Face Detection and Recognition Using OpenCV

Mrs. Madhuram.M[1], B. Prithvi Kumar[2], Lakshman Sridhar[3], Nishanth Prem[4], Venkatesh Prasad[5]

[1]*Assistant Professor, Department of Computer Science, SRM Institute of Science and Technology, Ramapuram, Chennai, India*
[2,3,4,5]*Department of Computer Science, SRM Institute of Science and Technology, Ramapuram, Chennai, India*

Abstract - *Face detection and recognition from an image or a video is a popular topic in biometrics research. Face recognition technology has widely attracted attention due to its enormous application value and market potential, such as real-time video surveillance system. It is widely acknowledged that the face recognition has played an important role in surveillance system as it doesn't need the object's co-operation. We design a real-time face recognition system based on IP camera and image set algorithm by way of OpenCV and Python programming development. The system includes three parts: Detection module, training module and recognition module.*

Key Words*: *Face detection, Face Recognition, OpenCV.

1. INTRODUCTION

Face detection and recognition is technology which is used to identify a person from a video or photo source. In the 1960s face recognition was introduced by Woodrow Wilson Bledsoe. Bledsoe developed a device that could classify photos of faces by hand using what's known as a RAND tablet, a device that people could use to input horizontal and vertical coordinates on a grid using a pen like stylus that emitted electromagnetic pulses. Ever since then the recognition system is being improved and optimized constantly, the technology becomes gradually mature and is more and more widely used in human daily life. It has been used increasingly for forensics by law enforcement and military professionals. In fact, facial recognition system was used to help confirm the identity of Osama bin Laden after he was killed in a U.S. raid. The face recognition system is also being increasingly used in the mobiles for device security. In this paper, we propose a face detection and recognition system using python along with OpenCV package. This system contains three modules which are detection, training and recognition. Basically, the detection module detects the face which gets into the field of vision of the camera and saves the face in the form of an image in JPG format. Then the training modules trains the system using Haar cascade algorithm which was proposed by Paul Viola and Michael Jones in their paper. This method consists of four steps-

1. Haar Feature Selection-

First step is to collect the Haar Features. A Haar feature considers adjacent 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.

2. Creating Integral Images-

Integral Images are used to make this process fast. Most of the calculated features are irrelevant.

3. Adaboost Training-

A concept called Adaboost which both selects the best features and trains the classifiers is used. This algorithm constructs a strong classifier using a linear combination of weighted simple weak classifiers.

4. Cascading Classifiers-

The cascade classifier consists of a number of stages, where each stage is a group of weak learners. These weak learners are simple classifiers called decision stumps. Each stage is trained using a method called boosting. Boosting provides the ability to train a highly accurate classifier by taking the weighted average of decisions made by the weak learners.

Finally, in the recognition module the principal components of the face from the new video are extracted. Then those features get compared with the list of elements stored during training and the ones with the best match is found and name of the person recognized is displayed. This monitoring system fulfills the basic needs of face detection and recognition system, also takes the cost into consideration to ensure the pervasive mode as economical as possible. Furthermore, it can also be combined with real-time analysis algorithms.

2. EXISTING SYSTEMS

A. Linear Discriminate Analysis

LDA is a method to find a linear combination of features which characterize or separate two or more classes of objects or events. Linear classifier can be obtained from the resultant. Large number of pixels are used to represent face in computerized face recognition. Before classification Linear discriminant analysis is used to reduce features and makes it more manageable. New dimensions are a linear combination of pixel values which forms a template.

B. Principal Component Analysis

PCA involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables Please purchase PDF Split-Merge on www.verypdf.com to remove this watermark. 5

called principal components. The variability in the data is accounted by the first principal components and the succeeding components accounts for further variability. For exploratory data analysis and for making predictive models PCA is the most used tool. The calculation of the eigen value decomposition of a data covariance matrix or singular value decomposition of a data matrix is done with the help of PCA. Eigenvector-based multivariate analysis is made easy with the help of PCA. The variance present in the data is best explained by revealing the internal structure of the data which is considered to be one of the important operations. If a multivariate dataset is visualized as a set of coordinates in a high-dimensional data space (one axis per variable, a lower-dimensional picture is supplied by PCA, a "shadow" of this object is visible when viewed from its (in some sense) most informative viewpoint.

C. Hidden Markov Model

A *hidden Markov model (HMM)* is a statistical model that can be used to describe the evolution of observable events that depend on internal factors, which are not directly observable. The observed event is called as a `symbol' and the factor underlying the observation is a `state'.. Hidden Markov models are especially known for their applications in temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, partial discharges and bioinformatics.

3. PROPOSED SYSTEM AND TECHNIQUE

The quality of image depends on plethora of factors that influence the system's accuracy. It is important to apply various pre-processing techniques to standardize the images that you supply to a face recognition system. Face recognition algorithm usually find it difficult to recognize a face under extreme light sensitive conditions. If the system was trained to recognize a person when they are in a dark room, then it is highly possible that it won't recognize them in a bright room. This problem is referred to as "lumination dependent". There are many other issues, such as the face should also be in a very consistent position within the images like the eyes being in the same pixel coordinates, consistent size, rotation angle, hair and makeup, emotion like smiling, angry, etc. Hence it is important to use a good image preprocessing filter. For simplicity, the face recognition system presented in this paper is Eigenfaces using grayscale images. This paper shows us that it is easy to convert color images to grayscale (also called 'grayscale') and then to apply Histogram Equalization. It is a very simple method of automatically standardizing the brightness and contrast of your facial images. For better results, apply more processing stages such as edge enhancement, contour detection, motion detection, etc. OpenCV uses a face detector algorithm called a Haar Cascade classifier. An image, can come from a file or from live video, the face detector examines each image location and classifies it as "Face" or "Not Face." Classification assumes a fixed scale for the face. Faces in an image can be smaller or larger, the classifier runs over the image several times, to search for faces across a range of images. The classification is fast, even when it's applied at several scales. The code for the recognition system is given below.

