An Approach To Locate And Detect Face In A Given Image

M. Rama Bai[1], CH Ramanitha[2]
[1]Professor, Dept of Computer Science and Engineering
[2]Dept of Computer Science and Engineering
[1, 2] Mahatma Gandhi Instituteof Technology, Telangana, India.

Abstract- *Facial detection could be a powerful and customary use-case of Machine Learning. It is wont to automatize the manual tasks like school attendance and law enforcement. Within the other hand, it may be used for biometric authorization. Face detection and picture or video recognition may be a popular subject of research on the world of biometrics. Face recognition during a real-time setting has an exciting area and a rapidly growing challenge. Face detection may be a biometric identification technology that that supports on human facial feature information. This study uses the Webcam or camera to collect images with faces and automatically detect faces in the images, and then carry out a series of technical processing on the detected faces. The standard face detection technology is mainly based on visible images or pictures, which is also familiar with the detection method. The purpose of this study is to solve the classic problem of face detection under different lights, and to develop an intelligent and efficient human face detection method on OpenCV technology. The Main objective of this work is to develop a framework for face detection using Haar Feature-based Cascade Classifiers.*

Keywords- Face Detection,Haar-Cascade, LBPHs, OpenCV

I. INTRODUCTION

The face detection process is a necessary step because it detects and locates human faces in images and videos. Face detection is procedure of identifying Human faces with different sizes in an image. It's many applications like financial transactions, monitoring systems, mastercard verification, ATM access, notebook computer access, video surveillance etc. Since the face is adynamic object with a high degree of appearance variability, so face detection may be a challenging field in computer vision. A large type of methods and techniques are presented during this subject. A number of them are supported principal feature or component analysis, some propose template matching. Color analysis, Hough transform, and neural network are the opposite employed techniques to detect human faces within the input image.

A Face Detection framework is presented in, which might detect faces with high rate and fast. The framework has three contributions, integral image to fast computation of the features, AdaBoost learning algorithm to pick out a small number of critical visual features, and cascade classifiers to discard background of the image. A cascade architecture is proposed in, which is made on convolutional neural networks. The proposed cascade network operates at multiple resolutions, quickly rejects the background regions and evaluates a little number of face candidates. Article, presents a self-adaptive face detection algorithm supported coloring for images with complex background. First histogram color model is made and then skin colour segmentation is implemented employing histogram back projection.

II. PROBLEM DEFINITION

The face capture process transforms analogue information (a face) into a set of digital information (data) based on the person's facial features. The face detection process is an essential step as it detects and locates human faces in images and videos. A face detection and recognition program is a software application for verifying a person and identifying him or her with a video or picture from a source to improve Accuracy. Whenever we implement a new system it is developed to remove the shortcomings of the existing system. The computerized mechanism has the more edge than the manual system. The existing system use images and videos for detecting faces. The existing system is based on manual system which takes a lot of time to get performance of the work using pictures. The Proposed system uses Webcam for detecting faces using OpenCV, in which we try to achieve high accuracy and efficient detection. In the proposed system utmost care would be that no information is repeated anywhere, in storage or otherwise. This would assure economic use of storage space and consistency in the data stored.

III. DESIGN METHODOLOGY

A.Face Detection

Face detection is a type of application classified under “computer vision” technology. It is the process in which algorithms are developed and trained to properly locate faces or objects (in object detection, a related system), in images. These can be in real time from a video camera or from photographs. An example where this technology is used are in airport security systems. In order to recognize a face, the camera software must first detect it and identify the features before making an identification. Likewise, when Facebook makes tagging suggestions to identify people in photos it must first locate the face. On social media apps like Snapchat, face detection is required to augment reality which allows users to virtually wear dog face masks using fancy filters. Another use of face detection is in smartphone face ID security. Before the face detection, we pretreated the image, and the specific operation process was shown in Figure 1.

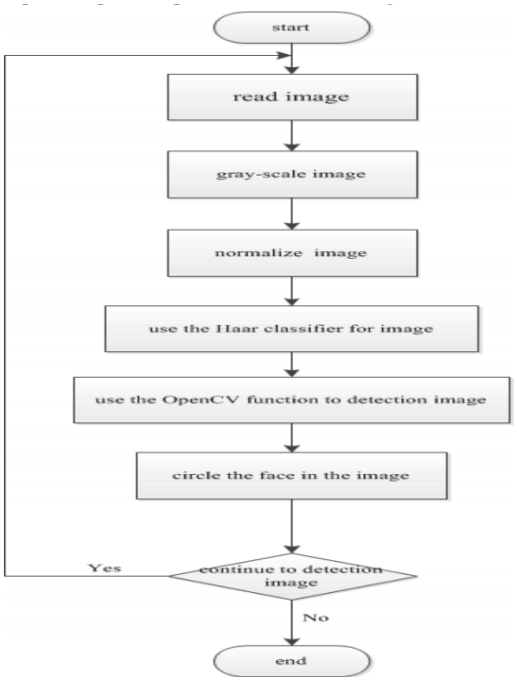

Figure 1: Block Diagram

B. Modules

OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. OpenCV provides a powerful infrastructure for computer vision applications.

