, proposing a system that would help the police find missing persons using face recognition. The system makes use of a deep convolutional neural network and a linear SVM classifier for face matching and recognition [1][2]. The classifier was trained to take the measurements from test images to give the closest match as output. A web interface was built using the Flask Application Programming Interface (API) interface to give a better user experience. When a missing person is found by the public, the face image of the person is uploaded to the database through the portal. The algorithm then analyses the face for a match, and details of the search result are sent back. If a match is not found, the administrator registers the missing person through the online portal. The proposed system did not use smart cameras in public places to detect faces of missing persons' in real-time. However, the proposed model had an accuracy of 99.8%.

In 2018, Peace Muyambo proposed the use of LBPH (Local Binary Patterns Histograms) algorithm for face recognition to find missing people [3]. The method was a hybrid that was proposed to tackle the problem of pose variation and illumination. It was suggested to improve the recognition rate and accuracy. One significant advantage of the proposed algorithm was its robustness to varied illumination. It also performed better in a controlled environment. The proposed algorithm was implemented on a camera to capture image frames, which was detected from video feeds. The result, however, showed a 67.5% face recognition rate.

In 2016, Professor Sumeet Pate and his team proposed a system that used Line Edge Method for recognising the faces of missing people and wanted criminals [1][4]. The proposed approach was robust to lighting and pose variations. The robustness was achieved at a computational cost because the algorithm detects facial points and calculates the Line Segment Hausdorff (LSH) distance in order to transform facial features into technical forms. The researcher and his team implemented the algorithm using remote CCTV cameras to detect and recognise the faces of missing persons. As soon as a face was recognised, the system sent an SMS alert to police stations close to the location of the person found. The proposed system had an accuracy of 85%.

In 2015, Thomas and Andrew proposed a system that makes use of a mobile-based web service to search for missing people [5]. Guardians register and report their missing ones on the platform to be contacted if found. The method was quite traditional and did not use any form of deep learning to facilitate the process. It was therefore inefficient, but very simple. The result, however, showed that cases of missing persons that were reported and logged on the platform had a 72% chance of being found.

In 2012, A. Mishra et al focused on building a system that would identify one or more faces in a given still image or video of a particular scene, by using correlation method to find the highest correlation score between the video or still image and stored images in the database [6]. The researcher viewed face recognition as a two-dimension problem for detecting and recognising faces in a projected set of array or face space, known as Eigenfaces, whereby variations between the face images are encoded. This technique is such that a new face can be identified in an unsupervised manner at once. The researcher also improved the common feature extraction technique by introducing a unique algorithm. When a new face image is fed into the system, it is compared with faces stored in the database for a match. In the proposed methodology, the set of images used for training is fed into the system to find the Eigenspace based on the Principal Component Approach. The average is taken to form the covariance matrix of eigenvectors, which is the mathematical representation of the various facial features. The Euclidean distance of the face image is measured by a linear combination of the face image and the Face space. A match is found when the vector falls within a particular threshold. The Eigenface approach is fast and straightforward and works well under constrained conditions. One limitation encountered with this approach was the difficulty in handling varying facial expressions of the same person.

In 2016, Chantar et al. proposed a real-time surveillance system for home monitoring through live video recording using OpenCV and Raspberry Pi [7]. Notifications were sent to the homeowner whenever movements are detected within the home. Intruders were detected at once as the algorithm had the capacity to distinguish known members from intruders. This proposed system had both face detection and recognition capabilities. The researchers in their work noted that images in the training data set should be taken with different facial expressions and varied lightings so that known members of the family or house are not identified as intruders. Also, whenever a face is brought in front of the camera, it is detected and compared with stored faces in the database. If a match is found, it continues the monitoring loop. Else, the current frame is captured and sent as an email alert to the homeowner. Sending Email Notifications was the final step. The algorithm sends a notification having both a message alert and the captured image of the intruder.

