Face Recognition for Student Attendance using Haar Cascades Algorithm

[1]Ibrahim Mohamed Ahmed Ali, [2]Abdel-Badeeh M. Salem

[1]Computer Science Department
Faculty of Computer and Information Sciences
Karary University, Khartoum, Sudan

[2]Computer Science Department
Faculty of Computer and Information Sciences
Ain Shams University, Cairo, Egypt

ibrahim1630@gmail.com, abmsalem@yahoo.com

Abstract

Face recognition is identification and authentication means, it is successfully employed in many tasks, this paper aims to provide a reliable and effective system to verify the identity of the students and then allow them to take attendance. The system provides service monthly reports that contain warning and deprivation lists, which will send to students via e-mail .We used Haar Cascades techniques that have been used in image recognition and image compression field, we reached attendance system using fingerprint face.

As a results the system detect and recognize face and ensure that the process of attendance is safe, as well as providing the time and effort of the teacher by exemption from the process of taking attendance manually, furthermore makes the student following up Attendance Record by himself, we achieved 100% face detection rate on the nineteen sample of students .in addition to attendance list for all subjects.

Keyword: *Face recognition, Haar Cascades, OpenCV, Python, Biometrics.*

1. Introduction

Biometric recognition systems are inherently probabilistic, and their performance needs to be assessed within the context of this fundamental and critical characteristic. Biometric recognition involves matching, within a tolerance of approximation, of observed biometric traits against previously collected data for a subject. Approximate matching is required due to the variations in biological attributes and behaviors both within and between persons. Consequently, in contrast to the largely binary results associated with most information technology systems, biometric systems provide probabilistic results.

There are numerous sources of uncertainty and variation in biometric systems, including the following:

a) Variation within persons.
b) Sensors.
c) Feature extraction and matching algorithms.
d) Data integrity [1].

Face recognition is one of the most challenging areas in the field of computer vision. Face detection is the first step for face recognition in order to localize and to extract the face region from the background. For face detection, active contour models referred as snakes, are being used to detect the edges and for locating the face boundary [2].

Modern face recognition has reached an identification rate of greater than 90% for larger databases with well-controlled pose and illumination conditions. While this is a high rate for face recognition, it is by no means comparable to methods using keys, batches or passwords, nor can it bear direct comparison with the recognition abilities of a human concierge [2].

Furthermore, the human face is not a unique, rigid object. Indeed, there are numerous factors that cause the appearance of the face to vary. The sources of variation in the facial appearance can be categorized into two groups: intrinsic factors and extrinsic ones for face recognition [4]. Facial feature extraction algorithm is widely used. The distinguishing features found by the algorithm are used to compare images. There exist several algorithms to extract features.

Automatic recognition is a vast and modern research area of computer vision, reaching from face detection, face localization, face tracking, extraction of face orientation and facial features and facial expressions. These will need to tackle some technical problems like illumination, poses and occlusions [3].

The development of face recognition over the past years allows an organization into three types of recognition algorithms, namely frontal, profile, and view-tolerant recognition, depending on both the kind of imagery (facial views) available, and according to the recognition algorithms. While frontal recognition certainly is the classical approach to tackle the problem at hand, view-tolerant algorithms usually treat it in a more sophisticated fashion by taking into consideration some of the underlying physics, geometry, and statistics [2].

The research problem is the time consuming and effort of taking attendance manually, furthermore it is difficult for students to follow up Attendance Record by themselves.

1. 1 Haar Cascades

Haar classification is a tree-based technique where in the training phase, a statistical boosted rejection cascade is created. Boosted means that one strong classifier is created from weak classifiers, and a weak classifier is one that correctly gets the classification right in at least above fifty percent of the cases. This buildup to a better classifier from many weak is done by increasing the weight (penalty) on misclassified samples so that in the next iteration of training a hypothesis that gets those falsely classified samples right is selected [5].

