Journal of the Nigerian Association of Mathematical Physics
Volume26, (*March, 2014*), pp 447 – 454

An Approach to the Design and Development of an Automatic Face Detection and Recognition System

Bello N. and Osaze R. I.

Department of Electrical/Electronic Engineering, University of Benin

Abstract

Computer vision is a field that includes methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. Computer vision involves transformation of data from a still or video camera into either a decision or a new representation. All such transformations are done for achieving some particular goal. The main aim of computer vision has been to replicate the abilities of human vision by electronically perceiving and understanding an image.This paper provides a practical implementation of the OpenCV libraries for face detection and recognition. Proposed methodology is a connection of two stages – Face detection using Haar Based Cascade Classifier and recognition using EigenFaces algorithm.

Keywords: Haar - Based Cascade Classifier, EigenFaces, OpenCV (open computer source libraries), EmguCV

1.0 Introduction

This paper is a guide towards developing a functional face detection and recognition system. Face Detection and Recognition is a method of detecting and identifying human faces in an image. Recently, face recognition has received lots of attention of researcher and it is still a growing area in computer vision.Automated face detection and recognition is an interesting area in computer vision with many commercial and law enforcement applications. User verification and user access control, crowd surveillance, enhanced human computer interaction all become possible if an effective face detection and recognition system can be implemented. While research into this area dates back to the 1960's, it is only very recently that acceptable results have been obtained. However, face detection and recognition is still an area of active research since a completely successful approach or model has not been proposed to solve the face detection and recognition problem[1].

The inadequacy of automated face detection and recognition systems is especially apparent when compared to our own innate face recognition ability. We perform face recognition, an extremely complex visual task, almost instantaneously and our own recognition ability is far more robust than any computer can hope to be.

The aim of the project is to develop a functional automatic face detection and recognition system capable of detection and recognizing faces in a static image or a video stream.

1.1 Approach and Methodology

1.2 Assumptions

Before continuing, it is helpful to assume the reader has knowledge of:

1. The C# programming language. Readers can search the internet for resources to learn C#.
2. Digital Image Processing in general; pixelization, template matching, thresholding to be more particular.
3. Interpretation of flowcharts and block diagrams.

2.2 Dependencies

The project depends on the dot net framework since it was developed in C# language. It also depends on OpenCV (Open Computer Vision) image processing library for image manipulation.

Corresponding author: ***Bello N.,*** E-mail: nosabello@yahoo.com, Tel.: +2348025373737

2.3 Methodology

The core of the project's operations (face detection and recognition) relies heavily on OpenCV library (which is a leading image processing and computer vision library).

The face detection and recognition system was designed using the MVVM (Model-View-ViewModel) method. This method separates the user interface from the source codes. The input unit of the system is either the internal webcam of a laptop or an attached digital camera that captures images of individuals approaching or facing the camera. The video frames are analyzed and converted from bitmap format (dot Net format) to BGR format which is a recognized format in OpenCV.

The BGR image is converted to gray image and the contrast is balanced using "gray._EqualizeHist()" a function in EmguCV. For computers with a graphic processing unit (GPU), the gray image is further converted to GPU gray image; otherwise it is passed on the face detecting unit which is the "Haarcascade" function in EmguCV (OpenCV in dot Net). A rectangle is drawn over the detected face and the face within the boundary of the rectangle is extracted for recognition.

The recognition process begins by first training the recognizer with faces within the blue rectangle in the trainer view. A minimum of six (6) faces with different facial expressions are used to train the recognizer and are stored in the database (Microsoft SQL server express) with their names attached to it.

The system compares and matches the extracted face(s) from detection using a function in EmguCV called "EMGU.CV.EigenFaceRecognizer". If the face(s) are found in the database, the system returns the face label which contains the name of the face and the status (either good, bad or suspicious represented by the colors green, red and orange respectively).

3.0 Design Specification

1. The dimension of a face for detection and recognition should be 80 pixels.
2. Face should be extracted irrespective of its color and background.
3. We are targeting frontal faces without any form of occlusion.

