Ormedian Research Institute

A REVIEW OF EXISTING FACE DETECTION & RECOGNITION ALGORITHMS AND THE PERFORMANCE EVALUATION OF HAAR CASCADE ALGORITHM ON IMAGES USING OpenCV

By

Adekola Oluwaseun Oluwafemi

`

March 12, 2023.

ABSTRACT

The development of several sectors and organizations has continually required the incorporation of some security measures into their systems to safeguard all resources concerned. Identity verification has become one of the means of providing access to some organizational facilities. For the need for a more reliable and effective system for verifying individuals' identity, several identity recognition systems have been discovered, one of which is Haar Cascade Classifier which has been extensively used for image detection and recognition purposes. This paper aims to discuss several concepts of Haar Cascade. This paper covers what Haar cascade is, its application, its implementation, and the pros and cons of using Haar Cascade Classifiers. The implementation to be carried out in this paper will focus on facial detection and recognition to enable us to understand the complexity in the process of image detection and recognition and to also understand the positive and negative contributions of different factors be it lighting variation, environmental conditions, and different facial expressions. This paper will as well evaluate the performance of the Haar Cascade classifier to detect unique facial characteristics and compare the results at every level of implementation to see how reliable and accountable the Haar Cascade is when it comes to the need for detecting and recognizing faces in images. OpenCV and Python programming language will be used to develop possible algorithms required in this paper for the final result to be judged upon.

Keywords: Haar Cascade Classifier, Face detection, Face recognition, Python Programming, OpenCV, Numpy

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Computer vision is a trending field in the world of technology today, it has proven to be one of the propelling fields for ensuring that highly reliable detection, recognition, and tracking systems are built with the right algorithms put in place. As true as this may be, one of the most challenging areas of computer vision is face recognition. Some of the factors responsible for this uncertainty are variation within individuals, sensors used, features extraction, algorithm employed, and data integrity. While data security has become very essential, several other techniques such as the use of keys and passwords proved to be reliable, they are still not completely free from being hacked if mistakenly exposed, but the advent of image recognition systems has provided more security features in the sense that human face tends to be classified as a unique characteristic of an individual which is not likely to be by-passed by another individual in the presence of the security system. Just like every other biometric system in which feature extraction is very important before further processing could be done, so it is in face recognition systems. For a face to be successfully recognized, it is important that the face detection process is first initialized for proper localization and extraction of facial features and regions from the background. During the process of recognition, matching should be within a tolerance of approximation to ensure that generated results remain desirable even if it is not completely accurate because the presence of noise and other factors such as variations in images could make recognition systems produce probabilistic results. While there are intrinsic and extrinsic factors responsible for variation in facial appearance, an attempt is continually made by researchers to ensure that facial detection and recognition algorithms are being improved upon. Some facial recognition algorithms have been developed over the past years and up till today to ensure that better accuracy is achieved. As improvement is continually made on each developed algorithm, factors such as poses, illumination, and occlusion remain part of the technical problems they all try to solve. Modern research areas of computer vision especially recent face recognition have been able to reach an identification rate of above 88% for larger databases that have well-controlled pose and illumination

conditions, and continuous improvement has made face recognition systems superior to password or key-protected systems.

With tons of research in the computer vision field, more sophisticated facial recognition algorithms have been developed to detect and recognize different facial positions such as frontal view, profile view, etc. In the attempt of building an application that detects and recognizes faces, Haar Cascade is one of the oldest and simplest algorithms used for facial detection and recognition. It is one of the easiest approaches used for detecting objects in an image. It is useful for the identification of custom objects and the classification of real-time objects. Haar Cascade is not limited to just one specific object when it comes to detection, it has the capability to detect several other objects such as persons, buildings, cars, etc. regardless of the scale of such an object.

LITERATURE REVIEW

2.1 INTRODUCTION

Several facial detection and recognition algorithms have been developed, each with different approaches for the extraction of information contained in an image and matching it with the input image. Some of the algorithms in existence that are typically used for detection and recognition include the following Eigenfaces (1991), Local Binary Patterns Histograms (LBPH) (1996), Fisherfaces (1997), Scale-Invariant Feature Transformation (SIFT) (1999), Speed Up Robust Features (SURF) (2006). Apart from these facial recognition algorithms, more advanced object recognition algorithms have been developed, for example, YOLOv3, Tiny-YOLOv3, and Faster R-CNN are used for more complex applications. In this paper, a brief discussion about each of these algorithms will be considered as they will form the basis of the related work which will help us better understand face recognition algorithms and Haar Cascade that will be implemented and demonstrated in this paper.

2.1.1 EigenFace

As the name implies, an Eigenface is just a name describing some sets of eigenvectors used for image recognition in some computer vision problems. It was an approach developed by Kirby and Sirovich which in 1991 was applied in face classification by Mattew Turk and Alex Pentland, a technique in which the eigenvectors of covariance matrix could be calculated in an automated face recognition system and still ensures dimension reduction. Its technique of recognizing faces starts by saving images i.e. gallery images seen by the system as collections of weights that describe how each eigenface can be attributed to such an image so that when a new face is examined for recognition on such a system, the closest match is checked for by comparing the collection of weights of the new face against all the weights available in the gallery set. As good as the Eigenface approach in facial recognition is in the aspect of dimension reduction and improving speed and efficiency of a system which makes it superior to some other algorithms, it begins to experience failure when the variation between the seen images and probe images becomes too large [1]. It can be concluded that the Eigenface approach aims at improving the accuracy of the image recognition system through the preprocessing method [2].

2.1.2 Fisherface

Fisherface is a very popular face recognition algorithm introduced in 1997 and is considered to be superior to some other face recognition techniques like Eigenface based on the fact that separation that exists between classes during the process of training is maximized. According to [3], the Fisherface algorithm in face recognition uses Principal Component Analysis (PCA) technique which is then followed by Linear Discriminant Analysis (LDA) technique. While PCA aims to reduce the space dimension of a face, LDA helps to extract the characteristic features of an image. Therefore the Fisherface algorithm ensures that the principal components that separate one different individual are carefully extracted. Hence, Fisherface is an improvement over the eigenface method. Even in the presence of varying illumination, the Fisherface algorithm still produces good results by following a systematic approach of retrieving data, image processing, feature generation, and recognition. While the Fisherface method offers some improvements, i.e. it is a modified version of the Eigenface method, it is not free from noise and blurring effects, and the process of computation required in the Fisherface algorithm is quite complicated and complex [4].