A Haar-like feature considers neighboring rectangular regions at a specific location in a detection window, sums up the pixel intensities in each region and calculates the difference between these sums. This difference is then used to categorize subsections of an image. OpenCv [3] provides haar cascade classifier Implementation with various trained classifier cascades [6].

The haar cascade approach has several advantages:

a) To handle the large databases haar cascade classifier is the best detector in terms of speed and reliability.
b) Even the image is affected by illumination, face detection results are more accurate using haar cascade classifier.
c) There is no restriction on wearing glasses.

Looking at the advantages of haar cascade classifier it is suitable to implement for face recognition system.

The implementation of face recognition technology includes the following stages:

1.1.1 Detection

When the system is attached to a video surveillance system, the recognition software searches the field of view of a video camera for faces. If there is a face in the view, it is detected within a fraction of a second. A multi-scale algorithm is used to search for faces in low resolution. The system switches to a high-resolution search only after a head-like shape is detected.

1.1.2 Alignment

Once a face is detected, the system determines the head's position, size and pose. A face needs to be turned at least 35 degrees toward the camera for the system to register it.

1.1.3 Normalization

The image of the head is scaled and rotated so that it can be registered and mapped into an appropriate size and pose. Normalization is performed regardless of the head's location and distance from the camera. Light does not impact the normalization process.

1.1.4 Representation

The system translates the facial data into a unique code. This coding process allows for easier comparison of the newly acquired facial data to stored facial data.

1.1.5 Matching

The newly acquired facial data is compared to the stored data and (ideally) linked to at least one stored facial representation.

1.2 Open CV

Open Source Computer Vision (OpenCV) was started at Intel in 1999 by Gary Bradsky, and was first released in 2000. OpenCV supports a wide variety of programming languages such as C++, Python, Java, etc., and is available on different platforms including Windows, Linux, Android, and ions [4].

OpenCV is a popular computer vision library. The cross-platform library sets its focus on real-time image processing and includes patent-free implementations of the latest computer vision algorithms. In 2008 Willow Garage took over support and OpenCV 2.3. OpenCV is released under a BSD license, so it is used in academic and commercial projects such as Google Streetview [7].

Image processing in OpenCV includes the following steps [4].

1.2.1 Changing Color spaces.

There are more than 150 color-space conversion methods available in OpenCV. The most widely used are BGR ←→ Gray and BGR ←→HSV. The function used is cv2.cvtColor (), cv2.inRange ()

1.2.2 Geometric Transformations of Images.

There exist different geometric transformations to images like scaling, translation, rotation, affine transformation, resizing, etc. These functions are available in cv2.getPerspective Transform.

1.2.3 Image Thresholding.

In Image thresholding, if pixel value is greater than a certain threshold value, it is then assigned a value (which may be white), or else it is assigned another value (which may be black). The function used is cv2.threshold.

1.2.4 Smoothing Images.

Image smoothing is achieved by convolving the image through a low-pass filter kernel which removes the noise. It actually removes high frequency content (e.g.: noise, edges) from the image therefore the edges are blurred when this is filter is applied.

1.2.5 Morphological Transformations.

Morphological transformations normally performed on binary images. It needs two inputs, one is the original image, and the second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators for transformations are Erosion and Dilation.

1.2.6 Template Matching.

Template Matching is a method used to search and find the location of a template image in a larger image. OpenCV comes with a function cv2.matchTemplate () for this purpose. It slides the template over the input image and compares the template and patch of input image under the template image.

1.3 Python.

Python is a powerful modern computer programming language. Python allows you to use variables without declaring them (i.e., it determines types implicitly),and it relies on indentation as a control structure [8].Python is a good choice for mathematical calculations, since we can write code quickly, test it easily, and its syntax is similar to the way mathematical ideas are expressed in the mathematical literature. By learning Python you will also be learning a major tool used by many web developers. Python is especially important to learn because it is used very widely as a scripting Language [9].

The remaining paper is setup as follows: Section 2 describes our proposed method and section 3 describes experimental results. Finally, conclusion and future work is discussed in last section.

