Facial Detection and Recognition using OpenCV on Raspberry Pi Zero

Gagandeep Singh Nagpal
Computer Science and Engineering
Guru Tegh Bahadur Institute of Technology
New Delhi, India
gagannagpal131@gmail.com

Gagandeep Singh
Computer Science and Engineering
Guru Tegh Bahadur Institute of Technology
New Delhi, India
igagandeep.s@gmail.com

Jappreet Singh
Computer Science and Engineering
Guru Tegh Bahadur Institute of Technology
New Delhi, India
jappreet97@gmail.com

Nishant Yadav
Computer Science and Engineering
Guru Tegh Bahadur Institute of Technology
New Delhi, India
tonishantrao@gmail.com

***Abstract*—In this modern era, security has become a major concern. The amount of criminal activities has increased exponentially over the last three decades. Apart from that, security in places such as airports and railways stations is cumbersome and often ineffective in various cases. The motivation for developing this system was the need for a system to identify suspicious individuals in an time, cost and resource effective manner. In this paper, we discuss a system which consists of a Raspberry Pi Zero connected to a Raspberry Pi camera module, Capacitive touch sensor and OLED display. This system uses Haar Cascade classifier for face detection in an image followed by Local Binary Pattern Histogram for facial Recognition (LBPH). The LBPH algorithm is implemented using OpenCV.**

Keywords—OpenCV; Facial Recognition; Computer Vision; Face Detection; Local Binary Pattern Histogram; Raspberry Pi Zero

I. INTRODUCTION

The definition of facial recognition refers to a subset of computer technology that is able to identify human faces in digital images. Face detection algorithms emphasise on detecting human faces within images which may contain other objects, other human parts and landscapes. Some common applications of face detection is people counting (marketing), photography and facial recognition. Face detection is the first and foremost step in case on facial recognition. Facial recognition is defined as the process of identifying or verifying a person from a digital image or video frame. Facial Recognition is a subject that researchers have focused on for years. In the past decades, the advances in technology allowed computing systems to perform face recognition. OpenCV (Open Source Computer Vision Library) is used commonly for the implementation of facial recognition algorithms. Some commonly used facial recognition algorithms using OpenCV are Eigenface, Fisherface and LBPH.

This paper presents a system consisting of a Raspberry Pi Zero, Raspberry Pi camera Module, Capacitive touch sensor and an OLED display. All of these components are connected to the Raspberry Pi Zero. The capacitive touch sensor is used to give input which causes the camera module to capture an image. The Haar Cascade classifier is used to detect any faces, if present in the image. The Local Binary Patten Histogram (LBPH) algorithm is used for the purpose of facial recognition from the system's face database. The processing of the image input is divided into two parts: Detection and Recognition. The detection part deals with the detection of faces in the image followed by the recognition part which deals with the recognition of the face present in the image.

A. System Specifications

Initially, the system specifications were analysed for optimal decision making during the design and implementation process.The system consists of Raspberry Pi Zero having a Broadcom BCM2835 SoC along with a ARM11 CPU running at 1 GHz and 512 MB of RAM. It is attached to a Raspberry Pi camera module with a resolution of 5 Megapixels which is capable of capturing an image of up to a resolution of 2592 x 1944 (pixels), an 0.96" OLED display with a resolution of 128 x 64 and a capacitive touch sensor having with the dimensions of 370 x 300 (mm). All of the components are connected to the Raspberry Pi, which is the control centre for all the components.These components provide a simple and efficient system design.

B.Facial Detection

In recent years face detection has been widely studied due to its applications in computer and human interaction. Face detection falls under the area of image processing. Image processing is essentially a tool for compressing, enhancing or

ISBN: 978-1-5386-4119-4/18/$31.00 ©2018 IEEE

extracting useful features from an image. This paper uses a facial recognition technique that can detect single or multiple faces in an image, eliminating unnecessary background noise from the image. A face detection algorithm essentially has to classify images into two categories depending upon whether an image has a face present in it or not. The motive of the face detection algorithm is to fully scrutinise the image, detect the presence of faces in the image and also eliminate the background from the image. There are two types of errors that occur in face detection namely false negative and false positive. Incase of a false positive, a face is detected in an image that does not consist of any face. Whereas a false negative is when the algorithm dismisses the presence in the image. Detection rate is the ratio between the number of faces detected by human and number of faces is correctly detected by system. The detection rate of the facial detection algorithm should maximum.

