International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 72-76 www.ijaem.net ISSN: 2395-5252

Security Using Facial Recognition

Pratham
Student, Guru Govind Singh Indraprastha University, Dwarka, Delhi
Student, Maharaja Agrasen Institute of Technology, Rohini Sec 22, Delhi.

Date of Submission: 05-12-2020 Date of Acceptance: 20-12-2020

ABSTRACT: This research paper is about "security using facial recognition" the objective of the project is to provide user an algorithm which will detects the face of the user and unlock the applications or website's according to that.
It will first take the training data as an input from the camera of the device and then will train the model from the input and will detect the face according to that training of the model.
The training of the data will be done using LBPH facial recognition algorithm.
To work on the project, we will use Facial Recognition Technique in the project.
KEYWORDS: Secuity, facial recognition, LBPH algorithm, Machine learning, haarcascade.

I. INTRODUCTION

[1]Security is a state where an environment is secured from malicious intent. The implementation of security is widely used from personal computer to user's smartphone. The development of security has changed significantly over the years, from simple passcode to using user's biometric features to give security. The advance technology in present day has made it possible to integrate biometric security with smartphones. However, there remains two problems when biometric security is to be implemented on mobile application. First, some biometric security requires external tools and these tools are mostly expensive. Second, most existing applications on the Android Market namely FaceLock[2] does provide user with Face Recognition features to secure access for their applications, Although, the non-paid version only allows user to lock only one applications, and for full function of the application, the user is required to pay for it. As a result, the user may not receive full functioning biometric security they require. Thus, an application called "Authentication Lock for Application Integration" are created to satisfy user's needs for biometric security for no payment. Authentication Lock for Application Integration is developed to enable face recognition security for user and secure their applications from unauthorized user. Authentication Lock has slight advantage over FaceLock in term of availability and functionality, the application is always available running in the background to provide security and to keep track of the locked application.

In order to understand how Face Recognition works, let us first get an idea of the concept of a feature vector. Every Machine Learning algorithm takes a dataset as input and learns from this data. The algorithm goes through the data and identifies patterns in the data. For instance, suppose we wish to identify whose face is present in a given image, there are multiple things we can look at as a
pattern:

- Height/width of the face.
- Height and width may not be reliable since the image could be rescaled to a smaller face. However, even after rescaling, what remains unchanged are the ratios – the ratio of height of the face to the width of the face won't change.
- Color of the face.
- Width of other parts of the face like lips, nose, etc.

Clearly, there is a pattern here – different faces have different dimensions like the ones above. Similar faces have similar dimensions. The challenging part is to convert a particular face into numbers – Machine Learning algorithms only understand numbers. This numerical representation of a "face" (or an element in the training set) is termed as a feature vector. A feature vector comprises of various numbers in a specific order. As a simple example, we can map a "face" into a feature vector which can comprise various features like:

- Height of face (cm)
- Width of face (cm)
- Average colour of face (R, G, B)
- Width of lips (cm)
- Height of nose (cm)

This

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 72-76 www.ijaem.net ISSN: 2395-5252

II. WORK FLOW OF SECURITY USING FACIAL RECOGNITION

There will be two modules in this project namely "Mod1" and "Mod2".

In the first module by the use of OpenCV library of Python:-

1. We will define the "Haarcascade_frontal_face" classifier as the face classifier to detect only the front face of the user for the face lock not background or any other feature
2. We will initialize webcam using" cv2.VideoCapture(0)".
3. We will start capturing images by converting it into "black and white color" and we will crop the image too
4. If face is detected according to face classifier then it will capture the image otherwise "Face Not Found" will be shown .
5. This process will go until 1000 pics are captured or enter key is pressed.
6. Store Pics to given path.

Mod2:-

1. Define the path where pics are and then read and load pics using cv2.imread() method.
2. Define the model which we have selected as " cv2.face.LBPHFaceRecognizer_create()".
3. Train the training data on this model using model.train().
4. Initialize webcam again and again do same cropping and converting of image into black and white.
5. Use "Haarcascade_frontal_face" classifier to detect and capture the face.
6. Use the captured image to predict whether the face matched with the training data or not.
7. If confidence is above 85% then "UNLOCK" otherwise "LOCKED".
8. "Enter" Key is used to exit.

