Face, Eyes, Nose and Lips Detection using MATLAB

MATLAB for Engineers - 2ECOE76

Akshat Shah - 19BCE246
Computer Science Department
Institute of Technology, Nirma University
Ahemdabad 382 481, India
19bce246@nirmauni.ac.in

Devansh Shah - 19BCE247
Computer Science Department
Institute of Technology, Nirma University
Ahemdabad 382 481, India
19bce247@nirmauni.ac.in

***Abstract*—In this study, we provide a neural network experiment for the facial recognition problem. The face is recognised using neural networks that learn the right classification of the coefficients obtained by the Eigenface technique. The network is trained on photographs from the face database before being used to recognise the faces that are shown to it. The task of training the neural networks was accomplished using the ACF detector. Apps like Image labeller from MATLAB were used to classify photos into four labels: 'face,' 'eyes,' 'nose,' and 'lips,' accordingly. There were about 160 photos considered for labelling and data training using positive and negative samples. Only a handful were chosen to use the confidence interval to represent the accuracy results.**

***Index Terms*—Detection, Image Labeller, MATLAB, ACF, Features, Computer Vision**

I. Introduction

FACE recognition is a challenge of visual pattern recognition. In more detail, a face recognition system with an arbitrary image input will search a database to identify people in the input image. A face recognition system consists of four modules: detection, alignment, feature extraction, and matching, with localization and normalisation (face detection and alignment) serving as processing steps preceding face recognition (facial feature extraction and matching)[1].

Face detection distinguishes the parts of the face from the backdrop. In the case of video, a face tracking component may be required to monitor the discovered faces. Face alignment tries to achieve more accurate localization and, as a result, normalise faces, whereas face detection offers coarse approximations of each identified face's location and scale. The input face image is normalised with regard to geometrical qualities, such as size and pose, utilising geometrical transforms or morphing, based on the position points for facial components such as eyes, nose, and mouth. After a face has been normalised geometrically and photometrically, feature extraction is used to generate meaningful information that may be used to discriminate between different people's faces while being stable in terms of geometrical and photometrical variances. The extracted feature vector of the input face is matched against those of enrolled faces in the database for face matching; when a match is established with appropriate confidence, it outputs the identification of the face; otherwise, it indicates an unknown face.

In the last 20 years, artificial neural networks have been successfully used to solve signal processing difficulties. Many distinct artificial neural network models have been proposed by researchers. The task at hand is to find the most appropriate neural network model that can dependably solve real-world problems [2].

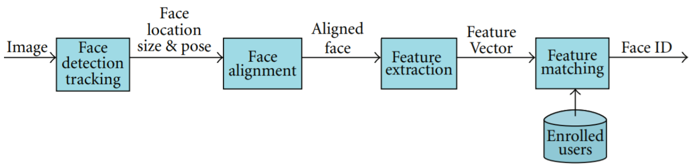


Fig. 1. Structure of a Face Recognition System

II. Detection of Object

Convolutional neural networks (CNNs, or ConvNets) are used to perform classification, object detection, and transfer learning, as well as to develop customised detectors. Object detection is a computer vision approach for detecting things in photos and videos. To obtain relevant results, object detection algorithms often use machine learning or deep learning. Humans can recognise and locate objects of interest in photos or video in a few of seconds. The purpose of object detection is to use a computer to imitate this intelligence. The optimum object detection strategy is determined by your application and the problem you're trying to address.

Deep learning techniques necessitate a large number of labelled training images, so using a GPU to speed up the training process is recommended. Convolutional neural networks (CNNs or ConvNets), such as R-CNN and YOLO v2, or single-shot detection are used in deep learning-based techniques to object detection (SSD). You can use transfer learning to train a bespoke object detector or use a pretrained object detector. Transfer learning allows you to start with a pretrained network and fine-tune it for your application. Deep Learning ToolboxTM is required for convolutional neural networks. A CUDA®-capable GPU is required for training and prediction. The use of a GPU is suggested, and Parallel Computing Toolbox is required.

For human face or upper-body detection, machine learning algorithms include aggregate channel features (ACF), support vector machines (SVM) classification utilising histograms of oriented gradient (HOG) features, and the Viola-Jones algorithm. You have the option of starting with a pre-trained object detector or creating a bespoke object detector tailored to your needs.