In this system, the first and foremost step after the image is captured by the camera module is the detection of faces in that image. A predefined Haar Cascade classifier XML file used for this purpose. Object Detection using Haar feature-based cascade classifiers is an effective object detection method which is implemented commonly using OpenCV. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. This trained classifier is used in the python code to detect the faces in the images captured by the Raspberry Pi camera module.

C. Facial Recognition

Facial recognition is the world's leading and rapid biometric technology. It uses the most accessible human body part : the face, in a non-intrusive manner. According to international statistics, most people aren't aware of the process of facial recognition occurring on them, making it one of the most non-intrusive processes with minimal delay. The facial recognition algorithm analyses a face's different characteristics from the input image. This biometric has been widely, and perhaps wildly, touted as a fantastic system for recognising potential threats such as terrorists, scam artists, etc but so far has not seen wide acceptance in high-level usage. It is projected that biometric facial recognition technology will soon overtake fingerprint biometrics as the most popular form of user identification and authentication. OpenCV is widely used for implementing facial recognition algorithms. Some widely used algorithms implemented using OpenCV are : -

- Eigenface :- It is based on principal component analysis that classify images to extract features using the same set of images. It is crucial that the eyes of the individuals match in each image and the images are under the same lighting conditions.
- Fisherface :- This algorithm is based upon linear discriminant analysis which is used for pattern recognition. It uses labels for classes as well as data point information. Different lighting conditions have minimal effect on its classification process.
- Local Binary Pattern Histogram :- In this algorithm, the image is divided into blocks of pixel size of 3 x 3. The intensity of luminosity of the surrounding pixels are compared with the central pixel and depending upon the difference, a value of 1 or 0 is given to each surrounding pixel. The result provides an eight bit number which is converted to a decimal. The luminosity of the image does not have any effect on the algorithm. Histograms are used to find the frequency of occurrences of values, making the process more efficient.

Local Binary Pattern Histogram algorithm was preferred out of three most widely accepted and used facial recognition algorithms listed on the OpenCV website, since LBPH gives a balance between accuracy, CPU utilisation and latency. The following considerations were taken into account while selecting LBPH as the appropriate algorithm for facial recognition :-

- Eigenface and Fisherface algorithms use Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) respectively. Both of these are feature extraction processes which convert a model from a higher dimensionality to a lower dimensionality. These processes are CPU intensive and require relatively higher processing power as compared to computational matrix mathematics used in LBPH.
- Eigenface algorithm uses PCA to find the axes of maximum variance. However, there is an issue when the cause of variation is an external source such as light. Hence the images need to have similar lighting conditions which isn't practically feasible.
- The algorithmic complexity of Fisherface algorithm caused the CPU to function at a 100% capacity for lengthier duration as compared to LBPH algorithm and an increased latency between the input and output of the system which wasn't practically deployable.

In the system, following the detection of faces in the captured image, the image is scanned through the system's database and the output is shown on the OLED display. The linear binary pattern histogram algorithm is used for facial recognition. The local binary pattern operator is an image operator which transforms an image into an array or image of integer labels describing small-scale appearance of the image. These labels or their statistics, most commonly the histogram, are then used for further image analysis. The original version of the local binary pattern operator works in a 3×3 pixel block of an image. The pixels in this block are thresholded by its centre pixel value, multiplied by powers of two and then summed to obtain a label for the centre pixel. This implementation utilises Local Binary Patterns Histograms to extract features from test face images and Manhattan distance to retrieve the correct match from the system's face database. If the image captured is within the specified range of the

image in the database, then the image is shown to be a match. Otherwise, the captured image isn't a match from the database.

II.LITERATURE SURVEY

Almost all the literature available describes facial detection and recognition on a software level. Only a handful of research papers have addressed the combination of hardware and software on a theoretical level. The practical implementation facial detection and recognition using the combination of hardware and software has most commonly been implemented on FPGA.

In [1], a survey conducted by M. Yang, D. Kriegman, and N. Ahuja face detection is divided into four categories: feature invariant methods, knowledge-based methods, template-based methods, and appearance-based methods.

