SCAD
COLLEGE OF ENGINEERING & TECHNOLOGY
ISO CERTIFIED | Affiliated to Anna University, Chennai
Recognized under Section 2(f), 12(B) of the UGC Act, 1956
Cheranmahadevi, Tirunelveli - 627414

# PROCEEDINGS

## of the

# International Conference on Trends in Electronics and Informatics (ICOEI 2019)

## 23-25, April 2019

## Technical Sponsors

IEEE

Proceedings of the Third International Conference on Trends in Electronics and Informatics (ICOEI 2019)
DVD Part Number: CFP19J32-DVD; ISBN: 978-1-5386-9438-1

# *Attendance Management System using Face Recognition and Spoof Detection*

Sheen Liz Shaju
*Electronics and Telecommunication,*
*Fr. C. Rodrigues Institute of Technology,*
*Vashi, India*
sheenlizshaju@gmail.com

Pratik Shetty
*Electronics and Telecommunication,*
*Fr. C. Rodrigues Institute of Technology,*
*Vashi, India*
pratikshetty@ymail.com

Akhil Singh
*Electronics and Telecommunication,*
*Fr. C. Rodrigues Institute of Technology,*
*Vashi, India*
singhakhil1061997@gmail.com

Mr. Hemant Meena
*Senior Software Engineer,*
*Zing HR,*
*Malad, India*
hemant.meena@zinghr.com

Mrs. Smita Hande
*Electronics and Telecommunication,*
*Fr. C. Rodrigues Institute of Technology,*
*Vashi, India*
smita.hande@fcrit.ac.in

***Abstract*- The most commonly used biometric identification and authentication system is face recognition. The frequent use of face recognition has raised concerns regarding biometric presentation attacks. Spoofing of identity to gain access to facilities or services is a major issue. Here, in this research work, we aim to design an attendance management system for the on-field employees, where an image (a face) and the current location of the employee will be sent from a mobile application for updating the attendance. This attendance management system should also detect spoofs when recognising the face to mark the attendance. Deep metric learning which is a combination of face recognition and deep learning, and Support Vector Machine (SVM) which is a machine learning classification algorithm are used to detect and recognize a face. A proposed method called average brightness method is used to differentiate between a genuine face and a spoofed image of a face. The attendance will be taken on an individual basis through an android application, which uses volley library, flask framework and firebase database; and each day's attendance will be updated to a portal after authentication of the face and current location.**

***Keywords*—attendance management system; face recognition; spoof detection; machine-learning; Support Vector Machine; deep metric learning; Android application; flask framework; firebase database.**

## I. INTRODUCTION

Attendance management is an important task in any organization, to check the regularity of individuals. This can be done using the paper or file-based approach, but most have adopted methods of automatic attendance using some biometric techniques, face recognition is one of the most successful applications of image processing and analysis.

The facial recognition is a process achieved only after detecting the face. There are many existing methodologies for the detection of a face. Some of them are skin colour based, characteristic or feature based, etc. OpenCV, aimed at real-time computer vision, provides pre-trained face detection classifiers like Haar (Cascade) Classifier [1] and Local Binary Patterns (LBP) Classifier [2]. There are three steps to computer coding facial recognition: data gathering, train the recognizer and recognition. OpenCV has built-in face recognizers [3]: EigenFaces, FisherFaces, Local Binary Patterns Histogram (LBPH). Though all the mentioned algorithms work almost real-time on CPU, have a simple architecture and detect faces at different scales, a major drawback is that these methods give a lot of false predictions, don't work on non-frontal images or don't work under occlusion [3]. The deep learning-based facial embeddings that are used in the proposed system are both highly accurate and capable of being executed in real-time [5].

A disadvantage of face recognition used in attendance management system is spoof attacks which can give access to important secure facilities or services. Existing methods of face spoof detection, particularly methods using texture features, commonly used features (e.g., LBP) that can capture facial details and differentiate one subject from the other, that is face recognition. When the same features are used to differentiate a genuine face from a spoof face, they either contain abundant information or unwanted information, which limits the generalization ability of existing methods [6].

The attendance management system that is implemented in this paper uses face recognition to automatically mark the attendance of employees working on-field in offices without a third person's intervention. This attendance is recorded by using an android application that captures an image of the employee, detects the face in that image, checks for liveness of the image (spoof detection) then compares the detected face with an already created database, verifies the location of the image taken and marks the attendance.

