***Face Detection and Recognition***

Piyush Kakkar

Information Technology Department

Maharaja Agrasen Institute of Technology

Delhi, India

ABSTRACT:- This paper presents a face detection and recognition system based on AdaBoost algorithm using Haar features. The system recognizes a human’s face by their names and identifying multiple faces in real time. A pc webcam is used to record real time video data and the system automatically detects and identifies faces in real time application. An accurate location of the face is still a challenging task. Viola-Jones framework has been widely used by researchers in order to detect the location of faces and objects in a given image. Face detection classifiers are shared by public communities, such as OpenCV.

KEYWORDS:- AdaBoost; face detection; Haar classifier; real-time; Viola-Jones; OpenCV.

# **Introduction**

The face is crucial for human identity. It is the feature which best distinguishes a person. Face recognition is an interesting and challenging problem, and impacts important applications in many areas such as identification for law enforcement, authentication for banking and security system access, and personal identification among others. Face recognition is an easy task for humans but its entirely different task for a computer. A very little is known about human recognition to date on How do we analyze an image and how does the brain encode it and Are inner features (eyes, nose, mouth) or outer features (head shape, hairline) used for a successful face recognition? Neurophysiologist David Hubel and Torsten Wiesel has shown that our brain has specialized nerve cells responding to specific local features of a scene, such as lines, edges, angles or movement. Since we don’t see the world as scattered pieces, our visual cortex must somehow combine the different sources of information into useful patterns.

Automatic face recognition is all about extracting those meaningful features from an image, putting them into a useful representation and performing some classifications on them. Face recognition based on the geometric features of a face is probably the most intuitive approach to Human identification. The whole process can be divided in three major steps where the first step is to find a good database of faces with multiple images for each individual. The next step is to detect faces in the database images and use them to train the face recognizer and the last step is to test the face recognizer to recognize faces it was trained for.

Nowadays, face detection is used in many places especially the websites hosting images like Picassa, Photobucket and Facebook. The automatically tagging feature adds a new dimension to sharing pictures among the people who are in the picture and also gives the idea to other people about who the person is in the image. In our project, we have studied and implemented a pretty simple but very effective face detection algorithm which takes human skin color into account.

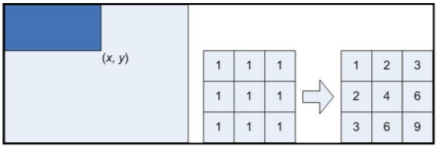
Our aim, which we believe we have reached, was to develop a method of face recognition that is fast, robust, reasonably simple and accurate with a relatively simple and easy to understand algorithms and techniques. The examples provided in this report are real-time and taken from our own surroundings.

# **Face Detection Using Haar Classifier Algorithm**

The face detection algorithm proposed by Viola and Jones is used as the basis of our design. The face detection algorithm looks for specific Haar features of a human face. When one of these features is found, the algorithm allows the face candidate to pass to the next stage of detection. A face candidate is a rectangular section of the original image called a sub-window. Generally, these sub windows have a fixed size (typically 24×24 pixels). This sub-window is often scaled in order to obtain a variety of different size faces. The algorithm scans the entire image with this window and denotes each respective section a face candidate.

## **Integral Image**

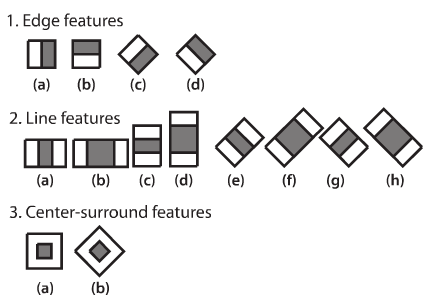
The integral image is defined as the summation of the pixel values of the original image. The value at any location (x, y) of the integral image is the sum of the image’s pixels above and to the left of location (x, y). “Fig. 1” illustrates the integral image generation.