In 2019, Santhoshkumar and Geetha focused on one of the applications of intelligent traffic management, which is vehicle recognition [8]. Unlike, the traditional machine learning-based algorithm for vehicle recognition, which had poor performance, the researchers proposed a deep convolution neural network to achieve better performance. The flaw of conventional machine learning algorithms like low recognition accuracy was overcome using a nine-layer neural network with frameworks such as Caffe and Torch to improve accuracy and performance. The first part consists of the convolution layers, responsible for feature extraction, the pooling layers, accountable for parameter tuning and speed, and the fully-connected layers. The second part has the activation function, and one of such activation functions is the Rectified Linear Unit (ReLU). ReLU is an activation function that saves a lot of computation time and effort, reduces parameters and solves the overfitting problem common with neural networks. It uses the dropout technique to set the output of each hidden layer to zero with a probability of 0.5. In the experiment, a dataset of 90000 vehicle images was used. The dataset was divided into 72000 training images and 18000 testing images with 998 classes.

Zhang et al. proposed a method face recognition that was based on Linear Discriminant Analysis (LDA). The LDA was applied on component-based face representation to extract discriminant features of the components and compared to the traditional Fisherface method [9]. These feature vectors were combined as a single vector and fed into the LDA for final face description. This technique showed high efficiency on face image dataset. In this method, a face image is divided into four overlapped components, and LDA is applied to each of the components to learn the most distinguishable feature vector. The experiment made use of 1196 images and showed a recognition rate of 91.8% using the LDA.

In 2019, Mansoor Iqbal et al. presented a deep Learning approach for face recognition based on angularly discriminative features [10]. The researchers came up with a hybrid angularly discriminative classification by combining additive cosine and multiplicative angular margin to improve angular softmax efficiency. Softmax played a vital role in increasing the discriminating capability of the algorithm. The proposed approach achieved a 99.77% accuracy when tested on Labeled Face in the Wild (LFW) dataset.

In 2019, Guo and Zhang presented a concise overview on specific problems in face recognition such as pose estimation, illumination, and varied expression[11]. It summarises 330 contributions in face recognition using deep learning. At the end of the experiment, the researchers observed that the deep learning approach still requires expert human intervention in tuning interdependent hyperparameters. Interdependent in the sense that, changing weight in one hidden layer can alter the weights in the next layers. This is so because the size and depth of the network interact with other hyperparameters, and it's usually difficult to tune one independent of the other. The two ways of tuning the hyperparameters are Regularisation and Pruning. Regularisation adjusts the network with a small number of parameters, while the second deletes redundant parameters and features. Another practical approach is the incremental way of starting off with a small network and gradually increasing the size of the network.

3 METHODOLOGY

In the last section, a detailed explanation of the key concepts on which this research study was based has been done, as well as detailed analysis of related works. Hence basic knowledge of the study's purpose is known. This section would provide information on the system design and methodology used to model the objectives of this research study in detail, as well as considerations for real-life implementation. The design set-up of a face recognition system is shown below using Raspberry Pi, Pi Camera, Pan and Tilt servo controller, Python and Computer vision. An essential add-on to the Raspberry Pi is the pan and tilt camera because it uses two servos that enable the camera to tilt left to right and then up and down simultaneously and this allows the detection of face movements by tracking and recognising in real-time. The components of the proposed face recognition system for missing persons is shown in Figure 1. The Raspberry Pi is a small single-board computer containing a multicore processor, Graphics Processing Unit (GPU), and Input/output (I/O) peripherals inside it. This unit was responsible for running the Haar cascade classifier based deep dataset face recognition algorithm for locating missing persons.

