# Facial Recognition using Machine Learning Algorithms on Raspberry Pi

Seema Singh
*Professor and Head,*
*Dept of Electronics and Telecommunication Engineering,*
BMS Institute of Technology and Management,
Bangalore, Karnataka
seemasingh@bmsit.in

Ramya R
*Dept of Electronics and Communication Engineering,*
BMS Institute of Technology and Management,
Bangalore, Karnataka
ramya.ravi2104@gmail.com

Sushma V
*Dept of Electronics and Communication Engineering,* BMS Institute of Technology and Management, Bangalore, Karnataka
vsushma2307@gmail.com

Roshini SR
*Dept of Electronics and Communication Engineering,* BMS Institute of Technology and Management, Bangalore, Karnataka
roshinirishab@gmail.com

Pavithra R
*Dept of Electronics and Communication Engineering,* BMS Institute of Technology and Management, Bangalore, Karnataka
rpavithra2797@gmail.com

***Abstract* – Facial recognition is a non-invasive method of biometric authentication and useful for numerous applications. The real time implementation of the algorithm with adequate accuracy is required, with hardware timing into consideration. This paper deals with the implementation of machine learning algorithm for real time facial image recognition. Two dominant methods out of many facial recognition methods are discussed, simulated and implemented using Raspberry Pi. A rigorous comparative analysis is presented considering various limitations which may be the case required for innumerable application which utilize facial recognition. The drawbacks and different use cases of each method is highlighted. The facial recognition software uses algorithms to compare a digital image captured through a camera, to the stored face print so as to authenticate a person's identity. The Haar-Cascade method was one of the first methods developed for facial recognition. The HOG (Histogram of Oriented Gradients) method has worked very effectively for object recognition and thus suitable for facial recognition also. Both the methods are compared with Eigen feature-based face recognition algorithm. Various important features are experimented like speed of operation, lighting condition, frontal face profile, side profiles, distance of image, size of image etc. The facial recognition model is implemented to detect and recognize faces in real-time by means of Raspberry Pi and Pi camera for the user defined database in addition to the available databases.**

***Keywords– Facial recognition, Haar Cascade, HOG, Comparison, Raspberry Pi***

## I. INTRODUCTION

Face is the most authentic biometric method to identify people. Face recognition is widely used in numerous applications of security like criminal detection, airport, face tracking, forensic etc. The advantage that face recognition offers over other biometric characters like palm print, finger print, iris etc., is that it is non-invasive. It can work even without the user's knowledge and can also be useful for security-based applications and other statistics-based applications.

Face biometrics is most promising but challenging area of research. Various limitations imposed for a face recognition system are variation in illumination, variation in pose, variation in expression, the age factor, occlusion etc. Different methods are proposed in literature which works well to overcome different limitations.

Face biometrics involves three basic steps, first being training the system with labelled images, secondly, classifying them into labelled classes and thirdly, storing them in the database. When a test image of known or unknown person is presented to the system, it is compared with the existing database and then classified.

There are many algorithms that can be used to achieve this. Two methods, namely HOG and the Haar-Cascade method have been considered, implemented and compared in terms of reliability, accuracy and speed.

Haar method is implemented on Raspberry Pi using Pi-camera as the real time image capturing device. The challenges with respect to implementation of HOG method on the same target is presented in the paper. Implementation is shown with user defined database and the results are discussed with real time images which overcomes all the above-mentioned challenges of facial recognition.

## II. RELATED WORK

Many researchers have attempted for facial recognition, keeping the speed of operation as the key factor. The viola jones object detection framework [1] was able to detect faces with minimum computation time and high detection rates. A boosted cascade of simple features approach was used for rapid object detection. Freund et al [3] introduced the new intermediate image representation called integral image which allows very fast feature evaluation. Feature selection process through Adaboost to form classifiers and combining the classifiers to form the cascaded structure to increase the speed of detection by focusing more in regions which is more likely to contain a face was used. Facial detection becomes