**Figure 1**: Integral image generation. The shaded region represents the sum of the pixels up to position (x, y) of the image. It shows a 3×3 image and its integral image representation.

## **Haar Features**

Haar features are composed of either two or three rectangles. Face candidates are scanned and searched for Haar features of the current stage. The weight and size of each feature and the features themselves are generated using a machine learning algorithm from AdaBoost. The weights are constants generated by the learning algorithm. There are a variety of forms of features as seen below in “Fig. 3”.



**Figure 2:** Common Haar Features

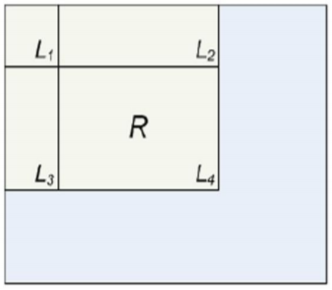
Each Haar feature has a value that is calculated by taking the area of each rectangle, multiplying each by their respective weights, and then summing the results. The area of each rectangle is easily found using the integral image. The coordinate of the any corner of a rectangle can be used to get the sum of all the pixels above and to the left of that location using the integral image. By using each corner of a rectangle, the area can be computed quickly as denoted by “Fig. 4”. Since L1 is subtracted off twice it must be added back on to get the correct area of the rectangle. The area of the rectangle R, denoted as the rectangle integral, can be computed as follows using the locations of the integral image: L4-L3-L2+L1.



**Figure 3:** Examples of Haar features. Areas of white and black regions are multiplied by their respective weights and then summed in order to get the Haar feature value.

## **Haar Feature Classifier**

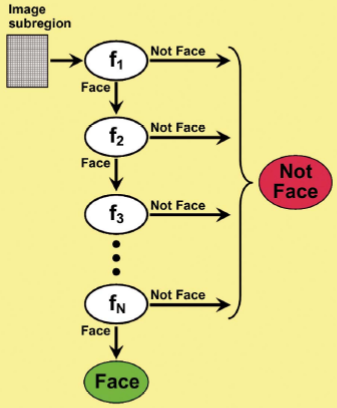
A Haar feature classifier uses the rectangle integral to calculate the value of a feature. The Haar feature classifier multiplies the weight of each rectangle by its area and the results are added together. Several Haar feature classifiers compose a stage. A stage comparator sums all the Haar feature classifier results in a stage and compares this summation with a stage threshold. The threshold is also a constant obtained from the AdaBoost algorithm. Each stage does not have a set number of Haar features. Depending on the parameters of the training data individual stages can have a varying number of Haar features. For example, Viola and Jones‟ data set used 2 features in the first stage and 10 in the second. All together they used a total of 38 stages and 6060 features [6]. Our data set is based on the OpenCV data set which used 22 stages and 2135 features in total.



**Figure 4:** Calculating the area of a rectangle R is done using the corner of the rectangle: L4-L3-L2+L1.

## **Cascade**

The Viola and Jones face detection algorithm eliminates face candidates quickly using a cascade of stages. The cascade eliminates candidates by making stricter requirements in each stage with later stages being much more difficult for a candidate to pass. Candidates exit the cascade if they pass all stages or fail any stage. A face is detected if a candidate passes all stages. This process is shown in “Fig. 5”.



**Figure 5:** Cascade of stages. Candidate must pass all stages in the cascade to be conclude

# **Implementation**

## **Import the required modules**

The Modules required to perform the facial recognition are cv2, os, image module and numpy. cv2 is the OpenCV module and contains the functions for face detection and recognition. OS will be used to maneuver with image and directory names. First, we use this module to extract the image names in the database directory and then from these names individual number is extracted, which is used as a label for the face in that image. Since, the dataset images are in gif format and as of now, OpenCV does not support gif format, Image module from PIL is used to read the image in grayscale format. Numpy arrays are used to store the images.

