GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

# Image-based Face detection

A Report for the Final Evaluation

*Submitted by*

**Ashutosh Mishra**
**( 1613107017 / 16SCSE107036 )**

***in partial fulfillment for the award of the degree***

**of**

**Bachelor of Technology**

**In**

**Computer Science and Engineering With Specialization of Cloud Computing and Virtualization**

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING**

*Under the Supervision of*

**Mr. Shubham Kumar**

**APRIL / MAY - 2020**

TABLE OF CONTENTS

## Abstract

Image recognition from image or image may be a popular topic in biometrics research. Many public places have surveillance cameras for image capture and these cameras have their significant value for security properties. it's widely acknowledged that the image detection/recognition have played a important role in closed-circuit television due to it's need the object's cooperation. the particular pros of image based identification over other biometrics are uniqueness and acceptable. As image can be a dynamic object having high degree of variability in its appearance, that creates image detection/recognition a difficult problem in computer vision. during this field, accuracy and speed of identification may be a main issue.

The goal of this research paper is to process various image detection/recognition and recognition methods, provide complete talks about and algorithm for image based image detection/recognition and recognition with higher accuracy, better response rate as an initial step for image surveillance. talks about and algorithm is supported performed tests on various image rich databases in terms of subjects, pose, emotions, race and lightweight.

## Introduction :

The goal of this article is to provide an easier human-machine interaction routine when user authentication is needed through face detection and recognition. With the aid of a regular web camera, a machine is able to detect and recognize a person's face; a custom login screen with the ability to filter user access based on the users' facial features will be developed. The objectives of this thesis are to provide a set of detection algorithms that can be packaged in an easily portable framework among the different processor architectures we see in machines (computers) today. These algorithms must provide at least a 95% successful recognition rate, out of which less than 3% of the detected faces are false positive is processed to crop and extract the person's face for easier recognition. Face Recognition where that detected and processed face is compared to a database of known faces, to decide who that person is.

Since 2002, face detection can be performed fairly easily and reliably with Intel's open source framework called OpenCV . This framework has an inbuilt Face Detector that works in roughly 90-95% of clear photos of a person looking forward at the camera. However, detecting a person's face when that person is viewed from an angle is usually harder, sometimes requiring 3D Head Pose Estimation. Also, lack of proper brightness of an image can greatly increase the difficulty of detecting a face, or increased contrast in shadows on the face, or maybe the picture is blurry, or the person is wearing glasses, etc. Face recognition however is much less reliable than face detection, with an accuracy of 30-70% in general. Face recognition has been a strong field of research since the 1990s, but is still a far way away from a reliable method of user authentication. More and more techniques are being developed each year.

The Eigenface technique is considered the simplest method of accurate face recognition, but many other (much more complicated) methods or combinations of multiple methods are slightly more accurate. OpenCV was started at Intel in 1999 by Gary Bradski for the purposes of accelerating research in and commercial applications of computer vision in the world and, for Intel, creating a demand for ever more powerful computers by such applications. Vadim Pisarevsky joined Gary to manage Intel's Russian software OpenCV team. Over time the OpenCV team moved on to other companies and other Research. Several of the original team eventually ended up working in robotics and found their way to Willow Garage. In 2008, Willow Garage saw the need to rapidly advance robotic perception capabilities in an open way that leverages the entire research Eigenfaces is considered the simplest method of

accurate face recognition, but many other (much more complicated) methods or combinations of multiple methods are slightly more accurate.

Most resources on face recognition are for basic Neural Networks, which usually don't work as well as Eigenfaces does. And unfortunately there are only some basic explanations for better type of face recognition than Eigenfaces, such as recognition from video and other techniques at the Face Recognition Homepage or 3D Face Recognition Wikipedia page and Active Appearance Models page . But for other techniques, you should read some recent computer vision research papers from CVPR and other computer vision conferences. Most computer vision or machine vision conferences include new advances in face detection and face recognition that give slightly better accuracy. So for example you can look for the CVPR10 and CVPR09 conferences .

## Existed System

Over the past decade face detection and recognition have transcended from esoteric to popular areas of research in computer vision and one of the better and successful applications of image analysis and algorithm based understanding. Because of the intrinsic nature of the problem, computer vision is not only a computer science area of research, but also the object of neuroscientific and psychological studies also, mainly because of the general opinion that advances in computer image processing and understanding research will provide insights into how our brain work and vice-verse. A general statement of the face recognition problem (in computer vision) can be formulated as follows: given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces.

Facial recognition generally involves two stages: Face Detection where a photo is searched to find a face, then the image and commercial community and began actively supporting OpenCV, with Gary and Vadim once again leading the effort. Intel's open-source computer-vision library can greatly simplify computer vision programming. It includes advanced capabilities - face detection, face tracking, face recognition, Kalman filtering, and a variety of artificial intelligence (AI) methods - in ready-to use form. In addition, it provides many basic computer-vision algorithms via its lower-level APIs. OpenCV has the advantage of being a multi-platform framework; it supports both Windows and Linux, and more recently, Mac OS X. OpenCV has so many capabilities it can seem overwhelming at first.

