Implementation of Face Recognition Algorithm on a Mobile Single Board Computer for IoT Applications

Swarnendu Guha
Department of ECE
University of Engg. and Management
Kolkata, India
swarnaguha@gmail.com

Amlan Chakrabarti
A. K. Choudhury School of I.T
University of Calcutta
Kolkata, India
amlanc@ieee.org

Sujoy Biswas
Department of ECE
Neotia Institute of Technology Management and Science
Kolkata, India
sbiswas@ieee.org

Soumen Banerjee
Department of ECE
University of Engg. and Management
Kolkata, India
soumen.banerjee@uem.edu.in

2020 IEEE 17th India Council International Conference (INDICON) | 978-1-7281-6916-3/20/$31.00 ©2020 IEEE | DOI: 10.1109/INDICON49873.2020.9342290

***Abstract*—This work focuses on a specific development aiming towards the detection of human face in an Internet of things (IoT) platform. IoT is a growing wireless based technology, which connects physical objects supported by different forms of electronic hardware. In this work, we propose an IoT application meant for security purpose, where human faces will be analyzed online. The application accepts the face image of a person and tries to find a match with any of the pre-stored facial images in the database. The detection will be automatically validated and the status message can be transferred in real-time to some specified IP address. Initial phase of our work describes the development of a Python program, which works on a Raspberry Pi IoT hardware bearing a small controller fitted with four USB modules. It is connected to the outer world through a Pi Camera, passive infrared (PIR) motion sensor, power supply, secure digital (SD) card, and a 40-pin GPIO port. The next phase of work focuses on estimating the resource requirement and its minimization. The proposed research aims towards a smart technology for robust face detection in an on-field environment.**

Keywords— — Face recognition, internet of things, raspberry pi, OpenCV

I. INTRODUCTION

Face detection and recognition is a biometrics authentication technique developed as a successful application of pattern recognition and image analysis. It has been a popular topic drawing enormous attention of the researchers dealing with the problem of face based image analysis in the domain of computer vision [1]. Literature survey shows that efforts have been made in order to implement different methods of face recognition employing different computing platforms and tools [2]-[6]. The studies have been further enhanced to recognize gestures [7] and some specific recognition problems like biometric iris recognition [8]. These algorithms require computational resources and as such, it is easy to implement it in hardware platforms that are capable of handling such problems. With the advent of IoT there is a growing need of fast and accurate recognition method based on mobile single board computing platforms. The primary challenge, therefore, lies in implementing of face recognition systems in such single board computer systems which offer limited computational resources. Limited computational resources of the single board computers seriously limit the performance of such algorithms. The recognition algorithm should be able to perform in a wide range of mobile single board computer platforms with minimum computational resources available to cater all users and to provide a cheap solution. This problem is addressed in this paper to implement the face recognition system in a mobile platform with reasonable performance.

In this work, implementation of face recognition algorithms has been explored in a Raspberry Pi board. The face detection and recognition methods have been optimized for low processing power of the hardware platform. Satisfactory detection and recognition performance with acceptable speed in the real time is achieved with low false positives and demonstrated in the paper.

II. BACKGROUND

In the modern world, we are getting dependent on electronic devices in almost all of our day to day actions and that has been forcing us to enter into an automated world. This in turn indicates a higher demand for quick and smart user authentication and identification. Recent cases of identity theft, robbery, murder and other malpractices have heightened the necessity for methods to prove that somebody is truly who that person claims to be. Face detection technology might be a solution to this problem since a face is undeniably connected with its owner except in cases of any identical twins. Face, being the most important part of a human body, its accurate identification plays an important role in the electronic security solution. Research by several groups has shown that human face expresses different emotions in various ways which indeed play very significant role in interacting with different people in our society. Our face actually conveys our identity and therefore, this can be treated as a key to modern electronic based security solution. For the same reason, face detection system is becoming a popular means across the globe for creating a reliable and safe technology. Compared to the commonly used biometric security