In [2], Yang and Huang used a face detection method, in this method set of rules are defined as a knowledge based method and it has rules on an input images first scanning is applied on face and all possible face can be found by scanning.

In [3], N.Stekas and G.Heuval implemented facial recognition with facial recognition on a custom designed System on Chip with an accuracy of 79 %.

In [4], D.Meena and R.Sharan implemented facial detection using Viola Jones algorithm and facial recognition using Eigenface algorithm on Windows Operating System using Matlab. Their algorithm provided accuracy of approximately 90% with some false positives on their database of over 1000 images.

In [5], C.Gao and S.Lu presented an approach to use an FPGA to accelerate the Haar feature classifier based face detection. They re-trained the Haar classifier with 16 classifiers per stage.

In [6], J.Cho, S.Mirzaei, J.Oberg, R.Kastner presented a hardware architecture for face detection based on system on AdaBoost algorithm using Haar features.

In [7], S. Dominic, M.Mohan, Aparna C, Ajeesh M.S., A.S.Nath and A.Antony have presented a system for the extraction of faces from a live video feed and facial recognition of the faces from the criminal database.

In [8], G.Xiang, Z.Quiyu, W.Hui and C.Yan presented a system of facial recognition using a variance of Local Binary Pattern Histograms. The method of regression is used to get facial features whose complexity is minimal.

In [9], S.Guennouni, A.Ahaitoufa and A.Mansouri have presented two major techniques of facial recognition. The two techniques are namely Edge-Orientation Matching and Haar-like feature selection combined cascade classifiers.

In [10], L.Cuimei, Qi.Zhiliang, J.Nan and W. Jianhua have shown a new human facial detection algorithm by primitive Haar cascade algorithm in combination with three additional weak classifiers. First, images of people are processed by a primitive Haar cascade classifier followed by a weak classifier based on face skin hue histogram matching is applied and a majority of non-human faces are removed. Finally, Finally, a mouth detection operation is utilised to the remaining non-human faces and the false positive rate is further decreased.

In [11], L.Lang and W.Gu studied an algorithm for facial detection using AdaBoost was studied. This algorithm generates a cascade classifier automatically that significantly reduces overtraining of the model.

In [12], E.Hjelmas and B.K.Low presented a comprehensive and critical survey of face detection algorithms ranging from simple edge-based algorithms to high-level approaches utilising advanced pattern recognition methods.

III.PROPOSED WORK

Fig.1 shows the block diagram of the system's components and their configuration and connection with other each . The system is designed to take touch input from the user on the capacitive touch sensor. This initiates an event which activates the Raspberry Pi camera module to capture three images simultaneously. The third image is used for further processing since it is the most illuminated image and is converted into grayscale and is processed further. The facial detection process initiates at this step and if a face is detected in the image, the facial recognition process is initiated. If a face isn't detected in the image, an output of "Not Detected" is shown on the OLED display. In the process of facial recognition, if the image is shown to be a match from the system's database, the name of the individual is shown on the OLED display. If the image isn't a match from the system's database, an output of "All Good" is shown on the OLED display.

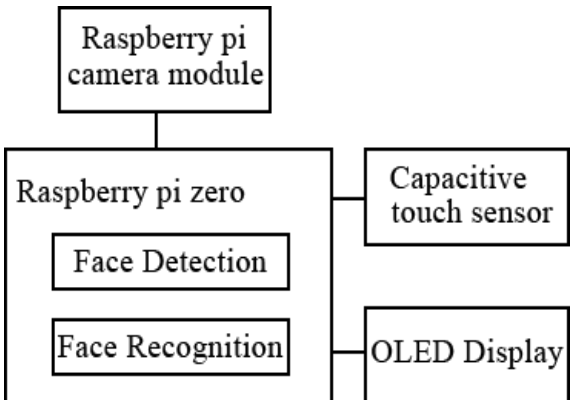

Fig 1. Block Diagram of the System

A. Image Dataset

The image dataset considered is of four persons having ten photos each in the dataset. The images for the preparation can be captured using any conventional camera or a web camera having a image resolution of 720p or above.

