*Nigerian Journal of Technology (NIJOTECH)*
*Vol. 39, No. 3, July 2020, pp.* **896 – 904**

Print ISSN: 0331-8443, Electronic ISSN: 2467-8821
www.nijotech.com
http://dx.doi.org/10.4314/njt.v39i3.31

# COMPARATIVE ANALYSIS OF SELECTED FACIAL RECOGNITION ALGORITHMS

**J. A. Popoola[1,*] and C. O. Yinka-Banjo[2]**
[1, 2,] DEPARTMENT OF COMPUTER SCIENCE, UNIVERSITY OF LAGOS, AKOKA YABA, LAGOS STATE, NIGERIA
***E-mail addresses:*** [1] *johnbimbo12@gmail.com*, [2] *cyinkabanjo@unilag.edu.ng*

**ABSTRACT**
***Systems and applications embedded with facial detection and recognition capabilities are founded on the notion that there are differences in face structures among individuals, and as such, we can perform face-matching using the facial symmetry. A widely used application of facial detection and recognition is in security. It is important that the images be processed correctly for computer-based facial recognition, hence, the usage of efficient, cost-effective algorithms and a robust database. This research work puts these measures into consideration and attempts to determine a cost-effective and reliable algorithm out of three algorithms examined.***

***Keywords:*** *Haar-Cascade, PCA, Eigenfaces, Fisherfaces, LBPH, Face Recognition.*

## 1. INTRODUCTION

The conventional means of access control relies on what an individual has rather than who he is [1]. An individual can have keys, passwords, PIN, security questions, birth day, last four digit of Credit/Debit Card of someone else, and successfully authenticate their way through using any of the listed means. It is evident that those means do not really define who we are. Of noteworthy is the fact that these means, if compromised, will grant an impostor access to our data and/or property any time they want.

Recently, the technology that allows identity verification of individuals became available based on a field called "biometrics". "Biometrics" is a word gotten from the Greek words "bios" and "metrikos"; *bios* means life while *metrikos* means measure. Faundez-Zanuy [2] defines recognition via biometrics as "those security applications that analyze human characteristics for identity verification or identification". In other words, physiological characteristics like facial features and fingerprints or behavioural qualities such as the pattern of keystrokes and handwriting are employed by this kind of access control to verify the identity of a living person.

### 1.1 Facial Detection and Recognition

In the past years, we have seen a considerable level of improvement in the quality of cameras, and there is enough evidence to suggest that this enhancement will continue. The major usage of cameras has been to capture and record various moments, as well as do video conferencing, but there can be other usages, one of which is computer vision.

Computer Vision, which is a field of Artificial Intelligence, aims at providing computers with the capability of visually understanding the world. It consists of tasks including methods for acquiring, analyzing and understanding digital images acquired from the real world, and extraction of high-dimensional data that can be used to produce numerical or symbolic information including, but not limited to, making decisions, etc.

The task of facial detection refers to a subset of computer vision capable of identifying the faces of people especially in digital images. Applications based on face detection use algorithms designed to detect human faces within larger images which most times might have other objects and landscapes as well as other human parts.

*** Corresponding author, tel: +234 – 703 – 395 – 4990**

## 2. LITERATURE REVIEW

### 2.1 Face Detection using Haar Cascades

Haar-like features have similarity with Haar wavelets. In object recognition, Haar-like features are features in a digital image that are made up of sets of two-dimensional Haar functions useful in encoding the local appearance of objects [3].

We can define a Haar wavelet as a mathematical function that outputs waves in the shape of a square having a beginning and an end. These square shapes are useful in recognizing signals.

Figure 1 shows an example of a Haar wavelet. A combination of several of these wavelets can be trained to identify lines, circles and edges that have different intensity of colors.

Figure 2 shows an example of a Haar-like feature. Detection of objects using Haar cascade classifiers is a method put forward by [4]. The approach, which is based on Machine Learning, introduces a method whereby we use both negative and positive images to train a cascade function. The idea is to eliminate negative examples using little processing. This trained function is eventually used in the detection and identification of objects in test images.

A rectangular Haar-like feature can be described as the difference of the sum of the pixels of areas inside the rectangle, which can be located anywhere in the original image, it can also be at any scale. This represents a 2-rectangle feature. *f,* which is the feature-value of a Haar-like feature having k rectangles is obtained as follows [5]:

$$f = \sum_{i=1}^{k} w^{(i)} \cdot \mu^{(i)} \qquad (1)$$

$$\sum_{i=1}^{k} w^{(i)} = 0 \qquad (2)$$

In (1), $\mu^{(i)}$ is the mean intensity of the pixels in image x enclosed by the *i*th rectangle. $\mu^{(i)}$ is the weight assigned to the *i*th rectangle. Conventionally, we set the weights assigned to the rectangles of a Haar-like feature to default integer numbers to satisfy (2).