A good understanding of how these methods work is the key to getting good results when using OpenCV. Fortunately, only a select few need to be known beforehand to get started. OpenCV's functionality that will be used for facial recognition is contained within several modules. Following is a short description of the key namespaces CXCORE namespace contains basic data type definitions, linear algebra and statistics methods, the persistence functions and the error handlers. Somewhat oddly, the graphics functions for drawing on images are located here as well. CV namespace contains image processing and camera calibration methods. The computational geometry functions are also located here. CVAUX namespace is described in OpenCV's documentation as containing obsolete and experimental code. However, the simplest interfaces for face recognition are in this module. The code behind them is specialized for face recognition, and they're widely used for that purpose.

ML namespace contains machine learning interfaces. HighGUI namespace contains the basic I/O interfaces and multi-platform windowing capabilities. CVCAM namespace contains interfaces for video access through DirectX on 32- bit Windows platforms.OpenCV uses a type of face detector called a Haar Cascade classifier. Given an image, which 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, say 50x50 pixels. Since faces in an image might be smaller or larger than this, the classifier runs over the image

several times, to search for faces across a range of scales. This may seem an enormous amount of processing, but thanks to algorithmic tricks, explained in the sidebar, classification is very fast, even when it's applied at several scales. The classifier uses data stored in an XML file to decide how to classify each image location. The OpenCV download includes four flavors of XML data for frontal face detection, and one for profile faces. It also includes three non-face XML files - one for full body (pedestrian) detection, one for upper body, and one for lower body.

## Proposed System

When image quality is taken into consideration, there is a plethora of factors that influence the system's accuracy. It is extremely important to apply various image pre-processing techniques to standardize the images that you supply to a face recognition system. Most face recognition algorithms are extremely sensitive to lighting conditions, so that if it was trained to recognize a person when they are in a dark room, it probably won't recognize them in a bright room, etc. This problem is referred to as "lumination dependent", and there are also many other issues, such as the face should also be in a very consistent position within the images (such as the eyes being in the same pixel coordinates), consistent size, rotation angle, hair and makeup, emotion (smiling, angry, etc), position of lights (to the left or above, etc). This is why it is so important to use a good image pre-processing filters before You'll need to tell the classifier where to find the data file you want it to use. The one I'll be using is called haarcascade_frontalface_default.xml. In OpenCV version 1.0, it's located at [OPENCV_ROOT]/data/haarcascades/haa rcascade_frontalface_default.xml where [OPENCV_ROOT] is the path to your OpenCV installation. For example, if you're on Windows XP, and you selected the default installation location, you'd use [OPENCV_ROOT] = "C:/Program Files/OpenCV" (If you're working with an older, 16-bit version of Windows, you'd use '\' as the directory separator, instead of '/'.) It's a good idea to locate the XML file you want to use, and make sure your path to it is correct, before you code the rest of your face-detection program. It is very easy to use a webcam stream as input to the face recognition system instead of a file list. Basically you just need to grab frames from a camera instead of from a file, and you run forever until the user wants to quit, instead of just running until the file list has run out. Grabbing frames from a webcam can be implemented easily using function

## Implementation

**Step 1: Using Cv2 for image capturing**

```
1)  import cv2, time
2)  img = cv2.VideoCapture(0)
3)  state , pix= img.read() #takes pixel value
4)  print(state) # prints true or false
5)  print(pix)
6)  time.sleep(5)
7)  cv2.imshow('my_image',pix) #to show image
8)  cv2.imwrite('Myimage.jpg',pix)
9)  cv2.waitKey(0) #used to close window here we have given 0 so if any key is pressed
    so it will close window
10) img.release() #stop capturing
11) cv2.destroyAllWindows()
12) import cv2, time
13) img = cv2.VideoCapture(0)
14) a=0 #use this to check total frame capture
15) while(True):
16) a=a+1
17) state,pix=img.read()#takes pixel value
    gray=cv2.cvtColor(pix,cv2.COLOR_BGR2GRAY)
18) cv2.imshow('my_image',pix) #to show image
19) key =cv2.waitKey(1) #used to close window here we have given 1 so defined key
    is pressed so it will close window
20) Tf key ==ord('a'):
21) break
22) print(a)
23) img.release() #stop capturing
24) cv2.destroyAllWindows()
```

**Step 2: Implementing the Face Detection algorithm**

```
25) import cv2, time
26) imagepath= 'F:/Code/Python/Cognizance Internship/Data sets/IMG_9468.jpg' algopath
    = 'F:/Code/Python/Cognizance Internship/Data
    sets/haarcascade_frontalface_default.xml' #create the object for haar-cascade
27) facecascade = cv2.CascadeClassifier(algopath)
28) image =cv2.imread(imagepath)
29) image = cv2.resize(image,(800,800)) faces = facecascade.detectMultiScale(image ,
    scaleFactor = 1.1 ,minNeighbors =5 ,minSize=(30,30),flags =
    cv2.CASCADE_SCALE_IMAGE)
30) print(len(faces))
31) cv2.imshow(",image)
32) cv2.waitKey(0)
33) cv2.destroyAllWindows()
34) print (faces)
35) for (x,y,w,h) in faces :
36) cv2.rectangle(image , (x,y) , (x+w , y+h) ,(0,255,0),2)
37) print(faces)
38) cv2.imshow(",image)
39) cv2.waitKey(0)
40) cv2.destroyAllWindows()
```