2. The Proposed System Methodology

The study was carried out using hybrid of qualitative research methodology at the karary computer and information collage in Khartoum Sudan. The main objective of the study was to develop an attendance system for the student. Actually, the specific phases used in developing the proposed system included: (1) Problem and need identification; (2) Requirement and system analysis; (3) Formalization or Modeling; (4) Design or conceptualization; (5) Implementation; (6) Testing and Maintenance.

2.1 Problem and Need Identification

The main objective of this phase was to identify, characterize, and define the problems the system will be expected to solve. The main problems identified include: difficult in ensuring the safety of attendance, a lot of time and effort by the teachers from the process of taking attend manually, and difficult to follow up Attendance Record by the student.

2.2 Requirements Analysis.

This phase involved getting to know and understand what the users needed the system to do for them and also stipulate what the system needed to function. Mainly, for user requirements, the system should automate the attendance to provide the students following up Attendance Record by themselves as well as providing the time and effort of the teacher by exemption from the process of taking attend manually. In the other hand, the system requirements focused on hardware, software, and human (end user) skills to get the fastest, most reliable and upgradeable computer system. This was categorized as: Hardware requirements (a computer); Software requirements (Eclipse, Python, OpenCV).

2.3 Formalization.

The system was modeled in two forms to facilitate understanding of how it will operate and how it arrives at its conclusion.

2.3.1. Enrollment mode.

Process of registering the biometric characteristics of individuals, to establish the "ground truth". Figure 1 shows the enrollment mode.

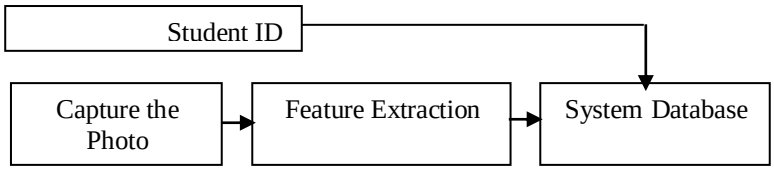


Figure 1. Enrollment mode

2.3.2. Verification mode

The presented face is compared with a single enrolled "template". With the presented token, the biometric template (or samples) associated with the user will be retrieved. The extracted feature from the input sample will be compared with the template from the database. The output of the comparison is an accept/reject decision figure 2 describes this mode.

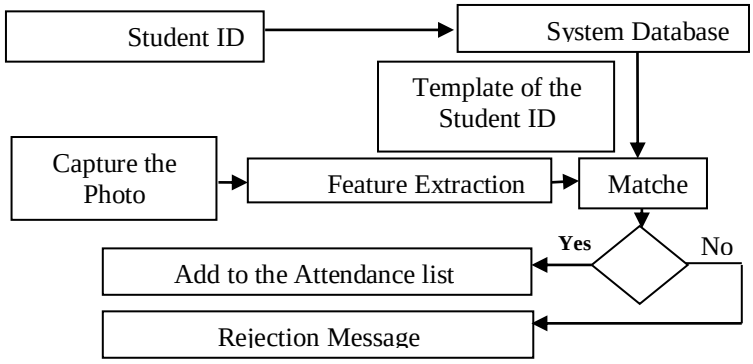


Figure 2. Verification mode

2.4 Implementation

This phase involves the actual coding of the system. Before writing the code we followed the installation steps.

a) Install eclipse then install PyDev plugin to be able to work with python.
b) Install python 2.7 and make sure pip is installed.
c) Install opencv 2.4 and link it to python.
d) Install all missing libraries in python using pip

figure 3 shows the installation steps.

