



International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS)

www.iasir.net

STAND ALONE FACE RECOGNITION SYSTEM USING PRINCIPLE COMPONENT ANALYSIS

¹Mrs.SUMA.M.O, ²Mrs.RASHMI.H.N, ³Mr.SRINIDHI B SESHADRI
^{1,2}Asst.Professor, RGIT , India

Abstract: Biometric access systems have come a long way. Methods such as voice based access, speaker recognition; finger print recognition, password key systems and face recognition are a few important modes of biometric access systems. Face recognition, verification of authenticity based on the matching of a person's face, is a complex algorithm involving several methods. This paper aims to implement a face recognition software code based on the method of Principal Component Analysis and to successfully implement this code on the Raspberry Pi platform for real time recognition.

In this paper, an attempt to implement face recognition algorithm on a hardware platform, which is simple, yet efficient in usage is taken up. The Raspberry Pi development platform used is based on a BCM2835 System-on-Chip sporting an ARM11 processor and Image capture device is based on a linux compliant USB webcam. The software codes for both detection and recognition of faces are written using Python.

- A detection rate of almost 92% was achieved when a distance threshold of 80% was used.
- Due to the detection algorithm, background noise was reduced and pre-processing is minimized.
- The system works best when the face is sufficiently illuminated and the person is frontal
- w.r.t. the camera.

I. INTRODUCTION

A Face Recognition System is a system which automatically identifies and/or verifies the identity of a person from digital images or a video frame from a video source [1].

Over the last ten years or so, face recognition has become a popular area of research in computer vision and one of the most successful applications of image analysis and understanding. Because of the nature of the problem, not only computer science researchers are interested in it, but neuroscientists and psychologists also. It is the general opinion that advances in computer vision research will provide useful insights to neuroscientists and psychologists into how human brain works, and vice versa.

Wide availability of powerful and low-cost desktop and embedded computing systems has created an enormous interest in automatic processing of digital images and videos in a number of applications, including biometric authentication, surveillance, human-computer interaction, and multimedia management. Research and development in automatic face recognition follows naturally [2].

A general statement of the face recognition problem (in computer vision) can be formulated as follows: Given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces.

This paper gives the implementation of face recognition as a standalone system without the requirement of a sophisticated computer system. On being successful, with further optimizations and improvements, the system may be implemented in real time systems requiring user authentication such as attendance systems, ATM security, Electronic Passports and Network security.

This paper is organized as follows. In Section II, we briefly describe the principle component analysis. Then, brief description on raspberry pi and its implementation in Section III. Section IV gives the design and development of system. Finally, brief summaries are given in Section V to conclude this paper.

II. PRINCIPLE COMPONENT ANALYSIS

The implementation of face recognition is modeled on the principles of the Karhunen-Loève transform which provides the basis for the calculation of the eigen vectors [principal components] of an image and thereby generating a template for an individual image. PCA was effectively used first by Matthew Turk and Alex Pentland during their tenure at the Media Laboratory, Massachusetts Institute of Technology. The two phases of the implementation are as follows:

Software: The software is initially modeled and tested in the MATLAB Integrated Development Environment for easy understanding of the algorithm's working. Subsequently, using open source imaging libraries such as open cv and GUI toolkits such as Tkinter, two GUIs are implemented using the Python IDLE; one for face detection and the other for face recognition.

Hardware: The hardware used to implement the above developed Python code is the Raspberry Pi which is a credit card sized computer development platform based on a BCM2835 System on Chip, sporting an ARM11 processor coupled with an on chip GPU for video streams, DSP and on chip RAM and ROM. The image capture device used is a standard linux compliant USB webcam. A PIC16f877A microcontroller is interfaced with the Raspberry Pi through I2C. This MCU drives a 16x2 character LCD to display the status messages during operation including the Grant/Denial of access based on the recognition output.