The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. It has C++, Python, Java and MATLAB interfaces and supports Windows, Linux, Android and Mac OS.

OpenCV uses two types of classifiers, Haar cascades and LBP (Local Binary Pattern).

Haar Cascade Classifiers

A Haar wavelet is a mathematical fiction that produces square-shaped waves with a beginning and an end and used to create box shaped patterns to recognise signals with sudden transformations. An example is shown in Figure 2. By combining several wavelets, a cascade can be created that can identify edges, lines and circles with different colour intensities. These sets are used in Viola Jones face detection technique and since then more patterns are introduced for object detection as shown in Figure 2. To analyse an image using Haar cascades, a scale is selected smaller than the target image. It is then placed on the image, and the average of the values of pixels in each section is taken. If the difference between two values pass a given threshold, it is considered a match. Face detection on a human face is performed by matching a combination of different Haar-like-features. For example, forehead, eyebrows and eyes contrast as well as the nose with eyes as shown below in Figure 3 single classifier is not accurate enough. Several classifiers are combined as to provide an accurate face detection system as shown in the block diagram below in Figure 4.

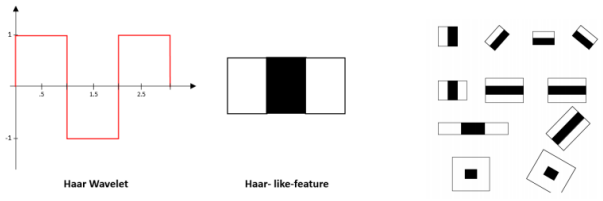

Figure 2: A Haar wavelet and Haar-like feature

A similar method is used effectively to by identifying faces and eyes in combination resulting better face detection. Similarly, in viola Jones method, several classifies were

combined to create stronger classifiers. ADA boost is a machine learning algorithm that tests out several week classifiers on a selected location and choose the most suitable.

Figure 3: Several Haar-like-features matched to the features of face

It can also reverse the direction of the classifier and get better results if necessary. Furthermore, Weight-update-steps can beupdated only on misses to get better performance. The cascade is scaled by 1.25 and re-iterated in order to find different sized faces. Running the cascade on an image using conventional loops takes a large amount of computing power and time.Viola Jones used a summed area table (an integral image) to compute the matches fast. First developed in 1984, it became popular after 2001 when Viola Jones implemented Haar-cascades for face detection.Using an integral image enables matching features with a single pass over the image.

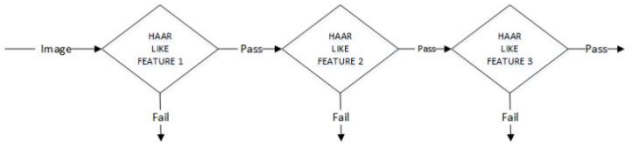

Figure 4: Haar-cascade flow chart

Local Binary Pattern Histogram (LBPH)

Local binary patterns were proposed as classifiers in computer vision and in 1990 By Li Wang. The combination of LBP with histogram oriented gradients was introduced in 2009 that increased its performance in certain datasets. For feature encoding, the image is divided into cells (4 x 4 pixels). Using a clockwise or counter-clockwise direction surrounding pixel values are compared with the central as shown in Figure 5. The value of intensity or luminosity of each neighbour is compared with the central as shown in Figure 5.

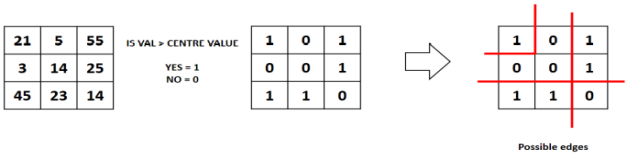

Figure 5: Local binary pattern histogram generating 8-bit number

Depending if the difference is higher or lower than 0, a 1 or a 0 is assigned to the location. The result provides an 8-bit value to the cell. The advantage of this technique is even if the luminosity of the image is changed as in Figure 6, the result is the same as before. Histograms are used in larger cells to find the frequency of occurrences of values making process faster. By analysing the results in the cell, edges can be detected as the values change. By computing the values of all cells and concatenating the histograms, feature vectors can be obtained. Images can be classified by processing with an ID attached. Input images are classified using the same process and compared with the dataset and distance is obtained. By setting up a threshold, it can be identified if it is a known or unknown face. Eigenface and Fisherface compute the dominant features of the whole training set while LBPH analyse them individually