Using a single classifier is not enough to effectively detect a human face, a combination of different Haar-like features is used such as the contrast between the eyes and forehead, the contrast between the nose and the eyes, and many others.

Figure 3 represents the decision-making process of a Haar-cascade used in determining if a feature being examined is present in the image. Each decision point has a specific feature that is checked for its existence in the input image. The Viola-Jones framework, which is among one of the first powerful real-time face detectors was successful because of three main ideas: Integral image, AdaBoost and attentional cascade structure.

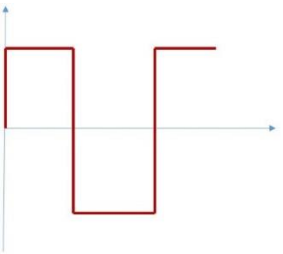

*Figure 1: A Haar Wavelet*

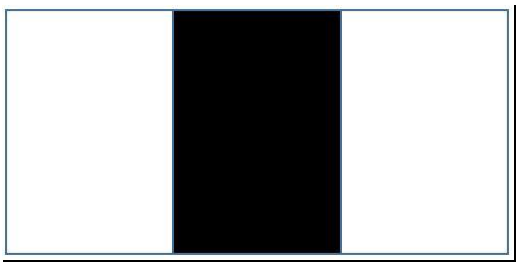

*Figure 2: A Haar-like feature*

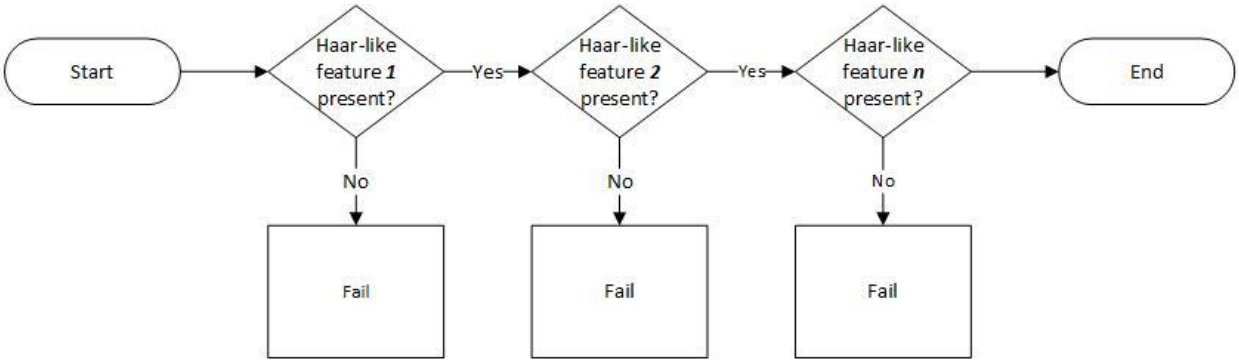


*Figure 3: Flowchart for Haar-cascade*

We can define the integral image as an algorithm used in efficient and fast calculation of the sum of rectangular subset of an image [5]. The integral image at a location *x,y* consists of the sum of the pixels above and to the left of *x,y,* inclusive. It is calculated as follows:

$$ii(x,y) = Ei(xi,yi) \quad (3)$$

In (3), *i(x,y)* is the original image and *ii(x,y)* is the integral image.

Figure 4 shows an integral image. With the use of four array references, we can calculate rectangle D's pixels sum. For instance, the value of the integral image at 1 is the sum of the pixels in rectangle A. At location 2, the value is A + B; at location 3, the value is A + C, and at location 4 it is A + B + C + D. As such, the sum within D can be computed as 4 + 1 – (2 + 3). Using this method, any rectangular sum can be computed in four array references.

### 2.2 Face Recognition using Eigenfaces

Eigenface is the name called a set of Eigenvectors applied in solving the problem of human face recognition. Eigenface approach was designed by [6] and was further made popular when Turk and Pentland [7] used Eigenfaces in face classification. Turk and Pentland [7] assumed that most faces lie in a low-dimensional subspace in a big image-space determined by *k* eigenvectors, i.e. *k* directions of maximum variance where $k <<< d$. For instance, if d is 10,000, *k* might be 100; this agrees to the fact that *k* is way smaller than *d*.