2.1.3 Scale-Invariant Feature Transform (SIFT)

According to [5], SIFT was invented in 1999, with other improvements being made to it by researchers as years passes-by to further develop an advanced version of SIFT called the Modified Scale-Invariant Feature Transform (MSIFT). In SIFT algorithm, three actions are performed i.e. detection, description, and matching of local features in an image. SIFT technique starts by extracting features of objects from reference images and stores these features in the database. When a new image is to be recognized, features of the objects in this image have to be compared with the features present in the database for proper matching based on the Euclidean distance of feature vectors. Objects with correct matching results are filtered out as good matches [6]. SIFT is considered one of the most accurate algorithms used for detection and recognition but it is complex and computationally heavy because the computation of gradients is time-consuming and it cannot be recommended for low-power devices.

2.1.4 Local Binary Pattern Histogram (LBPH)

LBPH has its root in the LBP, LBP is one of the cases of the texture spectrum model introduced in the 1990s. In an attempt to improve the detection and recognition performance, LBP was combined with the descriptors of the Histogram of oriented Gradients (HoG) to form LBPH in 2006. LBPH has now become

a powerful feature when it comes to cases of texture classification [7]. With LBPH, both frontal and side views of the human face can be recognized. While this is of great advantage, the LBPH has some limitations whenever an object in an image has some disorientation or lighting variations, these factors put challenges on the performance of LBPH during the detection and recognition process. It uses an operator called the LBP operator with some set threshold values to convert pixels in an image to binary values, Histogram extraction, and then the recognition phase follows [8]. LBPH is an algorithm used extensively in texture analysis, biometrics, and computer vision.

2.1.5 Speed-Up Robust Features (SURF)

In 2006, SURF was presented as a patented local feature detector and descriptor which is widely used for a variety of computer vision tasks. It is an algorithm that uses the same principle as SIFT except that it is several times faster and more robust than SIFT [9]. SURF in its process of detecting points of interest utilizes the integer approximation of Hessian Blob detector determinants using precomputed integral images. Just like the SIFT algorithm, it consists of three main steps which are interest points description, detection of the local neighbourhood, and matching. As good as it is, it has some limitations such as low matching accuracy in the presence of a large rotation angle and a large angle of view, its computation is also huge [10].

2.1.6 Faster R-CNN Face Detection

The Region-based Convolutional Neural Network (R-CNN) which was first proposed in 2013 was developed with the goal to receive images as inputs and then capture each of the objects in the images by producing surrounding boundaries in form of boxes around those objects. With further development and research, R-CNN has been made to detect some regions of interest (ROI) in images. This development has really helped in the aspect of face detection. Further advancement has been made to R-CNN which led to the development of the Faster R-CNN algorithm in 2015, the latter having the ability to run a neural network on an image once unlike the R-CNN that computes neural network features on each of the ROI involved. Just like the R-CNN, Faster R-CNN integrates ROI Pooling with neural networks for feature extraction from each of the regions of the proposal. Faster R-CNN saves time and has better accuracy [11].

2.2 OTHER RELATED WORK

In the work of [12], a comparison was made between the LBP and Haar Cascade on different databases to examine the speed at which each of the two algorithms detects objects using the OpenCV library of computer vision. It was gathered that LBP detects faster than the Haar Cascade but with lesser accuracy. According to [13], they proposed a paper in which two different algorithms, Haar Cascade and Edge Orientation Matching algorithms were compared, from the result analysis, they concluded that the detection accuracy in Haar Cascade Classifier is higher than in Edge Orientation Matching. Deep Learning face detectors and other complex algorithms for facial recognition are also out there trending as the latest algorithms for detecting objects in images.

HAAR CASCADE THEORIES & EXPERIMENTAL RESULTS

3.1 HAAR CASCADE THEORIES

In this paper, Haar Cascade will be experimented with face detection and recognition using the OpenCV computer vision library to evaluate the performance of Haar Cascade Classifiers and take into account some lapses associated with it. Haar Cascade classifier also regarded as the Viola-Jones algorithm is an algorithm that is applied when solving computer vision task that involves facial detection (finding faces in an image) or facial recognition (finding characteristics that best describes an image). It is one of the well-known algorithms applied in most real-time face recognition applications e.g. digital cameras. Several efforts have been incorporated into the Haar Cascade classifier in an attempt of improving its accuracy. To properly train features for efficient and effective detection and recognition, researchers are trying their best to ensure that the error rate is well corrected in this algorithm. With some level of performance of the Haar Cascade, it is still not free from errors of some kinds majorly due to some noise that is present in an image. The Haar Cascade algorithm has been modified to have three important components i.e. Haar Features, Integral representation, and Adaboost to offer the algorithm some level of improvement. The fundamental component responsible for detecting human face features and performing feature extraction is the Haar features while the Adaboost helps in eliminating redundancy by selecting the best features having the lowest error rate. For some useful reasons, Haar Cascade has been applicable in so many areas. Some of which are highlighted below.

3.2 APPLICATIONS OF HAAR CASCADE

- **Facial Recognition:** Haar Cascade could be used to set some security features for certain devices. The mobile phone we use today now comes with a facial recognition security feature that allows login of only the registered users.
- **Industrial Use:** With Haar Cascade algorithms, robots, and intelligent machines can recognize objects that are to be picked and transited from one point to another easily without supervision. This has helped in the use of machines to perform tasks repetitively.
- **Autonomous Vehicles:** with the help of Haar Cascade, the nature of the surroundings and objects such as pedestrians, streetlights, etc. are easy to identify by the autonomous vehicle operating in such an environment and this aids its decision-making.
- **Object Recognition:** The application of Haar Cascade is not only limited to facial detection or recognition, it has now been extended to fields of object recognition.

- **Robotics:** Manufacturing tasks are now being performed easily with smart automation and object recognition which makes it easy for robots to easily recognize nature and perform tasks safely.
- **Agriculture:** Haar Cascade application in agriculture helps recognize bugs that damage farm produces.

3.3 HOW HAAR CASCADE WORKS

Haar Cascade applies a function called cascading function with cascading window to compute features that are present in every window and further classify the positive data points (needed part of the object i.e. part of the image with faces) and the negative data points (unwanted parts i.e. part of the image without faces) by traversing the window across the input image. Hence, it assigns positive data points to the cascade windows which can be considered as part of our object but assigns negative data points to those cascade windows that cannot be a part of our object. While it tries to produce the best results, hyperparameters must be well-tuned to ensure that false/negative data points are rejected as fast as possible. Also in the process of classifying images, to compute and match features, this is done regardless of the image scale or location.

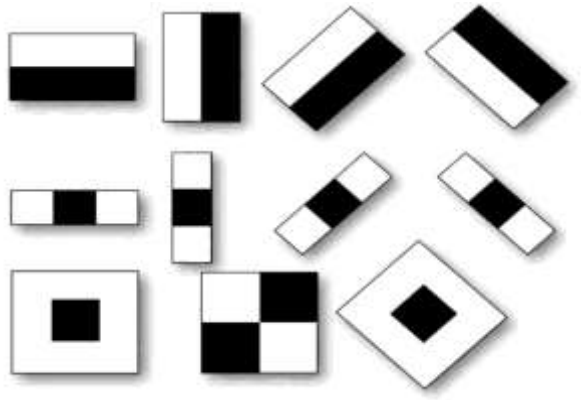


Figure 3.1: Typical Haar Features [13].