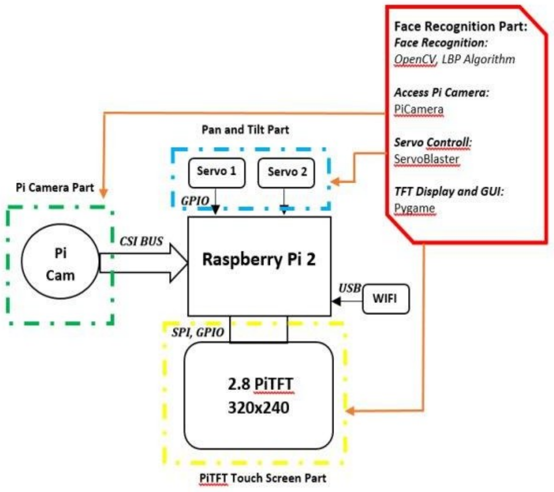

Figure 1. Face Recognition System for Locating Missing Persons: The Hardware

The Raspberry Pi Camera is a custom add-on board designed for the Raspberry Pi, which has a high-quality image sensor of 8 megapixels. This was used to capture face images needed for processing by the Raspberry Pi. The High Definition Multimedia Interface (HDMI) screen display is a touch screen monitor which comes with a touch pen for easy use. It works with Raspberry Pi 4 and 3B+. The HDMI interface was used to display detected and recognised face images. The Cana Kit 2.5A Raspberry Pi power supply delivers a full 2.5A and an output voltage well within the USB minimum voltage specifications. The Pan-Tilt HAT Servo Controller was the unique addition for face movement tracking. The HAT and its on-board microcontroller allow independent driving of the two servos (pan and tilt), thereby rotating the Pi cameras.

3.1 Software Design Specification

Python is a high-level language for several programming applications such as machine learning, data science and web development. It has useful advanced scientific and powerful functional libraries that were useful for this research study. Open CV is a Python library for real-time computer vision. It was the library responsible for providing computer vision function of the research study. The Python Integrated Development Environment used for this research study was Virtual Code (VS Code) which supports Python and OpenCV functionalities. Linux is an Operating System (OS) for industrial and security applications, which was very vast in support features and applications for this research study.

3.2 Steps involved

In the course of the study, it was challenging to control two servo motors to enable the Pi camera pan and tilt in real-time. The algorithm had to be optimized to make it possible for the camera to keep up with the two servos. The prototype of the Pan-tilt Raspberry Pi Camera enabled the tracking and recognition of face images in public places through the implementation of the following steps:

- A File folder served as the database for the faces of missing persons.
- The Pi camera first detects and captures a face, while implementing the tracker that follows face movement (180 degrees left to right and 180 degrees up and down).
- The captured frame from the real-time feed through the Pi camera was sent to the database and compared with all other faces in the database file manager system.
- The algorithm matches the input frame from the Pi camera with the face of the missing person previously stored in the database.
- After matching, the frame was sent to the email address of the nearest security station to the location of the missing person.

The images shown in Figure 2 formed the deep dataset of missing persons that need to be recognized. The deep dataset contained fewer unique images with a variety of pose, expressions, and illumination. The database was populated using sing the Pi camera to enroll face images of volunteers that were assumed to be the face of missing persons. This is so because there are really no standard databases of missing persons available for use.

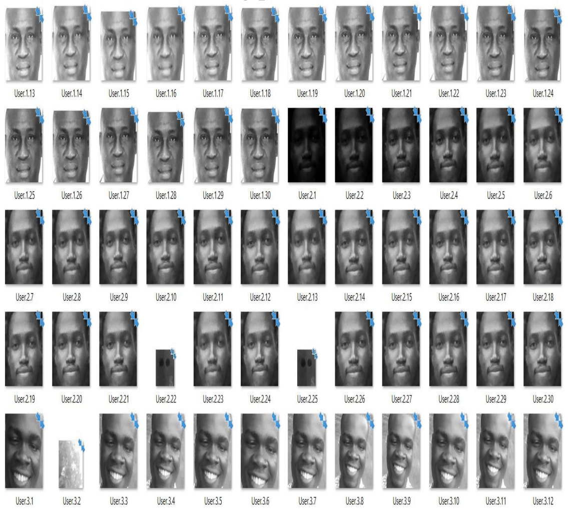

Figure 2: Face-Image Deep Dataset for Training

Before Beginning with capturing input frames using a Raspberry Pi camera, the workflow consists of detecting faces, embedded computing, and comparing the vector to the database via a voting method, as shown in Figure 3. A 128-d vector was also computed using a deep neural network to quantify each of the face images in the database. Inside the code loop, there are a few important events.