The next step is to use PCA (Principal Component Analysis) to find the vectors (also called the eigenfaces) that span that subspace. Next, represent all face images in the dataset as linear combinations of those eigenvectors. In the example case, that will be the coefficient of the 100 eigenvectors. When this is done, we represent the entire dataset (say 10,000) by just the coefficient of these eigenvectors, multiply those coefficients times the eigenvectors, the total sum of which will be the new image. This results in a tremendous data reduction.

### 2.3 Face Recognition using Fisherfaces

When we need to classify images, it is imperative to find a subspace that aggregates sample vectors belonging to the same class in a single spot. This technique is called Discriminant Analysis (DA), and its most popular derivative is Linear Discriminant Analysis (LDA), gotten from a suggestion by R.A. Fisher in 1936. When we apply LDA to find the subspace that represents most of the variance of the input data (i.e. image data), the result generated, which are the basis vectors which define that space, are FisherFaces.

### 2.4 Face Recognition using Local Binary Patterns Histogram (LBPH)

[8] proposed the idea of Local Binary Patterns as a visual classifier for texture classification, and in 2009, it was combined with histograms of oriented gradients (HOG) in order to improve on its level of performance. The idea that birthed this algorithm was that it was possible to describe a 2-dimensional surface texture using the local spatial patterns and the contrast of the gray scale.

Four parameters are used by LBPH and they are:

- Radius: this is the radius of the central pixel, most times set to 1,
- Neighbours: this is the number of neighbors of the central pixel, usually 8,
- Grid X: the number of horizontal cells,
- Grid Y: the number of vertical cells.

The procedure for the algorithm is as follows:

- Divide the view being examined into cells e.g 4x4 pixels for each cell
- Take a pixel and conduct a comparison between the pixel and each of its 8 neighbours in a clockwise or anti-clockwise direction
- Wherever the value of the centre pixel is greater than that of its neighbor, mark such pixel as 0, otherwise mark as 1. These series of 0s and 1s will result in an 8-digit binary number.
- Represent the output of the cell as a histogram, while taking note of the frequency of the occurrence of each number.
- Normalize and concatenate the cells' histograms.

The resulting output is a feature vector for the view being examined.

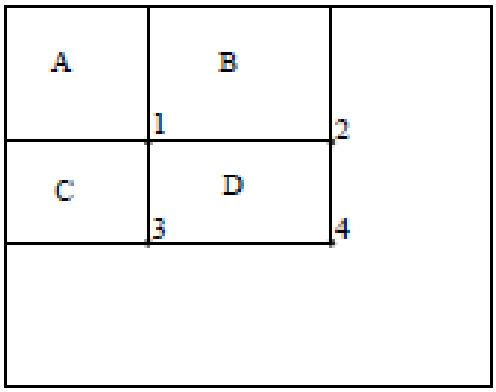


*Figure 4: Integral Image*

### 2.5 Comparison

#### *2.5.1 Eigenfaces*

On frontal faces, the accuracy of PCA is over 90%. Eigenfaces' accuracy is dependent on different factors such as positioning of the head and varying light intensity. However, it is fast on recognition and easy to implement. It also results in a tremendous data reduction as PCA is used to find the vectors that span the subspace.

#### *2.5.2 Fisherfaces*

The main concept behind Fisherface is Eigenface. While PCA is based on the search for the highest variance in the matrix, Linear Discriminant Analysis, on which Fisherfaces is based, works on the idea of labels such that when the required dimension is projected onto the image, the difference between the means of classes can be maximized [9]. Hence, LDA maximizes the ratio between the class scatter and its matrices. Thus, LDA is able to differentiate between feature classes better than PCA while also requiring a lesser amount of space. This makes LDA a better approach than PCA.

#### *2.5.3 Linear Binary Patterns Histogram (LBPH)*

LBPH is based on local binary operator. Unlike PCA and LDA, it considers results in the form of binary numbers. Its popularity and wide acceptance is due to its discriminative power and computational simplicity. The unique features such as computational simplicity and monotonic grey-scale changes makes it possible to analyze images in real-time.

## 3. ISSUES IN FACIAL DETECTION AND RECOGNITION

### 3.1 Illumination/Light Intensity

Illumination presents a significant obstacle for application of facial detection and recognition, as the effects are quite complex. Changing the direction of illumination on a familiar image is capable of shifting the location and shape of the shadows, as well as cause relative changes in highlights and contrast.