978-1-7281-3261-7/19/$31.00 ©2019 IEEE

challenging when it has to detect faces in unconstrained images subjected to different face orientations and illuminations. The facial detection for faces subjected to orientations from +90 to -90 degree using HOG descriptor is proposed in [4]. Dalal et al [5] produced excellent results for pedestrian detection using HOG and also showed detection performance is better than Haar wavelet-based detector. Chang et al [6] proposed the use HOG descriptors for facial recognition and it achieved almost the same recognition rate with lesser computation time than the Gabor descriptor and better accuracy than the LBP (Local binary pattern) descriptor. The process of facial recognition consists of three main steps: detection and normalization which is done roughly, feature extraction and precise normalization, finally identify the faces with the available database. In most of the applications it is required that all the steps are carried out in a small fraction of time, thus there is a need to separate each subtask. It becomes difficult to recognize the faces from the 2D images which are converted from the 3D images as it reduces the efficiency. However, most of the steps involved in the recognition process can be combined depending on the application according to which the required feature extraction techniques can be used. Zhao et al [11] compared the performance of different recognition algorithms. As it is clearly evident from the literature, the cascade method of Haar and HOG are explored by the researchers to a great extent. However, the important factor of its implementation in target hardware needs to be studied. The performance of the algorithms in the target hardware majorly decides its usage in any application. This work aims to implement both HOG and Haar algorithm for user created database in Raspberry Pi. This helps to analyse the performance of these algorithms for any application which requires a portable device for facial recognition.

## III. METHODOLOGY

### A. Haar Model

Haar-Cascade is one of the machine learning algorithms used widely to detect faces in an image. Haar is predominantly a texture based detection algorithm. It distinguishes the lighter and darker regions of the face. The first step of the algorithm is to collect haar-features. It considers adjacent rectangular region at a specific location of the detection window and sums up the pixel values of the darker region and this value is subtracted from the pixel sum of the lighter region. Such as the eye region is darker when compared to the Nose Bridge and upper cheek region in all faces. It is one of the strong two-feature classifiers.

The value of a two-rectangle feature is the subtraction of the sum of pixels of one rectangular region from the other. The regions are similar in dimensions and are horizontally adjacent to each other. It may be either horizontal or vertical as shown in Figure 1. When it comes to a three-rectangle feature the computation is done by calculating the sum of pixels within two outside rectangles subtracted from the sum of pixels in the centre rectangle. And, in a four-rectangle feature the value is computed by taking the difference between diagonal pairs of rectangles [1].

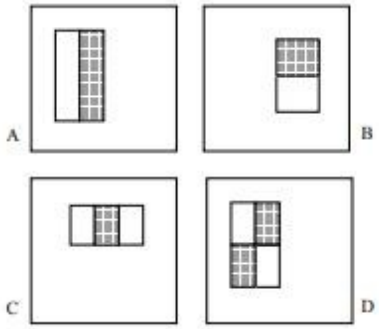


Fig 1. (A) and (B) Two-rectangle features, (C) Three-rectangle feature, (D) Four-rectangle feature.

For the faster computation of the rectangular features, intermediate representation of an image called integral image is used, the integral image of any particular location, for example x, y contains the sum of pixel values to the left and above the location of x, y.

$$ii(x, y) = \Sigma\, i(x', y') x' \leq x, y' \leq y$$

ii(x,y) is the integral image of the original image i(x',y') [1]

Majority of the extracted features are irrelevant and the best features need to be selected for further processing. There will be 180000 features associated with each sub window amongst which the best features are selected by Adaboost training [1]. Boosting is a technique which builds strong classifier by considering the weighted average of the decision made by weak learners [2]. These relevant features form the classifier. Then the classifiers are cascaded to achieve increased detection performance by reducing the false rate.

The overall detection process forms a degenerate decision tree, which is known as "cascade". The first classifier's positive result triggers the evaluation of the second classifier, both of which have been configured to obtain high detection rates. Second classifier's positive result triggers a third classifier, and so on. Instant rejection of the sub-windows [1] happens in case of negative result at any point. Thus, it does early rejection of the non-face regions in an image. This method has been explored well by the researcher. The hardware implementation and validation is done using Raspberry Pi to reinforce the same.

### B. HOG Model

Histogram is the optical representation of distribution of the data; it shows sample amount of data and the number of repeated times of those data values. An image histogram acts as a pictorial representation of the tonal distribution in the image which is in its digital form. As it is the graphical representation, it plots the number of pixels for each tonal value in the digital image. Tonal distribution is the relative darkness or lightness of an area that varies from bright white of a light source to dark region.

An image gradient is a change in the direction in intensity in an image. The calculated image gradients can be used for

extracting information from the images. Those gradient images are developed from the original image generally with the help of filters by the usage of convolution like Sobel filter. Each pixel value of a gradient image calculates the intensity value change of that same pixel value in the original image, in a given direction.

Calculation of gradient in image needs to be carried out. The gradient is the vector $(g_x, g_y)$, where an image is a discrete function of (x, y).