B. Trainer File

A python script is written which is executed manually once the images for the preparation of the dataset are captured (multiples images of individuals). This python script converts the images into grayscale followed by converting all the images into an numpy array. The Haar Cascade frontal face classifier file is used to detect the presence of faces in all of the images. Each of the face is scanned and is appended to their respective IDs. Thus, a model is trained consisting of images of faces. This model is saved as a trainer file with the extension ".yml". This trainer file is loaded each time at the boot up time of the Raspberry Pi and is used by the local binary pattern histogram algorithm for the purpose of facial recognition. This trainer file is used to match the faces present in the system's face database and is responsible for the latency between the boot up and input. Hence, the number of photos per person used for training the model is to be selected carefully to strike a balance between accuracy and latency.

C. Automation

The Raspberry Pi is configured in a manner such that, upon booting of the Raspberry Pi, the python script responsible for desired functionality gets invoked automatically. A Shell Script is written which contains the name along with the path of our python file which is to be invoked upon start up. The shell script is placed within the directory "~/.config/lxsession/LXDE/autostart", which is responsible for the invocation of graphical user interface elements and application software at the time of boot up.

D. No. of Photos Vs. Size of trainer file

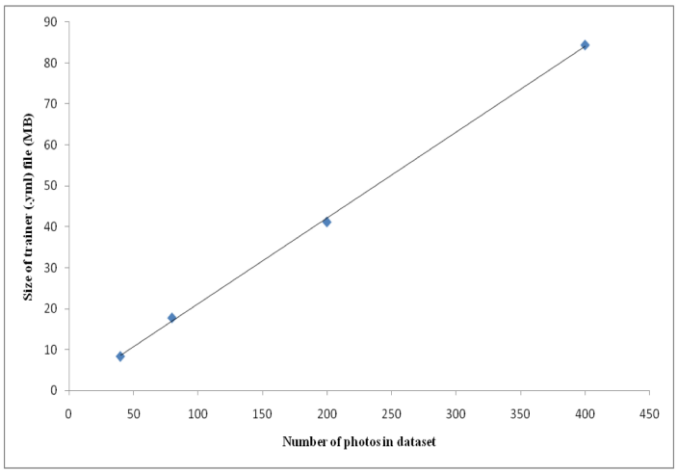

Fig 2. Relationship between the Number of photos in the dataset and size of trainer (.yml) file

As observable from Fig. 2, the size of the trainer file and the number of photos in the dataset follows a linear relation. As the number of photos in the dataset increase, the size of the trainer file also increases and vice versa. Initially the number of images of each person in the database was set to 100, Thus the total number of images for four persons was 400. After extensive testing, the number of images per person was set to 10. Thus making total number of images for four persons equal to 40. This reduction in the number of images did not cause any noticeable reduction in the accuracy of facial recognition. Thus, providing comparable results with a dataset which is significantly reduced in size.

E. Size of trainer file Vs. Loading time of trainer file

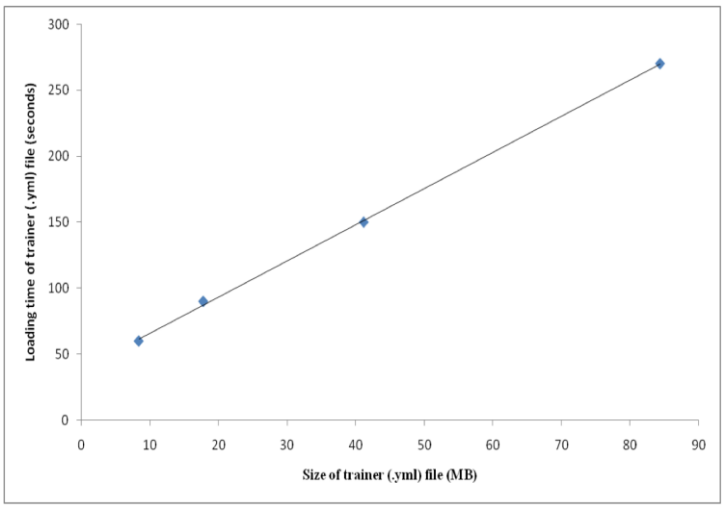

Fig 3. Relationship between the Size of trainer (.yml) file Vs. Loading time of trainer (.yml) file

As observable from Fig.3, similar to the number of photos in a dataset, the loading time of the trainer file also follows a linear relationship with the size of trainer file. As the size of the trainer file increases, the loading time of the file also increases and vice versa. The trainer file is loaded upon the boot up of the Raspberry Pi. Thus, a larger file causes an increasing delay between the boot up of the system and the input by the user. The size of the trainer file was reduced to cause a reduction in the loading time of the file. Thus, bringing the latency between the boot up and input to a minimal level.