### 3.2 Pose

The pose problem is a situation where the same face has different appearances because of changes in the viewing condition. In a situation whereby the viewing angle of the observer changes or there is rotation in the position of the head, the pose of the face automatically changes. The facial recognition system still works fine with little change in pose but it becomes a challenge when the angle of rotation increases, and thus causes a no-match or mismatch with the available image in the database.

### 3.3 Facial Expressions

As human facial expressions and mood are not static, the differences pose a challenge for a Facial Recognition System to match the exact face stored in the database accurately.

### 3.4 Ageing.

Ageing is a natural phenomenon. With increasing age comes changes in the appearance of humans, which directly affects the efficiency of the Facial Recognition System.

### 3.5 Occlusion

The presence of items like glasses, beard, moustache, etc. on faces makes it challenging for the system to identify a face correctly. This is called occlusion.

### 3.6 Low Resolution

A situation when the resolution of the image to be recognized is lower than 16x16 will result in the problem of low resolution. Small scale camera applications found in CCTVs of supermarkets, streets, banks, etc. are prone to delivering pictures of this quality. Due to the distance between the camera and the individual, the face region captured is usually smaller than 16x16. A low resolution image like such has limited information, as most of the important details are lost, and this in turn drastically reduces the recognition rate [10].

## 4. METHODS AND IMPLEMENTATION

This section explains the methodology and procedures followed during implementation.

### 4.1 Technologies Used

1. Python Programming Language: The Python programming language is an interpreted, high-level, general-purpose programming language. Python has provision for high-level data structures such as list, dictionaries, dynamic typing and dynamic binding, classes, exceptions and automatic memory management [11].
2. Open Computer Vision (OpenCV) Library: OpenCV is part of the python distribution. It is an open source computer vision library designed to allow computer vision applications have a common infrastructure. The library has over 2500 optimized

computer vision and machine learning algorithms, which can be used to detect faces, recognize faces, identify and track moving objects and track camera movements among other usages. The OpenCV library is extensively used by technology giants like Google, Amazon, IBM, Honda, etc [12].

### 4.2 Process for Face Detection

Figure 5 represents what happens during the detection process. The detection is done using Haar Cascades. There are pre-trained classifiers in OpenCV, which can handle the recognition of the eyes, the face and other objects. A classifier is an algorithm that detects whether a particular feature is an object or not. This implies that it detects the object for which it has been trained to detect. Classifiers can be trained to detect dogs, chairs, bikes, etc. In this work, the classifier detects if a face is in an image form or not.

The result is stored in an XML format. We downloaded two cascades from the OpenCV repository: **haarcascade_frontalface_default.xml** pre-trained to detect a face and **haarcascade_eye.xml** to detect the eyes. Another important library imported is the NumPy (Numerical Python) library used for scientific computing in Python.

The detection process begins by creating the input feed using the computer webcam. This feed is converted to grayscale using the function *cvtColor()* of the OpenCV library. The output of this conversion is an array of the *RGB* (Red, Green, Blue) colour model, with different intensities of each colour specified as an integer between 0 and 255. To ensure dark images are not captured, an average of the array values is taken and a minimum acceptable value of 110 is set. The *CascadeClassifier.detectMultiScale()* module of OpenCV is invoked if this threshold value is met or passed, which returns the location of objects. The eye cascade is then run on the location data returned, and if eyes are detected, a marker with (x,y,w,h) coordinates is placed around a detected face. This module takes the following parameters: *image* (the converted video feed), *scaleFactor* (indicates the intensity of the reduction of the image size at each scale), *minNeighbors* (the higher the value, the lower the number of detections) and *minSize* (minimum object size, default is (30,30)).

### 4.3 Obtaining Image Data

We developed an application that captures 50 images per individual, with 3 seconds between captures, this made up the training set. At the startup of the application, a console prompt requests for the name of the subject about to be captured, which is stored alongside an auto-incremental integer ID in a text file. The detection system is invoked and if the level of illumination meets the specified minimum, the image is captured, cropped and saved with the ID appended to the name in the text file. This continues throughout a loop until 50 satisfactory pictures have been gotten.