- Extraction of the missing person's name from the path
- Loading the image and converting the image to Red, Green and Blue (RGB) components
- Computing of face embedding and adding it to Known Encodings with the name added to a corresponding list element in Known Names.

The face image encodings along with the names are exported for use in the face recognition script. A dictionary was constructed having two keys along with respective values of Known Encodings and Known Names. This line was stored in a data variable and exported. After running the encode_faces.py, the script was called encodings. Pickle was then created, containing the 128-d face embedding for each face in the dataset. The pi_face_recognition.py script contained the Haar cascade algorithm used to detect, localise, and recognise the face. This is Haar cascade algorithm was created using two modules from imutils and OpenCV. Image frames were looped over in the video stream from the Pi camera; therefore, the block of code below was implemented:

- Load the facial encodings data
- Use Haar cascade to instantiate the face detector
- Use the Pi camera connected to the Raspberry Pi to initialise video stream
- Allow the Pi camera to boot
- Start frames per second

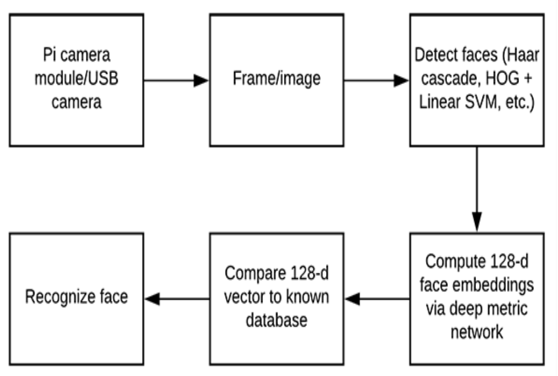
Pi camera module/USB camera
Frame/image
Detect faces (Haar cascade, HOG + Linear SVM, etc.)
Compute 128-d face embeddings via deep metric network
Compare 128-d vector to known database
Recognize face

Figure 3: The Process: Haar Cascade Classifier Based Deep-Dataset Face Recognition Algorithm for Locating Missing Persons

3.3 Capturing Frame and Recognizing Faces from Pi Camera

This deals with capturing frames from a live video stream. In capturing frames, a particular frame is selected and pre-processed. The Haar cascades face detection algorithm was used. This is the trained face detection algorithm of OpenCV that needs parameter tuning. The Haar cascade algorithm is a method for detecting objects in framed images in multiple real-time scales called the detectMultiScale method. Parameters of the detectMultiScale method are:

- MiniSize: The minimum object size in the frame beyond which objects smaller than the minimum size is ignored by the algorithm.
- Scale Factor: This parameter holds a value that specifies how the image is reduced in size at each image scale.
- MiniNeighbours: This parameter specifies the maximum number of faces a detecting frame can contain. The output of the face detector is a rectangular shaped frame, called rect. This rectangular box is bounded by four closed lines wherein the detected face in an image frame is contained, located and tracked. This leads to the computing of the 128-d encodings of the face image for its unique identification. The process involves:

1. Checking for face matches between the input frame and the stored database images.
2. Implementing a voting system that determines the face it will most likely be, in the event of a tie after matching. Here, the rectangular-shaped frame is drawn to surround the faces having the name of the person around it.

3.4 Tuning the Pan and Tilt Proportional-Integral-Derivative (PID) Independently

Tuning a PID, in this case, ensured that the servos track the face image that has been detected. In implementing this Pan and Tilt module, there were four major processes carried out.

- A process for finding the face in the frame.
- A process for calculating the panning (left and right) and tilting (up and down) angles with a PID.
- A process for driving the two servos..

3.5 PID Controller

The PID or Proportional-Integral-Derivative controller is a closed-loop feedback system, as shown in Figure 4.