F. Flow Of Control

Fig. 4 demonstrates the flow of control of the system in a lucid manner. The flow starts from the image input into the system, followed by the process of facial detection. If a face isn't detected , the flow goes to the output display. If a face is detected, the initiation of the face recognition process occurs. Following the face recognition process the flow enters the display output, this is the final step of the system. The output on the display depends upon whether the face has been recognised from the images present in the database or not.

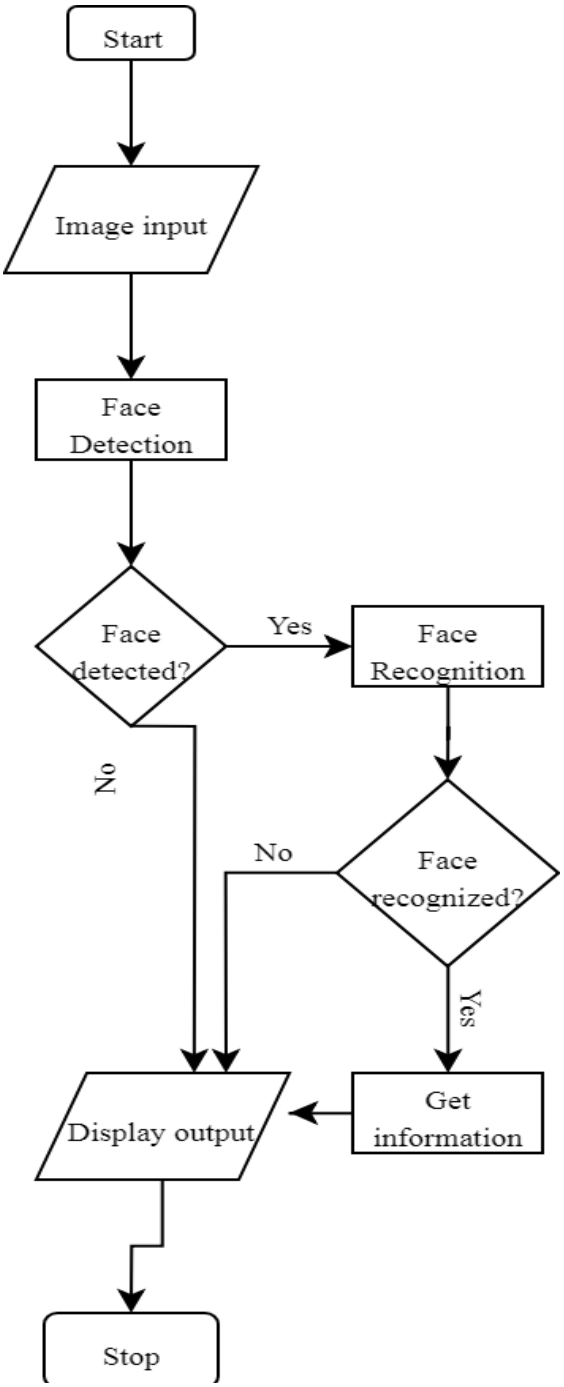

Fig 4. Flowchart of the system

IV.RESULT

The average time taken for the system to capture an image followed by face detection, recognition and the display of output on the OLED display is approximately 2 - 3 seconds. The most accurate results are produced under the following conditions :-

- Ideal lighting conditions
- Distance between the camera and individual is approximately 1 m.

Under these ideal conditions, the percentage face detection was found to be 80% and the accuracy of facial recognition was also found to be 80%.

Considering all of the different test cases under various lighting conditions and distances between the user and Raspberry Pi camera, the average percentage of face detection is found to be in the range of 50 - 60 % and the average accuracy of facial recognition is found to be in the range of 55 - 65 %.