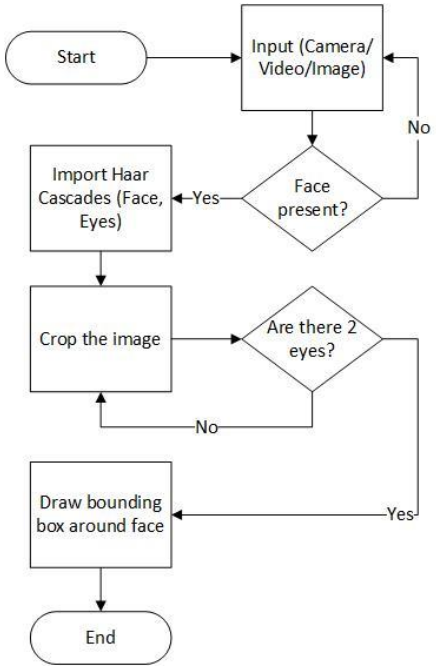


*Figure 5: Flowchart for face detection*

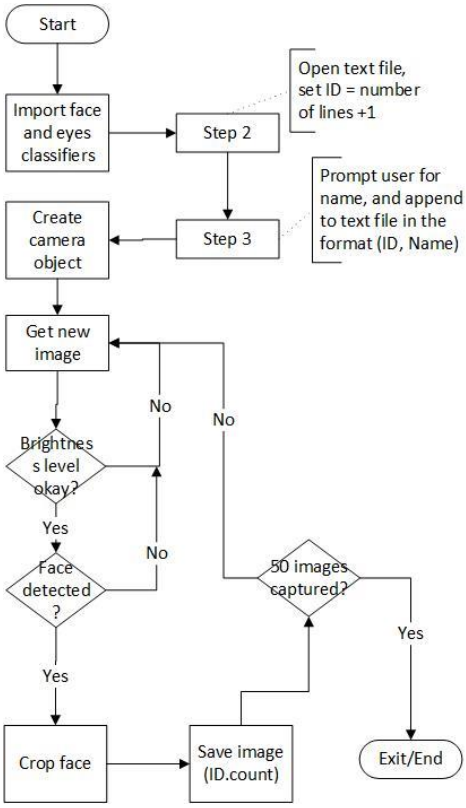


*Figure 6: Flowchart for image data collection*

Figure 6 shows the process flow for image data acquisition. It can be seen that an iteration happens until 50 images are captured for each subject.

### 4.4 Process for Facial Recognition

#### *4.4.1 Training the classifiers*

We created a method to retrieve each image's filename and their ID in the textfile, stored in two different lists using same index and thereafter converted to NumPy arrays. The arrays were passed to the *FaceRecognizer.train()* methods in each of the following recognizer objects. The features extracted from the dataset were generated and stored in XML files using *FaceRecognizer.save(FILENAME).*

We create a recognizer object by invoking the *FaceRecognizer* class in OpenCV for each of the three algorithms as follows:

*cv2.face.EigenFaceRecognizer_create(15)* for EigenFace, where 15 is the number of components for the PCA to create Eigenfaces,

*cv2.face.FisherFaceRecognizer_create(2)* for FisherFace, where 2 is the number of components for the LDA to generate Fisherfaces,

*cv2.face.LBPHFaceRecognizer_create(1,1,7,7)* for LBPH, where the parameters represent the

- radius (of the pixel at the centre),
- neighbours (the number of sample points needed to build the pattern. The higher the number, the slower the computation will be),
- number of cells in the x axis
- number of cells in the y axis

#### *4.4.2 Face Recognition*

Figure 7 depicts the process flow followed in the recognition module. The recognition process for the three algorithms follow similar patterns. We began by importing the face and eyes Haar classifiers, in order to ensure a face object is fed into the video feed. Next, we created a recognizer object and imported the training data for each of the algorithms. The training data fed into the recognizer object using FaceRecognizer.read(PATH_TO_TRAINING_DATA) method, where FaceRecognizer represents the recognizer object, after which the video feed was started.

Similar to the detection process, the feed is converted to grayscale. Next was to detect the face and the eyes by invoking the detectMultiScale() method of each classifier object. Once this was positive, the ID of the image as well as the confidence in the video feed was determined using the FaceRecognizer.predict() method of the recognizer object. The ID was passed to a lookup table that checks the text file containing names and IDs, and returns the respective name for the matching ID.

For each algorithm, if the confidence is higher than the set threshold, the ID is -1. The IDs in the text file starts from 1, and hence -1 will return "Unknown Face".

## 5. RESULTS

In order to effectively run this application, the required minimum hardware specification are:

- Processor: Intel Core i5
- RAM: 8GB

Figure 8 shows the subjects used; each subject captured 50 times in varying pose. Their names are appended here for test confirmation purpose.