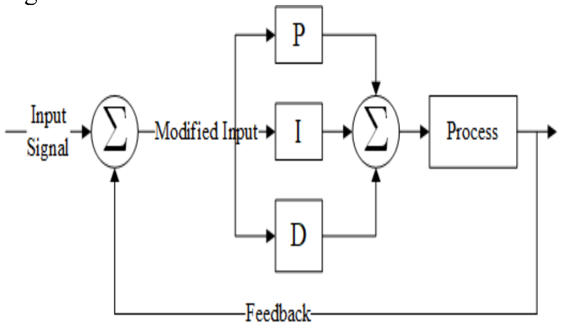

Figure 4: The Pan Tilt Servo PID Controller

The goal of the PID controller was to arrive at an optimum value by continuously taking in the output from the feedback and compensating for the error. It calculates and outputs a value which was used as an input to an electromechanical process.

- P (Proportional): Proportional posits that the output will be proportionally large when the current error is large to cause a significant correction.
- I (Integration): For integration, the error values are integrated over time. It only takes less significant corrections to reduce the error over time. The value will grow as long as the error remains. Conversely, the term will not grow if the error is eliminated.
- D (derivative): The derivative is a dampening method that anticipates the future and controls a value in the event of an overshoot. In the event that Proportional and Integral terms cause overshooting, D controls the value as it approaches the output.

3.9 Detector

The detector was initialised with two update methods:

- PID update: This is a method that performs servo angle calculations to keep the face centrally aligned in the pi camera.
- ObjCenter update: This method finds the face and returns centre coordinates.

The ObjCenter method accepts the image frame parameter and the frameCenter. The frame was converted to grayscale to get an even background for recognition because of external constraints such as illumination and brightness. At this point, face detection was performed using the Haar cascade detecting method, called detectMultiScale method.
As soon as the face was detected, the centre coordinates of the face was calculated. After this was done, the rect function drew a rectangle around the detected face the purpose of tracking. Once the face is bounded by the detector frame, the servo runs to follow the motions of the detected face.
To create the complete project on Face Recognition for missing persons, three distinct phases of work were carried out:

- Data Gathering
- Training the Recogniser
- Face Recognition

The Pi Camera has been enabled in the Raspberry Pi. It was tested by running a script in python to display and run the video stream. The code captures a frame in the Video Stream, that was generated from the Pi Camera. The captured frame was displayed in both RGB colour and Gray format.

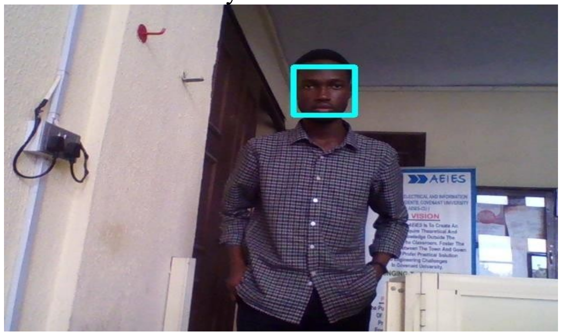

Figure5: Face Detection: Pi Camera Test

The Image in Figure 5 is a frame captured from the Pi Camera when the python test script was run. The rectangular box is the rect function that was drawn on detected faces

3.10 *Face Detection*

Face detection is the essential task in face recognition because the algorithm must capture a face first in order to recognise it. Face detection using Haar feature-based classifiers was a useful object detection method for the research study. This was a deep learning-based approach where the Haar cascade function was trained by a large number of face images. It had a high degree of accuracy because it was trained on both positive and negative images. The trained cascade function was used to detect faces brought before the camera. The Haar Cascade for implementing the face detector is shown in the first algorithm.

```
Algorithm 1: Face Detection
Input: Video Stream
Output: Rect. Function bounded face
Initialisation; Import Libraries
Run Camera = Cv2.VideoCapturer
Load Cascadepath = Cv2.CascadeClassifier
1: while Camera = Cv2.VideoCapture do
2:   Run cascade path = Cv2.CascadeClassifier
3:   Activate "rect" function
4: end while
5: return DetectedFaces
```

The second to the last line in Face detector algorithm was used to load in the Haar feature classifier in XML format. The last line of code added was to call the detectMultiScale function, a classifier function. The called function detected faces in the video stream.