Figure 3.2: Haar feature traversing an image.

3.4 SIMPLE HAAR CASCADE FACE DETECTION OPERATION

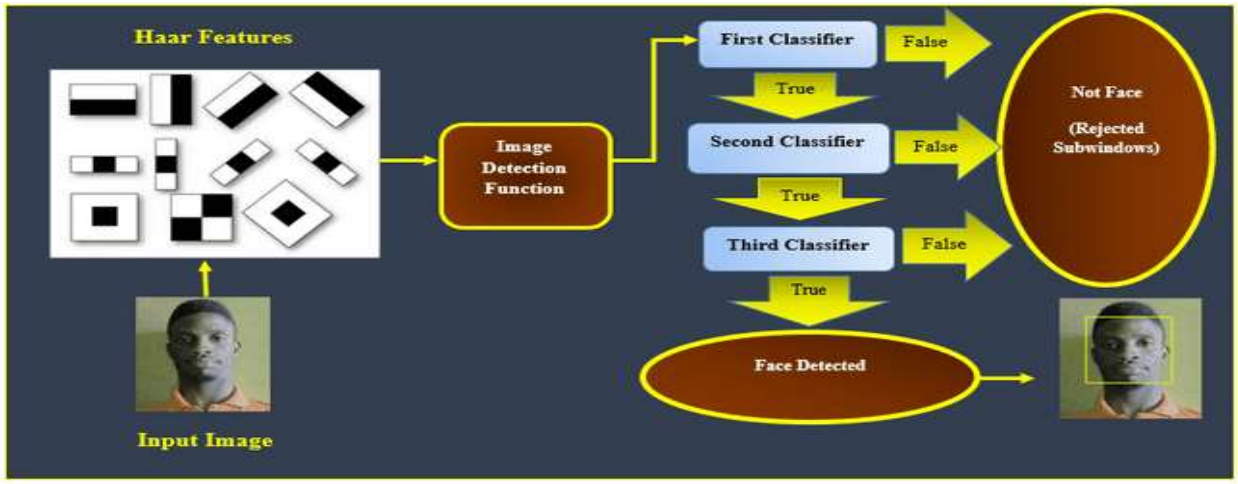


Figure 3.3: Haar Cascade Operation.

3.5 HAAR CASCADE FACE DETECTION FLOWCHART

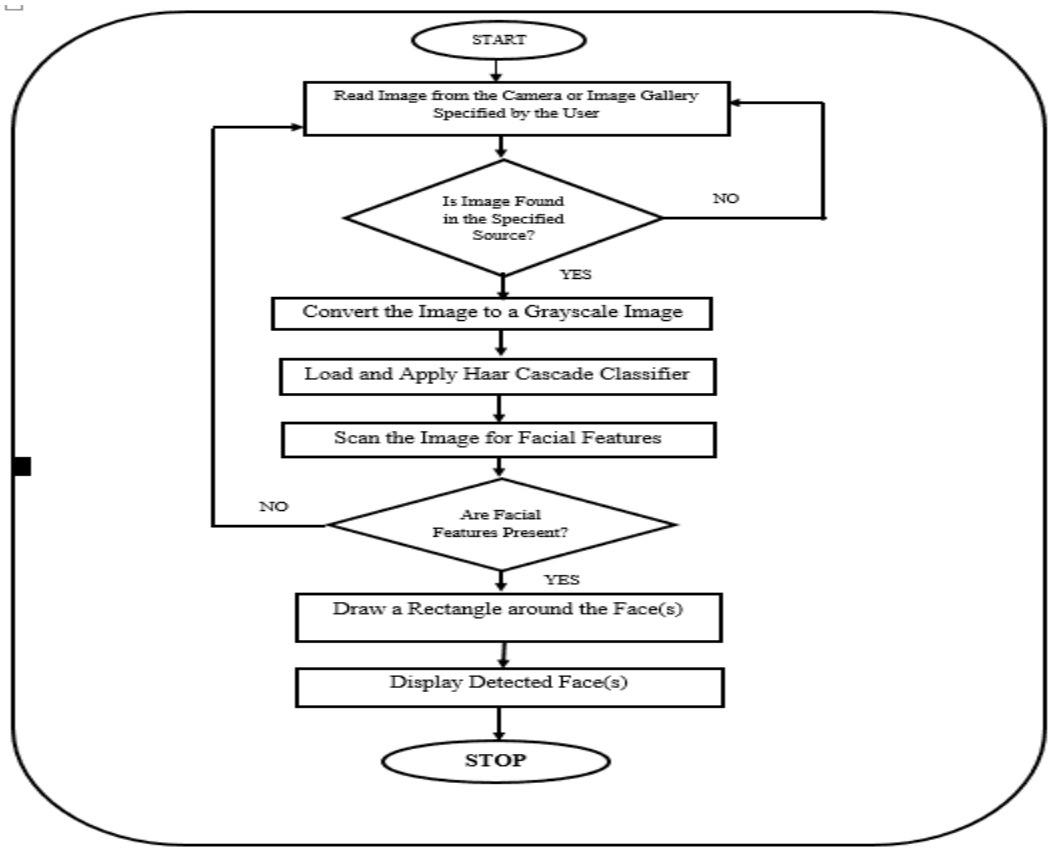


Figure 3.4: Haar Cascade Face Detection Flowchart.

3.6 EXPERIMENTS

Haar Cascade Classifier will be used to experiment two different cases in this paper to see the performance of the Haar Cascade algorithm. The two cases to be examined are:

- Face detection by reading images directly from the Image Gallery or Camera
- Face recognition by testing unseen Images on custom datasets Using Haar Cascade

Applying Haar Cascade to face detection tasks requires that the XML format of the pre-trained Haar Cascade must be downloaded before it can be integrated into the user-written codes to get results. With this being done, many other settings are still required but the most fundamental one is the detectMultiScale() function that has to be used to detect objects of different sizes in an input image and it must return the list of rectangles which has detected objects in it. But in this paper, detectMultiScale() function will be used to detect faces that are present in an image. A few parameters to be passed into this function basically helps in defining an image as well as fine-tuning the image until results are yielded. Parameters to be set are:

ScaleFactor: This parameter is essentially used for the scale pyramid. It specifies how the image size should be reduced at each image scale. ScaleFactor value set at 1.1 works perfectly well for most detection cases.

MinNeighbors: This specifies the number of neighbours to be contained in each rectangle. This is a very important factor in determining the quality of the results to be generated. With higher values set for minNeighbors, detection quality becomes lesser while some considerably smaller values usually within 3 to 6 could produce the best detection results.

minSize: Used to specify the minimum allowable size of an object.

maxSize: Used to specify the maximum allowable size of an object.