A multipurpose image recognition application was developed to test if the system could detect the identity of the subjects using a totally different picture of them. Three identical applications were written to run through the three face recognition algorithms respectively, using different number of components to train in each iteration, starting from 1 to 150, and tested against a photo.

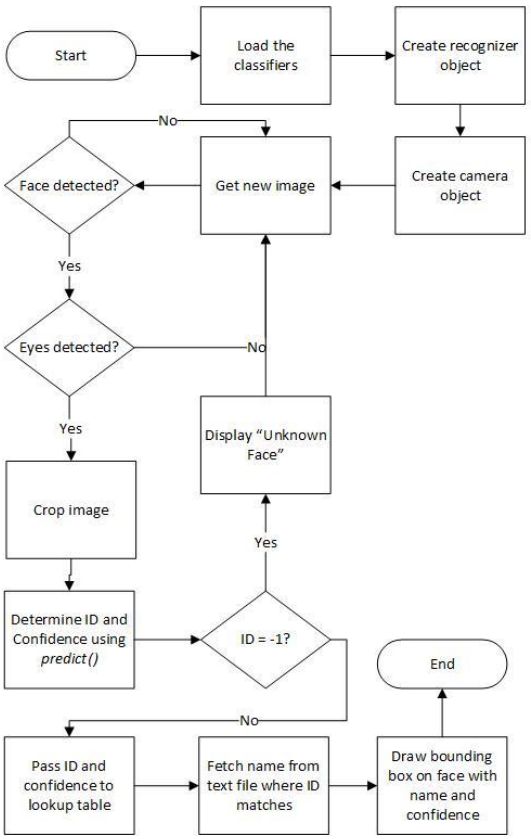


*Figure 7: Flowchart for image recognition*

*Figure 8: Subjects used in training the application*

ID: 1, Name: John ID: 2, Name: Dipo ID: 3, Name: David ID: 4, Name: Funbi

### 5.1 Eigenface Test Result

Figure 9 shows Eigenfaces test result. We notice that in the graph "ID vs Number of Components", when the number of components was 0-1, it reported ID 4. Increase in the number of components resulted in a constant value of 3. Confidence increased with increase in the number of components in the second graph "Confidence vs Number of Components".

### 5.2 Fisherfaces Test Result

Figure 10 shows Fisherfaces test result. Fisherface confidence increases until the number of components is 5 and will be used as the ideal value.

### 5.3 LBPH Test Result

Figure 11 shows LBPH test results. Since the image size is 110 X 110, maximum radius is 54. The ID is steady all the way to 50.