3.11 Gathering the Deep Dataset

A database folder was created and named deep-dataset, where a group of sample faces of missing persons were stored in grey, each with a unique face ID. Faces were stored in groups of 30 for each face. The available dataset was not many but had more variations in pose and illumination, hence the term, deep-dataset. This was for the next phase, where the Haar Cascade Classifier model was trained on the face images. The data gathering algorithm is shown in Algorithm 2. The deep-dataset gathering algorithm was written as a script responsible for capturing faces and storing the face images in the database. Since the missing person was obviously not available, a photo from the parents or guardian was fed into the system for training and recognition.
The data acquisition algorithm could capture user ID. The input of the user ID was an integer value. After capturing the face image (n) number of times, the captured faces images were stored in the database folder, named 'dataset' directory. Three unique faces were captured, and the result from running the second algorithm is shown in Figure 2.

```
Algorithm 2: Deep Dataset Gathering
Input: Video Stream
Output: Greyscale scaled face image
Initialisation; Import Libraries
Run Camera = Cv2.VideoCapturer
Load Cascadepath = Cv2.CascadeClassifier
Define Face ID = Face ID Number
Run Count
Set Scale
1: while Face Detector = True do
2:   Get frame = Cv2.Flip (Frame, -1, Face)
3:   Convert Frame to a grayscale image
4:   Scale grayscale face image
5:   for Count > 30 do
6:       Capture face image sample
7:       Assign Name/ID
8:       Save scaled grayscale face images
9:       Increment Count
10:      if Count > 30 then
11:          Break
12:      end if
13:  end for
14: end while
```

3.12 Trainer

The next step was to train the OpenCV Recognizer by running the training script. This script moved all the face images from the database folder to the trainer to train the model to recognise the same faces when detected in a real-time video stream. After training the model, the result was a file that was saved on a 'trainer' directory folder

```
Algorithm 3: Training using Haar Cascade Classifier
Input: Scaled grayscale face images
Output: Trained Haar Cascade Classifier
Initialisation; Import Libraries
Run Camera = Cv2.VideoCapturer
Load Cascadepath = Cv2.CascadeClassifier
Define Face ID = Face ID Number
Run Recognizer – Cv2.Face.LBPHFFaceRecognizer
Define Image Label
1: for Image Label do
2:   Convert Image to Feature Array
3:   Append ID to Feature Array
4: end for
5: return face Samples; IDs
```

The LBPH (Local Binary Patterns Histograms) was used as the Recognizer and this was called from the OpenCV package. The Recognizer took all photos on the "dataset/" database directory and returned 2 arrays: "IDs" and "Faces". The arrays were used to train the face recogniser. Whenever a kidnapped case was reported, the photo was uploaded to the database and used to train the model for real-time video stream recognition.

4 RESULT AND DISCUSSION

The Results obtained Implementing, and testing are explained using the following:

- Results (Accuracy level) from the Recognition Test
- The result from the Notification Test
- The result from the Gmail Inbox Notification Test

- The result from the Email Notification Content Test.

4.1 Results from the Recognition Test

In this step, the recogniser script ran and loaded the trainer.yml file for recognition. The system was actively on real-time surveillance, and all faces within the viewing range of the Pi camera was detected and recognised. The recogniser made a "prediction" by returning its name label and confidence level in percentage showing how confident the recogniser was that a match was found.

```
Algorithm 4: Recogniser Accuracy
Input: Face Images
Output: Recognizer Accuracy
Initialisation; Import Libraries
Run Camera = Cv2.VideoCapturer
Load Cascadepath = Cv2.CascadeClassifier
Define Face ID = Face ID Number
Run Recognizer – Cv2.Face.LBPHFaceRecognizer
Read Image
1: while Rect, Frame = Read do
2:   Get grayscale image
3:    for grayscale face image do
3:         Run Recognizer.Predict
4:         if Confidence is <= 100 then
5:             State ID = Name
6:             Confidence = Value in %
8:         else
9:              ID = Unknown
10.             Confidence = 0%
11.        end if
12.    end for
13: end while
5: return RecognizerAccuracy
```