In the proposed system, the first stage of processing is face detection which is achieved using HOG (Histogram of Oriented Gradients) face detector [4]. The second stage that includes face recognition is done using dlib face recognition tool [7]. And lastly, spoof detection is executed using a proposed method called average colour method, which is used to discriminate spoofed faces from genuine ones. Image uploaded by the employee on the Android application is sent to a server. Once the image/images, as per requirement, hits the server, the following functions such as adding a new entry to the database, recognizing the face to identify the employee, differentiating genuine faces from the spoofed ones, creating and training the required models for machine learning, verifying location, etc. takes place at the back end. A corresponding response is reverted to the android application and the attendance portal is updated.

This research work aims to create an efficient, time-saving, low-cost, Smart Attendance System using Image Processing, and further managing the records using a portal.

The remainder of the paper is organized as follows: Section II discusses the proposed approach to an attendance management system. Section III gives a brief insight into the implementation of the proposed methods of face detection, face recognition and spoof detection. Section IV and V includes the results of the implemented methods and the conclusions based on the results respectively.

## II. PROPOSED SYSTEM

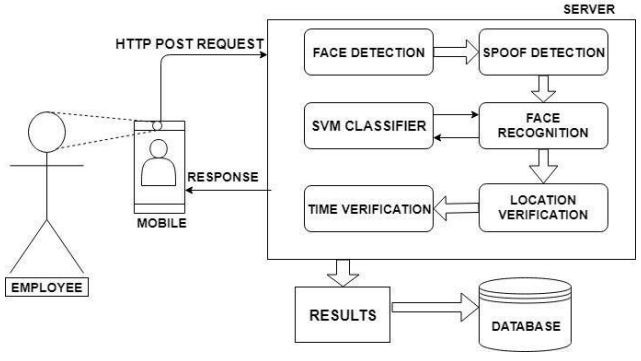


Figure 1: Basic architecture of the proposed system.

Figure 1 shows the basic architecture of the proposed attendance management system. Image captured from the android application is sent to a server using HTTP post request.

All the processing like face detection, recognition, spoof detection and location and time verification, takes place in the server. The authentication of the above processes is executed using a trained model that is generated by the python code present in the server. As a response, the mobile user, that is the employee, is notified about the successful authentication through the application and the attendance will be marked if the employee's authentication is carried out properly.

### *A. Face recognition with OpenCV, Python and deep learning:*

Facial recognition in this attendance management system is achieved using deep learning based facial embeddings which are both highly accurate and capable of being executed in real-time. Deep learning and face recognition work together using a technique called deep metric learning. This technique trains a network to accept a single input image and gets a real-valued feature vector as the output. Facial recognition via deep metric learning involves a triplet training step [5].

In order to perform face recognition with Python and OpenCV, two additional libraries: dlib and face_recognition are installed. The dlib library, maintained by Davis King, contains the implementation of "deep metric learning" which is used to construct face embeddings used for the actual recognition process. The face_recognition library, created by Adam Geitgey, is based on dlib's facial recognition functionality, making it easier to work with.

The basic pipeline for recognising faces might work in the following way: find the face in the image, analyse facial features, compare against known faces and finally make a prediction. In other words, several machine learning algorithms are chained together. Each step of face recognition is discussed below [8]:

#### *1. Finding all the faces:*

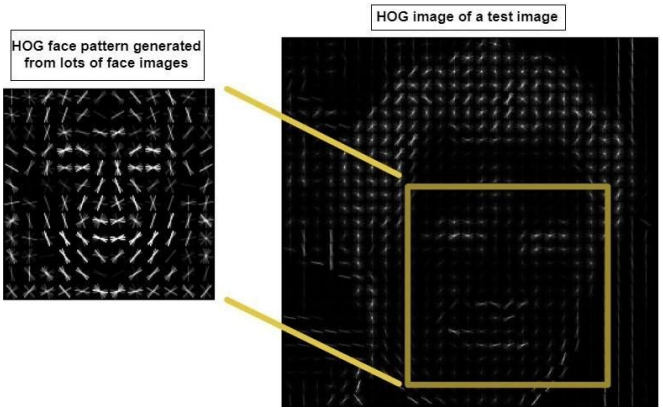


Figure 2: HOG representation [8]

This is the first step in the pipeline. Faces have to be located before trying to tell them apart. A method called Histogram of Oriented Gradients (HOG) finds faces in a black and white image, colour data isn't required to find faces.