The direction of gradient is given by,

$$\Theta = \tan^{-1}\{g_x/g_y\}$$

And the magnitude is given by,

$$G = \sqrt{(g_{x2} + g_{y2})}$$

The edges or borders present in an image are obtained by the derivatives, which show the quick transitions from one hue to another of a color. The derivative of a matrix is calculated by an operator called the Laplacian. One needs to calculate first two derivatives, called derivatives of Sobel, each of which takes into account the gradient variations in a certain direction, one horizontal and other vertical.

Calculation of histogram of Oriented Gradients is done using following steps:

**Step 1: Pre-processing**

In the pre-processing step, the bounding section of the object is extracted. In this case, images with the human faces are considered. If images of only individual faces are considered, then pre-processing is assumed to be done by default.

**Step 2: Developing the gradient image**

To develop a HOG descriptor, horizontal and vertical gradients are measured first and the result obtained is used to measure the magnitude and direction of the gradient image.

The image in figure 2 shows the gradient,

Fig 2. Left: Magnitude of x-gradient. Center: Magnitude of y-gradient. Right: Magnitude of the resultant gradient.

The x-gradient highlights on vertical values and y-gradient highlights on horizontal values. The magnitude of gradient highlights wherever there is intense or sharp change in the intensity. None of them gets highlighted when the region is smooth i.e. when there is no sharp change in the intensity.

For images that have colour in them, the gradient values are calculated for the three channels i.e. Red, Green and Blue. The magnitude of gradient at a pixel is the highest value of the magnitude amongst the three channels, and the angle is the angle corresponding to the highest gradient.

**Step 3: Developing Histogram of Gradients in 8□8 cells.**

Here, the image gets a division of 8x8 cells and a histogram of gradients is developed for those divided in 8x8 cells of the image. An 8x8 image patch contains 8x8x3 = 192-pixel values. The gradient of this patch contains 2 values (magnitude and direction) per pixel which adds up to 8x8x2 = 128 numbers. Hence those 128 numbers are viewed using 9-bin histogram. That 9-bin histogram can be used to store as a row of 9 numbers. Figure 3 shows an 8x8 patch in the image and its gradients.

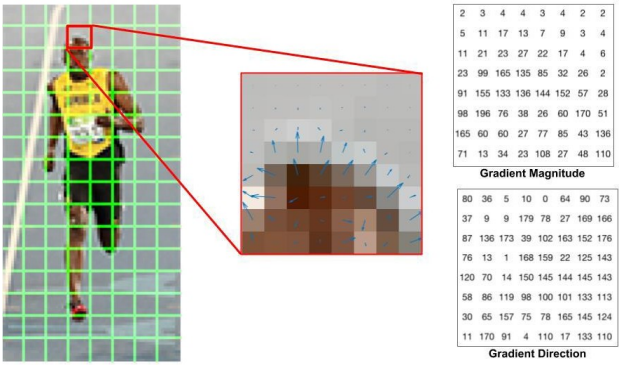


Fig 3. Center: The RGB patch and gradients represented using arrows. Right: The gradients in the same patch represented as numbers.

The next step is to create a histogram of gradients in these 8x8 cells. The histogram contains 9 bins corresponding to angles 0, 20, 40… 160.

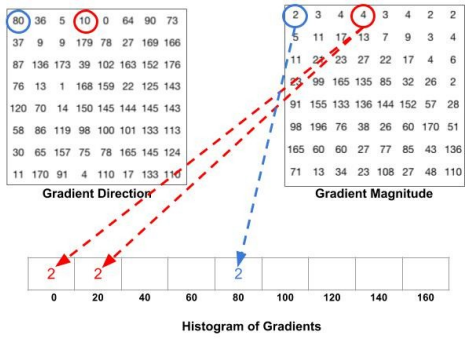


Figure 3 illustrates the process. It shows the magnitude and direction of the gradient of the same 8×8 patch which

was shown in Figure 2. A bin is selected based on the direction, and the vote (the value that goes into the bin) is selected based on the magnitude.

For example, focus on the pixel encircled in blue colour is focused first. It has an angle (direction) of 80 degrees and magnitude of 2. So it adds 2 to the 5th bin. The gradient at the pixel encircled using red colour has an angle of 10 degrees and magnitude of 4. Since 10 degrees is half way between 0 and 20, the vote by the pixel splits evenly into the two bins. If the angle is greater than 160 degrees, it is between 160 and 180, and the angle wraps around making 0 and 180 equivalents.