A new array was included to display "names", instead of numbered IDs. The algorithm was such that the name IDs were trained and matched with the positions of the images. A detected face called the and called the important function, The recogniser.predict (). The recogniser.predict () was a function that took the captured face as a parameter, analysed and returned its probable owner, indicating its ID name-label and how confident the recogniser was in relation to this match. The recogniser could predict a face, put a text over the image with the probable name label and how much, in terms of the "probability" in percentage that the match was correct. If not, an "unknown" label was put on the face. After running the Recogniser script, the model accurately predicted the name for ten missing persons with an average confidence level of 72.9%. The result of the two control tests for Daniella and Emmanuel is also shown in Figure 6 where the system was able to recognise the two missing persons with accuracies of 60% and 70% respectively. Figure 7 shows the recognition accuracy of ten subjects. The average experimental accuracy of 72.9% was due to variations in illumination, positioning of the face, and facial expressions. he simulation result was got after implementing our proposed Haar Cascade algorithm in OpenCV (Open Computer Vision), a Phython powered platform for research in Computer Vision.

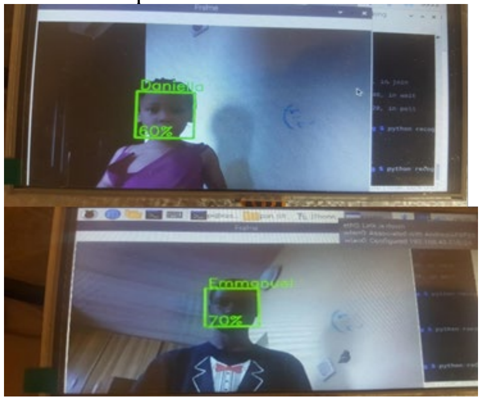

Figure 6. Sample Face Recognition Accuracy on Control Deep-Dataset of Two Missing Persons

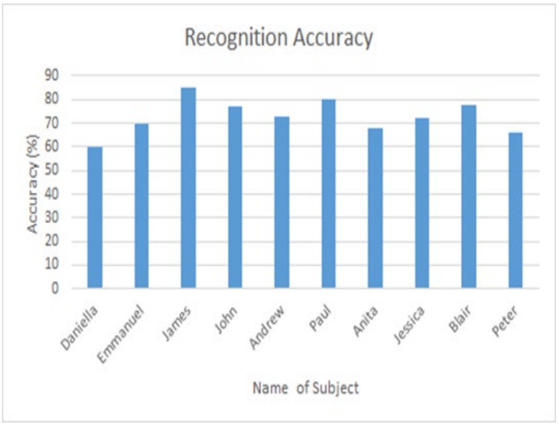

Figure 7. Recognition Accuracy for the Control Test

4.2 Accuracy

The deep dataset as shown in Table 1 comprises of thirty images for each of the ten missing persons under test. The thirty images were used to train the Haar Cascade Classifier based neural network to carry-out the recognition task. Five face image of each subject was used for testing.

Table 1: Recognition Accuracy Result Table

S/No	Subject	Training Data	Testing Data	Recognition Accuracy (%)
1	Daniella	30	5	60
2	Emmanuel	30	5	70
3	James	30	5	85
4	John	30	5	77
5	Andrew	30	5	73
6	Paul	30	5	80
7	Anita	30	5	68
8	Jessica	30	5	72
9	Blair	30	5	78
10	Peter	30	5	66
	Average Recognition Accuracy			72.9%

4.3 Result of the Email Notification Test

In this step, the result from the Recognizer.py script was sent as an email notification using another script. The recognised image frame from the image window contained both the name of the missing person and the confidence level in percentage. The algorithm that executes the email notification function is shown as Algorithm 5.

```
Algorithm 5: Email Notification
Input: Face Images
Output: Recognizer Accuracy
Initialisation; Import Libraries
Run Cv2, imwrite
1: while Recognizer Accuracy >= Threshold do
2:     Create Alert Message
3:     Specify the email address
4: end while
5: return MissingPersonFoundAlert
```