4.0 Algorithm

4.1 Face detection using Haar Cascades

Face detection was implemented using Haar - Based Cascade Classifier. Object Detection using Haar feature-based cascade classifiers is a machine learning based approach where a cascade function is trained from a lot of positive and negative images[2].

It is then used to detect objects in other images. This approach to detecting objects in images combines four key concepts:

- Simple rectangular features, called Haar features
- An integral Image for rapid feature detection
- The AdaBoost machine-learning method
- A cascaded classifier to combine many features efficiently
-

Figure 2b shows the first two features from the original Viola-Jones cascade superimposed on a face. The first one keys off the cheek area being lighter that the eye region. The second uses the fact that the bridge of the nose is lighter than the eyes.

The presence of a Haar feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present. To determine the presence or absence of hundreds of Haar feature at every image location and at several scales efficiently, Viola and Jones used a technique called an Integral Image[3]. In general, "integrating" means adding small units together. In this case, the small units are pixel values. The integral value for each pixel is the sum of all the pixels above it and to its left. Starting at the top left and traversing to the right and down, the entire image can be integrated with a few integer operations per pixel.

As Figure 3 shows, after integration, the value at each pixel location, (x, y), contains the sum of all pixel values within a rectangular region that has one corner at the top left of the value in this rectangle, you would only need to divide the value at (x, y) by the rectangle's area[4].

To select the specific Haar features to use, and to set threshold levels, Viola and Jones use machine learning method called **AdaBoost**. Adaboost (Adaptive Boosting) is a machine learning algorithm that can be used in conjunction with many other learning algorithms to improve their performance[5].It combines many "weak" classifiers to create one "strong" classifier. "Weak" here means the classifier only gets the right answer a little more often than random guessing would. AdaBoost selects a set of weak classifiers to combine and assigns a weight to each classifiers, this weighted combination is the strong classifier.

Viola and Jones combined a series of AdaBoost classifiers as a filter chain, shown in Figure 4, each filter is a separate AdaBoost classifier with a fairly small number of weak classifiers.

Figure 4 shows the classifier cascade is a chain of filters. Image sub regions that make it through the entire cascade are classified as "Face". All others are classified as "Not Face".

The filters at each level are trained to classify training images that passed all previous stages. (The training set is a large database of faces, maybe a thousand or so.) During use, if any one of these filters fails to pass an image region, that region is immediately classified as "not Face". When a filter passes an image region, it goes to the next filter in the chain. Image region, that pass through all filters in the chain are classified as "Face".

4.2 Face Recognition using Eigenfaces

The whole recognition process involves two steps [6]:

- Initialization process
- Recognition process

The Initialization process involves the following operations:

1. Acquire the initial set of face images called training set, usually found in the data base.
2. Calculate the eigenfaces from the training set, keeping only the highest eigenvalues. These M images define the face space. As new faces are experienced, the eigenfaces can be updated or recalculated.
3. Calculate the corresponding distribution in M-dimensional weight space for each known individual, by projecting their face images on to the "face space".

These operations can be performed from time to time whenever there is a free excess operational capacity. This data can be cached which can be used in the further steps eliminating the overhead of re-initializing, decreasing execution time thereby increasing the performance of the entire system.

Having initialized the system, the next process involves the steps:

1. Calculate a set of weights based on the input image and the M eigenfaces by projecting the input image onto each of the eigenfaces.
2. Determine if the image is a face at all (known or unknown) by checking to see if the image is sufficiently close to a "free space".
3. If it is a face, then classify the weight pattern as either a known person or as unknown.

4.2.1 Generating Eigenfaces

Assume a face image I(x,y) be a two-dimensional M by N array of intensity values, or a vector of dimension MxN. The Training set used for the analysis is of size 110x129, resulting in 14,190 dimensional space. A typical image of size 256 by 256 describes a vector of dimension 65,536, or, equivalently, a point in 65,536-dimensional space. For simplicity the face images are assumed to be of size NxN resulting in a point in N^2 dimensional space. An ensemble of images, then, maps to a collection of points in this huge space.