4. SAMPLE CODE

A. Training Set

```
def assure_path_exists(path):
 dir = os.path.dirname(path)
 if not os.path.exists(dir):
 os.makedirs(dir)
recognizer = cv2.face.LBPHFaceRecognizer_create()
Detector=cv2.CascadeClassifier("haarcascade_frontalface_default.xml");
def getImagesAndLabels(path):
 imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
 faceSamples=[]
 ids = []
for imagePath in imagePaths:
 PIL_img = Image.open(imagePath).convert('L')
 img_numpy = np.array(PIL_img,'uint8')
 id = int(os.path.split(imagePath)[-1].split(".")[1])
 faces = detector.detectMultiScale(img_numpy)
 for (x,y,w,h) in faces:
 faceSamples.append(img_numpy[y:y+h,x:x+w])
 ids.append(id)
 return faceSamples,ids
faces,ids = getImagesAndLabels('dataset')
recognizer.train(faces, np.array(ids))
assure_path_exists('trainer/')
recognizer.save('trainer/trainer.yml')
```

B. Face Datasets

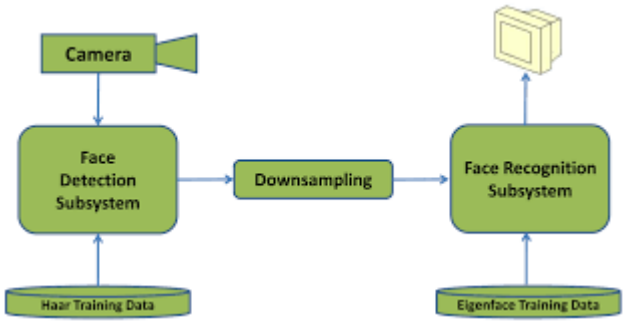
```
def assure_path_exists(path):
 dir = os.path.dirname(path)
if not os.path.exists(dir):
 os.makedirs(dir)
vid_cam = cv2.VideoCapture(0)
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
face_id = 1
count = 0
```

```
assure_path_exists("dataset/")
while(True):
 _, image_frame = vid_cam.read()
 gray = cv2.cvtColor(image_frame, cv2.COLOR_BGR2GRAY)
 faces = face_detector.detectMultiScale(gray, 1.3, 5)
 for (x,y,w,h) in faces:
 cv2.rectangle(image_frame, (x,y), (x+w,y+h), (255,0,0), 2)
 count += 1
 cv2.imwrite("dataset/User." + str(face_id) +  str(count) + ".jpg", gray[y:y+h,x:x+w])
 cv2.imshow('frame', image_frame)
 if cv2.waitKey(100) & 0xFF == ord('q'):
 break
elif count>100:
break
vid_cam.release()
cv2.destroyAllWindows()
```

C. Face Detection

```
def assure_path_exists(path):
 dir = os.path.dirname(path)
 if not os.path.exists(dir):
 os.makedirs(dir)
recognizer = cv2.face.LBPHFaceRecognizer_create()
assure_path_exists("trainer/")
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);
font = cv2.FONT_HERSHEY_SIMPLEX
cam = cv2.VideoCapture(0)
while True:
 ret, im =cam.read()
gray = cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
 faces = faceCascade.detectMultiScale(gray, 1.2,5)
 for(x,y,w,h) in faces:
 cv2.rectangle(im, (x-20,y-20), (x+w+20,y+h+20), (0,255,0), 4)
 Id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
if(Id == 1):
 Id = "ARJUN {0:.2f}%".format(round(100 - confidence, 2))
 cv2.rectangle(im, (x-22,y-90), (x+w+22, y-22), (0,255,0), -1)
 cv2.putText(im, str(Id), (x,y-40), font, 1, (255,255,255), 3)
 cv2.imshow('im',im)
 if cv2.waitKey(10) & 0xFF == ord('q'):
break
cam.release()
cv2.destroyAllWindows()
```

5. SYSTEM ARCHITECTURE



6. CONCLUSIONS

In this paper we have developed a system for face detection and recognition using opencv. It is used to detect and recognize human faces. The images of the persons are the datasets which are defined and trained before recognizing.

Haar cascade algorithm is used for detection.

For better face recognition and detection small features can be improved. In the coming future, as technology advances, more advance features will be added to the system.

REFERENCES

[1] https://www.youtube.com/watch?list=PLQVvvaa0QuDfKTOs3Keq_kaG2P55YRn5v&v=OGxgnH8y2NM

[2] https://opencv.org/

[3] https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html

[4] https://pythonprogramming.net/haar-cascade-object-detection-python-opencv-tutorial/

[5] .Open Source Computer Vision Library Reference Manual-intel [Media]

[6] Tej Pal Singh, “Face Recognition by using Feed Forward Back Propagation Neural Network”, International Journal of Innovative Research in Technology & Science, vol.1, no.1

[7] N.Revathy, T.Guhan, "Face recognition system using back propagation artificial neural networks", International Journal of Advanced Engineering Technology, vol.3, no. 1, 2012.

[8] M.A.Turk and A.P. Pentaland, "Face Recognition Using Eigenfaces", IEEE conf. on Computer Vision and Pattern Recognition, pp. 586-591, 1991