Every 16x16 pixels is compared to the surrounding 16x16 pixels and is replaced with the strongest gradient, an arrow that shows the flow from light to dark pixels across the entire image. This gives a simpler representation that captures the basic structure of a face. The newly generated HOG pattern of the test image and a known HOG pattern that was extracted from a bunch of other training faces are compared. The region that is found to be the most similar is marked as the location of the face, that is, the face is detected as shown in figure 2. The HOG representation captures the major features of the image regardless of image brightness [9].

*2. Encoding faces:*

The encodings of a face are found to measure an unknown face by extracting a few basic measurements like the size of each ear, the spacing between the eyes, the length of the nose, etc. and find the known face with closest measurements. The most accurate approach to do this is to let the computer figure out the measurements to collect by itself. An image of a face, a complicated raw data is reduced into a list of computer-generated numbers. This is achieved by training a Deep Convolutional Neural Network. The neural network is trained to reliably generate 128 measurements for each person. These 128 measurements of each face are called embeddings. Thus, reducing complicated raw data into a list of computer-generated numbers. In this work, pre-trained network by OpenFace is used to get the 128 measurements for each face.

*3. Creating and predicting labels from the encoding:*

Support Vector Machines (SVM) is a discriminative classifier that attempts to draw a straight line separating the two sets of data, which will be used to create a model for classification [10]. Given a set of training data (here images of faces), each marked as belonging to a category, an SVM training algorithm builds a model that assigns new sets to one category or the other. An SVM model is a representation of a set of data as points in space, mapped so that the sets of the separate categories are divided by a clear gap that is as wide as possible. New sets are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

There are different implementations of this algorithm, Linear SVC (Support Vector Classifier) being one of them. The methods: predict(X) is used to predict class labels for samples in X and fit (X, y[,same_weight]) is used to return the mean accuracy on the given test data and labels [11]. The SVM classifier is trained to take measurements from a new test image and tell which known person is the closest match.

*B. Spoof Detection:*

A spoof attack is an attempt to gain access to someone else's privileges by using illegitimate means. Some examples of spoof attacks are print attack, replay/video attack, 3D mask attack, etc. [12].

To detect spoofing attacks, the average brightness of the processed version of the gray scaled image and that of the hue component of the HSV colour model of the image are used. Standard values for both these variables (the average brightness) are set, which decides whether the face is genuine or spoofed.

## III. IMPLEMENTATION

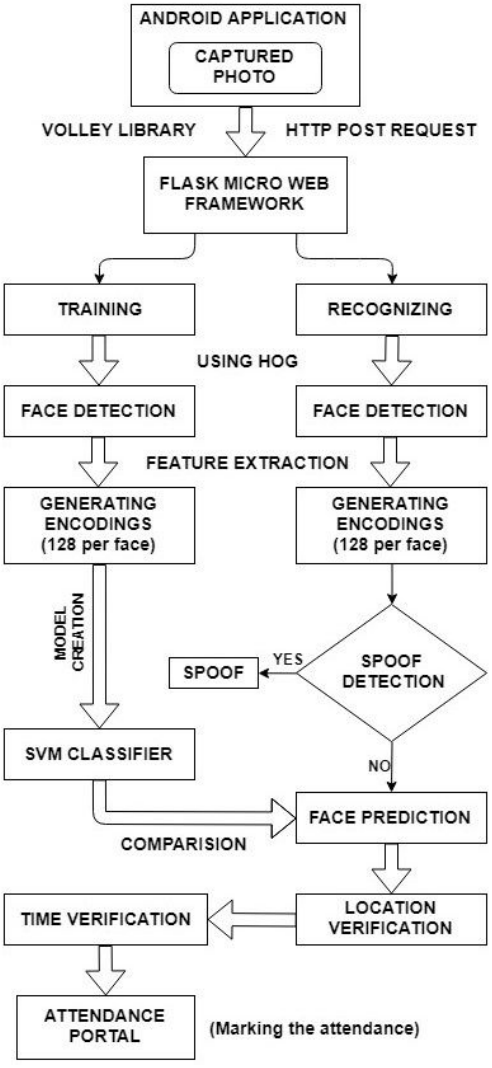


Figure 3: Block diagram of the proposed system

In the proposed attendance management system, the attendance is marked using face recognition along with spoof detection for better authentication. Figure 3 shows the block diagram of the implemented work. It begins by capturing a photo through the android application. The captured image is

sent to the server (the backend of the application) by exerting volley library via HTTP post method. The HTTP request is received by the Flask micro web framework. The server-side processing is done by Flask web server and web application where the python scripts are executed.