Experiment 1: ***Face detection by reading images directly from the Image Gallery or Camera Using Haar Cascade Classifier Algorithm***

Haar Cascade classifier which has been pre-trained to detect human faces will be applied in this section to observe how accurately a human face can be detected in an image through the use of the Haar Cascade algorithm. Getting to work with this algorithm requires some steps and key parameters are put in place to get the implementation done correctly.

A. Detection of Faces from Images in the Image Gallery

The Haar Cascade algorithm will be applied to different images in this section to see the possible effects of Haar Cascade as regards face detection.

Case one

CV > facedetect.py

```
1  import cv2 as cv
2  img = cv.imread('Images/seun.jpg')
3  gray_img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
4  haar_cascade = cv.CascadeClassifier('ha_faceclas.xml')
5  face_detect = haar_cascade.detectMultiScale(gray_img, scaleFactor = 1.1, minNeighbors = 3)
6  for (x, y, w, h) in face_detect:
7      cv.rectangle(img, (x, y), (x+w, y+h), (0,255,225), 2)
8  cv.imshow("Detected Faces", img)
9  print("Number of faces detected is {}".format(len(face_detect)))
10 cv.waitKey(0)
```

Figure 3.5: Algorithm for face detection using images from the Image Gallery.

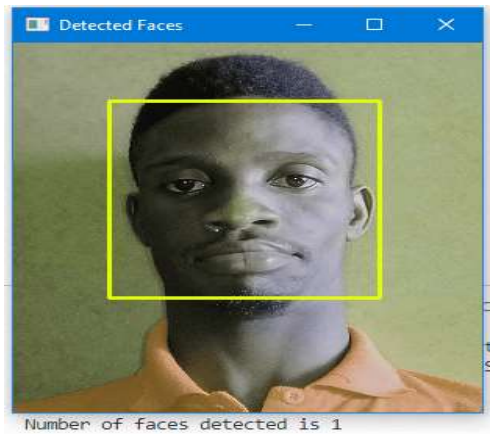


Figure 3.6: Case One Face Detection Test Result.

Case Two: *With minNeighbors = 1*

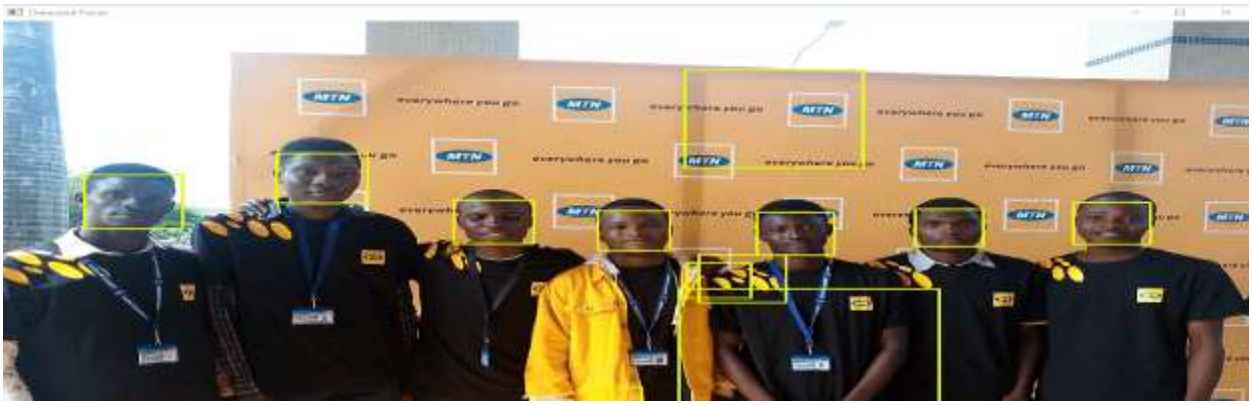


Figure 3.7: Case Two Face Detection Test Result.

Case Three: *With minNeighbors = 2*

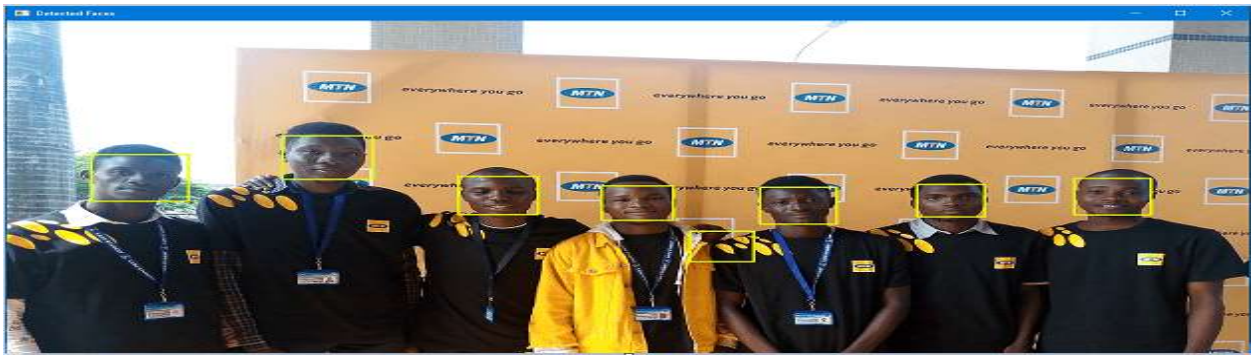


Figure 3.8: Case Three Face Detection Test Result.

Case Four: *With minNeighbors = 3*

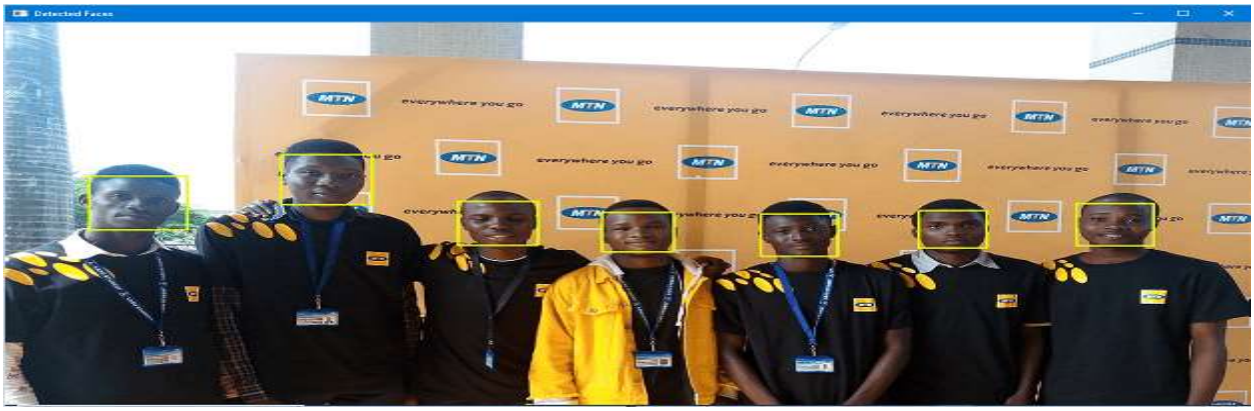


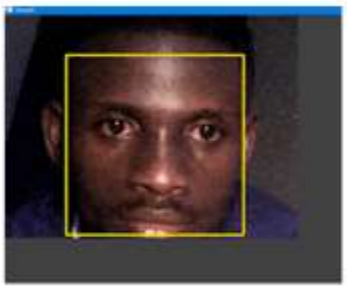
Figure 3.9: Case Four Face Detection Test Result.