## **Load the face detection Cascade**

To Load the face detection cascade the first step is to detect the face in each image. Once we get the region of interest containing the face in the image, we use it for training the recognizer. For the purpose of face detection, we will use the Haar Cascade provided by OpenCV. The haar cascades that come with OpenCV are located in the directory of OpenCV installation. haarcascade frontalface default.xml is used for detecting the face. Cascade is loaded using the cv2 CascadeClassifier function which takes the path to the cascade xml file. if the xml file is in the current working directory, then relative path is used.

## **Create the Face Recognizer Object**

The next step involves creating the face recognizer object. The face recognizer object has functions like FaceRecognizer.train to train the recognizer and FaceRecognizer.predict to recognize a face. OpenCV currently provides Eigenface Recognizer, Fisherface Recognizer and Local Binary Patterns Histograms Face Recognizer. We have used Fisher Face Recognizer to perform face recognition. When Linear Discriminant Analysis(LDA) is used to find the subspace representation of a set of face images, the resulting basis vectors defining that space are known as Fisherfaces. Using FisherFaces it is possible to describe the texture and shape of a digital image. This is done by dividing an image into several small regions from which the features are extracted that can be used to get a measure for the similarity between the images.

## **Prepare the training set and Perform the training**

To create the function to prepare the training set, we will define a function that takes the absolute path to the image database as input argument and returns tuple of 2 list, one containing the detected faces and the other containing the corresponding label for that face. For example, if the ith index in the list of faces represents the 4th individual in the database, then the corresponding ith location in the list of labels has value equal to 4.

Now to perform the training using the Face Recognizer. Train function. It requires 2 arguments, the features which in this case are the images of faces and the corresponding labels assigned to these faces which in this case are the individual number that we extracted from the image names.

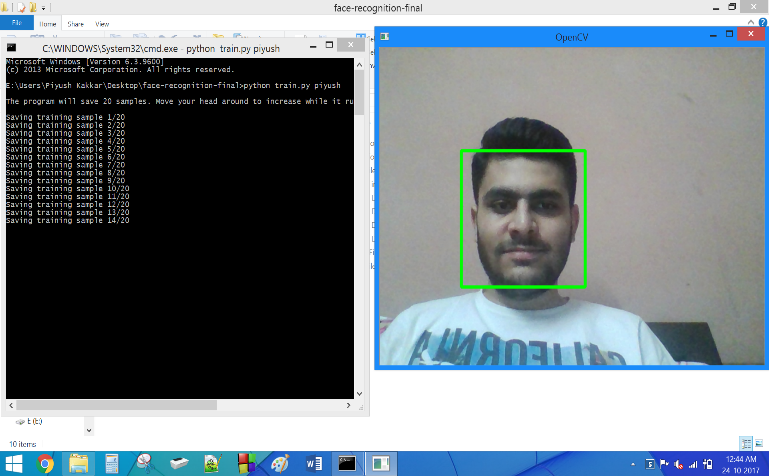
## **Testing**

For testing the Face Recognizer, we check if the recognition was correct by seeing the predicted label when we bring the trained face in front of camera. The label is extracted using the os module and the string operations from the name of the sample images folder. We also display the confidence score for each recognition. Lower is the confidence score better is the prediction.

# **Results**

## **Training**

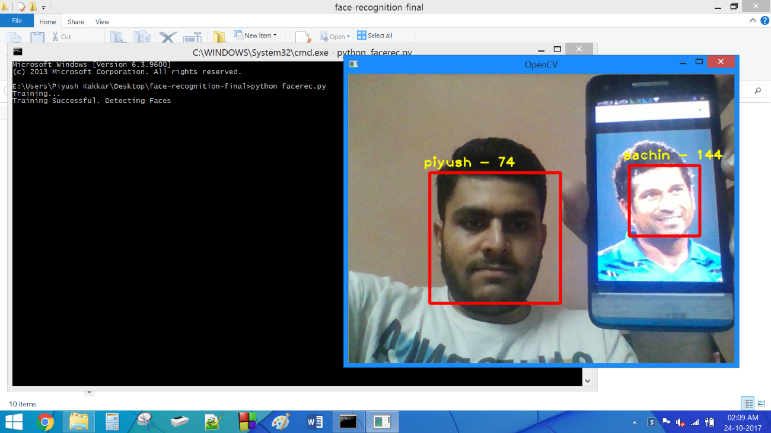
Running training script to train our model to recognize a particular person. Training script will capture 20 images of face. These images are converted to grayscale and are scaled to region where face is present.