For the application to recognize and predict the face in a captured image, the machine should first undergo training. For training, 20 images of each employee are obtained which will be further used to create the model. The face is detected from the captured image using HOG. The isolated face is centred by aligning, posing and projecting by marking 68 specific points called landmarks. Features that the computer considers prominent are extracted, thus generating 128 encodings per detected face. These 20x128 encodings are classified into a single label using SVM Classifier. Similar such labels are generated for each employee. All these encodings along with the label are saved in a CSV file and a model is devised. Now the machine is ready for recognition of faces

Once the Flask micro web framework receives the captured photo of the employee, first the face is detected using HOG. The 128 encodings are generated.

Spoof detection is carried out using the average brightness algorithm as explained in figure 4. The image is converted from BGR to GRAY. The histogram of the gray scaled image is equalized using Contrast Limited Adaptive Histogram Equalization. This image is blurred and smoothened using bilateral filtering to remove noise while preserving the edges. Each bit of the array of the blurred image is inverted using bitwise_not operation. The image is also converted from BGR to HSV, which is split into its components – hue, saturation, value (h, s, v). The average brightness of the gray scaled image and that of the hue component of the image is calculated respectively. Standard values for both the variables are set, which decides whether the face is genuine or spoofed.

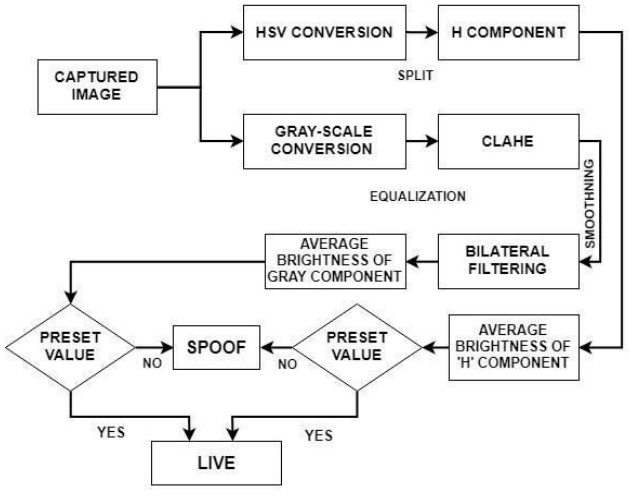


Figure 4: Average brightness algorithm

If the image is a spoofed one, the process is terminated by displaying that it is a spoof. On the other hand, if a genuine face is detected, the person's name (label) is predicted using the model created during training. The attendance portal is updated after location and time verification. Thus, marking the attendance of the employee.

## IV. RESULTS

This section shows the results of the proposed system which includes face detection, spoof detection and face recognition.

### A. *Face Detection*

Figure 5 shows face detection on a test image. The rectangle box shows the detected region of the face. 128 encodings for this region is generated.

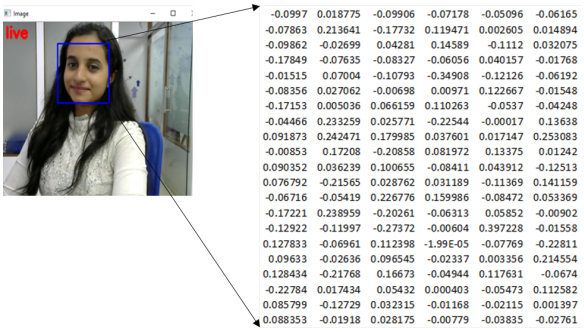


Figure 5: Face detection and the generated encodings of a test image.

### B. *Spoof Detection*

Figure 6 shows spoof detection on two test images. The first image is a spoofed test image of print attack type. This image is detected as "fake". The second is a genuine test image. This is detected as "live" by the algorithm.

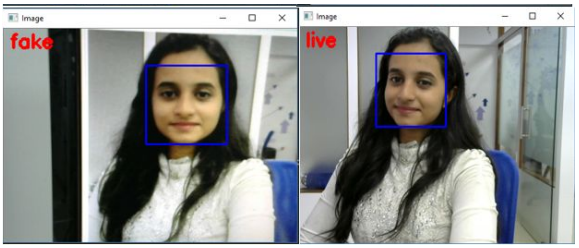


Figure 6: Spoof detection.