B. Detection of Faces directly from a Camera/Video Frame

CV > camfacedetect.py > ...

```
1  import cv2 as cv
2  faceDetector = cv.CascadeClassifier('ha_faceclas.xml')
3  img = cv.VideoCapture(0)
4  while img.isOpened():
5      rect, frame = img.read()
6      gray_image = cv.cvtColor(frame, cv.COLOR_BGR2GRAY)
7      faces = faceDetector.detectMultiScale(gray_image, scaleFactor = 1.1, minNeighbors = 3)
8      for (x, y, w, h) in faces:
9          cv.rectangle(frame, pt1 = (x, y), pt2 = (x+w, y+h), color =(0,225,255), thickness = 3)
10     cv.imshow("window", frame)
11     if cv.waitKey(1) & 0xFF == ord('e'):
12         break
13 frame.release()
14 cv.destroyAllWindows()
15
```

Case Five



Output

Figure 3.10: Algorithm for Face Detection Directly Using Camera.

Table 3.1: Results and Observations from Face Detection Test

Case	Observation
Case one	Reading an image having a single face seems easier to detect when the ScaleFactor = 1.1 and the minNeighbors = 3. After testing this on several other images involving only one face.
Case Two	In situations in which there is more than one individual in the picture, extra tuning may be required to get all faces detected. In case Two, with minNeighbors = 1 indicated in the detection output that more than 7 faces were detected. This implies how weird the result produced could be sometimes when the image has a lot of noise in it.
Case Three	With minNeighbors = 2, the noise was reduced with the image that 8 faces were detected instead of 7 but this is more reasonable than the result obtained in case two. But still, hyperparameter tuning is still required.
Case Four	With minNeighbors set = 3, all faces were detected accurately without any other false facial features constituted windows.
Case Five	Reading a frame directly from a camera also requires careful tuning. It was observed that human motion, change in orientation, and illumination revealed the presence of some noise in the image while capturing. But when the movement became stable with a reasonable amount of illumination and other factors. The region with the facial feature was clearly specified by rectangular boxes indicating that a face has been detected.

Experiment 2: ***Face recognition by testing unseen Images on custom datasets Using Haar Cascade***

In this section, a total of 480 Images with 120 Images per four different Nigerian artistes (Davido, Olamide, Tiwa Savage, and Wizkid) have been collected and grouped into three different essential folders i.e. Training set, Validation set, and Testing Set with each artiste's images constituting 84, 18, and 18 image files in the folders respectively. 70% for the Training set, 15% for validating set, and 15% for the Testing set. To successfully implement the face recognition goal in this paper, the selected training data were trained on the Haar Cascade and saved in "yml" format in order to make classification easier and also make such a program a reusable one on any other platform. After training, a set of image files were validated on this training dataset to see how accurately this system can recognize already stored image files. Also upon testing, unseen images are tested on the pre-trained dataset to see how well the system can accurately recognize new image files by matching characteristic features of this new image and those stored in the database.

The training data folder samples for each artiste are shown below

a. Davido

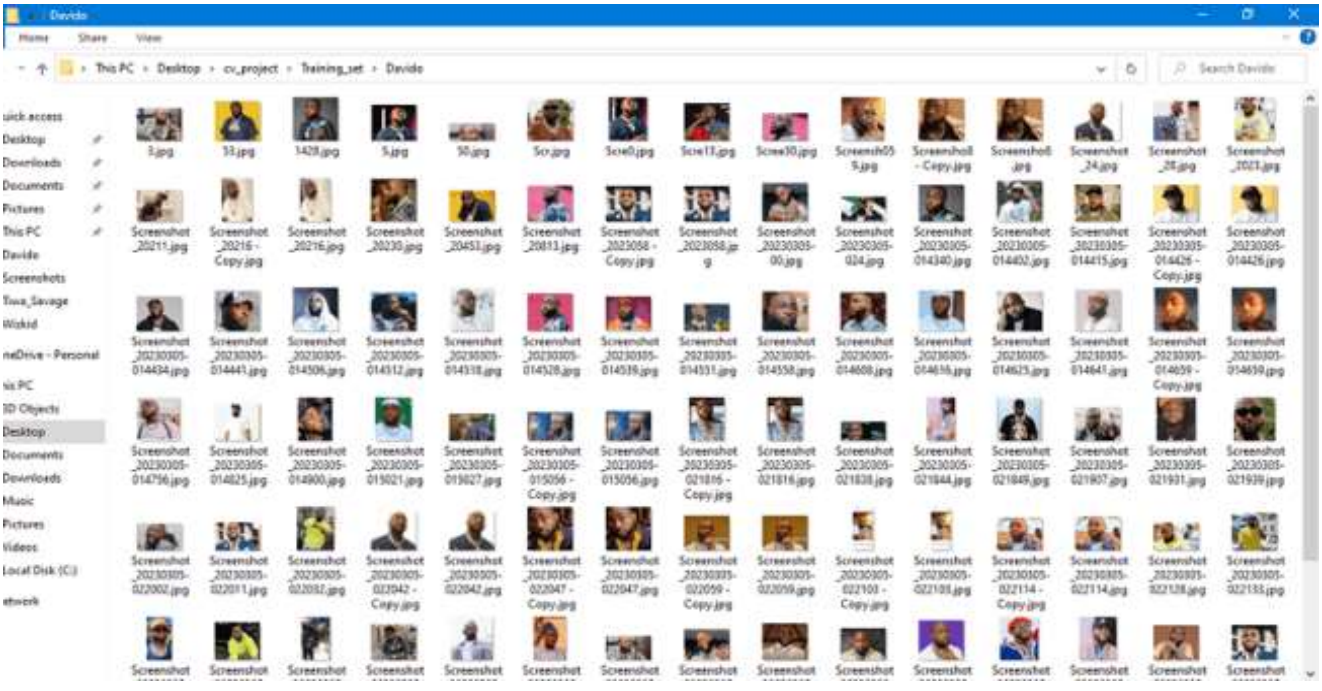


Figure 3.11: Items in the Training Set Folder 1.