The contributions of all the pixels in the 8×8 cells are added up to create the 9-bin histogram. For the patch of figure 3, the block normalization is shown in Figure 4.

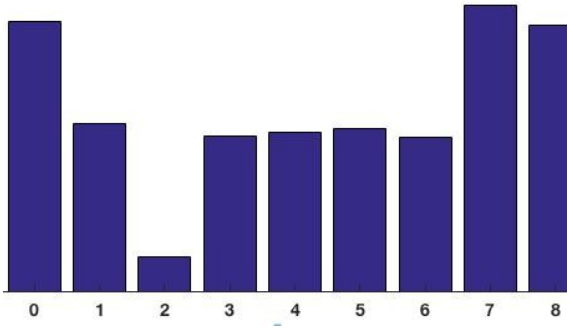


Fig 4. 16x16 Block Normalization

**Step 4- 16x16 Block Normalization**

Gradients of an image are sensitive to overall lighting. If the image is made darker by dividing all pixel values by 2, the gradient magnitude will change by half, and therefore the histogram values will change by half. Ideally, descriptor needs to be independent of lighting variations. The histogram is normalized so that they are unaffected by lighting variations due to camera changes.

For the image, normalization is done over a bigger sized block of 16×16. A 16×16 block has 4 histograms which can be concatenated to form a 36 x 1 element vector and it is normalized just the way a 3×1 vector is normalized. The window is then moved by 8 pixels and a normalized 36×1 vector is calculated over this window and the process is repeated over the entire image.

**Step 5: Calculation of the HOG feature vector**

The complete image patch, the 36×1 vector are concatenated into one giant vector. This forms the final feature vector. Each 16×16 block is represented by a 36×1 vector. After concatenation of all into one giant vector, a 36×105 = 3780-dimensional feature vector is obtained.

## IV. IMPLEMENTATION

HOG and Haar algorithm is implemented to compare its reliability, accuracy and speed. The Haar method was implemented on the Raspberry Pi 3 Model B+ with 1.4GHz Cortex A53 with 1GB RAM. There is a considerable lag in the real-time output. However, it works real-time on a traditional laptop CPU giving instant efficient output.

HOG method was also tried on the Raspberry Pi but it was seen to be too computationally intensive and crashed multiple times. Hence HOG was implemented on a system with Intel i5 processor and the real-time video, though choppy was enough to check accuracy and reliability.

## V. RESULTS

This implementation of the HOG method on a system with i5 processor which has support of the Nvidia Geforce Graphic Processing Unit (GPU) was carried out. Figure 5 shows the training on the user defined database using HOG algorithm. Figure 6 shows the result where the algorithm is able to identify human faces correctly in a group of images also.

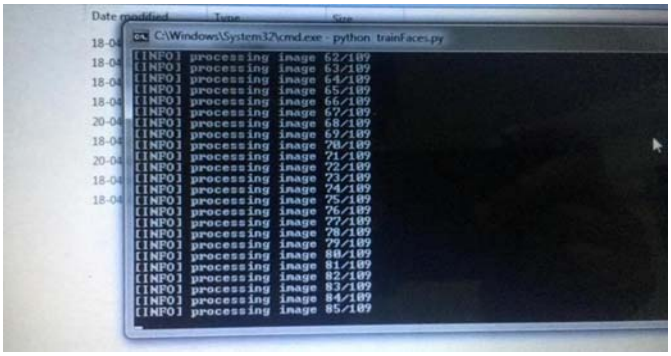


Fig 5. Training using the HOG algorithm

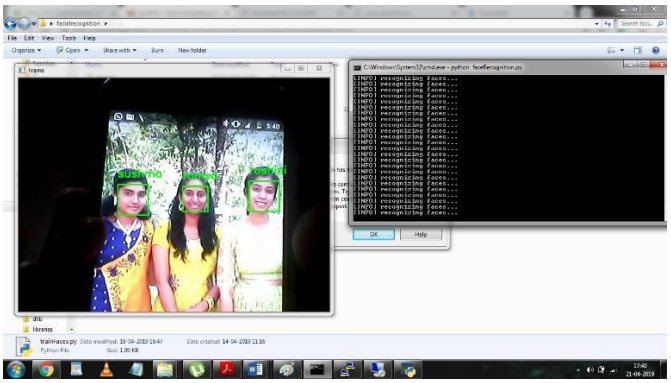

Fig 6.Correct labels assigned for the faces identified in the group images by a model trained using HOG method.