**Figure 6:** Output: Training

## **Recognizing Face with Confidence value**

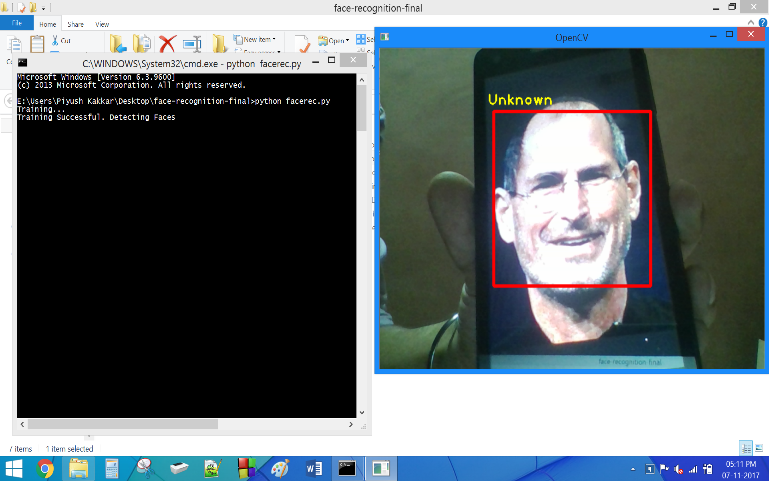
Running face recognition script when the model is trained for a particular person. At that time the face in the screen will be recognized and name is displayed along with the confidence value. Lower is the confidence value better is the face recognition.



**Figure 7:** Output: Recognized Face with Confidence Value

## **Recognizing Unknown Face**

Running face recognition script when the model is not trained for a particular person. At that time the face in the screen will be considered as unknown face.



**Figure 8:** Output: Unknown Face

# **Conclusion**

In this project, we are able to detect and recognize faces in real time obtained from camera. We have used Haar feature-based cascade classifiers approach for face detection. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Several advantages of this algorithm are: Efficient feature selection, Scale and location invariant detector, instead of scaling the image itself, we scale the features Such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, legs).

It also has some disadvantages: Detector is most effective only on frontal images of faces, it can hardly cope with 45° face rotation both around the vertical and horizontal axis and Sensitive to lighting conditions.

##### **References**

1. Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Transactions on Pattern Analysis and Machine Intelligence. 19, pp. 711-720. IEEE Computer Society
2. Bornet, O. (2005, May 19). Learning Based Computer Vision with Intel's Open Source Computer Vision Library. Retrieved April 2007, 2007, from Intel.com Website: http://www.intel.com/technology/itj/2005/volume09issue02/art03\_learning\_vision/p04\_face\_dete ction.htm
3. Brunelli, R., & Poggio, T. (1993). Face Recognition: Features versus templates. IEEE Transaction on Pattern Analysis and Machine Intelligence , 15 (10), 1042-1052.
4. Open Computer Vision Library Reference Manual. Intel Corporation, USA, 2001.
5. Viola, P. and Jones, M. Rapid object detection using boosted cascade of simple features. IEEE Conference on Computer Vision and Pattern Recognition, 2001.
6. http://docs.opencv.org/trunk/d7/d8b/tutorial\_py\_face\_detection.html
7. http://opencv-python-tutroals.readthedocs.io/en/latest/index.html
8. http://ieeexplore.ieee.org/document/7800424/