978-1-7281-6916-3/20/$31.00 ©2020 IEEE

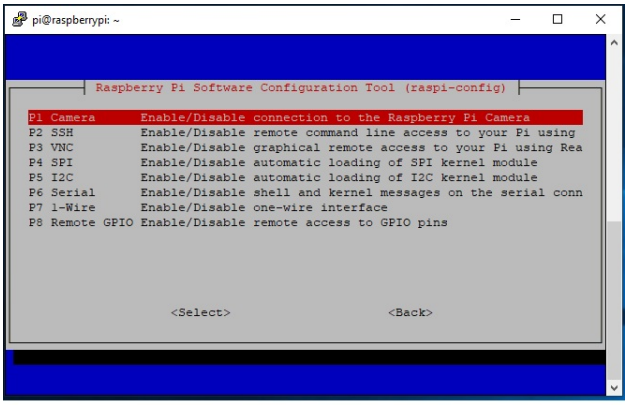


Fig.1 Enabling and testing the Pi Camera

mechanisms such as fingerprints or palm identification, face detection extends a wide range of benefits. It avoids physical proximity of a person and has been proved effective especially in the present situation of the global pandemic due to COVID-19 pandemic.

The image of a person's face will be matched using the process of matrix difference between two images that have been compared with each other. If the process of comparison matches with any facial image stored in the database, the detection will be automatically validated and a message will be transferred immediately to some specified destinations or smart phones over the internet.

III. Related Works (Face Recognition Techniques)

Haar feature-based cascade classifiers have been used for object detection. It is actually based on adaptive machine learning technique and the cascade function is trained using several positive and negative images. The features are then extracted from it. In order to recognize the faces advanced methods are required. To begin with, the implementation of face detection methods Local Binary Patterns Histograms (LBPH) was first explored here to detect and recognize the faces in the images and streaming video in real time. The method showed that for face detection, HOG (Histogram of Oriented Gradients) image descriptor and a LSVM (Linear Support Vector Machine) can be applied to train an accurate object classifier. This method is implemented in this work as an improvement over the LBPH method and is demonstrated to provide better performance.

The Local Binary Pattern Histogram (LBPH) based face detection scheme has been developed and it is found to work well to detect correct faces. In the training phase of the face recognition implementation, we take all user data from our dataset and train the OpenCV Recognizer. This is executed directly by a specific OpenCV function. The result will be a *.yml* file that will be saved in the *"trainer/"* directory. We created a subdirectory to store the trained data. We use it as a recognizer, i.e., the LBPH (Local Binary Patterns Histograms) face recognizer, which is already included in the OpenCV package.