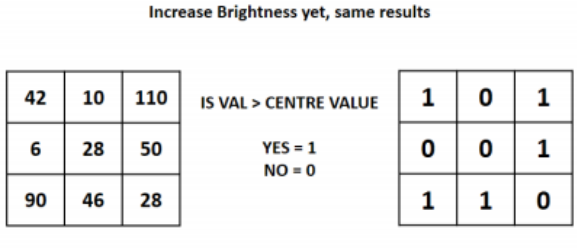

Figure 6:LBPH with brightness

C. Methodology

First stage was creating a face detection system using Haar-cascades. Although, training is required for creating new Haar-cascades, OpenCV has a robust set of Haar-cascades that was used for the project. Using face-cascades alone caused random objects to be identified and eye cascades were incorporated to obtain stable face detection. The flowchart of the detection system can be seen in Figure 7. Face Classifier objects are created using classifier class in OpenCV through the **cv2.CascadeClassifier()** and loading the respective XML files. A camera object is created using the **cv2.VideoCapture()** to capture images.

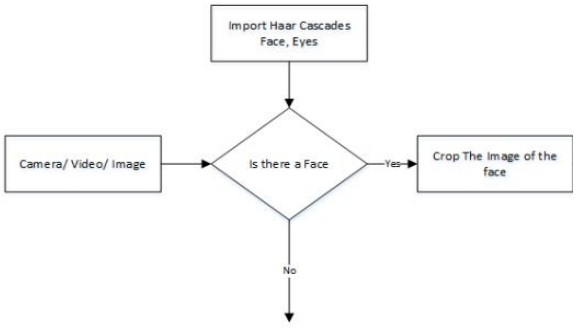

Figure 7: The Flow chart of the face detection application

By using the **CascadeClassifier.detectMultiScale()** object of various sizes are matched and location is returned. Using the location data, the face is cropped for further verification. Eye cascade is used to verify there are two eyes in the cropped face. If satisfied a marker is placed around the face to illustrate a face is detected in the location. We shall be using the **detectMultiscale()** module of the classifier. This function will return a rectangle with coordinates(x,y,w,h) around the detected face. This function contains important parameters which have to be tuned according to the data.

- **scaleFactor:** The value indicates how much the image size is reduced at each image scale. A lower value uses a smaller step for downscaling. This allows the algorithm to detect the face. It has a value of x.y, where x and y are arbitrary values you can set.
- **minNeighbors:** This parameter specifies how many "neighbors" each candidate rectangle should have. A higher value results in less detections but it detects higher quality in an image. You can use a value of X that specifies a finite number.
- **minSize:** The minimum object size. By default it is (30,30). The smaller the face in the image, it is best to adjust the minSize value lower.

IV. TRAINING AND VALIDATION

Testing is a very important module in the software development to verify, validate and provide quality and service for different components of software. It is used to minimize the risks by efficient use of resources in the development life cycle. This module can be employed at any point of the development process. It is efficient for the testing phase to be implemented at initial level to lower down the risks of defects and failures.

A. Test Cases

Test Case 0
The following Figure 8 shows the negative test cases of the model.

Figure 8: Negative Test Case

Test Case 1
The following Figure 9 shows the positive test cases of the model.

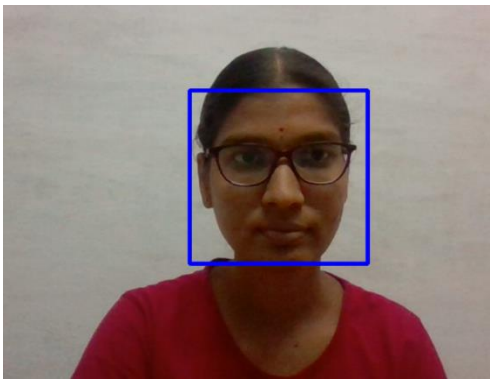

Figure 9: Positive Test Case

B. Results

Running this code shows an application shown in Figure 10 that can easily detect frontal faces by drawing blue colored bounding boxes around that specific area:

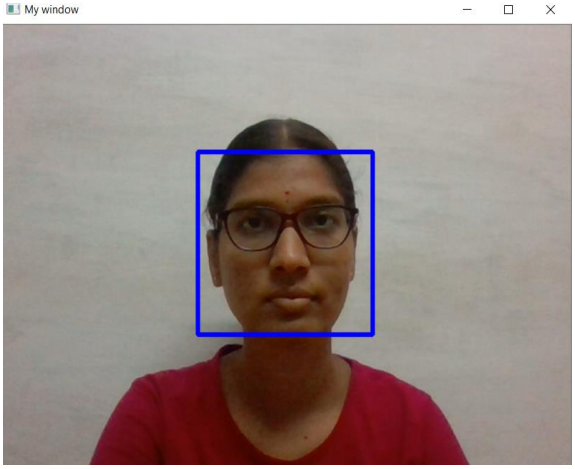

Figure 10: Face Detection

The face is detected by drawing the colored bounding boxes around the Faces and more faces can be detected with Webcam as shown in the Figure 11.