The Training set images used for the analysis purpose are shown in the Figure 5 and the corresponding eigenfaces from the training sets are shown in the Figure 6.

Let the training set of face images be $\Gamma_1\ \Gamma_2 ... \Gamma_M$. (1)

The average face of the set is defined by

$$\Psi = \frac{1}{M} \sum \Gamma_k \quad (2)$$

Each face differs from the average by the vector $\Phi_i = \Gamma_i - \Psi$. (3)

Figure 7 below shows an averageΨ of the Eigenfaces shown in figure 6,

This set of very large vectors is then subject to principal component analysis, which seeks a set of M orthonormal vectors, u_k, which best describes the distribution of the data. The *kth*vector is $u_k$chosen such that,

$$\lambda_k = \frac{1}{M} (u_k^T \Phi_n)^2$$

The vectors $u_k$and $\lambda_k$scalars are eigenvectors and eigenvaues,respectively, of the covariance matrix

$$C = \frac{1}{M} \sum_{n=1}^{M} \Phi.\Phi^T \quad (4)$$

$$= A.A^T \quad (5)$$

where the matrix $A = [\Phi_1, \Phi_1, \Phi_1 ..... \Phi_M]$

The matrix C, however, is N^2x N^2, which could be enormous. For example, images of size 64 x 64 create the covariance matrix of size 4096 x 4096. It is not practical to solve for the eigenvectors of C directly.A Computationally feasible method is to be funded to calculate theseeigenvectors.

If the number of data points in the image space is M ($M<N^2$),there will be only M-1 meaningful eigenvectors, rather than N^2. Theeigenvectors can be determined by solving much smaller matrix of theorder M^2xM^2which reduces the computations from the order of $N^2$to M,pixels. Therefore we construct the matrix L

$L=A.\ A^T$ (6)

Where $L_{mn}= \Phi^T_m . \Phi_n$

And find the M eigenvector $u_l$of L . These vectors determine linear combination of the M training set face images to form the eigenfaces v_l

$v_l = \sum u_{lk}.\Phi_k$

where $l= 1….M$ (7)

Once the eigenfaces are created, identification becomes a pattern recognition task. The eigenfaces span an N^2-dimensional subspace of the original A image space. The M' significant eigenvectors of the L matrix are chosen as those with the largest associated eigenvalues. In the test cases, based on $M = 6$ face images, $M' = 4$ eigenfaces were used. The number of eigenfaces to be used is chosen heuristically based on the eigenvalues. A new face image (I) is transformed into its eigenface components (projected into "face space") by a simple operation,

$\Omega_k = v^T(\Gamma_k - \Psi)$ (8)

where $k = 1…….M'$ k

This describes a set of point-by-point image multiplications and summations. Figure 2.5.5 shows three images and their projections into the seven-dimensional face space, the weights form a vector

$\Omega^T = [\Omega_1\ \Omega_2…….. \Omega_M\ ']$

that describes the contribution of each eigenface in representing the input face image, treating the eigenfaces as a basis set for face images.

(9)

The vector is used to find which of a number of predefined face classes, if any, best describes the face. The simplest method for determining which face class provides the best description of an input face image is to find the face class k that minimizes the *Euclidean distance*

$\varepsilon_k = \| \Omega - \Omega_k\|$

where $\Omega_k$is a vector describing the *kth* face class.

A face is classified as belonging to class k when the minimum $\varepsilon_k$is below some chosen threshold θ_ε , otherwise the face is (10)
classified as "unknown". The distance threshold, θ_ε, is half the largest distancebetween any two face images, Mathematically can be expressed as,

$\theta_\varepsilon = ½ \max_{jk}\{ \| \Omega - \Omega_k\| \}$

where $j,k = 1……M$

(11)

Recognition process can formulated as:

If

$\varepsilon \geq \theta_\varepsilon$ then input image is not a face

$\varepsilon < \theta_\varepsilon , \varepsilon_k \geq \theta_\varepsilon$ then input image contains an unknown face

$\varepsilon < \theta_\varepsilon , \varepsilon_{k'} = \min\{\varepsilon_k\} < \theta_\varepsilon$then image contains face of individual k'

The above discussed methodology has been implemented in OpenCV, a free software[7].EmguCV (OpenCV in .Net), made it possible for us to implement face detection using Viola and Jones algorithm and face recognition using eigenfaces (Principal component analysis) algorithm in C#.