III. ACF for Face Detection

A. *Aggregate Channel Features Detector Overview*

The Aggregate Channel Features (ACF) object detection code is implemented in the detector section of this toolkit. The ACF detector is a sliding window detector that is both quick and effective (30 fps on a single core). It's based on the Viola Jones (VJ) detector, but with a 1000-fold reduction in false positives (at the same detection rate). ACF works best for semi-rigid object detection (e.g. faces, pedestrians, cars).

B. *Image Labeller*

You may label ground truth data in a collection of photographs with the Image Labeler app. • Rectangular regions of interest (ROI) labels, polyline ROI labels, pixel ROI labels, polygon ROI labels, and scene labels can all be defined using the app. Use these labels to label your ground truth data interactively.

• Label your ground truth data with built-in detection or tracking techniques.

• Create, import, and apply your own unique automation algorithm to label ground truth automatically.

• Use a visual report to assess the performance of your label automation techniques.

• Create a groundTruth object from the labelled ground truth. This item can be used to test a system or to train an object detector or semantic segmentation network.

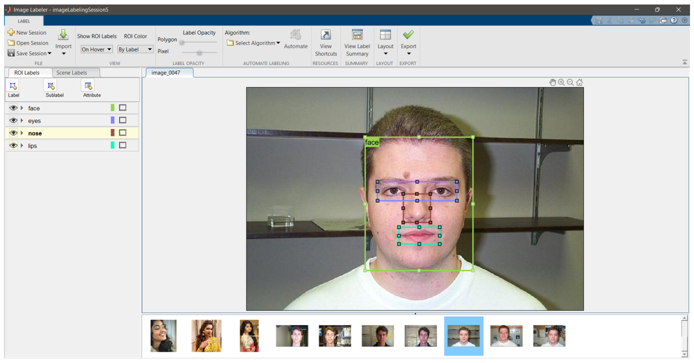


Fig. 2. Image Labeller

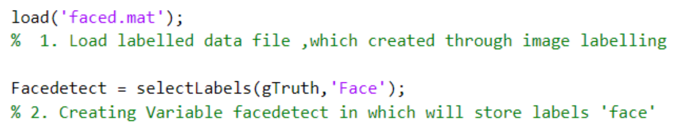
As shown in the above output, each and every image is being labelled with 4 type of labels:

- Face
- Eyes
- Nose
- Lips

The Image Labeler programme is compatible with all image file types supported by the imread function, as well as the DICOM (Digital Imaging and Communication in Medicine) format. Create an imageDatastore and utilise the ReadFcn attribute to read additional file formats.

If a picture has a size larger than 8000 pixels or is a multiresolution image, the Image Labeler app allows you to convert it to a blocked image when loading it. A blocked image is a huge image that has been broken down into tiny blocks to fit into memory. After the Image Labeler has converted the huge image to a block, you may use the programme to process it like any other image. While using restricted pictures allows you to analyse photographs in the app that you wouldn't be able to otherwise, it does come with certain drawbacks.

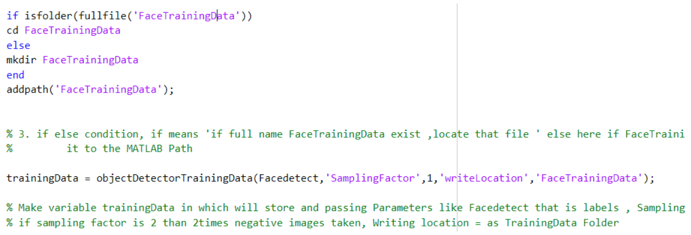
This file is exported to different .mat files for further processes of selecting labels. The exported file is loaded into the training MATLAB file.



```
load('faced.mat');
%  1. Load labelled data file ,which created through image labelling

Facedetect = selectLabels(gTruth,'Face');
% 2. Creating Variable facedetect in which will store labels 'face'
```

Fig. 3. Code Snippet for Training Model

'SelectLabels' stores a new groundTruth object, or array of groundTruth objects, containing only the labels specified by labelNames.('face' here) into Facedetect variable.



```
if isfolder(fullfile('FaceTrainingData'))
cd FaceTrainingData
else
mkdir FaceTrainingData
end
addpath('FaceTrainingData');


% 3. if else condition, if means 'if full name FaceTrainingData exist ,locate that file ' else here if FaceTraini
%         it to the MATLAB Path

trainingData = objectDetectorTrainingData(Facedetect,'SamplingFactor',1,'writeLocation','FaceTrainingData');

% Make variable trainingData in which will store and passing Parameters like Facedetect that is labels , Sampling
% if sampling factor is 2 than 2times negative images taken, Writing location = as TrainingData Folder
```

Fig. 4. Code Snippet for storing the Training Data

A directory in which the training data will be stored is made explicitly named 'FaceTrainingData'.