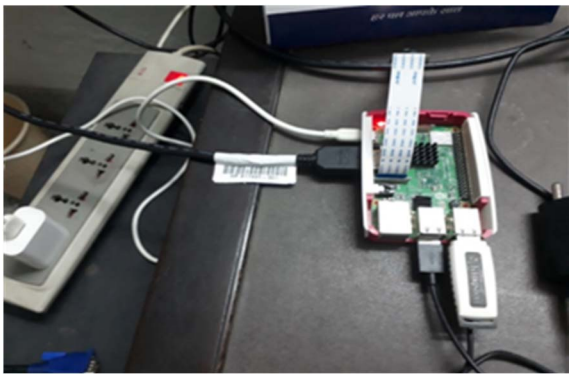


Fig.2 Raspberry Pi device in working condition

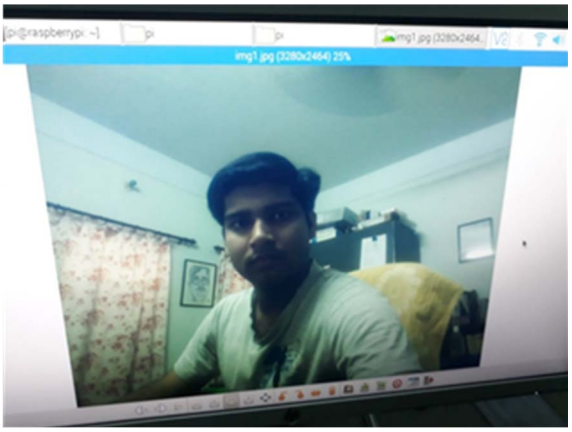


Fig.3 A color image captured by the Pi camera

IV. System Design and Implementation

A. *Hardware requirements*

i) Raspberry PI: Model 3B+

ii) Raspberry PI Camera

iii) 16GB MicroSD Card

iv) Card reader

v) Power adapter for RPI

B. *Creating the Working Platform*

i) Installing the Raspbian OS

ii) Configuring Raspberry Pi for remote access

iii) Updating the System

iv) Installing the Dependencies

v) Installing Python 3

vi) Creating Python Virtual Environment

vii) Installing the Pi Camera Library

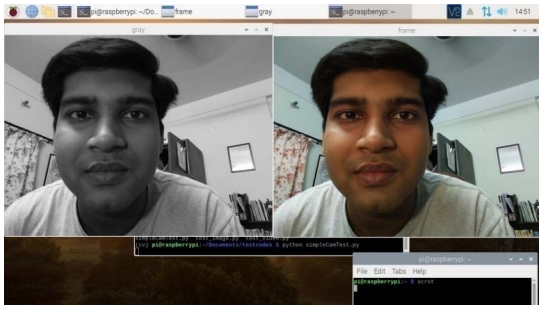


Fig.4 Color to grey scale conversion using the code

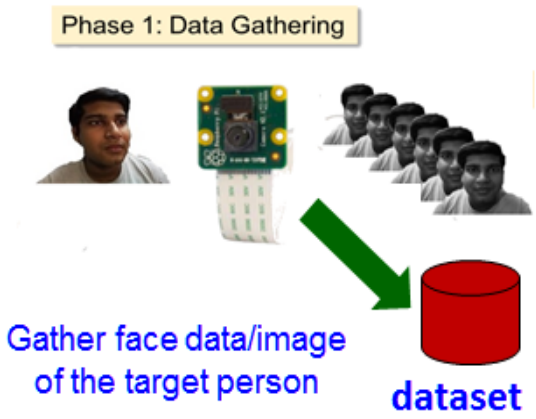


Fig.5 Collection of image datasets captured through Pi camera

viii) Installing OpenCV

C. Testing Pi Camera and Capturing Color Images

The raspberry pi camera is connected to the board and *raspi-config* is used to enable the camera from the option of *Interfacing Options > Pi Camera.*

The photograph of the working setup is shown in the Fig. 2. The camera is now tested for capturing still image and its sample output is shown in Fig.3.

D. Accessing the Camera with Python Code

The Pi Camera is now accessed with a python code and installed Pi Camera library. A simple python test code is then used to access the camera, capture an image and convert it to gray scale format. The output of the code is shown in Fig.4.

V. Face Recognition Implementation

We need to create a dataset to store a group of gray-scale photographs for each image ID. This follows creation of a project directory and then a subdirectory for storing the facial samples, name as "dataset". An input command has been added to capture the user id, which is simply an integer such as 1, 2,

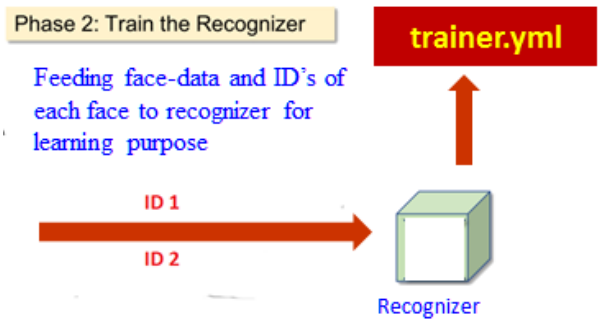


Fig.6.The training phase yielding .yml file.

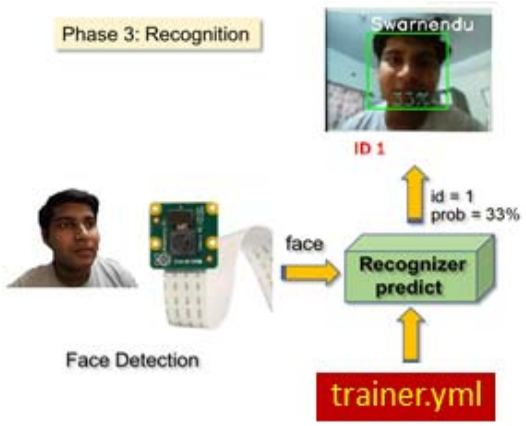


Fig.7 This indicates 33% close approximation in correctly identifying a test face

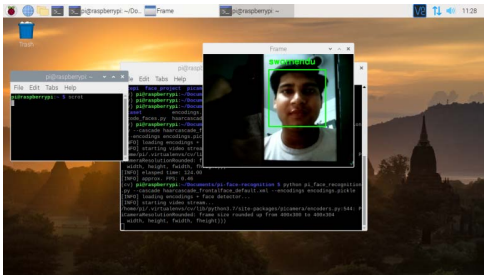
etc. Each captured frames is saved as a file on the "dataset" directory. In here, we captured 30 samples from each id which can be changed on the last command "elif".