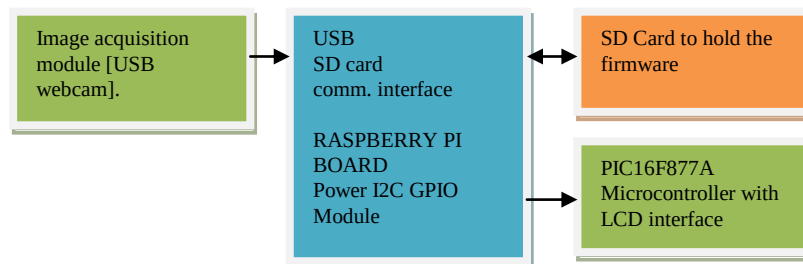


Figure1: system block diagram

Principal component analysis (PCA) is a mathematical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it be orthogonal to (i.e., uncorrelated with) the preceding components. Principal components are guaranteed to be independent only if the data set is jointly normally distributed. PCA is sensitive to the relative scaling of the original variables.

A. Eigen Faces Approach

Eigenfaces is a known term that represents the significant features or principal components of human faces. Each feature or component is represented as an eigenvector. These eigenvectors do not correspond to the physical entities at the face (e.g., eye, nose, mouth, etc.). In the process of recognizing human faces, each face is projected into a set of Eigenface features. The eigenvectors corresponding to these features are weighted. Usually, the sum of these weights is a good representation of a given face. Recognition approaches that use Eigenfaces for recognition ignore the third dimensional information and base their models on two-dimensional information only. The foundation of using Eigenfaces in face recognition is based on the fact that each image can be represented as a matrix. A matrix has a set of eigenvectors that represent the principal components of the matrix. Eigenfaces are the eigenvectors of the covariance matrix of all faces. Similar faces can be described in a space with lower dimensionality. Most of the research on Eigenfaces is concerned with one pose and on rotating the image about the image centre. A collection of different pictures of different angles or emotions of a same person is needed to estimate the actual face. The more sample picture of a person is given; the better accuracy will be found to recognize that person[7].

B. Overview of the algorithm

First, the original images of the training set are transformed into a set of eigenfaces E . After that the weights are calculated for each image of the training set and stored in the set W . Upon observing an unknown image X , the weights are calculated for that particular image and stored in the vector WX . Then, WX is compared with the weights of images, of which one knows for certain that they are faces (the weights of the training set W). One way to do it would be to regard each weight vector as a point in space and calculate an average distance D between the weight vectors from WX and the weight vector of the unknown image WX (the Euclidean distance described in appendix A would be a measure for that). If this average distance exceeds some threshold value, then the weight vector of the unknown image WX lies too "far apart" from the weights of the faces. In this case, the unknown X is considered to not a face. Otherwise (if X is actually a face), its weight vector WX is stored for later classification. The optimal threshold value has to be determined empirically.



Figure 2: The set of training images considered



Figure 3 Eigenfaces of original images

C. Recognition Procedure

- A new face is transformed into its eigenface components. First we compare our input image with our mean image and multiply their difference with each eigenvector of the L matrix. Each value would represent a weight and would be saved on a vector Ω .

$$\Omega^T = [\omega_1, \omega_2, \dots, \omega_M] \quad \dots 1$$

- We now determine which face class provides the best description for the input image. This is done by minimizing the Euclidean distance

$$\varepsilon_k = \|\Omega - \Omega_k\|^2 \quad \dots 2$$

- The input face is considered to belong to a class if ε_k is below an established threshold $\theta\varepsilon$. Then the face image is considered to be a known face. If the difference is above the given threshold, but below a second threshold, the image can be determined as a unknown face. If the input image is above these two thresholds, the image is determined NOT to be a face.
- If the image is found to be an unknown face, we could decide whether or not we want to add the image to our training set for future recognitions. We would have to repeat the training steps to incorporate this new face image.

III. RASPBERRY PI PLATFORM

A. Overview

The Raspberry Pi is a credit-card-sized single-board development platform envisaged and developed in the UK by the Raspberry Pi Foundation.

The Raspberry Pi is manufactured through licensed manufacturing deals with Element 14/Premier Farnell and RS Components. Both of these companies sell the Raspberry Pi online.