A table needs to be created for storing the co-ordinates of the label in rectangular format for each image.

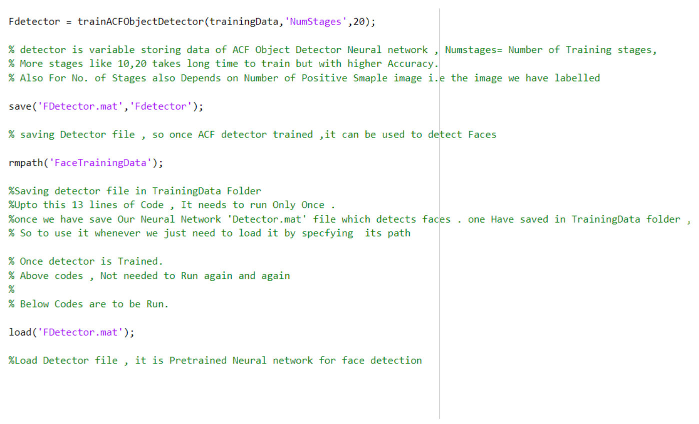
objectDetectorTrainingData() returns a table of training data from the specified ground truth. gTruth is an array of groundTruth objects. You can use the table to train an object detector using the Computer Vision Toolbox™ training functions. And it is stored in variable 'trainingdata'. The table is created as shown below:

C. *Train ACF Object Detector*

detector = trainACFObjectDetector(trainingData) returns a trained aggregate channel features (ACF) object detector. The function uses positive instances of objects in images given in the trainingData table and automatically collects negative instances from the images during training. To create a ground truth table, use the Image Labeler or Video Labeler app.

detector = trainACFObjectDetector(trainingData,Name,Value) returns a detector object with additional options specified by one or more Name,Value pair arguments. Use the *trainACFObjectDetector* with training images to create an ACF object detector that can detect stop signs. Test the detector with a separate image.

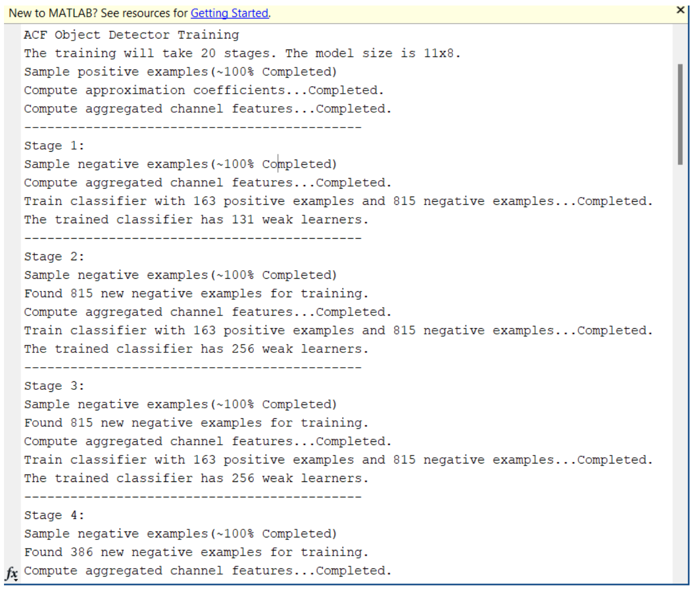
We are making a detector object for our training data and storing it in ‘Fdetector’. And we are processing the same for 20 stages for a higher accuracy. It would take 5-10 mins for complete data training.



```
Fdetector = trainACFObjectDetector(trainingData,'NumStages',20);

% detector is variable storing data of ACF Object Detector Neural network , Numstages= Number of Training stages,
% More stages like 10,20 takes long time to train but with higher Accuracy.
% Also For No. of Stages also Depends on Number of Positive Smaple image i.e the image we have labelled

save('FDetector.mat','Fdetector');

% saving Detector file , so once ACF detector trained ,it can be used to detect Faces

rmpath('FaceTrainingData');

%Saving detector file in TrainingData Folder
%Upto this 13 lines of Code , It needs to run Only Once .
%once we have save Our Neural Network 'Detector.mat' file which detects faces . one Have saved in TrainingData folder ,
% So to use it whenever we just need to load it by specfying  its path

% Once detector is Trained.
% Above codes , Not needed to Run again and again
%
% Below Codes are to be Run.

load('FDetector.mat');

%Load Detector file , it is Pretrained Neural network for face detection
```