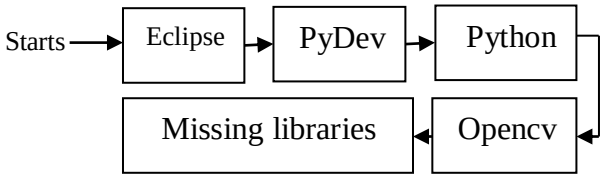
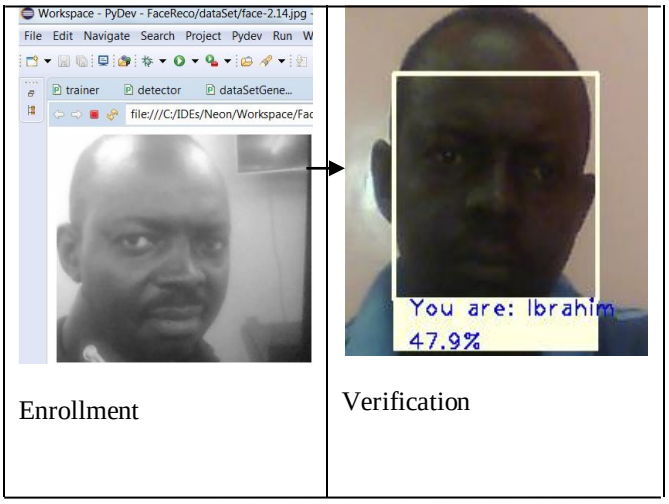


Figure 3. Installation steps

The system consists of three main components detection, training and matching figure 4 shows sample of image in enrollment mode and the same image in verification mode. As well as figure 5 shows screen shot of eclipse workspace linked with PyDev an listing the classifier, dataset, trainer and python libarary.



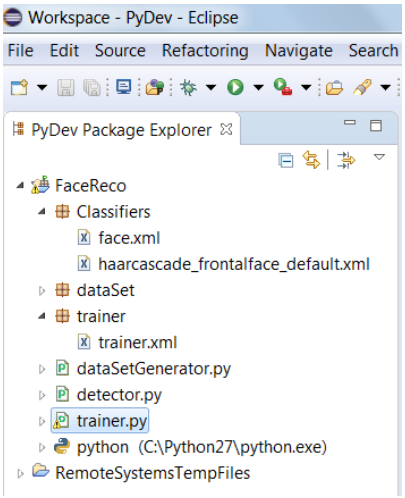


Figure 4. Enrollment and Verification

Figure 5. Eclipse workspace

The codes were developed and customized in python. It runs on a Windows 7 platform, running eclipse neon editor linked with opencv library.

3. Result

The system connect all gathered information and performs inferences through its knowledge process to output a verification of the student. In the detection face we had enrollment 50 image of student in the database, any image saved in 20 intra class variation, the total image in the database is 1000.

After training the data set the system gave 100% of recognition images and a live faces figure 6.

The verification describes the name and the ratio. Furthermore, the system evaluate and send reports to the students by e-mail, students can follow up attendance record by themselves.

Figure 6. Intra class variation in verification mode

A sample of the prolog coding of the system rules can be seen in Appendix A and B.

4. Conclusion and future work

Face Recognition has been playing a vital role in the active research area, especially in computer vision research In this paper we provided a reliable and effective system to verify the identity of the students and then allow them to take attendance .From the experimental point of view it is clear that haar cascade classifier has shown excellent performance for the images which contain Intra class variation samples. In most face recognition systems photos can be used to recognize faces which is a problem, to enhancement this work we need to develop algorithms that recognize if this is a live face or a picture.

References

[1]. Oseph N. Pato and Lynette I. Millett, "Biometric recognition challenges and opportunity", National Academy of Sciences, Washington, 2010.

[2]. https://www.cse.iitk.ac.in/users/biometrics/ "Face Recognition System", 15 dec 2018.

[3]. Sushma J., Sarita S. and Rakesh S. Jadon,"COMPARISON BETWEEN FACE RECOGNITION ALGORITHM-EIGENFACES, FISHERFACES AND ELASTIC BUNCH GRAPH MATCHING, Journal of Global Research in Computer Science, Volume 2, No. 7, July 2011.

[4]. Rabia Jafri and Hamid R. Arabnia, "A Survey of Face Recognition Techniques", Journal of Information Processing Systems, Vol.5, No.2, June 2009.

