Smart Door Lock Using Simultaneous Face Liveness and Face detection Algorithms

K Shanmukha
VR Siddhartha Engineering College(A)
Vijayawada, India
kshanmukha1501@gmail.com

Shaik Sameer
VR Siddhartha Engineering College(A)
Vijayawada, India

Dr G.Anuradha
Associate Professor, CSE Dept
VR Siddhartha Engineering College(A)
Vijayawada, India
ganuradha@vrsiddhartha.ac.in

***Abstract*-- The rapid growth of technology in the modern society has raised many questions on the terms like security and privacy. It may be either data security or home security. Doors act as defense mechanisms against burglars. Therefore, door security system is very much needed to enhance home security. Authentication is a key factor which helps for the identification of authorized people and helps in eradicating fraudulent activities, robberies, and many other social crimes. Most of the crimes are due to the vulnerabilities in the door locking systems which can be easily accessible by the outsiders. To overcome this challenge, this research work utilizes the advanced Internet of Things (IoT) and AI technology to improve home security. It is essential to let the right people inside like friends and family and keep the intruders out and to ensure this, a facial recognition-based door lock system is used. In addition to houses, this system can be implemented at the workplace, college campus, etc. Facial recognition is the easiest way to unlock the door as one doesn't need to make any physical efforts to open the door lock and just needs to look into the camera. The electromagnetic lock on the door will unlock if the image is present in the database. A security alert message in the form of an email is sent to the owner of the house in the case of an intruder. *But* there always lies a chance of spoofing of an authentic user t manipulate the recognition algorithm. A secure system needs Liveness detection in order to guard against such spoofing. In this work, face liveness detection approaches are categorized based on the various types of techniques used for liveness detection. This categorization helps understanding different spoof attacks scenarios and their relation to the developed solutions. A review of the works regarding face liveness detection works is presented which is done using a combination of Blinking analysis and Texture Analysis of any presented face in front of the camera.**

▶ ***Keywords---Home Automation, Facial recognition using AI, Liveness Detection, Local Binary patterns, Blinking Analysis, Face Recognition***

1. INTRODUCTION

The general public has immense need for security measures against spoof attack. Biometrics is the fastest growing segment of such security industry. Some of the familiar techniques for identification are facial recognition, fingerprint recognition, handwriting verification, hand geometry, retinal and iris scanner. Among these techniques, the one which has developed rapidly in recent years is face recognition technology and it is more direct, user friendly and convenient compared to other methods. Therefore, it has been applied to various security systems. But, in general, face recognition algorithms are not able to differentiate 'live' face from 'not live' face which is a major security issue. It is an easy

way to spoof face recognition systems by facial pictures such as portrait photographs. In order to guard against such spoofing, a secure system needs liveness detection. Biometrics is the technology of establishing the identity of an individual based on the physical or behavioural attributes of the person. The importance of biometrics in modern society has been strengthened by the need for large-scale identity management systems whose functionality depends on the accurate deduction of an individual's identity on the framework of various applications. Some examples of these applications include sharing networked computer resources, granting access to nuclear facilities, performing remote financial transactions or boarding a commercial flight . The main task of a security system is the verification of an individual's identity. The primary reason for this is to prevent impostors from accessing protected resources. General techniques for security purposes are passwords or ID cards mechanisms, but these techniques of identity can easily be lost, hampered or may be stolen thereby undermine the intended security. With the help of physical and biological properties of human beings, a biometric system can offer more security for a security system.

Liveness detection has been a very active research topic in fingerprint recognition and iris recognition communities in recent years. But in face recognition, approaches are very much limited to deal with this problem. Liveness is the act of differentiating the feature space into live and non-living. Imposters will try to introduce a large number of spoofed biometrics into system. With the help of liveness detection, the performance of a biometric system will improve. It is an important and challenging issue which determines the trustworthiness of biometric system security against spoofing. In face recognition, the usual attack methods may be classified into several categories. The classification is based on what verification proof is provided to face verification system, such as a stolen photo, stolen face photos, recorded video, 3D face models with the abilities of blinking and lip moving, 3D face models with various expressions and so on. Anti-spoof problem should be well solved before face recognition systems could be widely applied in our daily life.

Raspberryi-pi(Model 4B) is used to implement both the Liveness detection module and Face Recognition module in this project. The micro-controller board is equipped with an IR Sensor which readily detects any approaching face(i.e; within 10 cm of the sensor). When motion is detected in front of the camera, the facial recognition modules get activated.

There are many algorithms have been developed for face recognition algorithm, including appearance based , active appearance , support vector machines (SVM) , Bayesian model , deep learning neural network , and texture based . Face representation using Deep Convolutional Neural Network (DCNN) embedding is the method of choice for face recognition DCNNs map the face image, typically after a pose normalisation step , into a feature that should have small intra-class and large interclass distance. There are two main lines of research to train DCNNs for face recognition. Some train a multi-class classifier which can separate different identities in the training set, such by using a softmax classifier , and the others learn directly an embedding, such as the triplet loss . Based on the large-scale training data and the elaborate DCNN architectures, both the softmax-loss-based methods and the triplet-loss-based methods can obtain excellent performance on face recognition. However, both the softmax loss and the triplet loss have some drawbacks. In this paper, we propose an Additive Angular Margin Loss (ArcFace) to further improve the discriminative power of the face recognition model and to stabilise the training process. As illustrated in Figure 2, the dot product between the DCNN feature and the last fully connected layer is equal to the cosine distance after feature and weight normalisation. We utilise the arc-cosine function to calculate the angle between the current feature and the target weight. Afterwards, we add an additive angular margin to the target angle, and we get the target logit back again by the cosine function. Then, we re-scale all logits by a fixed feature norm, and the subsequent steps are exactly the same as in the softmax loss.