The Gmail emailing web service was used here as the third-party service to receive the security alerts of missing persons found. An inbox held the notification alerts sent to it. At a glance, the subject of the email is shown as 'ALERT: Missing Person(s) Found!'. The message content is important because it holds vital information about the location of the missing person found and the recognised image frame containing the missing person's name and confidence level in percentage. The notification system integrated into the machine learning algorithm proved to be useful in sending notification alerts to the security teams as soon as the face recognition technology recognises a missing person. This will help security teams locate and quickly track missing persons found by this technology.

5. CONCLUSION

The use of face recognition to find persons proved to be a powerful technological tool for law enforcement in identifying missing persons in public places. Persons reported missing to law enforcement agents can be identified in public places easily with a reasonable level of accuracy. Artificial Intelligence proved to be the best and most preferred algorithm for this solution, as its purpose serves to detect and recognise faces in public areas in real-time. Before faces are recognised, facial features are extracted and are then matched with a series of face images stored in the database. Once a match is found, the system sends a security notification alert to law enforcement through email. The proposed solution is able to search through real-time video surveillance for matches, helping police officers save time and narrow down their searches.

ACKNOWLEDGMENT

The authors acknowledge the sponsorship of the Covenant University Centre for Research, Innovation and Discovery (CUCRID), Ota, Ogun State, Nigeria under the aegis of intervention funds for research and development for the 2019/2020 academic session.

REFERENCES

[1] S. Balaji, R. Saravanan, P. Pavadharani, I. Deepalakshmi, S. Suvetha, "A SURVEY ON FACIAL RECOGNITION TECHNIQUES AND STIPULATED ALGORITHMS."

[2] A. Charity, K. Okokpujie, N.-O. Etinosa, "A bimodal biometric student attendance system," in 2017 IEEE 3rd International Conference on Electro-Technology for National Development (NIGERCON), IEEE: 464–471, 2017.

[3] P. Muyambo, "An Investigation on the Use of LBPH Algorithm for Face Recognition to Find Missing People in Zimbabwe," International Journal of Engineering and Advanced Technology (IJEAT) Volume, 7, 2018.

[4] K. Okokpujie, O. Modupe, E. Noma-Osaghae, O. Abayomi-Alli, E. Oluwawemimo, "A bimodal biometric bank vault access control system," International Journal of Mechanical Engineering and Technology, 9(9), 2018.

[5] T.M. Omweri, A.M. Kahonge, "Using a Mobile Based Web Service to Search for Missing

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

People–A Case Study Of Kenya," 2015.

[6] A. Mishra, M. Swain, B. Dash, "An Approach to Face Recognition Of 2-D Images Using Eigen Faces and PCA," Signal & Image Processing, 3(2), 143, 2012.

[7] A.P. Chantar, C.G. Bhaktha, S.G. Manjunath, H.M. Kumar, "Home Surveillance System using Raspberry Pi," JRJET. April, 2017.

[8] R. Santhoshkumar, M.K. Geetha, "Deep Learning Approach for Emotion Recognition from Human Body Movements with Feedforward Deep Convolution Neural Networks," Procedia Computer Science, 152, 158–165, 2019.

[9] W. Zhang, S. Shan, W. Gao, Y. Chang, B. Cao, "Component-based cascade linear discriminant analysis for face recognition," in Chinese Conference on Biometric Recognition, Springer: 288–295, 2004.

[10] M. Iqbal, M.S.I. Sameem, N. Naqvi, S. Kanwal, Z. Ye, "A deep learning approach for face recognition based on angularly discriminative features," Pattern Recognition Letters, 128, 414–419, 2019.

[11] G. Guo, N. Zhang, "A survey on deep learning based face recognition," Computer Vision and Image Understanding, 189, 102805, 2019.