Fig. 5. Code Snippet for Detector

Data Training Stages



New to MATLAB? See resources for Getting Started.

```
ACF Object Detector Training
The training will take 20 stages. The model size is 11x8.
Sample positive examples(~100% Completed)
Compute approximation coefficients...Completed.
Compute aggregated channel features...Completed.
--------------------------------------------
Stage 1:
Sample negative examples(~100% Completed)
Compute aggregated channel features...Completed.
Train classifier with 163 positive examples and 815 negative examples...Completed.
The trained classifier has 131 weak learners.
--------------------------------------------
Stage 2:
Sample negative examples(~100% Completed)
Found 815 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 163 positive examples and 815 negative examples...Completed.
The trained classifier has 256 weak learners.
--------------------------------------------
Stage 3:
Sample negative examples(~100% Completed)
Found 815 new negative examples for training.
Compute aggregated channel features...Completed.
Train classifier with 163 positive examples and 815 negative examples...Completed.
The trained classifier has 256 weak learners.
--------------------------------------------
Stage 4:
Sample negative examples(~100% Completed)
Found 386 new negative examples for training.
Compute aggregated channel features...Completed.
```

Fig. 6. Training Stages

D. *Prediction Time*

Now, we will import any random image to our workspace and then try to predict the perfect borderlines/labels for the same. After importing the detector file first, detect fucntion is used to detects objects within image I using the input aggregate channel features (ACF) object detector. The locations of objects detected are returned as a set of bounding boxes.

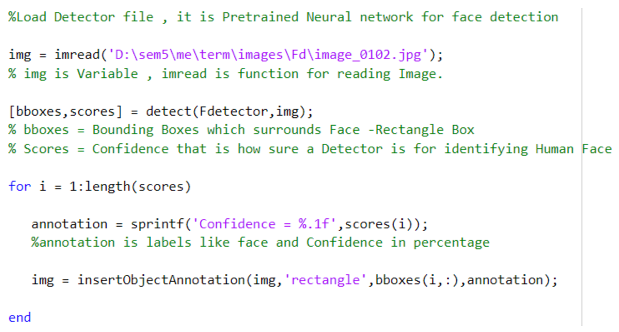
Here , [bboxes,scores] will also give the detection scores for each bounding box.

Then , we will create another different image with lables on it for better visualization purpose., which is possible using ‘insertObjectAnnotation’ function.

This function returns a truecolor image annotated with shape and label at the location specified by position.

Syntax: *insertObjectAnnotation(I,shape,position,label)*

Data Training Stages



```
%Load Detector file , it is Pretrained Neural network for face detection

img = imread('D:\sem5\me\term\images\Fd\image_0102.jpg');
% img is Variable , imread is function for reading Image.

[bboxes,scores] = detect(Fdetector,img);
% bboxes = Bounding Boxes which surrounds Face -Rectangle Box
% Scores = Confidence that is how sure a Detector is for identifying Human Face

for i = 1:length(scores)

   annotation = sprintf('Confidence = %.1f',scores(i));
   %annotation is labels like face and Confidence in percentage

   img = insertObjectAnnotation(img,'rectangle',bboxes(i,:),annotation);

end
```