The Raspberry Pi has a Broadcom BCM2835 system on a chip (SoC), which includes an ARM1176JZF-S 700 MHz processor (The firmware includes a number of "Turbo" modes so that the user can attempt overclocking, up to 1 GHz, without affecting the warranty), VideoCore IV GPU, and was originally shipped with 256

megabytes of RAM, later upgraded to 512MB. It does not include a built-in hard disk or solid-state drive, but uses an SD card for booting and long-term storage.

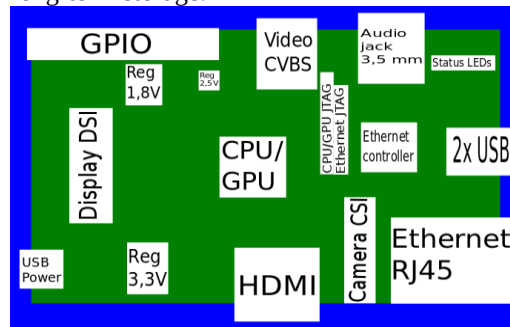


Figure 4. Block Diagram of Raspberry Pi PCB

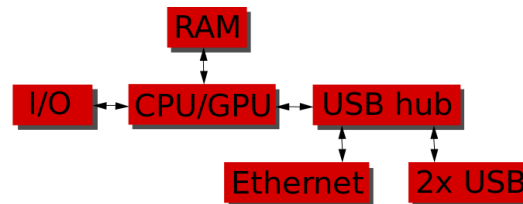


Figure 5. Raspberry Pi Functional Block Diagram

B. Software Architecture.

The Raspberry Pi uses Linux kernel-based operating systems. Raspbian, a Debian-based free operating system optimized for the Raspberry Pi hardware, is the current recommended system, released in July 2012.

The GPU hardware is accessed via a firmware image which is loaded into the GPU at boot time from the SD-card. The firmware image is known as the binary blob, while the associated ARM coded Linux drivers were initially closed source. This part of the driver code was later released, however much of the actual driver work is done using the closed source GPU code. Application software uses calls to closed source run-time libraries (Open Max, Open GL ES or open VG) which in turn calls an open source driver inside the Linux kernel, which then calls the closed source Videocore IV GPU driver code. The API of the kernel driver is specific for these closed libraries. Video applications use OpenMAX, 3D applications use OpenGL ES and 2D applications use OpenVG which both in turn use EGL. OpenMAX and EGL use the open source kernel driver in turn.

On 19 February 2012, the Raspberry Pi Foundation released its first proof of concept SD card image that could be loaded onto an SD card to produce a preliminary operating system. The image was based upon Debian 6.0 (Squeeze), with the LXDE desktop and the Midori browser, plus various programming tools. The image also runs on QEMU allowing the Raspberry Pi to be emulated on various other platforms.

IV. DESIGN AND DEVELOPMENT OF THE SYSTEM

A. Software Design

The software code for the Principal Component analysis was modeled initially in MATLAB. Once tested and debugged, the code was rewritten in Python. The next two sub sections deal with the software development using MATLAB and its subsequent porting to the Python IDE.

MATLAB, which stands for MATrix LABoratory, is a state-of-the-art mathematical software package, which is used extensively in both academia and industry. It is an interactive program for numerical computation and data visualization, which along with its programming capabilities provides a very useful tool for almost all areas of science and engineering. Unlike other mathematical packages, such as MAPLE or MATHEMATICA, MATLAB cannot perform symbolic manipulations without the use of additional Toolboxes. It remains however, one of the leading software packages for numerical computation.

Guessing from its name, MATLAB deals mainly with matrices. A scalar is a 1-by-1 matrix and a row vector of length say 5, is a 1-by-5 matrix. We will elaborate more on these and other features of MATLAB in the sections that follow. One of the many advantages of

MATLAB is the natural notation used. It looks a lot like the notation that you encounter in a linear algebra course. This makes the use of the program especially easy and it is what makes MATLAB a natural choice for numerical computations.

B. Python Implementation

The code for both face detection and recognition was written in python 2.7. Graphical User Interfaces [GUI] are created for both applications. Several Libraries are used along with python to incorporate specific tasks.