A. MOTIVATION

Visualizing the future of Home Automation by implementing an secured face detection system for unlocking doors. Using advanced Artificial Intelligence techniques to classify and recognize faces which aims at high security. Implementing anti-spoofing techniques using Heuristic and Texture analysis on faces in order to accurately differentiate real and fake faces(images, video spoofing, etc…).Implementing an effective face-recognition algorithm which can recognize faces with high accuracy which provides an fast and interaction minimal method for unlocking doors. (i.e. no need for any human action to unlock doors).

Realizing the potential of the technology in future home/office door lock solutions by implementing the project on live datasets with (<95%) accuracy.

B. PROBLEM STATEMENT

Build, Design and implement an intuitive and Advanced IoT/AI based Lock Management System (LMS) which aims at improving upon current home automation solutions whilst simultaneously providing high-end security and reliability to its users. Also verify the use of simultaneous face-liveness and face-recognition algorithms as an effective solution for home/office door lock management scenarios.

C. SCOPE

This project is developed to provide an intelligent solution for door lock in the SCHOLARSHIP SECTION of Velagapudi Ramakrishna Siddhartha Engineering College, Kanuru and can further be implemented onto other areas of interest of the college.

D. OBJECTIVES

The main objectives are:

a. Implement a reliable Door Lock solution using Raspberry-pi and IR sensor and PI-cam
b. Provide a highly accurate face-recognition algorithm which can be conveniently used in multiple scenarios with high accuracy(>95%).
c. Prevent the vulnerability of the proposed system i.e; implement an anti-spoofing Face-liveness detection module which can effortlessly differentiate between live and spoofed faces (either photo or video).

ADVANTAGES:

a. By using this system, the process of door lock management becomes extremely convenient and effortless for the user.
b. Avoid using keys, which can be easily forgotten in the amidst of day-to-day activities
c. Actively notifies the owner if anyone tries to unlock the door via face-recognition by sending the owner, a sms text with the exact time of event occurred.

2. PROJECT DESIGN

In this section, the hardware design, software design, and algorithm design will be presented in more details.

2.1 ***Hardware Design***

Figure 1 shows the proposed block diagram of face recognition system using Raspberry Pi. The Raspberry Pi is connected to the camera module which are shown in Figure 3. Raspberry Pi has the same processing capability of a single core processor with built-in graphics which can support up to 1080p video standard using its High Definition Media Input (HDMI) port. The ultrasonic sensor is used to detect the human presence. The relay circuit is connected to Raspberry Pi and provides an interface to the high voltage magnetic lock. The status could be shown using LED indicators and/or further send SMS using GMS module, or email using internet connection, or as part of smart home system as proposed in . Finally, the overall power required is less than 1 A, so that a compact power supply, i.e. smartphone charger, could be used as the power supply.

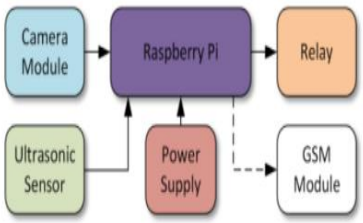


Figure 1. Block Diagram of Proposed Hardware

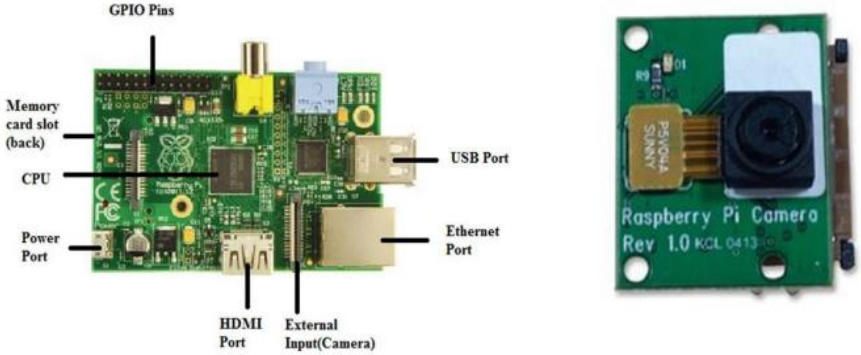


Figure 2. Raspberry Pi and Camera Module

In this project, the external input port was attached with a camera made for Raspberry Pi called the Rpi Camera Board. It features a 5MP camera with a 1080p resolution and is capable of delivering high data rates of pictures and videos. Although other cameras also work together with the Raspberry PI board, some developers argued that using the RPi camera board provides faster processing compared to cameras that are attached using the USB port as described in . This port shares the power with the other input output modules such as a keyboard and mouse, so an additional device requires an USB splitter or additional powered hubs to connect them.