Fig. 7. Code Snippet for Detection

Data Training Stages

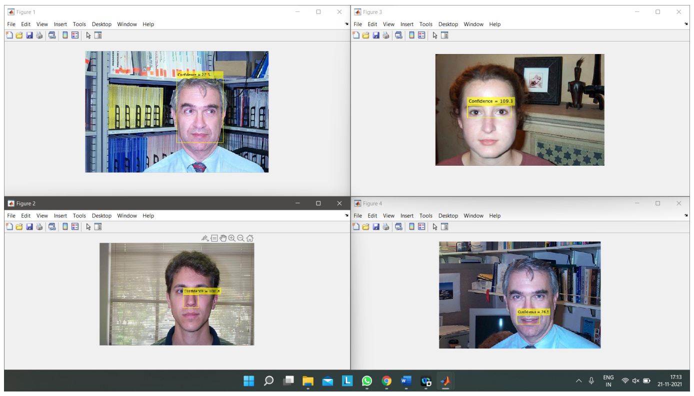


Fig. 8. [Detected Output] Face, Eyes, Nose and Lips

E. *AdaBoost and ANN for Face Detection*

The initial phase in a face recognition system is face detection processing. The phase will determine the system’s performance, hence it is the most crucial step in the recognition system. Many researchers have presented many techniques to carry it out efficiently. Face detection technologies can be divided into four categories:

1) Knowledge-based approaches
2) Invariant feature-based methods
3) Template matching-based methods
4) Machine learning-based methods

We concentrate on machine learning approaches in this work because they remove subjective thinking aspects from human experience. Furthermore, they make final conclusions based solely on training data. As a result, provided the training data is well-organized and sufficient, these systems can attain great performance without the use of humans [5].

The AdaBoost technique is one of the most popular and efficient learning machine-based algorithms for detecting faces.

Viola and Jones created a face detection system that uses AdaBoost learning to create nonlinear classifiers. The following three essential difficulties are solved by AdaBoost:

1) learning useful features from a huge feature set

2) building weak classifiers, each based on one of the selected features

3) boosting the weak classifiers to build a strong classifier

Viola and Jones employ a number of strategies to efficiently compute a large number of such characteristics at different scales and locations, which is critical for real-time performance. Furthermore, the calculation will be considerably more efficient thanks to the cascade of powerful classifiers that make up the cascade tree. Their technique is the first to recognise faces in frontal views in real time. However, there are certain flaws in their system. Because the detection findings are based on weak classifiers, there are often a lot of false positives. To reduce the percentage of false positives, the amount of strong classifiers and Haar-like features in the cascade tree must be increased, however this will result in a considerable increase in performance time, and the detection rate can be reduced. To address the problem, we need combine AdaBoost with other machine learning approaches in order to attain the same face detection ratios while reducing the amount of false positives and running time[6].

Artificial neural networks (ANNs) [7] are a common approach that achieves the same result. The word ”ANN” refers to a method for solving issues by imitating the actions of neurons. In more detail, ANNs can be best described as ”computational models” with specific qualities such as the ability to adapt or learn, generalise, cluster, or organise data, and a parallel processing operation. Many of the above listed characteristics, on the other hand, can be attributed to non-neural models. To detect faces, a hybrid technique combining AdaBoost and ANN is presented, with the goal of reducing performance time while still obtaining the target face detection rate. ABANN is the name of our hybrid model. This is the face detection model that combines AB and ANN. In this model, ABs are responsible for immediately rejecting nonface images, while ANNs filter false negative images to improve results. Face/nonface is the end result. The three-layer feedforward neural network with back propagation algorithm is the neural network of choice here. The number of input neurons T is equal to the length of the extracted feature vector, and the number of output neurons is 1 (C = 1). This will return true if the image contains a human face, and false otherwise. The number of hidden neurons H will be chosen based on the results of the experiment; it will be determined by the images in the sample database set. The input of ANN is the result image (20 x 20 pixels) of AB. The ANN's output is a real value between -1 (false) and 1 (true) (true). The preprocessing and ANN steps are illustrated in Figure 3(b). The original image is decomposed into a pyramid of images as follows: 4 blocks 10×10 pixels, 16 blocks 5×5 pixels, and 5 overlapping blocks 20×6 pixels. Thus, the ANN will have 4 + 16 + 5 = 25 input nodes. Its goal is to find out important face features: horizontal blocks to find out mouths and eyes, square blocks to find out each of the eyes, noses, and mouths. The system uses one hidden layer with 25 nodes to represent local features that characterize faces well. Its activation function is Tanh function with the learning rate = 0.3. In detail, a model of cascade of classifiers includes many strong classifiers, and ANN is combined with the strong classifiers to be a final strong classifier of the system to achieve better results in Figure. For example, AB includes 5 strong classifiers, called AB5, which will be combined with ANN, the sixth strong classifier, to be ABANN5 [7].

The image results of the step will be the inputs of the face alignment step. The next section elaborates our proposed method.