[5]. Staffan Reinius," Object recognition using the OpenCV Haar cascade-classifier on the iOS platform", Uppsala university, Department of Information Technology, 2013.

[6]. Vandna S., Dr. Vinod S. and Bhupendra S. , "FACE DETECTION BY HAAR CASCADE CLASSIFIER WITH SIMPLE AND COMPLEX BACKGROUNDS IMAGES USING OPENCV IMPLEMENTATION", International Journal of Advanced Technology in Engineering and Science, Volume No.01, Issue No. 12, December 2013.

[7]. Philipp Wagner," Face Recognition with OpenCV2", April 9, 2012.

[8]. S.R. Doty," Python Basics", 2008.

[9]. http://cs.brown.edu/courses/cs016/static/files/lectures/slides/pythonIntro," Introduction to Python – 2018", 21 des 2018.

Appendix A: Sample python Code of detector

```
import cv2 , os
import numpy as np
from PIL import Image

recognizer = cv2.createLBPHFaceRecognizer()
recognizer.load('trainer/trainer.xml')
cascadePath = "Classifiers/haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

cam = cv2.VideoCapture(0)
font = cv2.cv.InitFont(1, 1, 1, 0, 1, 1) #Creates a font
offset =10
while True:
    ret, im =cam.read()
    if ret:
        gray=cv2.cvtColor(im,cv2.COLOR_RGB2GRAY)
        faces=faceCascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=5,
minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE)
        if len(faces)==0:
            cv2.imshow('im',im)
            cv2.waitKey(10)
        else:
            for(x,y,w,h) in faces:
                nbr_predicted, conf = recognizer.predict(gray[y:y+h,x:x+w])

                if(nbr_predicted==1 or nbr_predicted==3 or nbr_predicted==5):
nbr_predicted='Abubaker'
                elif(nbr_predicted==2 or nbr_predicted==4):
nbr_predicted='Ibrahim'

                cv2.rectangle(im,(x-offset,y-
offset),(x+w+offset,y+h+offset),(225,255,255),2)
                cv2.rectangle(im,(x-
offset,y+h+offset),(x+w+offset,y+h+offset+30),(225,255,255),-1)

                cv2.cv.PutText(cv2.cv.fromarray(im),str("You are:
"+nbr_predicted), (x,y+h+20),font, 255) #Draw the text
                cv2.cv.PutText(cv2.cv.fromarray(im),str(conf)[:-8]+"%",
(x,y+h+40),font, 255) #Draw the text
                cv2.imshow('im',im)

                cv2.waitKey(10)
```

Appendix B : Sample python Code of trainer

```
import cv2
import os
import numpy as np
from PIL import Image

recognizer = cv2.createLBPHFaceRecognizer()
cascadePath = "Classifiers/haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
path = 'dataSet'
```

```
def get_images_and_labels(path):
image_paths = [os.path.join(path, f) for f in os.listdir(path)]
# images will contains face images
images = []
# labels will contains the label that is assigned to the image
labels = []
for image_path in image_paths:
# Read the image and convert to grayscale
image_pil = Image.open(image_path).convert('L')
# Convert the image format into numpy array
image = np.array(image_pil, 'uint8')
# Get the label of the image
nbr = int(os.path.split(image_path)[1].split(".")[0].replace("face-", ""))
#nbr=int(''.join(str(ord(c)) for c in nbr))
print nbr
# Detect the face in the image
faces = faceCascade.detectMultiScale(image)
# If face is detected, append the face to images and the label to labels
for (x, y, w, h) in faces:
images.append(image[y: y + h, x: x + w])
labels.append(nbr)
cv2.imshow("Adding faces to traning set...", image[y: y + h, x: x + w])
cv2.waitKey(10)
# return the images list and labels list
return images, labels

images, labels = get_images_and_labels(path)
cv2.imshow('test',images[0])
cv2.waitKey(1)

recognizer.train(images, np.array(labels))
recognizer.save('trainer/trainer.xml')
cv2.destroyAllWindows()
```