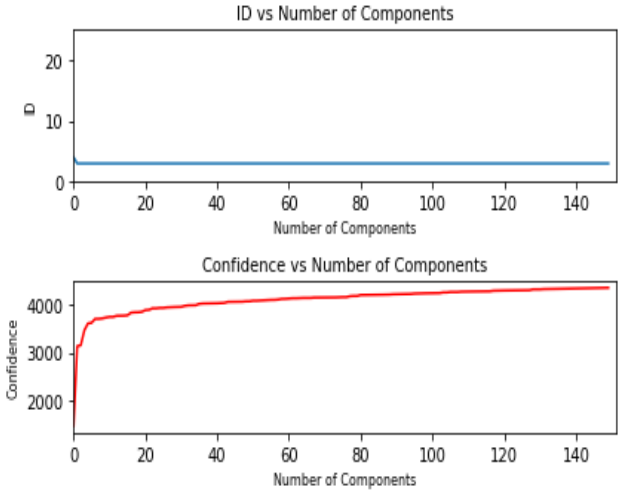


*Figure 9: Eigenfaces test result*

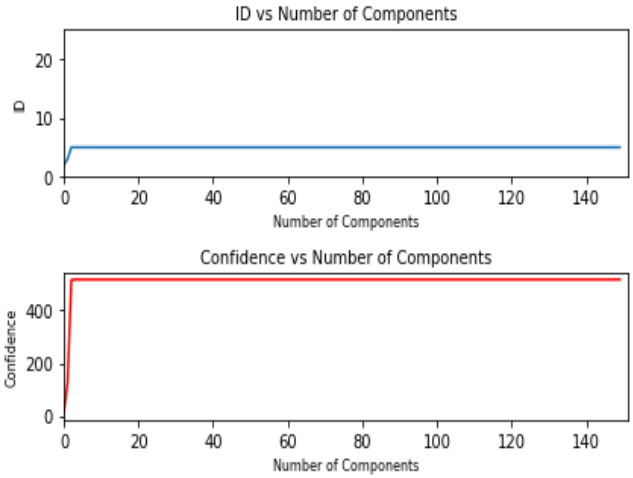


*Figure 10: Fisherfaces test result*

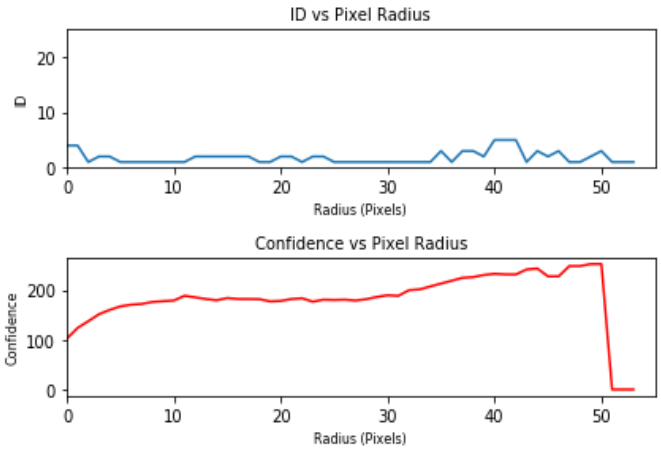


*Figure 11: LBPH test result*

### 5.3 Test to recognize old picture of subjects

The algorithms were tested on photos taken between 8 and 10 years ago and they were able to recognize the subject being tested.

Figures 12 and 13 show pictures taken 8 and 10 years ago respectively. The test reveals Eigenface performed lowest among the three, and indicates an unwavering level of confidence displayed by LBPH

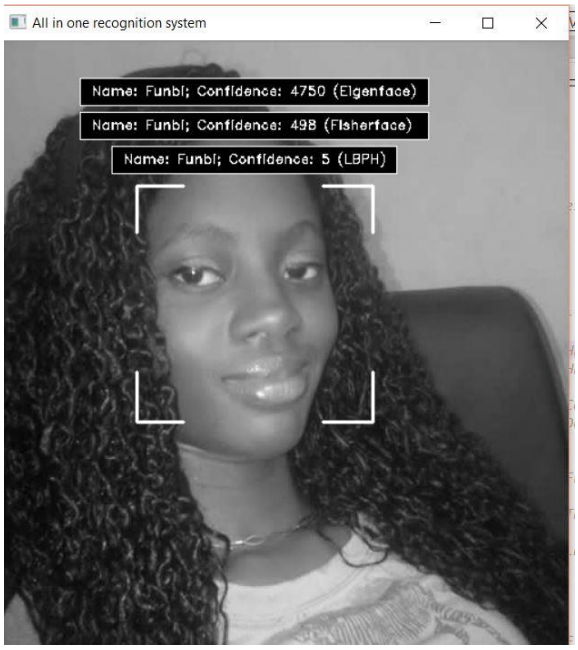


*Figure 12: Picture taken 8 years ago (Name: Funbi)*

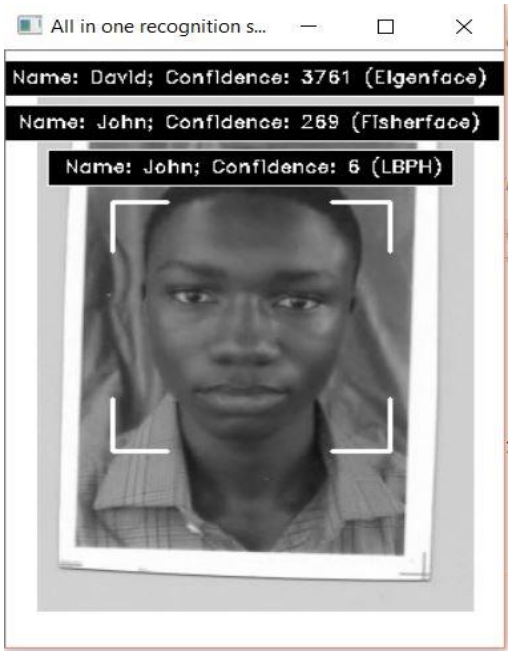


*Figure 13: Picture taken 10 years ago (Name: John)*

## 6. ANALYSIS

From Table 1, the result shows LBPH outperforms other algorithms with confidence factor in range 5-6 and has minimum noise interference. The outcome derived from the implementation reveals the existence of a trade-off between the correct recognition rate and the threshold value. The results shows that the error rate reduces with increase in threshold value possibly resulting in misclassifications.