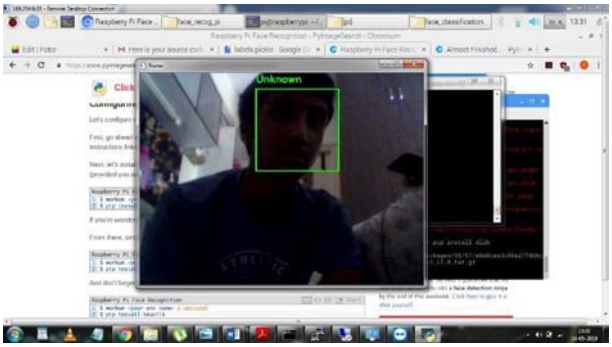


Fig 7. This method also gives the label UNKNOWN for a person whose images have not been pre-trained

It is important that the algorithm gives an unknown label for the face which is not trained with. One of such case is shown in Figure 7. HOG method works efficiently by detecting edges and therefore was 98% accurate for profile faces, specifically. However, generalized accuracy is quite lower than this.

The Haar cascade algorithm is trained on Raspberry Pi 3 Model B+ with a 64-Bit quad processor and the raspberry pi camera is used to capture the faces real time for the recognition process. The training procedure on Raspberry Pi is shown in Figure 8. The algorithm is capable of recognizing celebrity facial images (shown in Figure 9) in addition to the real time images (from video) captured through interfaced camera. Haar method was found to be better in the cases where lighting was differentiated with shade. It is efficient to recognize frontal faces. It is because the frontal image has bridge of the nose and cheek bones which cast shadows on the rest of the face leading to difference in shading. Haar method lacks efficiency in terms of detecting edges.

The accuracy of Haar method is limited in comparison to the HOG method. Haar cascade provides accuracy of about 50 percent whereas in case of HOG method the accuracy can be increased up to 80 percent for a versatile set of images.

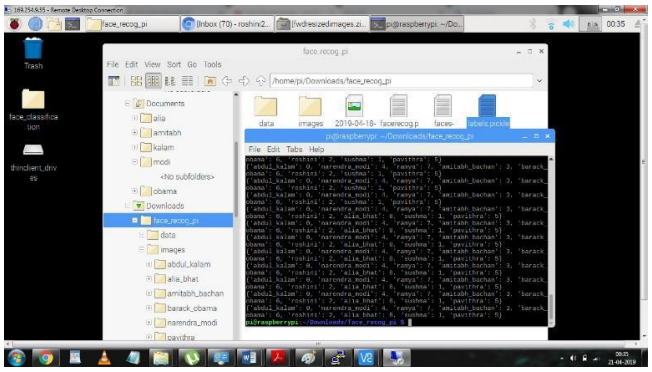

Fig 8. The training process of the Haar cascade method on raspberry pi is shown.

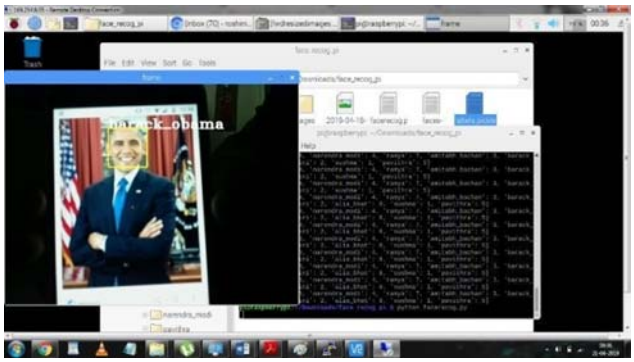

Fig 9. Raspberry pi detection of Barrack Obama using Haar method

## VI. Conclusion

Face recognition is one of the biometric which authenticates the identity of the person non-invasively. Two algorithms of face recognition that is Harr cascade classifier and HOG method was discussed, simulated and implemented for the user defined database. Intensive comparative analysis was carried out to find the advantages and drawbacks of each method for different orientation, lighting condition of faces. Haar features were better at differentiating light from shade but not orientation of edges. Therefore, Haar is better suited to detect frontal faces. A frontal image of the face has the bridge of the nose and cheek bones which casts shadows on the rest of the face leading to difference in shading. HOG method, on the other hand, performs well with detecting edges and hence better suited to detect shapes. Therefore, profile faces can be detected with better accuracy using this method. The outline of features such as the nose and chin need to be taken into consideration for this type of recognition process. Haar features (and other wavelets) are "texture features" as opposed to HOG. They are good at detecting similar texture but lack orientation information. HOG provides more orientation details.