5.0 Testing and Result

Testing of the overall system was performed by using a test set of five (5) different images. Each image was run through the system in three (3) different lighting conditions. A face is considered accurately detected if a steady rectangle is drawn over the face region and it is consider accurately recognized if a steady rectangle is drawn over the face region bearing the name and status of the face.

The results of the tests for face detections and recognition are presented in table 1 and 2 respectively.

5.1 Test/Result Images

Considering the test images above, the system failed to recognize two faces in figure 8 because, the face of the first person from the left was not found in the data base while a shadow was casted on the face of the person at the middle. In figure 9, the system failed to recognize one face because of poor lighting condition.
There was a complete detection and recognition in figure 10 because testing was done in a very good lighting condition.

6.0 Discussion

Table 1 shows the summary of the results for face detection under three (3) different lighting conditions using 5 different test subjects. The table shows the success/failures for the three different lighting conditions. All the faces were detected for condition A because of good lighting and clarity of the images. In condition B, three (3) faces were detected and 2 failed because of limited source of source of light and the output of the camera was reduced because light was focused on the lens of the camera. None of the faces were detected for condition C because there was no sufficient light to enable the image sensor of the digital camera to efficiently form an image for detection.

Table 2 shows the summary of the results for face recognition under same lighting conditions using 5 different test subjects. The table shows the success/failures for the three different lighting conditions. Four (4) faces were recognized for condition A because of good lighting and clarity of the images. In condition B, three (3) faces were recognized and 2 failed because of limited source of source of light and inefficiency of the recognizing under low lighting condition. None of the faces were also recognized for condition C because there was no sufficient light to enable the image sensor of the digital camera to efficiently form an image for detection and recognition.

7.0 Conclusion

In this paper, we succeeded in implementing face detection using Haar – Based Cascade Classifier (Viola and Jones algorithm) and recognition using EigenFaces method.

So far we have implemented an application that can capture images from real life and display them as a video stream, perform face detection and recognition using Viola and Jones algorithm and Eigenfaces algorithm respectively. Under good lighting condition, the system gives a good detection and recognition accuracy (75-85%). The algorithm however has its downsides as it is not able to detect or recognize face(s) in poor lighting condition and when the face is tilted or facing away from the camera. It also gives poor results for recognition when the faces share some similar features.