The software implementation in python involves face detection, initially, which detects the face region and returns an image comprising of only the detected region. This is helpful for the subsequent recognition process since less amount of pre-processing is required.

C. Development

The USB webcam sends a video stream of image frames received by the RPi through the USB interface. Once received, the python GUIs implement the face recognition and detection algorithms. The SD card holds the necessary software to implement these algorithms. It also holds the database of Eigenfaces. Once authenticated, the access grant message is displayed on screen.

At each stage of implementation, from the system initialization to the validation, status messages are displayed on a 16x2 character LCD. The RPi, through its GPIO pins is connected through a two-wire I2C interface to the PIC16F877A microcontroller. The microcontroller receives the status signals from RPi through I2C and appropriately drives the LCD, displaying the correct status message.

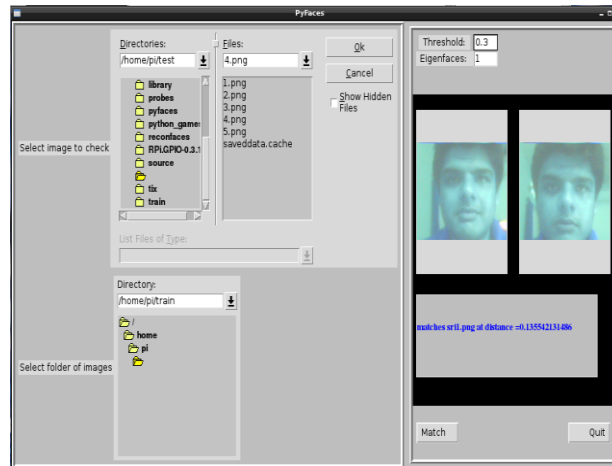


Figure 6. The GUI for facial recognition being tested on one of the database images. The match message is displayed with the distance measure

V. CONCLUSION

PCA based facial detection and recognition system uses the Raspberry Pi development platform based around a BCM2835 System-on-Chip sporting an ARM11 processor. Image capture device used here is a linux compliant USB webcam. The software codes for both detection and recognition of faces are written using Python.

On implementation, the following results have been obtained:

- A detection rate of almost 92% was achieved when a distance threshold of 80% was used.
- Due to the detection algorithm, background noise was reduced and pre-processing is minimized.
- The system works best when the face is sufficiently illuminated and the person is frontal w.r.t. the camera.

There are several avenues of future work-scope. Experimentation with the other face recognition algorithms such as ICA, Markov models, 3D Modelling etc is one such avenue. The frame rate and the pixel size can be increased to improve the performance. The algorithm may be optimized to reduce the recognition timing.

On the hardware side, instead of using the Raspberry Pi itself, the Processor of the SOC may individually be used to reduce the system cost. This may involve a transition to C coding for the processor but is effective in reducing costs nonetheless. A touch screen LCD may be interfaced with the Pi to enhance User Interface capabilities. Since the current project involves a remote desktop access to the RPi, several such systems may be linked through network clouds.

VI. REFERENCES

- [1]. Stan Z Li and Anil K Jain, Handbook of Face Recognition, Springer Science+Business Media, 2005.
- [2]. Matthew Turk and Alex Pentland, Eigen Faces for Recognition, Journal of Cognitive Neuro Science, MIT,1991, Volume 3, Number 1.
- [3]. Hyeonjaon Moon and P Jonathon Phillips, Computational and Performance aspects of PCA based face recognition algorithms, Perception, 2001, Volume 30, pages 303-321.
- [4]. Sarala A. Dabhade and Mrunal S. Bewoor, Real Time Face Detection and Recognition using Haar - Based Cascade Classifier and Principal Component Analysis, International Journal of Computer Science and Management Research, Aug 2012, Vol 1, Issue 1.
- [5]. Paul Viola and Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features, Accepted Conference on Computer Vision and Pattern Recognition 2001.
- [6]. Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman, Rapid Object Detection using a Boosted Cascade of SimpleFeatures, IEEE Transactions on Patten Analysis and Machine Intelligence, July,1997, Vol 19, No 7.
- [7] The OpenCV Library Documentation, www.opencv.org