b. Olamide

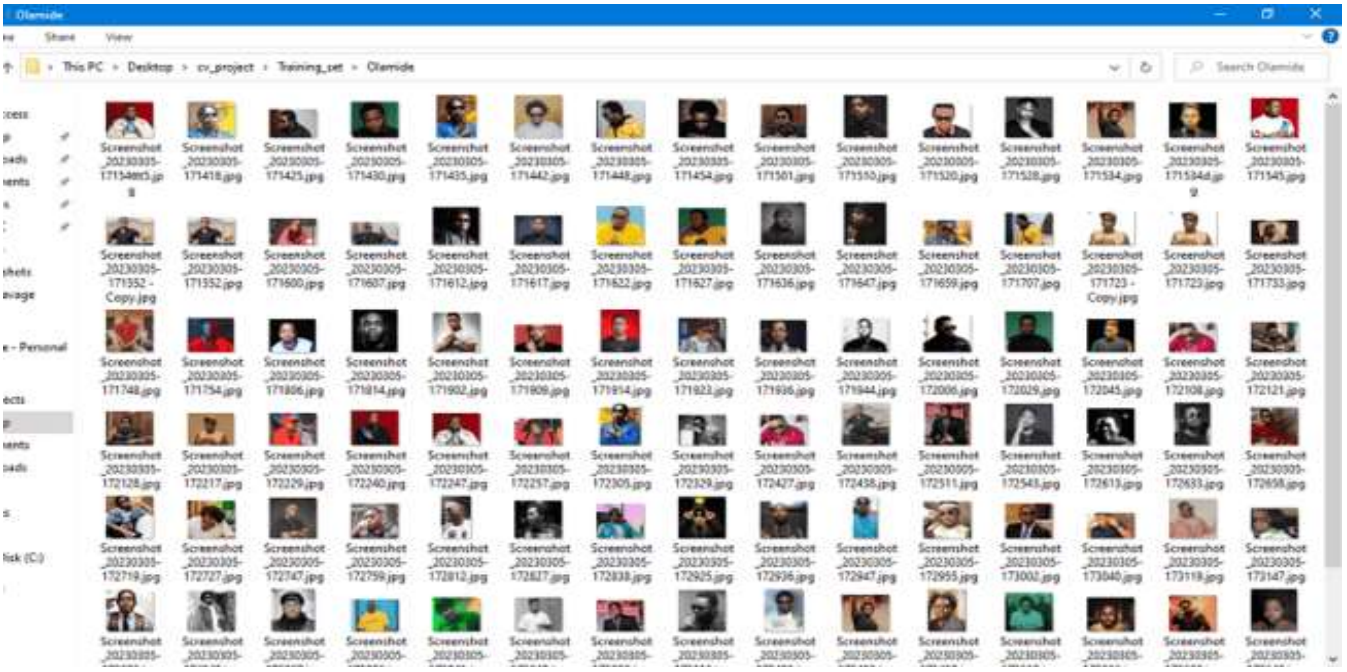


Figure 3.12: Items in the Training Set Folder 2.

c. Tiwa Savage

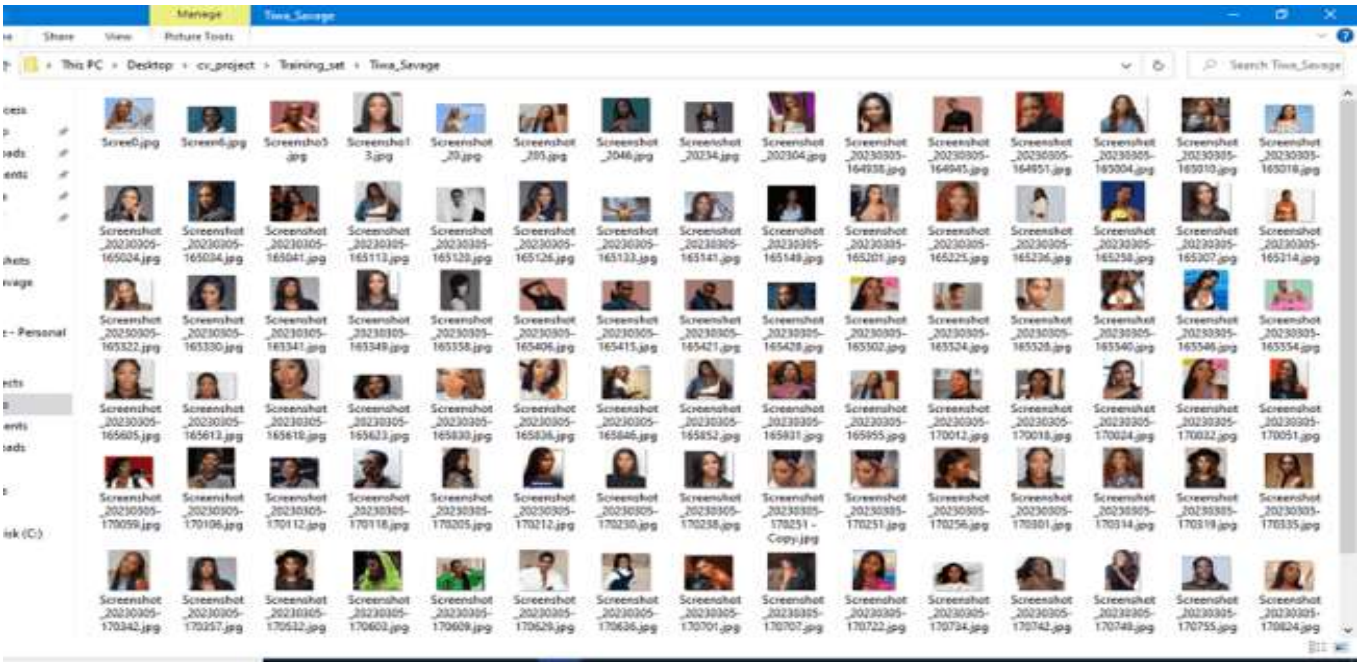


Figure 3.13: Items in the Training Set Folder 3.