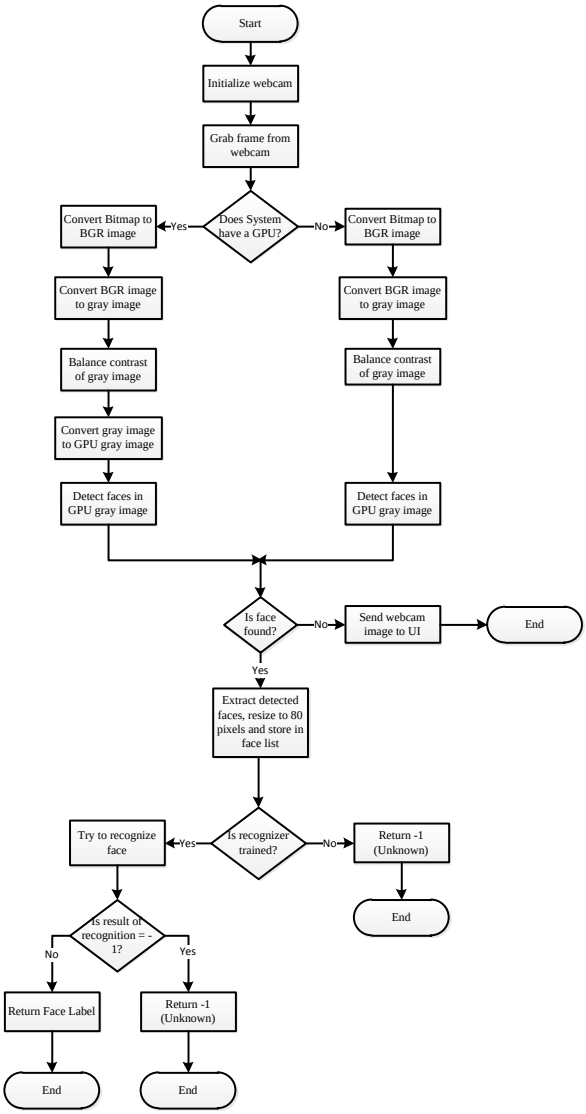

Fig. 1: Flow Chart

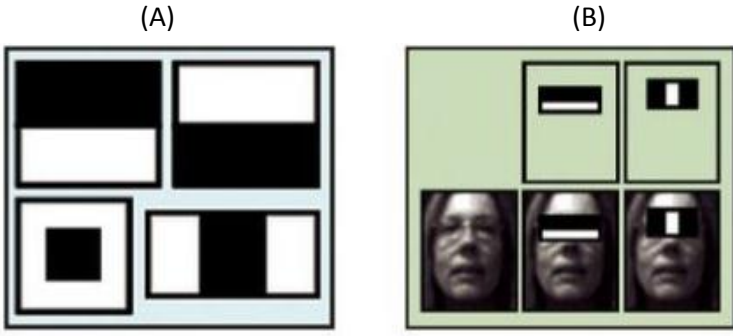

Fig. 2: Haar Features used in OpenCV

a. Examples of the Haar features used in OpenCV b. The first two Haar features in the original Viola and Jones cascade

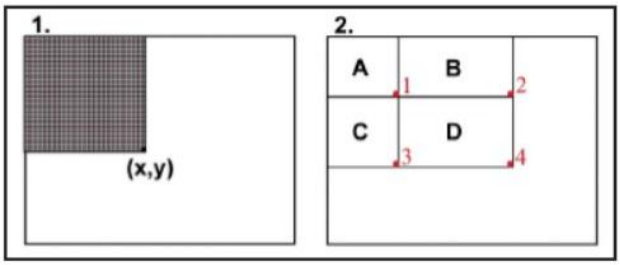

Fig. 3: Integral Image

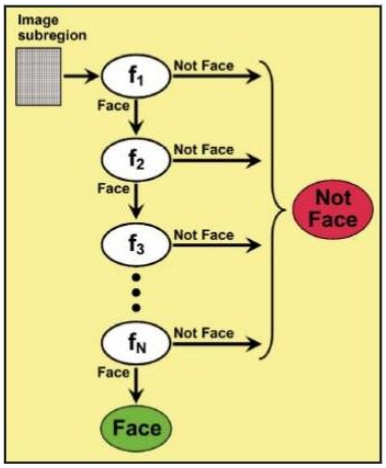

Fig. 4 Classifier Cascade

Fig. 5Training images in data base

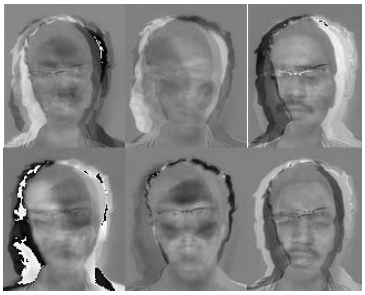

Fig. 6Eigenfaces of the corresponding training images

Fig. 7: The Average Face for the training set shown in Figure 6

Table 1: Face Detection Results

Condition of image	Total faces tested	Successful detection	Failed detection
A	5	5 (100%)	0
B	5	3 (60%)	2 (40%)
C	3	0	3 (100%)

Table 2: Face Recognition Results using 6 Training Images.

Condition of image	Total faces tested	Successful recognition	Failed recognition
A	5	4 (80%)	1 (20%)
B	5	3 (60%)	2 (40%)
C	3	0	3 (100%)

Condition A = Frontal view faces in a good lighting environment
Condition B = Frontal view faces in an average lighting environment
Condition C = Frontal view faces in a very poor lighting environment

Below are some images obtained from testing the system in different lighting conditions.