2.2. Software Design

Figure 3 shows the flowchart of the proposed face recognition system. First, it reads ultrasonic sensor to detect the human presence, which could be set to check every second (configurable). If the human presence is detected, then the camera will capture the face image. The face detection routine will

localize and segment the face region only. The face image is then fed in to the face recognition routine. If the recognized face is detected, the system will unlock the door by turning of the magnetic lock. After 30 seconds (configurable), the system will lock again the door by turning on the magnetic lock. The system will then start from the beginning.

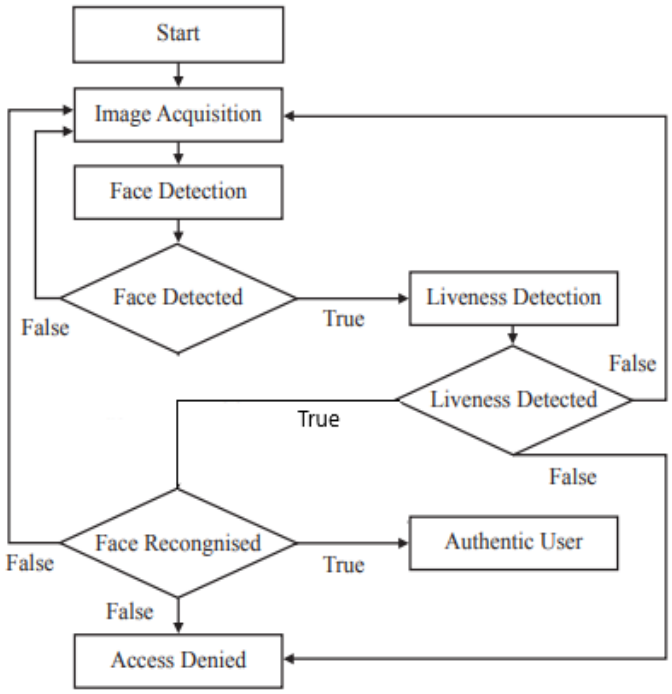


Figure 3. Flowchart of Face-Detection and Face-Recognition system

2.3 Algorithm Design and methodology

2.3.1 Face-Recognition Algorithm

The advantages of the proposed ArcFace can be summarised as follows:

Engaging: ArcFace directly optimises the geodesic distance margin by virtue of the exact correspondence between the angle and arc in the normalised hypersphere. We intuitively illustrate what happens in the 512-D space via analysing the angle statistics between features and weights.

Effective: ArcFace achieves state-of-the-art performance on ten face recognition benchmarks including large-scale image and video datasets.

Easy: ArcFace only needs several lines of code as given in Algorithm 1 and is extremely easy to implement in the computational-graph-based deep learning frameworks, e.g. MxNet , Pytorch and Tensorflow . Furthermore, contrary to the works in , ArcFace does not need to be combined with other loss functions in order to have stable performance, and can easily converge on any training datasets.

Efficient: ArcFace only adds negligible computational complexity during training. Current GPUs can easily support millions of identities for training and the model parallel strategy can easily support many more identities

ARCFACE APPROACH:

The most widely used classification loss function, softmax loss, is presented as follows:

$$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}, \qquad (1)$$

where xi ∈ R d denotes the deep feature of the i-th sample, belonging to the yi-th class. The embedding feature dimension d is set to 512 in this paper following . Wj ∈ R d denotes the j-th column of the weight W ∈ R d×n and bj ∈ R n is the bias term. The batch size and the class number are N and n, respectively. Traditional softmax loss is widely used in deep face recognition . However, the softmax loss function does not explicitly optimise the feature embedding to enforce higher similarity for intraclass samples and diversity for inter-class samples, which results in a performance gap for deep face recognition under large intra-class appearance variations (e.g. pose variations and age gaps) and large-scale test scenarios (e.g. million or trillion pairs).

For simplicity, we fix the bias bj = 0 as in . Then, we transform the logit as WT j xi = kWjk kxik cos θj , where θj is the angle between the weight Wj and the feature xi . Following , we fix the individual weight kWjk = 1 by l2 normalisation. Following , we also fix the embedding feature kxik by l2 normalisation and re-scale it to s. The normalisation step on features and weights makes the predictions only depend on the angle between the feature and the weight. The learned embedding features are thus distributed on a hypersphere with a radius of s.

$$L_2 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s\cos\theta_{y_i}}}{e^{s\cos\theta_{y_i}} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}. \qquad (2)$$

Algorithm 1 The Pseudo-code of ArcFace on MxNet

Input: Feature Scale s, Margin Parameter m in Eq. 3, Class Number n, Ground-Truth ID gt.