Training:

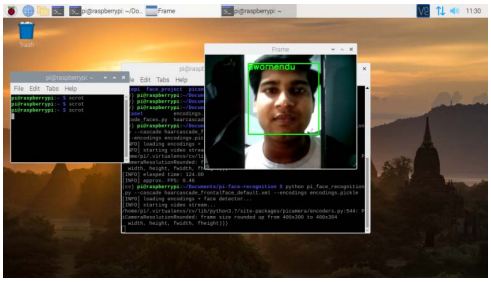
In the training phase, all user data have been taken from our dataset and utilized to train the OpenCV Recognizer. A specific OpenCV function has been used for the same resulting in a ***.yml file***. This is saved in the *"trainer/"* directory. A subdirectory stores the trained data. Local Binary Patterns Histograms (LBPH) face recognizer has been used in our work and this is readily available in the OpenCV package. Fig.6. depicts the scheme for 'train the recognizer'.

Recognizer:

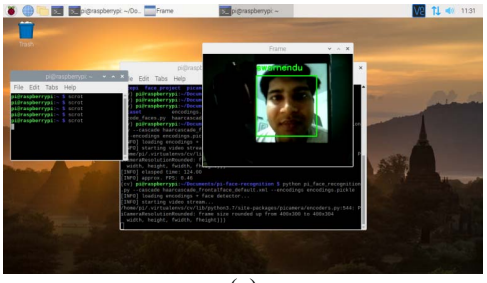
Now it is the final phase of our work where we intend capturing a images of fresh faces using our own pi- camera. The system



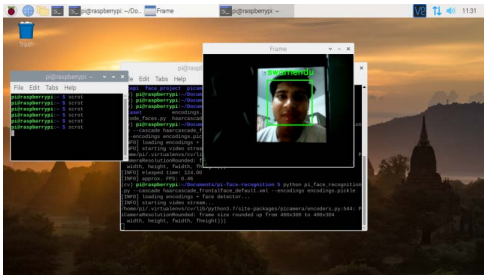
(a)



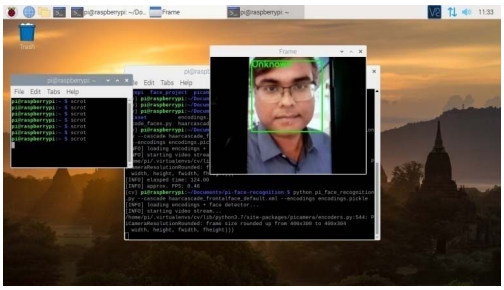
(b)



(c)



(d)



(e)

Fig.8 Test results viewed on the screen under different levels of illuminations: (a)-(d): correctly recognizes the faces of 'Swarnendu' with correct ID; (e) face is unknown to the system and hence the recognizer returns "Unknown"

recognizer should make a prediction if the face of that person is already trained before. This should also return to its ID showing the order of confidence indicated by the recognizer.Fig.7 helps in visualizing the outcome of a test in our project.

VI. Face Recognition Using Haar Cascades, HOG & Linear SVM

We have used five still pictures captured from our Raspberry Pi camera module for data source. A deep neural network is generally used for computing a 128-d vector, i.e., a list of 128 floating-point values. This quantifies each face in the dataset. Firstly, we need to import the required packages. From there, we handle our command line arguments with *argparse*:

`--dataset`: Path to dataset

`--encoding`: face encodings written to the file

`--detection-method`: Before we can encode faces in images we first need to detect them. Our detection method is HOG.

Inside the loop, we execute:

i) Finding out person's name from the path
ii) Convert to RGB image
iii) Locating face in the image
iv) Computing the face
v) Embedding and adding

The encodings are exported to disk. It is used in facial recognition script. After running the same, pickle file is obtained at our disposal. The file then bears 128-d face embedding.