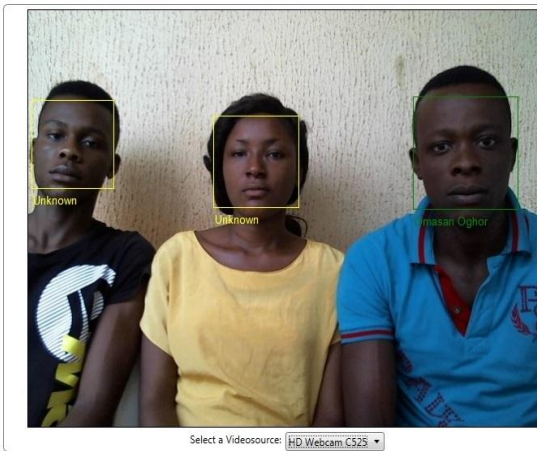

Fig. 8: Two failed recognition

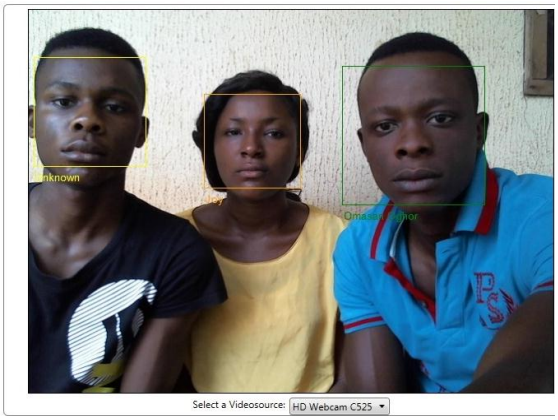

Fig. 9: One failed recognition

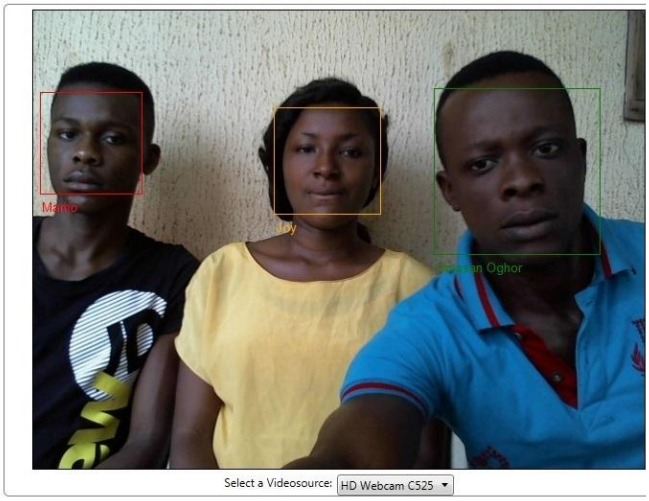

Fig. 10: Complete Detection and Recognition

References

[1] Lalendra S.B., "Frontal View Human Face Detection and Recognition". Thesis submitted to the department of Statistics and Computer Science, University of Colombo, May 2000.

[2] Paul V. & Michael J., "Rapid Object Detection using a Boosted Cascade of Simple Features" 2001.

[3] Roberto S.E., "Face Detection in C#", July 2012. PFC_ROBERTO_JAVIER_SOTO_ENTRENA.pdf, page 12 – 15.

[4] Roberto S.E., "Face Detection in C#", July 2012. PFC_ROBERTO_JAVIER_SOTO_ENTRENA.pdf, page 13.

[5] Adaboost, http://en.m.wikipedia.org/wiki/Computer_vision. Site accessed at 11am, 05-01-2014.

[6] Vijaya Y.L., Chandra K. B.T., International Journal of Recent Trends in Engineering, Vol. 1, No. 1, May 2009. "Facial Recognition using Eigenfaces by PCA"

[7] Reilly O., Bradski G. and Kaebler A., "Learning OpenCV", September 2008