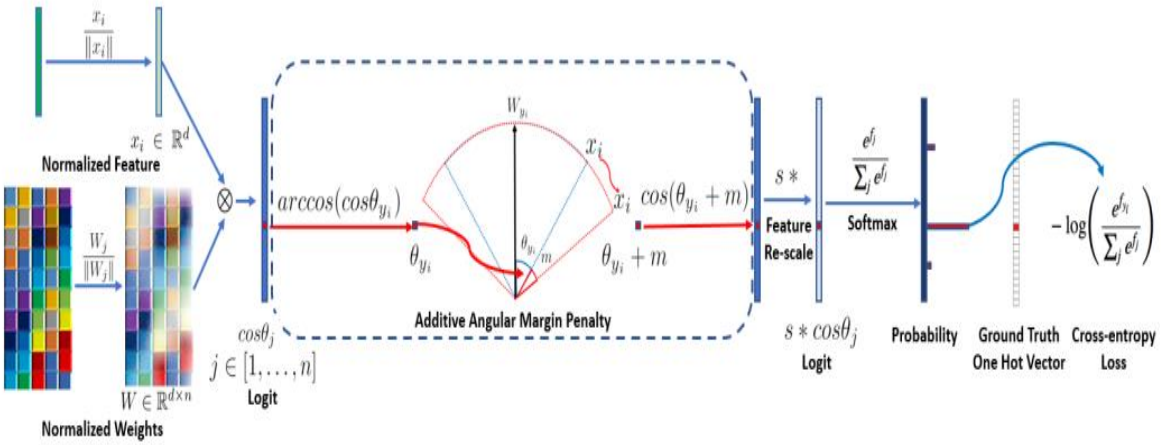
```
x = mx.symbol.L2Normalization (x, mode = 'instance')
W = mx.symbol.L2Normalization (W, mode = 'instance')
fc7 = mx.sym.FullyConnected (data = x, weight = W, no_bias = True, num_hidden = n)
original_target_logit = mx.sym.pick (fc7, gt, axis = 1)
theta = mx.sym.arccos (original_target_logit)
marginal_target_logit = mx.sym.cos (theta + m)
one_hot = mx.sym.one_hot (gt, depth = n, on_value = 1.0, off_value = 0.0)
fc7 = fc7 + mx.sym.broadcast_mul (one_hot, mx.sym.expand_dims (marginal_target_logit - original_target_logit, 1))
fc7 = fc7 * s
```

Output: Class-wise affinity score $fc7$.

As the embedding features are distributed around each feature centre on the hypersphere, we add an additive angular margin penalty m between xi and Wyi to simultaneously enhance the intra-class compactness and inter-class discrepancy. Since the proposed additive angular margin penalty is equal to the geodesic distance margin penalty in the normalised hypersphere, we name our method as ArcFace.

$$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))}+\sum_{j=1,j\neq y_i}^{n}e^{s\cos\theta_j}}. \quad (3)$$

Figure 4. Training a DCNN for face recognition supervised by the ArcFace loss. Based on the feature xi and weight W normalisation, we get the cos θj (logit) for each class as WT j xi. We calculate the arccosθyi and get the angle between the feature xi and the ground truth weight Wyi . In fact, Wj provides a kind of centre for each class. Then, we add an angular margin penalty m on the target (ground truth) angle θyi . After that, we calculate cos(θyi + m) and multiply all logits by the feature scale s. The logits then go through the softmax function and contribute to the cross entropy loss.



We select face images from 8 different identities containing enough samples (around 1,500 images/class) to train 2- D feature embedding networks with the softmax and ArcFace loss, respectively. As illustrated in Figure 4, the softmax loss provides roughly separable feature embedding but produces noticeable ambiguity in decision boundaries, while the proposed ArcFace loss can obviously enforce a more evident gap between the nearest classes.

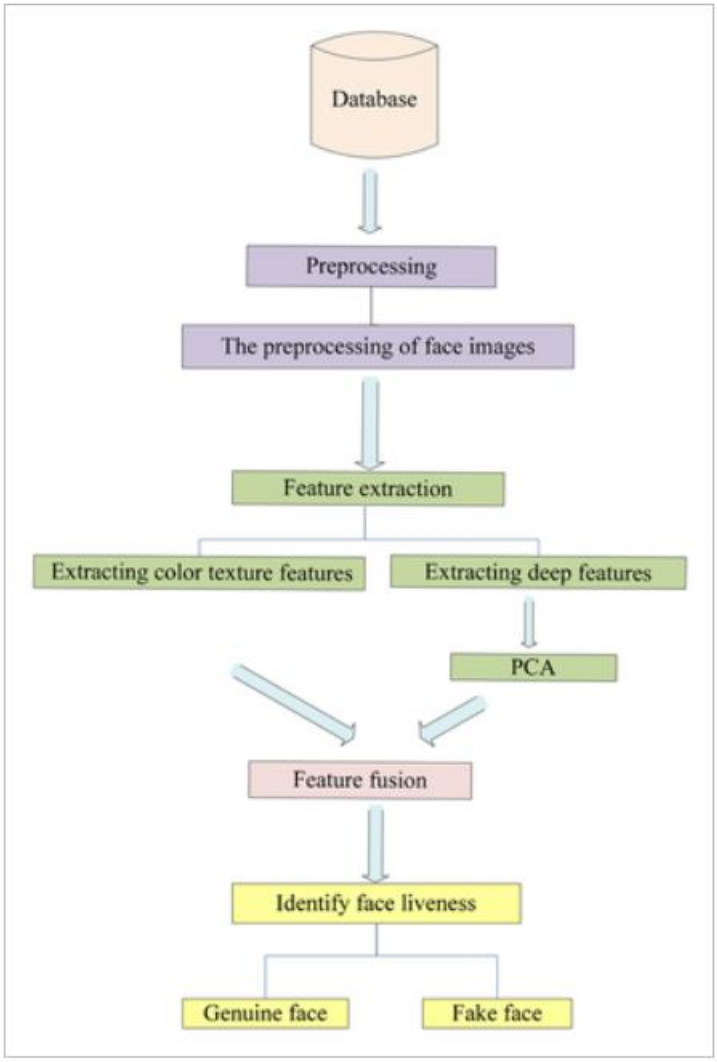
2.3.2 Face-Liveness detection algorithm

For facial liveness detection, the presented method is mainly divided into four parts: (i) *Pre-processing*. It mainly pre-processes face images in datasets to prepare for the next extraction of features.