d. Wizkid

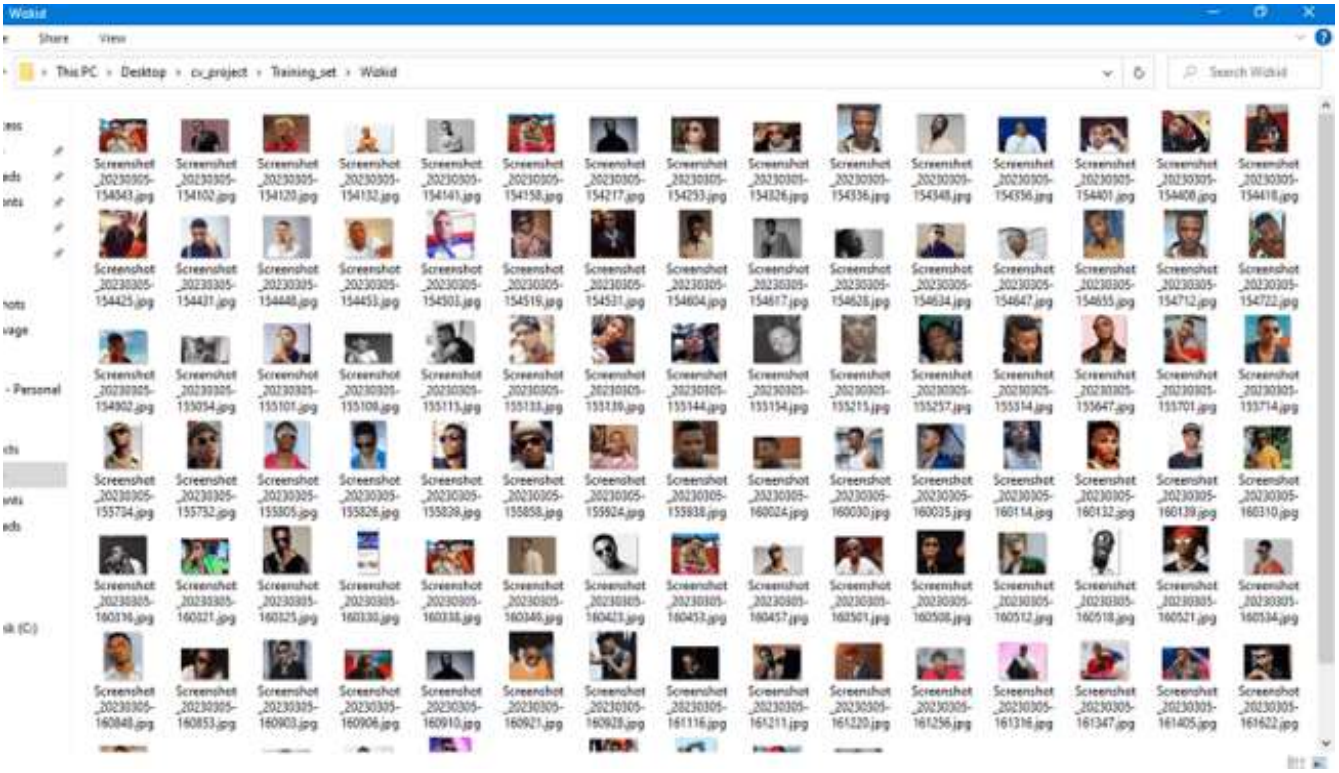


Figure 3.14: Items in the Training Set Folder 4.

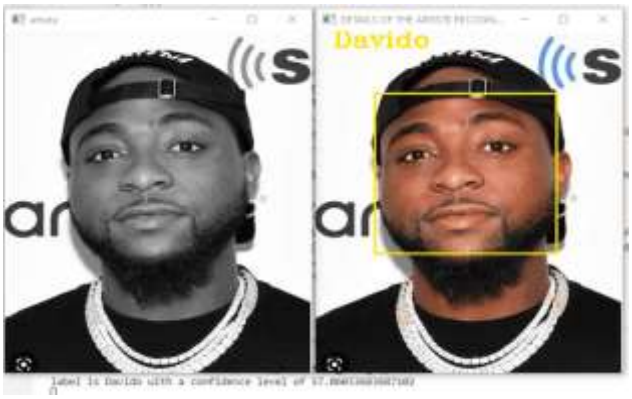
```
CV > face_recog.py > ...
1   import cv2 as cv
2   import numpy as np
3
4
5   haar_cascade = cv.CascadeClassifier('ha_faceclas.xml')
6   artistes = ['Davido', 'Olamide', 'Tiwa Savage', 'Wizkid']
7   face_recognizer = cv.face.LBPHFaceRecognizer_create()
8   face_recognizer.read('face_trained.yml')
9   image =cv.imread(r'C:\Users\SHOPINVERSE\Desktop\cv_project\validation_set\wizkid\Screenshot_20230305-16166.jpg')
10  gray_image = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
11  cv.imshow("artiste", gray_image)
12  face_detect= haar_cascade.detectMultiScale(gray_image, scaleFactor = 1.1, minNeighbors = 4)
13  for (x,y,w,h) in face_detect:
14      faces_roi = gray_image[y:y+h, x:x+w]
15      label, confidence = face_recognizer.predict(faces_roi)
16      print("label is {}".format(artistes[label]), "with a confidence level of {}".format(confidence))
17      cv.putText(image, str(artistes[label]), (20, 20), cv.FONT_HERSHEY_COMPLEX, 1.0, (0, 225, 255), thickness=2)
18      cv.rectangle(image, (x,y), (x+w, y+h), (0, 225, 255), thickness=2)
19  cv.imshow("Face Detected", image)
20  cv.waitKey(0)
```

Figure 3.15: Face Recognizer Algorithm for Validating and Testing.

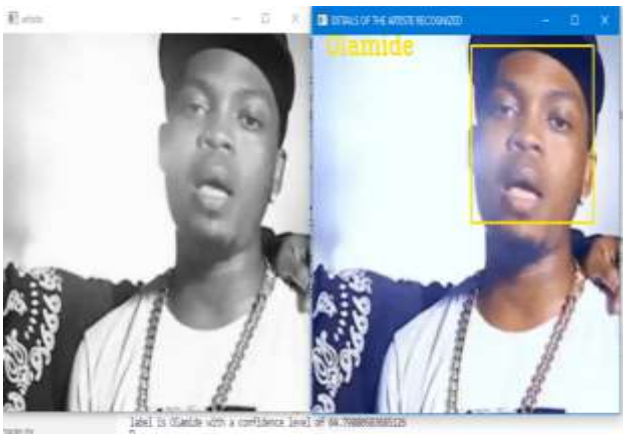
3.6.1 Results Obtained from the Face Recognition Experiments

Case One

Case Two



Case Three



Case Four

Case five

Case Six

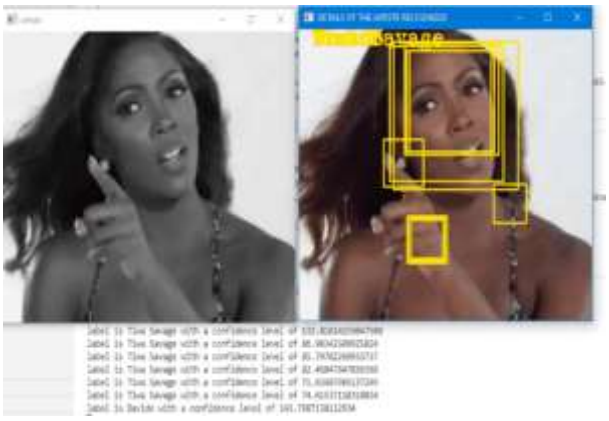


Figure 3.16: Face Recognition Test Results.