The main advantages that the Haar method provides is that it works almost real time on our traditional CPUs due to its simple architecture and it also detects faces at different scales. This scaling can be controlled and tuned according to the specific requirements. As we scale down, the computational intensity increases. But some of the limitations of this method includes the fact that it performs poorly for non-frontal and occluded faces. Furthermore, the number of false predictions is numerous that is it can provide accuracy of up to 50% whereas HOG accuracy could be up to 80%.

HOG method offers advantages like being able to detect multiple faces in group images, being non sensitive to uniform change in luminosity and being a light weight model. But it has its own limitations. Though it can handle more occlusion than the Haar method, it cannot detect faces that are small or too far away. Although it is better than Haar for non-frontal faces, it still doesn't perform well enough for extremely non-frontal faces and side profiles.

HOG offers better accuracy compared to Haar with a considerably smaller sized dataset for training. But Haar is more compatible with Raspberry-pi than HOG, because HOG requires more computation which is not supported by Raspberry Pi. Hence HOG method was implemented on laptop with Nvidia Geforce GPU support. Both methods have different characteristics and therefore may be chosen accordingly for specific applications.

Further work may be carried out in line with bigger/versatile database. The target hardware may also be explored with higher performance. Integration of both algorithms can give a robust solution for facial recognition problem.

## ACKNOWLEDGMENT

The authors would like to thank the management of BMS Institute of Technology and Management for their continuous support and encouragement.

## REFERENCES

[1] Paul Viola and Michael Jones, Rapid Object Detection using a Boosted Cascade of Simple Features,2001.

[2] Mathworks, "Train a Cascade Object Detector" Mathworks, 2017

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learn-ing and an application to boosting. In Computational Learning Theory: Eurocolt '95, pages23–37. Springer-Verlag, 1995

[4] Rekha N, Dr .M.Z.Kurian, Face Detection in Real Time Based on HOG, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3 Issues 4, April 2014

[5] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection", CVPR, pp. 886-893, Vol. 1,2005.

[6] SHU Chang, DING Xiaoqing, FANG Chi, Histogram of the oriented Gradient for Face Recognition, TSINGHUA SCIENCE AND TECHNOLOGY, ISSN 1007-0214 15/15 pp216-224 Volume 16, Number 2, April2011.

[7] Harshal V. Khodaskar, Shashank Mane "Human Face Detection & Recognition Using Raspberry Pi" International Conference on Science and Engineering for Sustainable De-velop--ment International Journal of Advanced Engineering, Management and Sci-ence(ICSESD-2017).

[8] R. Chellappa, C.L. Wilson, and S. Sirohey, "Human and machine recognition of faces: A survey," Proc. IEEE, vol. 83, pp. 705–740, 1995.

[9] K. H. Wanjale, Amit Bhoomkar, Ajay Kulkarni, Somnath Gosavi, V.I.I.T., Pune "Use of Haar Cascade Classifier for Face Tracking System in Real Time Video". International Journal of Engineering Research & Technology (IJERT), April2013.

[10] Sheenamol Yoosaf, Anish M P, "Face Detection & Smiling Face Identification Using' Ada-boost & Neural Network Classifier", International Journal of Scientific & Engineering Re-search, Volume 4, Issue 8, August2013.

[11] W.Zhao, R.Chellapa, P.J Philips, A. Rosenfeld, "Face Recognition: A Literature Survey", ACM Computing Surveys, vol. 35, no. 4, December 2003, pp.399-458.

[12] A. Albiol, D. Monzo, A. Martin, J Sastre, "Face recognition using HOG–EBGM". Elsevier- Pattern Recognition Letters, 29(10), 1537–1543, 2008.

[13] Déniz, O., Bueno, G., Salido, J., & De la Torre, F. "Face recognition using Histograms of Oriented Gradients".Elsevier- Pattern Recognition Letters, Volume 32, Issue 12, Pages 1598-1603,1 Sep 2011.

[14] Mukesh B.Rangdal, Dinesh B. Hanchate, "Animal Detection Using Histogram Orinted Gradient", IJRITCC, Volume: 2 Issue: 2 ISSN: 2321-8169 -178-183, February 2014.

[15] Paul Viola and Michael Jones, "Robust Real-time Face Detection". 8th IEEE International Conference on Computer Vision (ICCV'01), 2001.