(ii) *Feature extraction*. It consists of extracting deep features and colour texture features.

(iii) *Feature fusion*. After reducing the dimensions of deep features, two different features are fused in a parallel connection.

(iv) *Identify face liveness*. It is the SVM classifier to complete the task of distinguishing real faces from fake faces. The general steps of our algorithm have been shown in Fig. 5



2.3.2.1 Pre-processing

Pre-processing of face images

For face images in face spoofing datasets, Viola–Jones algorithm [**36**] is employed to detect face area. However, the deviation may appear when we use the Viola–Jones algorithm to locate the face region. Hence, the face alignment method Frames Per Second via Regressing Local Binary Features (FPS-LBF) [**37**] is adopted to optimise face frame. It locates face area through 68 key points. Besides, the method extracts accurate face images and retains some face background frames to improve the detection performance of the algorithm. Also, we normalise the face images into 64 × 64 pixels images.

2.3.2.2 Feature extraction

Extracting deep features

In the proposed method, ResNet proposed in [**38**] is adopted to extract deep features. As we all know, the level of features in CNN increases as the depth of the network increases, and extremely deep networks have extremely powerful expressive power. However, the deeper the network, the harder it is to train. Kaiming He *et al*. proposed building blocks in its network framework to solve the problem of gradient dispersion and degradation caused by network deepening. Therefore, the ResNet is able to train without any difficulty and implement a deeper network structure to achieve a lower training error and test error. Besides, it has fewer parameters and higher accuracy, which meets our need to face anti-spoofing. Moreover, ResNet-18 converges more quickly as compared to plaint net. The network structure of ResNet-18 is shown in Fig. **6**.

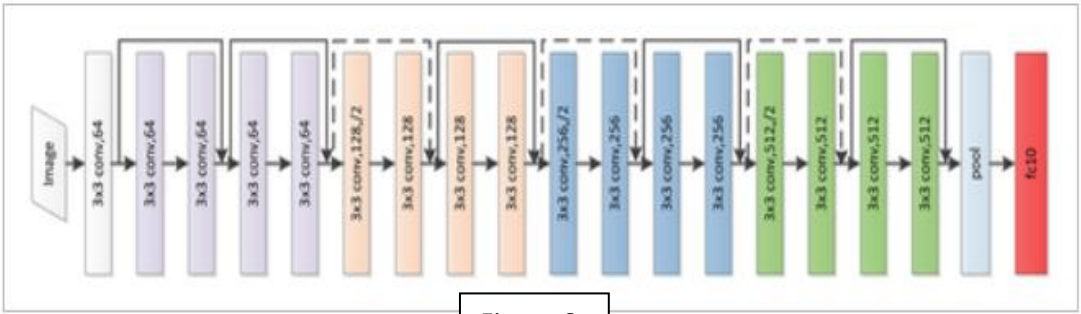


Figure 6

In ResNet, a building block is defined as:

$$\boldsymbol{y} = \boldsymbol{x} + F(\boldsymbol{x},\ \{w_l\}), \tag{1}$$

where y and x denote the output and input vectors of the layers considered. W(i) is initialised when training the network denotes the convolution kernel of the *l*th weight layer. F(x) denotes the residual function. For the example in Fig. 3, it has two layers, can be expressed as follows, where σ represents the non-linear activation function ReLU

$$F = w_2\sigma(w_1\boldsymbol{x}). \tag{2}$$

Therefore, the output
vectors Y of any weight layer can be expressed as follows:

$$\boldsymbol{Y} = \boldsymbol{x} + \sum_{i=l}^{L-1} F(\boldsymbol{x},\ w_i) \tag{3}$$

In the
ResNet, we need to back propagate the gradient of Loss *L* and compute the gradients of parameters to update *w*.Through back propagation, parameters are constantly updated. Finally, the best training

model is obtained. The ResNet is designed for various image recognition while face anti-spoofing detection is a two-category problem. Thus, for face liveness detection, we first fine-tune the designed network model. By a training set composed of genuine and spoofed images, the model is fine-tuned. Considering the difference between below four data sets, we choose Replay-Attack, CASIA-FASD and MSU MFSD to make up a new training set. The training set of the three data sets is taken 50%, respectively, to form the training set mentioned above before we start training the model. The softmax classifier is removed from ResNet-18 and the output of the last fully connected layer is changed from 1000 to 2 in ResNet-18 when training samples and fine tuning. The original input image is brought into the trained CNN to calculate the feature vector of the fully connected layer. Finally, principal component analysis (PCA) [39] is chosen to decrease the dimension of the deep feature vector obtained in the above steps. Also, the result is used in the process of the fusion of two features.