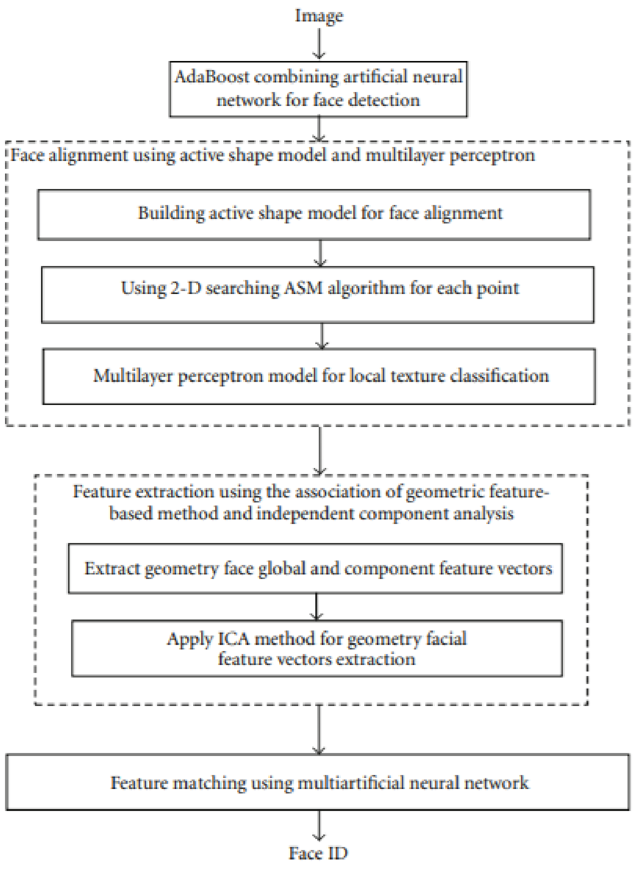


Image
AdaBoost combining artificial neural network for face detection
Face alignment using active shape model and multilayer perceptron
Building active shape model for face alignment
Using 2-D searching ASM algorithm for each point
Multilayer perceptron model for local texture classification
Feature extraction using the association of geometric feature-based method and independent component analysis
Extract geometry face global and component feature vectors
Apply ICA method for geometry facial feature vectors extraction
Feature matching using multiartificial neural network
Face ID

Fig. 9. Proposed Models for Steps of a Face Recognition System

IV. CONCLUSION

This research introduced new models for all stages of human face identification in two-dimensional digital photos. We can infer that the ACF object detector module provides good accuracy and precision for discriminating between items as well as human faces and distinct face parts when it comes to face detection. On the other hand, training the data takes a long period, which is not practical in today's generation. A model combining three-layer feed forward artificial neural network and AdaBoost was presented for detecting human faces in a face detection module utilising ANN and Adaboost. The tests were carried out on a challenging face detection database that has been extensively explored (MIT + CMU database). The results demonstrate that ABANN not only achieves an

AdaBoost detector's approximate detection rate and processing time, but also decreases false detections. The shortcomings of AdaBoost and the ANN detector were addressed by ABANN.

ACKNOWLEDGMENT

We, Akshat Shah - 19BCE246 and Devansh Shah - 19BCE247 would like to express our heartfelt gratitude to our teacher, Prof. Twinkle Bhavsar, as well as our Institute of Technology, Nirma University, for providing us with the wonderful opportunity to work on this wonderful project on the topic of Face, Eyes, Nose and Lips Recognition using MATLAB, which also enabled us to conduct extensive research and learn about a variety of new things. We owe them a debt of gratitude.

REFERENCES

[1] S. Z. Li and A. K. Jain, Handbook of Face Recognition, Springer, New York, NY, USA, 2004.
[2] C. Bishop, Pattern Recognition and Machine Learning, Springer, New York, NY, USA, 2006.
[3] P. Dollr, Z. Tu, P. Perona and S. Belongie ”Integral Channel Features”, BMVC 2009.
[4] Markus Weber, Frontal Face Database, California Institute of Technology, 1999.
[5] M. H. Yang, D. J. Kriegman, and N. Ahuja, “Detecting faces in images: a survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 1, pp. 34–58, 2002.
[6] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 511–518, December 2001.
[7] H. A. Rowley, Neural Network Based Face Detection, Neural network Based Face Detection, School of Computer Science, Computer Science Department, Carnegie Mellon University , Pittsburgh, Pa, USA, 1999.