**Step 3: Face Recognition through web cam**

```
41) import cv2, time
42) img = cv2.VideoCapture(0)
43) state , pix= img.read() #takes pixel value
44) print(state) # prints true or false
45) time.sleep(3)
46) cv2.imshow('my_image',pix) #to show image
47) cv2.imwrite('Myimage.jpg',pix)
48) cv2.waitKey(0) #used to close window here we have given 0 so if any key is pressed
    so it will close window
49) img.release() #stop capturing
50) cv2.destroyAllWindows()
51) imagepath= 'F:/Code/Python/Cognizance Internship/Myimage.jpg'
52) algopath = 'F:/Code/Python/Cognizance Internship/Data
    sets/haarcascade_frontalface_default.xml' #create the object for haar-
    cascade facecascade = cv2.CascadeClassifier(algopath)
53) image =cv2.imread(imagepath)
54) image = cv2.resize(image,(800,800))
55) faces = facecascade.detectMultiScale(image , scaleFactor =
    1.1 ,minNeighbors =5 ,minSize=(30,30),flags =
    cv2.CASCADE_SCALE_IMAGE)
56) print(len(faces))
57) for (x,y,w,h) in faces :
58) cv2.rectangle(image , (x,y) , (x+w , y+h) ,(0,255,0),2)
59) print(faces)
60) cv2.imshow(",image)
61) cv2.waitKey(0)
62) cv2.destroyAllWindows()
```

## Result

In this currentwork we developed the system to process the image detection/recognition and recognition methods which are toked as to be a bench mark. Some methods performed consistently over different well arranged datasets whereas other methods behave very randomly however supported average experimental results performance is evaluated, five well arranged datasets been used for this purpose. image detection/recognition and recognition method's result summery is provided in table 1 and table 2 respectively whereas well arranged datasets summery is provided in table 3. In this currentsystem Haar-like [7] have may features reported relatively well but it's much false detection/recognition than Local binary patterns [8] which might be consider being a future add surveillance to scale back false detection/recognition in Haar-like [7] have may features and for the popularity part gabor [11] is reported well as it's qualities overcomes well arranged datasets complexity.

## References


[1] image Recognition Data, University of Essex, UK, image 94, http://cswww.essex.ac.uk/mv/all images/images94.html.

[2] image Recognition Data, University of Essex, UK, image 95, http://cswww.essex.ac.uk/mv/all images/images95.html.

[3] image Recognition Data, University of Essex, UK, image 96, http://cswww.essex.ac.uk/mv/all images/images96.html.

[4] image Recognition Data, University of Essex, UK, Grimace, http://cswww.essex.ac.uk/mv/all images/grimace.html.

[5] Psychological Image sets at Stirling (PICS), Pain Expressions, http://pics.psych.stir.ac.uk/2D_image_sets.htm.
[6] K. T. Talele, S. Kadam, A. Tikare, Efficient image detection/recognition using Adaboost, "IJCA Proc on International Conference in Computational Intelligence", 2012.
[7] T. Mita, T. Kaneko, O. Hori, Joint Haar-like have may features for image detection/recognition, "Proceedings of the Tenth IEEE International Conference on Computer Vision", 1550-5499/05 ©2005 IEEE.
[8] T. Ahonen, A. Hadid, M. Peitikainen, image recognition with local binary patterns. "In Proc. of European Conference of Computer Vision", 2004.
[9] M. A. Turk and A.P. Pentland, image recognition using eigenimages, "Proceedings of the IEEE", 586-591, 1991.
[10] J Lu, K. N. Plataniotis, A. N. Venetsanopoulos, image recognition using LDA-based algorithms, "IEEE Neural Networks Transaction", 2003.
[11] L. Wiskott, M. Fellous, N. Krger, and C. Malsburg, image recognition by elastic bunch graph matching, "IEEE Trans", on PAMI, 19:775–779, 1997.
[12] I. Kukenys, B. McCane, it it always supportsVector Machines for Human image detection/recognition, "Proceedings of the New Zealand Computer Science Research Student Conference", 2008.
[13] M. M. Abdelwahab, S. A. Aly, I. Yousry, Efficient Web-Based Facial Recognition System Employing 2DHistogram of Oriented Gradients, arXiv:1202.2449v1 [cs.CV].
[14] W. Zhao, R. chellappa, P. J. Phillips, image recognition: A literature survey, "ACM Computing Surveys (CSUR)", December 2003.
[15] G. L. Marcialis, F. Roli, Chapter: Fusion of image Recognition Algorithms for image-Based Surveillance Systems, Department of Electrical and Electronic Engineering-Univ- ersity of Cagliari- Italy.
[16] A. Suman, Automated image recognition: Applications within law enforcement. Market and technology review, "NPIA", 2006.