2.3.2.3 Extracting colour texture features

Owing to the great success of colour texture in face anti-spoofing detection, we use the colour local binary pattern [40] to extract colour-textured characteristic in this study. Instead of uniform LBP, we use the RI-LBP operator to extract texture features in this study. Then, for the pixel (x,y) extracted from an image, and the RI-LBP operator can be written as follows:

$$\mathrm{LBP}_{R,P}(x,\ y) = \begin{cases} \sum_{n=0}^{P-1} \delta(r_n - r_c) & \text{if } U \le 2, \\ P+1 & \text{otherwise}, \end{cases} \tag{4}$$

$$U = |\delta(r_{P-1} - r_c) - \delta(r_0 - r_c)| + \sum_{n=1}^{P} |\delta(r_n - r_c) - \delta(r_{n-1} - r_c)|, \tag{5}$$

$$\delta(x) = \begin{cases} 1 & x \ge 0, \\ 0 & x < 0. \end{cases} \tag{6}$$

For the central pixel (x,y) and its P adjacent pixels situated at a circle with radius R ($R > 0$), its colour component values are represented as r_n(n=0,1,…..p-1) , where x is the sign function. denotes different colour space. The red–green–blue [illegible] ents are transformed into the hue (H), the saturation (S) and the value (V) correspond[illegible] $H_S^{(i)}, \{i = 1 : M\}$ nce in HSV colour space. In YCbCr colour space, Cb and Cr are, respectively, the blue chrominance component and the red chrominance component while Y is the luminance component. Extracted from the M channel of S, the RI-LBP histogram is represented by . In S, the RI-LBP texture features of I are described as follows:

$$H_S = [H_S^{(1)} \dots H_S^M]. \quad (7)$$

Finally, the obtained histogram of each channel in YCbCr and HSV colour space is normalised and connected into a feature vector, i.e. a RI-LBP texture feature vector of the entire image.

2.3.2.4 Feature fusion

After getting the deep features and colour texture features, we choose the way of parallel fusion. The features to be fused are expanded and normalised. Assume CNN feature vectors and RI-LBP feature vector are, respectively, FV_{CNN} and $FV_{RI\text{-}LBF}$. Then the normalised combined feature vector is represented as follows:

$$\boldsymbol{FV}_i^n = \frac{\boldsymbol{FV}_i - \mu_i}{\sigma_i}, \ i = \text{CNN}, \ \text{RI} - \text{LBP}. \quad (8)$$

The mean and variance of each group of features are statistically analysed and obtained in the training dataset. Then two eigenvectors are connected to form a new eigenvector in which the length of the newly generated feature vector is equal to the sum length of the feature vectors to be connected. The fused feature vector is expressed as follows:

$$\boldsymbol{FV} = \left[\boldsymbol{FV}_{\text{CNN}}^n \boldsymbol{FV}_{\text{RI-LBP}}^n\right]^{\text{T}}. \quad (9)$$

2.3.2.5 Identify face liveness

After obtaining the fused eigenvectors, we use SVM [**41**] classifiers with Radical Basis Function (RBF) cores for classification. Generally speaking, the SVM is a typical two-class model. The aim of it is to map input vectors into a high-dimensional space with kernel functions and generate an optimal hyper plane which makes margin largest. An optimal hyper plane $w^{\text{T}}x + b = 0$ can be implemented after training input images, where *w* denotes weight and *b* denotes bias. Also, *y* denotes the label of the input image. Whether the classification is correct by observing whether the symbol $w^{\text{T}}x + b$ is consistent with the symbol of the class mark *y* real face or fake face, where $y = 1$ denotes real face and $y = -1$ denotes fake face. Therefore, the positive and negative of $y(w^{\text{T}}x + b)$ can be used to determine or indicate the correctness of the classification. In the proposed approach, SVM is used to learn a face spoofing classifier from training data, which is able to identify face liveness.

3. IMPLEMENTATION

The box contains the Raspberry Pi, IR Sensor, reset button, LEDs and the power input and output. The prototype will be placed beside a door and connected to a magnetic lock as shown in Figure which is turned off/unlock when the authorized user accesses the system. If an unauthorized personnel tries to access the door will stay locked and the user image will be stored in the memory. The prototype box will be placed around 1.6m from the ground and magnetic lock will be placed on top of the door. The power supply will be taken from a wall plug and these are the only input and output from the box. On the software part, Raspbian OS is used as the operating systems for Raspberry Pi. Next the Python and OpenCV library was installed for the algorithm implementation. To train the faces into the library, we use the "train.py" algorithm in the OpenCV library. The training data should be loaded into the script. These images will be captured using the code "capturepositives.py". This code will continuously capture images into the training data folder. The training data also requires a negative training images that are built into the Raspberry pi library. Sets of 10 images for each person is trained at a time and "train.py" script is executed. Sample of the training images is shown in Figure. The training data given will produce an output named "training.xml" file which contains the positive data processed into it. This process can also be done using a full fledged computer to shorten the training time. An average of 10 minutes were required using a smaller picture to process using the Raspberry Pi. Finally, ports are initialized using the terminal in root mode to access the GPIO pins. The initial set up was done using a servo but this set up can be replaced to magnetic lock or solenoid lock by simply changing the Pulse Width Modulation (PWM) pin to any other GPIO pins and initialize the pin in the script. To run the code, the terminal window on the Raspberry Pi is opened and the python code "box.py" is executed. When the red light is turned on the system is ready to execute face recognition.