The next steps follow:

i. importing packages and parse command line arguments;
ii. importing a couple of modules: *VideoStream* and FPS from *imutils*;
iii. importing face_recognition and cv2 (OpenCV);
iv. parse two command line arguments:

`--cascade`: Path to OpenCV's Haar cascade

`--encodings`: Path towards serial database of facial encodings

This led to capturing frames using pi-camera and preprocessing, which includes resizing the matrix as well as converting to gray-scale and RGB. Haar Cascades have been used in our system.

Parameters to the `detectMultiScale` method include:

`gray`: gray-scale image.

`scaleFactor`: Parameter to specify the order of size reduction of the image

`minNeighbors`: Parameter to specify the retaining neighbors /rectangles.

`minSize`: Minimum size that is possible in recognizing the face.

The detection result appears as 'rects' which means rectangular boundaries surrounding the face, i.e., the location on the frame. Reordering of the coordinates is followed by computing the 128-d `encodings` for each face.

The following steps are executed:

- Check for `matches`
- If `matches` are detected, a voting system will be operated to decide and identify the 'face' or the most likely face. This process uses the stored dataset for the maximum number of matches.

VII. Result

In the Raspberry Pi terminal, the following commands havebeen taken from [6] and executed.

```
$ python pi_face_recognition.py --cascade haarcascade_frontalface_default.xml \

--encodings encodings.pickle

loading encodings + face detector

start video streaming

elapsed time: 124.00

approx. FPS: 0.46
```

Fig.8 shows the screen captures of the recognized and unknown faces with different levels of illuminations and it produces recognition without any false positives. The matching obtained in the recognition process varied from 68% to 80% accuracy.

VIII. Conclusion

We started the work with a twofold aim:

i. Development of Python based code for online face detection, and
ii. Integration with an IoT-Based System.

The execution part was not very smooth indeed as we had to create the Linux based working platform for Python OpenCV from the scratch. This gave us enough scope of learning and handling multiple software and hardware related issues. Those are indeed the background steps to implement image processing for practical applications. 'Face detection' is a very common task which is executed by a normal human brain without any significant effort. But execution of the same task with the help of a machine is a big challenge. A flawless detection involves several stages of fine 'decision making' especially with the help of Artificial Intelligence (AI). This, as per our understanding, provides enough scope of advancing the detection quality or accuracy. We intend extending the same to improve the present development in the future. The real challenge is identifying accurate features of online captured faces and their comparison with the stored images. We have tried to explore the best possible coding with Python programming language. Developing an operational system is the target for the future and for the same; we need to integrate it with IoT based systems.

References

[1] J. Yan, X. Zhang, Z. Lei, and S-Z. Li, "Face detection by structural models," ***Image Vis. Comput***. pp. 1-10, 2013.

[2] T. Kanade, Picture Processing System by Computer Complex and Recognition of Human Faces, Department of Science, Kyoto University, 1973.

[3] C. Kotropoulos, I. Pitas, Rule-based face detection in frontal views, ***Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-97)***, vol. 4, 1997, pp. 2537–2540, (IEEE).

[4] E. Osuna, R. Freund, F. Girosit, "Training support vector machines: an application to face detection,"***Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition***, 1997, pp. 130–136.

[5] H.A. Rowley, S. Baluja, T. Kanade, "Neural network-based face detection," ***IEEE Transactions on Pattern Analysis and Machine Intelligence*** 20 (1) (1998) 23–38.

[6] P. Viola and M. Jones, "Rapid object detection using aboosted cascade of simple features", "***Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision andPattern Recognition***", pp. I-511-I-518 vol, 2001.

[7] Khari, M., Garg, A. K., Crespo, R. G., &Verdú, E. (2019). ***Gesture Recognition of RGB and RGB-D Static ImagesUsing Convolutional Neural Networks***. International Journal of Interactive Multimedia & Artificial Intelligence, 5(7).

[8] M., Gupta, R., Khari, M., & Crespo, R. G. (2019) ***Biometric iris recognition using radial basis function neural network***,Soft Computing, 23(22), 11801-11815.

[9] Open source materials: https://www.pyimagesearch.com/