Table 3.2: Results and Observations from the Face Recognition Test

Case	Observation
Case One	The image used in case one of the face recognition experiments is also an existing image file in the training folder, case one was only a validation case and the result showed a confidence level of 0.0, which implied that the image is a perfect match of an existing image in the training database. Even after validating other seen images, the results were all perfectly matched.
Case Two	In this case, the facial recognizer developed was able to recognize the artiste "***Davido***" when an unseen image of this artiste was tested on the system. It correctly recognized the artiste with a confidence level of 57.06%.
Case Three	Artiste ***Olamide*** was correctly recognized when an unseen image of him was tested on the face recognizer. The prediction was accurate enough with a confidence level was 64.70%.
Case Four	The confidence level produced in this case was 115.64. Although the system predicted correctly that the unseen image brought to the recognizer for testing was ***Tiwa Savage***. While this was true, the weird confidence level value produced is one of the discrepancies associated with the use of Haar Cascade in detecting faces.
Case Five	In case Five, an unseen image of Wizkid was also correctly recognized by the system. The prediction was so accurate with a confidence level of 85.16%.
Case Six	While Haar Cascade could correctly predict results in some cases does not mean it is free from inaccuracies. In case 6, the recognizer was classifying Tiwa Savage as two different artistes i.e. Davido with a confidence level of 143.79% and Tiwa Savage with a confidence level between 70% - 85%. While we could consider this result as a weird result it is preferable to consider the artiste with a confidence level in the range below 100% as the correct prediction.

DISCUSSION

In the experimental processes, it is very important to tune hyperparameters manually and accordingly to get the best prediction results. Setting the minNeighbors parameter appropriately is one of the very crucial parts of the detectMultiScale() function used in the Haar Cascade algorithm. Hence, the Haar Cascade algorithm doesn't just work automatically, it might sometimes work with automatic settings for simple and flexible images but for complex images, detection or recognition requires a lot of manual tuning of hyperparameters. As good as Haar Cascade is, it may not really be an option to use in all cases, although it works well in some cases. The merits of the Haar Cascade Classifier are not only limited to its ease of use alone. Its speed and reliability when it comes to detecting and recognizing faces from a large dataset are the reasons behind its popularity. It still predicts well on faces with glasses. While its merits are important, its demerits must not be ignored either. It requires manual tuning of parameters, false-positive detection is higher, and it is also less accurate compared to deep-learning face detectors.

CONCLUSION

Haar Cascade is one of the oldest and easy to use face detection algorithms with its capacity expanded to detect different facial features. In this paper, the working, merits, and demerits of different face detection/recognition systems have been discussed. The Haar Cascade Classifier particularly has been implemented in this paper to evaluate its performance on different image files and datasets. Algorithms developed in this paper to confirm how well Haar Cascade can perform yielded positive results. While this paper can be commended for its analysis, it is important to appreciate accuracies at all levels, though the high false positive detection associated with Haar Cascade is now the reason behind the development of more advanced object detection algorithms such as Deep Learning object detectors, YOLO, SSD, etc.

REFERENCES

[1] Turk, Matthew A; Pentland, Alex P (1991). Face recognition using eigenfaces (PDF). Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–591. doi:10.1109/cvpr.1991.139758. ISBN 0-8186-2148-6.

[2] Yambor, Wendy S.; Draper, Bruce A.; Beveridge, J. Ross (2002). "Analyzing PCA-based Face Recognition Algorithms: Eigenvector Selection and Distance Measures" (PDF). Empirical Evaluation Methods in Computer Vision. Series in Machine Perception and Artificial Intelligence. Vol. 50. WORLD SCIENTIFIC. pp. 39–60. doi:10.1142/9789812777423_0003. ISBN 978-981-02-4953-3. ISSN 1793-0839.

[3] Mustamin Anggo and La Arapu. “Face Recognition Using Fisherface Method” Published under license by IOP Publishing Ltd. Journal of Physics: Conference Series, Volume 1028, 2nd International Conference on Statistics, Mathematics, Teaching, and Research 2017 9–10 October 2017, Makassar, Indonesia. Citation Mustamin Anggo and La Arapu 2018 J. Phys.: Conf. Ser. 1028 012119. DOI 10.1088/1742-6596/1028/1/012119.

[4] Thakur, S., Sing, J.K., Basu, D.K., Nasipuri, M. (2009). Face Recognition Using Fisher Linear Discriminant Analysis and Support Vector Machine. In: et al. Contemporary Computing. IC3 2009. Communications in Computer and Information Science, vol 40. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-03547-0_30.

[5] Lowe, David G. (1999). "Object recognition from local scale-invariant features" (PDF). Proceedings of the International Conference on Computer Vision. Vol. 2. pp. 1150–1157. doi:10.1109/ICCV.1999.790410.

[6] Lowe, David G. (2004). "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision. 60 (2): 91–110. CiteSeerX 10.1.1.73.2924. doi:10.1023/B:VISI.0000029664.99615.94. S2CID 221242327.

[7] "A HOG-LBP Human Detector with Partial Occlusion Handling", Xiaoyu Wang, Tony X. Han, Shuicheng Yan, ICCV 2009

[8] Sánchez López, L. Local Binary Patterns Applied to Face Detection and Recognition. 2010. Available online: https://upcommons. upc.edu/handle/2099.1/10772 (accessed on 20 April 2021).

[9] Bhatia, Richa (September 10, 2018). "What is the region of interest pooling?". *Analytics India*. Retrieved March 12, 2020.

[10] A. M. Romero and M. Cazorla, "Comparativa de detectores de características visuales y su aplicación al SLAM ", X Workshop de agentes físicos, Setiembre 2009, Cáceres

[11] J an Knopp, Mukta Prasad, Gert Willems, Radu Timofte, and Luc Van Gool, "Hough Transform and 3D SURF for Robust Three Dimensional Classification", European Conference on Computer Vision (ECCV), 2010

[12] K. Kadir, M. K. Kamaruddin, H. Nasir, S. I. Safie and Z. A. K. Bakti, "A comparative study between LBP and Haar-like features for Face Detection using Opencv," 2014 4th International Conference on Engineering Technology and Technopreneuship (ICE2T), Kuala Lumpur, 2014, pp.

[13] S. Guennouni, A. Ahaitouf, and A. Mansouri, "Face detection: Comparing Haar-like combined with cascade classifiers and Edge Orientation Matching," *2017 International Conference on Wireless Technologies, Embedded and Intelligent Systems (WITS)*, Fez, Morocco, 2017, pp. 1-4, doi: 10.1109/WITS.2017.7934604.