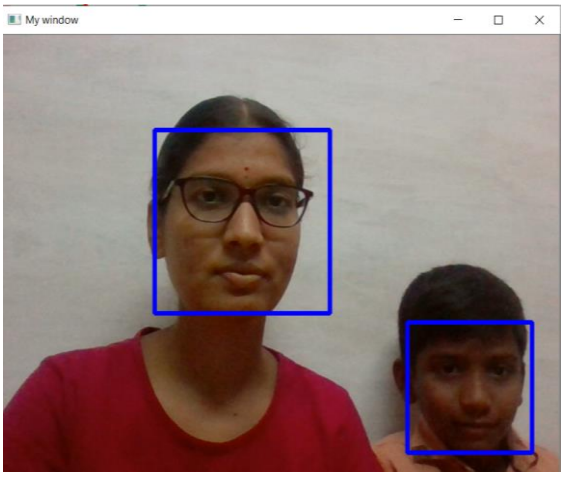

Figure 11: Multiple Faces Detected

The application is also accurate enough to detect minuscule faces on the cards and other photographs is in Figure 12:

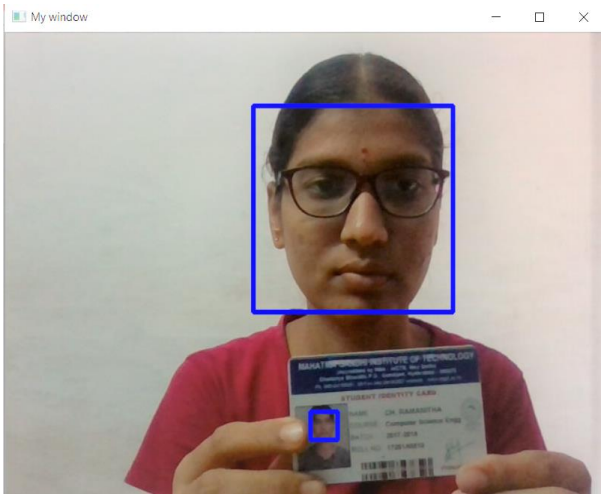

Figure 12: Detected face on Card

Most of the companies, or even in many conferences, you are supposed to carry an ID card in order to get entry. But what if we could figure out a way so that you don't need to carry any ID card to get access? Face Detection helps in making this process smooth and easy. By using this application the person just looks at the camera and it will automatically detect whether he/she should be allowed to enter or not.

V. CONCLUSION

This work to Locate and recognize face in a given Image using Python and OpenCV with Webcam is used to detect human faces with Webcam. This work used the rich library set of OpenCV for robust facedetection. For training the model with the feature set of a face Haar frontal face XML file is used. The coloured bounding boxes are drawn or generated, when faces are being detected with webcam. There are a number of detectors other than the face, which can be found in the library. We can explore the experiment with them and create detectors for eyes, license plates, etc. As shown here the most successful application of face detection would probably be photo taking as demonstrated. When you take a photo of our friends, the face detection algorithm built into your digital camera detects where the faces are and adjusts the focus accordingly to recognise the face.

REFERENCES

[1] Malina Khan, Sudeshna Chakraborty, Rani Astya, ShavetaKhepra, **"**Face Detection and Recognition Using OpenCV", International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2019.

[2] Amir Nobahar Sadeghi Nam, "Face Detection", International Journal of Innovative Science and Research Technology, Volume 5, Issue September – 2020

[3] Sidra Mehtab, Jaydib Sen, "Face Detection Using OpenCV and Haar Cascades Classifiers",from Article, 2020

[4] Venkatesh Chandra, "Real-Time Webcam Face Detection system using OpenCV in Python", 2020

[5] Haoxiang Li, Zhe Lin, Xiaohui Shen, Jonathan Brandt, Gang Hua, "A Convolutional Neural Network Cascade for Face Detection", 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015

[6] P. Viola, M.J. Jones, "Robust Real-Time Face Detection", International Journal of Computer Vision, 57.2 (2004): 137-154.

[7] Liu, Qiong, and Guang-zheng Peng. "A robust skin color based face detection algorithm", 2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010). Vol. 2. IEEE, 2010.