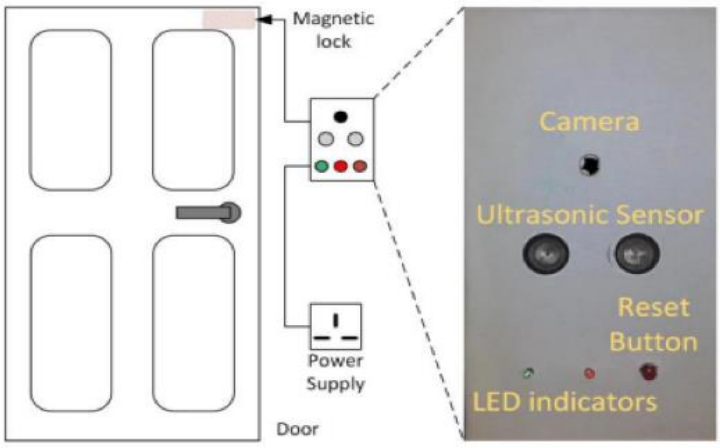


Figure 8. Complete Prototype of Face Recognition Security System using Raspberry Pi

3. *OUTPUT*

Fig. 9.Sample Images placed for training the dataset

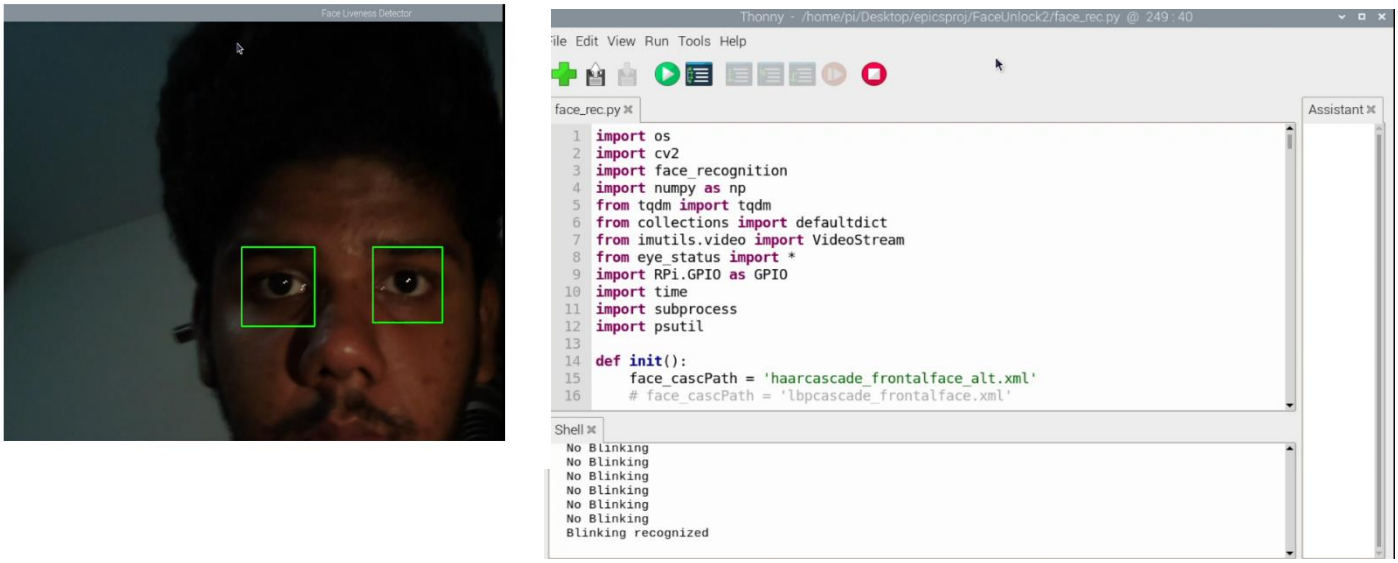


Fig 10. (On left) Face Liveness algorithm running Texture and Eye Blink analysis to evaluate the realness. (On Right) The output shell confirming the liveness of the face proceeding to the face-detection algorithm