The first observation we see in Table 2 is the percentage accuracy that increases as the number of pictures per person increases. In addition, when we trained the algorithms with 50 pictures per person, we notice a tremendous recognition accuracy among all three algorithms. This implies the importance of having a robust training set for the algorithms.

## 7. CONCLUSION

This work is an attempt to evaluate the strength and weakness of three facial recognition algorithms, namely Eigenfaces, Fisherfaces and Linear Binary Patterns Histogram (LBPH), and also the use of OpenCV classifiers and Haar Cascades in detecting human faces. It started with the history of Facial Recognition Technology, highlighting the major milestones in the course of advancement of this technology. Similar works done in this line were examined, with emphasis on their methodologies and results. Next, the technologies made used of were explained in line with the methodology used.

The use of Haar Cascades to detect human faces worked very well in this work, HP integrated TrueVision HD Camera was used to capture subjects, and the speed was good enough with no visible lag. Also, faces could still be detected even when the subject wore glasses.

*Table 1: Analysis of results*

| Criteria | Eigenfaces | Fisherfaces | LBPH |
|---|---|---|---|
| Confidence Factor (Based on output) | 2000-3500 | 200-500 | 5-6 |
| Threshold | 4000 | 500 | 7 |
| Principle of dataset generation | Component-Based | Component-Based | Pixel Based |
| Underlying Method | PCA | LDA | Histogram |
| Background Noise | Maximum | Medium | Minimum |
| Efficiency | Low | Higher than Eigenface | Highest |

*Table 2: Percentage Accuracy per Algorithm*

| | Training 20 pictures per subject | | | Training 50 pictures per subject | | |
|---|---|---|---|---|---|---|
| | Correct | Error | % Accuracy | Correct | Error | % Accuracy |
| Eigenfaces | 15 | 5 | 75 | 48 | 2 | 96 |
| Fisherfaces | 17 | 3 | 85 | 50 | 0 | 100 |
| LBPH | 19 | 1 | 95 | 50 | 0 | 100 |

Among the three algorithms tested, LBPH had the best performance, and when combined with Haar-cascades, it creates an economical and worthwhile face detection and recognition application.

This solution is cost-effective primarily because 50 or more pictures per subject might not always be available in a real-life application, and as such, the effectiveness and accuracy of the solution becomes undependable. However, with LBPH algorithm and its high level of accuracy, less robust training set can be used and results satisfactory.

This application can be embedded in classroom CCTV to automatically take attendance, thereby eliminating the need for manual approach, which is susceptible to errors and fraud. It can also be implemented in public places to alert security agencies when a recognized criminal is identified.

## 6. REFERENCES


[1] S.-H. Lin, "An Introduction to Face Recognition Technology," *Informing Science - Special Issue on Multimedia Informing Technologies - Part 2,* vol. 3, no. 1, pp. 1-7, 2000.

[2] M. Faundez-Zanuy, "Biometric security technology," *IEEE Aerospace and Electronic Systems Magazine,* vol. 21, no. 6, pp. 15-26, 2006.

[3] C. Papageorgiou, M. Oren and T. Poggio, "A general framework for object detection," in *ICCV '98: International Conference on Computer Vision*, Washington DC, USA, 1998.

[4] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features," in *Computer Vision and Pattern Recognition*, 2001.

[5] F. Jalled and I. Voronkov, "Object Detection Using Image Processing," arXiv, 2016.

[6] L. Sirovich and M. Kirby, "Low-dimensional procedure for the characterization of human faces," *Journal of the Optical Society of America,* vol. 4, no. 3, pp. 519-524, 1987.

[7] M. A. Turk and A. P. Pentland, "Face Recognition Using Eigenfaces," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Maui, HI, USA, 1991.

[8] D.-c. He and L. Wang, "Texture Unit, Texture Spectrum, And Texture Analysis," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 28, no. 4, pp. 509-512, 1990.

[9] L. Dinalankara, "Face Detection & Face Recognition Using Open Computer Vision Classifiers," *ResearchGate,* 2017.

[10] S. Ohlyan, S. Sangwan and T. Ahuja, "A Survey On Various Problems & Challenges In Face Recognition," *International Journal of Engineering Research & Technology (IJERT),* vol. 2, no. 6, pp. 2533-2538, 2013.

[11] M. F. Sanner, "Python: a programming language for software integration and development," *Journal of Molecular Graphics & Modelling,* vol. 17, no. 1, pp. 57-61, 1999.

[12] OpenCV, "About - OpenCV library," 2019. [Online]. Available: https://opencv.org/about.html.