TABLE I. Size and Loading time of trainer file corresponding to the number of images in the face dataset

Number of photos in dataset (4 persons)	Size of the trainer (.yml) file (MB)	Loading time of trainer (.yml) file (seconds)
400	84.4	270
200	41.2	150
80	17.8	90
40	8.4	60

As seen in Table I, the initial number images in the dataset was set to 400, which resulted in a trainer file of a colossal size of 84.4 MB, which caused the loading time of the file to be 270 seconds. Upon reducing the number of photos to 200 followed by 80 and finally 40 (10 images per person), the size and loading time of the trainer file was significantly reduced to 8.4 MB and 60 seconds respectively. There was not any noticeable difference in the performance of the system upon reducing the number of images in the dataset.

However, upon reducing the images of each individual from 100 to 10, the latency between the boot up and the system being ready for input is reduced significantly which has resulted in a more fast and efficient system for face recognition and facial recognition.

The use of Raspberry Pi Zero has provided a compact, efficient and easy to assemble system in comparison to other custom designed System on Chip (SoC).

V.SUMMARY AND CONCLUSION

Security is our day to day concern and this product through its mobility and effectiveness (60% accuracy in normal condition and 75 % in ideal condition) can definitely make a difference in the society. The system was developed using latest hardware which allows multiple functionality with a modest amount of efficiency. There are innumerable features and

modifications that can be added to this project such as a smartphone application to support the hardware, improved camera module for clearer images, custom designed System on Chip (SoC) to increase efficiency and reduce latency, which instigates colossal amount of future scope of this project.

ACKNOWLEDGMENT

We would like to express our great gratitude towards our mentor, Ms.Geetika Bhatia, Assistant Professor at Guru Tegh Bahadur Institute of Technology, New Delhi; who has given us immense support and suggestions. Without her help we could not have presented this paper up to the present standard. We also take this opportunity to thank everyone who provided us with the required support for this paper or have helped us in other aspects throughout our studies at Guru Tegh Bahadur Institute of Technology.

REFERENCES

[1] Yang, M. H., Ahuja, N., & Kriegman, D. (2000). Face detection using mixtures of linear subspaces. In Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on (pp. 70-76). IEEE.

[2] Yang, G., & Huang, T. S. (1994). Human face detection in a complex background. Pattern recognition, 27(1), 53-63.

[3] N.Stekas, D.Heuvel, “Face Recognition Using Local Binary Patterns Histograms (LBPH) on an FPGA-Based System on Chip (SoC), ” IEEE, Chicago, IL, USA, Aug 2016.

[4] D.Meena, R.Sharan, “An Approach to face detection and recognition, ” IEEE, Jaipur, India, December 2016

[5] C. Gao and S. Lu, “Novel FPGA based Haarclassifier face detection algorithm acceleration,” In Proceedings of International Conference on Field Programmable Logic and Applications, 2008.

[6] J.Cho, S.Mirzaei, J.Oberg, R.Kastner, 2009, “FPGA -Based face detection system using Haar Classifiers” , Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays, P 103 -112

[7] J.Cho, S.Mirzaei, J.Oberg, R.Kastner, 2009, “FPGA -Based face detection system using Haar Classifiers” , Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays, P 103 -112

[8] G.Xiang, Z.Quiyu,W.Hui, C.Yan, “Face recognition based on LBPH and regression of Local Binary features, ” IEEE, Shanghai, China, July 2016.

[9] S.Guennouni, A.Ahaitouf, A.Mansouri,“Face detection: Comparing Haar-like combined with cascade classifiers and Edge Orientation Matching, ” IEEE, Fez, Morrocco, April 2017.

[10] L.Cuimei, Qi.Zhiliang, J.Nan, W. Jianhua,“Human face detection algorithm via Haar cascade classifier combined with three additional classifiers, ” IEEE, Yangzhou, China, October 2017.

[11] L.Lang,W.Gu, “Study of Face Detection Algorithm for Real-time Face Detection System, ” IEEE, Nanchang, China, May 2009.

[12] E.Hjelmas, B.K.Low, 2001, “Face Detection: A Survey”, Computer Vision and Image Understanding, V 83 P 236 -274