### C. *Face Recognition*

Figure 7 shows face recognition of a test image. The predicted label is displayed. the outcome of face recognition after predicting the label.

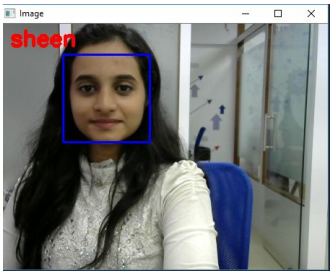


Figure 7: Face Recognition (prediction).

## V. CONCLUSION

Using HOG for face detection worked extremely well. It is the fastest method on the CPU. It works very well for frontal and slightly non-frontal faces and under small occlusion. This method can detect a face under most circumstances. The detected region often excludes part of the forehead and even part of the chin. This method does not work very well under substantial occlusions, side face and extreme non-frontal faces, like looking down or up.

Even after detecting the face, the face could be tilted or away from the actual required position of the eyes, nose and mouth. This could make it difficult to extract features and recognize the face. For this purpose, face landmark estimation is employed in which the face undergoes affine transformation (rotated, scaled, sheared, etc.) to centre the eyes and mouth to the best possible way.

The proposed average brightness method for spoof detection uses a pre-set threshold value for the standard brightness level of the surrounding to compare with the calculated average brightness of the test image. This works well for a specific brightness intensity level. When this level changes according to change in environment, the efficiency of the algorithm reduces. This can be rectified by making the threshold value dynamic. The algorithm is limited to print attack type of spoofs. It is not effective on video attacks, replay attacks and 3D mask attack.

Face recognition is achieved by using the dlib face recognition library of OpenCV. When presented with a face image, the tool correctly identifies if it belongs to the same person or a different person, that is the prediction is accurate up to 99.38% of the time.

## *REFERENCES*

[1] S. Emami and V. P. Suciu, “Facial Recognition using OpenCV” *Journal of Mobile, Embedded and Distributed Systems*, vol. 4, no. 1, pp. 38-43, March 2012. [Online]. Available: http://www.jmeds.eu/index.php/jmeds/article/download/Facial_Recognition_using_OpenCV. [Accessed March 11, 2019].

[2] Laura Sánchez López. *Local Binary Patterns applied to Face Detection and Recognition.* Final research project. Universitat politècnica de catalunya; 2010.

[3] L. Dinalankara, “Face Detection & Face Recognition Using Open Computer Vision Classifies,” *ResearchGate*, 2017. [Online]. Available: https://www.researchgate.net/publication/318900718_Face_Detection_Face_Recognition_Using_Open_Computer_Vision_Classifies. [Accessed: 14- Mar- 2019].

[4] V. Gupta, “Home,” *Learn OpenCV*, 22-Oct-2018. [Online]. Available: https://www.learnopencv.com/face-detection-opencv-dlib-and-deep-learning-c-python/. [Accessed: 09-Feb-2019].

[5] A. Rosebrock, "Face recognition with OpenCV, Python, and deep learning - PyImageSearch", *PyImageSearch*, 2019. [Online]. Available: https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/. [Accessed: 28- Feb- 2019].

[6] Di Wen, Hu Han and A. Jain, "Face Spoof Detection With Image Distortion Analysis", *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 4, pp. 746-761, 2015. Available: 10.1109/tifs.2015.2400395.

[7] *dlib C Library*. [Online]. Available: http://www.dlib.net/face_detector.py.html. [Accessed: 09-Feb-2019].

[8] A. Geitgey, "Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning", *Medium*, 2019. [Online]. Available: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. [Accessed: 28- Feb- 2019].

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, USA, 2005, pp. 886-893, doi: 10.1109/CVPR.2005.177.

[10] N. Cristianini and J. Shawe-Taylor, “Support Vector Machines,” in *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*, Cambridge, UK: Cambridge University Press, 2000, ch. 6, pp. 93-124.

[11] "sklearn.svm.LinearSVC — scikit-learn 0.20.2 documentation", *Scikit-learn.org*, 2019. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html. [Accessed: 28- Feb- 2019].

[12] Zinelabidine Boulkenafet, Jukka Komulainen, and Abdenour Hadid,” FACE ANTI- SPOOFING BASED ON COLOR TEXTURE ANALYSIS,” in IEEE International Conference on Image Processing (ICIP), Quebec City, CAN, September 27-30, 2015.