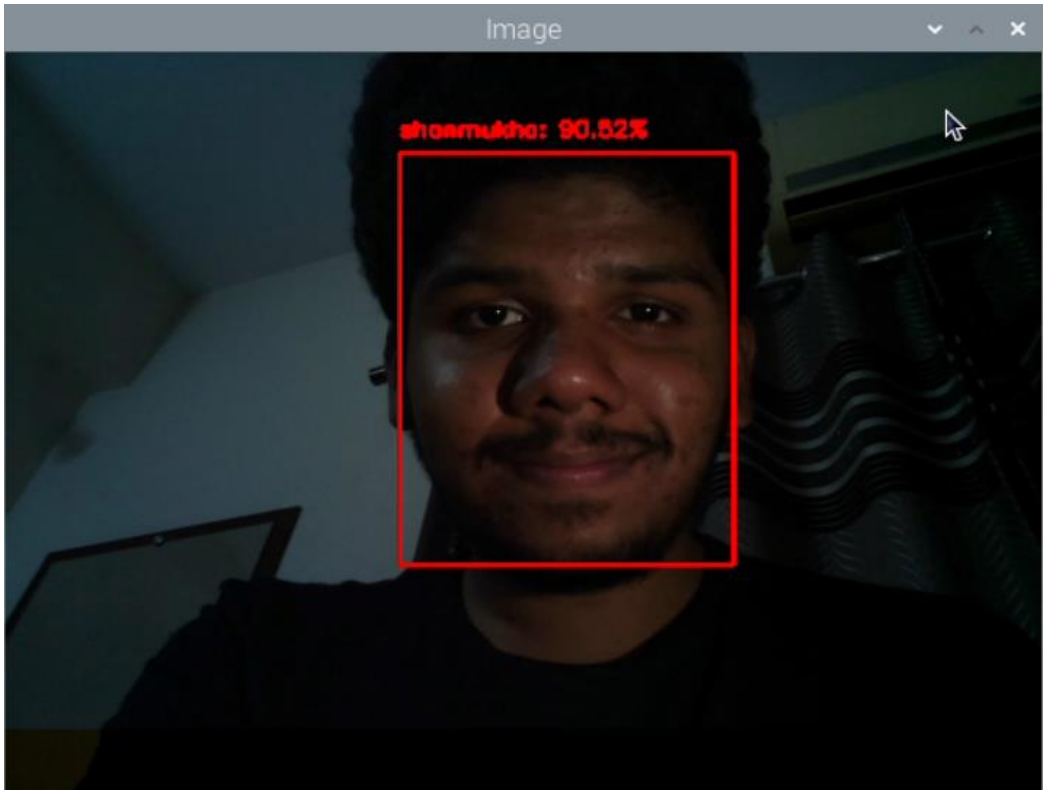


Fig 11. Face Recognition module detecting and recognizing the user's face with (>90%) accuracy.

4.1. Output of the Training Stage

Figure 9 shows the result of the training process, in which the positive eigenfaces were produced for each recognized face. The feature extraction will collect the data on the sample image using PCA and all the information is sent to another program for face recognition. The face recognition algorithm will compare this information with previously collected data from the library compiled during image training process and outputs the result if the person is recognized or not.

4.2 Face Recognition Experiment

When the "face_rec.py" algorithm is ran, it will load the previous trained data. When a positive image is identified, the green LED will light up signaling that the face is recognized and the door can be opened. Figure 11 shows the output from the python program. If an image is captured but the face could not be identified then the red LED will keep lighting up signaling that the face is not detected. To capture another image the reset button is pressed again. If a negative image is detected, the red LED light will be turned on and the user is denied permission as the lock will be kept close.

The performance evaluation includes the recognition time, memory management, accuracy and power management. To measure the recognition time required, the system is evaluated ten times and the average was taken. It was found that the average recognition time was 15 seconds. It is worth mentioning that the time measured did not consider the background process, such as checking internet connection, system update, and other services The system is tested using same person and different people including the authorized person and unauthorized person and the results show that the system recognizes the users 9 times out of 10. Therefore, it has an efficiency of 90% with a 10% tolerance. However, this only happened when the user changed some of the facial expressions. The experiment

was done using a brightly lit room with enough space between the user and the Pi camera. The best distance the user should stand to get the best accuracy is about 0.5m from the camera. Taking the picture from a closer distance will not get a good recognition as the whole face should be able to be seen inside the picture. When the captured image resolution is low, the accuracy seemed to drop because the image captures has not enough data to be processed. By using a hierarchical approach to this problem is reduced. The user image is captured in high resolution then shrunk to a smaller size before comparing to the training image. This produced a good recognition rate as well.

Using a training image of high resolution can also be used to get better accuracy but the Raspberry Pi's limited processing power causes the system to hang if the image above 720p is used for training. Therefore a smaller image is the best way to get the results. One of the limitation of the system is that the user must appear exactly as the training image captured by them. For example, if the intended user does not wears a spectacle when taking sample image then he cannot wear them during authorization. Sometimes, it is also sensitive when the user smiles wider than usual due to the size of mouth changing differently than the training image.

The Raspberry Pi has a good power management system but the internal power supply is enough to the board itself but it is inefficient to provide power to external sources. Unlike the GPIO pins in the Arduino, the raspberry Pi has limited power to supply to the these pins making developers rely on different alternatives such as combining two boards together. In this project the GPIO pins are the only source used to power up all the hardware to keep its simplicity and it is was insufficient to provide power to all the components. A low power relay was used to replace a normal relay but this can only be used for small voltage load. This is one of the limitation in the current Raspberry Pi board.

4. Conclusion

This project has presented a face recognition security system using Raspberry Pi. Python and OpenCV was used to implement the feature extraction and classifier, in which we used ARCFACE and SVM. Also a real-time face liveness detection module has been implemented which simultaneously analyses static TEXTURE and EYE-BLINK to determine the authenticity of the presented face.The proposed algorithm can rule out both spoofed images and videos by utilizing a combination of two algorithms with an effective detection rate of**(>94%).** The prototype design for real world implementation has been elaborated, in which the output of face recognition algorithm will lock or unlock the magnetic lock placed at the door using relay circuit. We have discussed the limited processing capability of Rasperry Pi which affect the image resolution to be captured, processing time, as well as memory and power management. The recognition rate was found to be around **(>90%)** when tested with three persons. This proposed system could be connected using Internet to the smart home system for the added security capability. Further research includes optimization of hierarchical image processing, use different features extraction and classifer, or use parallel Raspberry Pi clusters to speedup the computation.