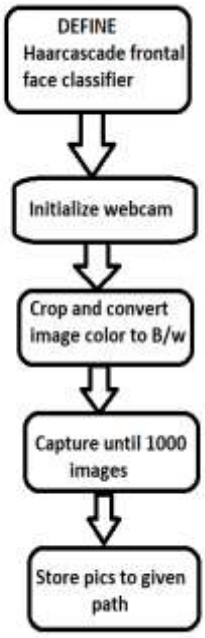

Figure 1 : Work flow of Module 1

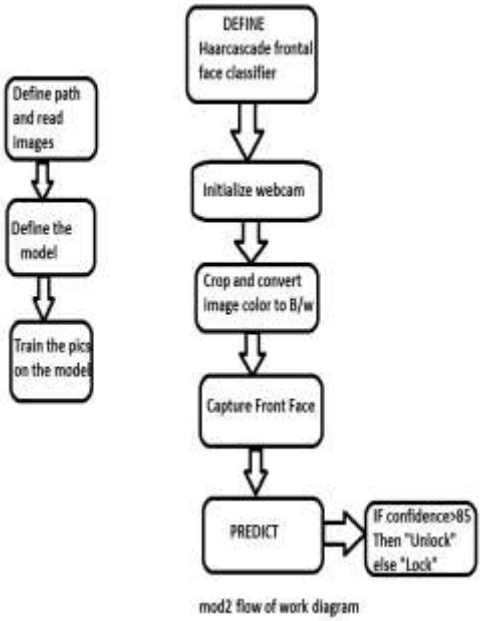

Figure 2: Work flow of Module 2

III. LBPH ALGORITHM

Human beings perform face recognition automatically every day and practically with no effort.

Although it sounds like a very simple task for us, it has proven to be a complex task for a computer, as it has many variables that can impair

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 72-76 www.ijaem.net ISSN: 2395-5252

the accuracy of the methods, for example: illumination variation, low resolution, occlusion, amongst other.

In computer science, face recognition is basically the task of recognizing a person based on its facial image. It has become very popular in the last two decades, mainly because of the new methods developed and the high quality of the current videos/cameras.

Note that face recognition is different of face detection:

- Face Detection: it has the objective of finding the faces (location and size) in an image and probably extract them to be used by the face recognition algorithm.
- Face Recognition: with the facial images already extracted, cropped, resized and usually converted to grayscale, the face recognition algorithm is responsible for finding characteristics which best describe the image.

The face recognition systems can operate basically in two modes:

- Verification or authentication of a facial image: it basically compares the input facial image with the facial image related to the user which is requiring the authentication. It is basically a 1x1 comparison.
- Identification or facial recognition: it basically compares the input facial image with all facial images from a dataset with the aim to find the user that matches that face. It is basically a 1xN comparison.

There are different types of face recognition algorithms, for example:

- Eigenfaces (1991)
- Local Binary Patterns Histograms (LBPH) (1996)
- Fisherfaces (1997)
- Scale Invariant Feature Transform (SIFT) (1999)
- Speed Up Robust Features (SURF) (2006)

Each method has a different approach to extract the image information and perform the matching with the input image. However, the methods Eigenfaces and Fisherfaces have a similar approach as well as the SIFT and SURF methods.

Today we going to talk about one of the oldest (not the oldest one) and more popular face recognition algorithms: Local Binary Patterns Histograms (LBPH).

Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets.

Using the LBP combined with histograms we can represent the face images with a simple data vector.

As LBP is a visual descriptor it can also be used for face recognition tasks, as can be seen in the following step-by-step explanation.

Now that we know a little more about face recognition and the LBPH, let's go further and see the steps of the algorithm: [3]

1. Parameters: the LBPH uses 4 parameters:

- Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- Grid X: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- Grid Y: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

Don't worry about the parameters right now, you will understand them after reading the next steps.

2. Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 11, pp: 72-76 www.ijaem.net ISSN: 2395-5252

The image below shows this procedure:

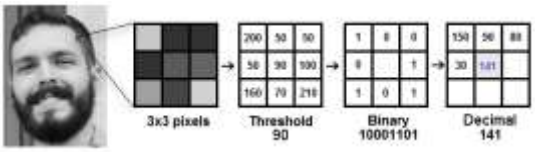

Figure 3: LBPH algorithm uses a concept of a sliding window, based on the parameters radius and neighbors.

Based on the image above, let's break it into several small steps so we can understand it easily:

- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g., 10001101). Note: some authors use other approaches to concatenate the binary values (e.g., clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.
- Note: The LBP procedure was expanded to use a different number of radius and neighbors, it is called Circular LBP.

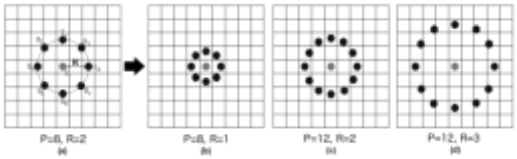

Figure 4: Diagram showing the number of pixels around the central pixel with the size of sliding window

It can be done by using bilinear interpolation. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. Extracting the Histograms: Now, using the image generated in the last step, we can use the Grid X and Grid Y parameters to divide the image into multiple grids, as can be seen in the following image:

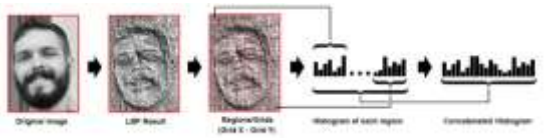

Figure 5: Diagram showing the formation of histogram from the the X and Y grids which is obtained from the LBP result

Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

5. Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: euclidean distance, chi-square, absolute value, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

- So, the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated

distance, which can be usedas a 'confidence' measurement. Note: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.

- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

IV. RESULT AND OUTPUT

As discussed that this project has two modules named Mod1 and Mod2. Mod1 will capture and store the 1000 images of user after extracting the frontal face only. Each and every image will be stored with a unique name in the local memory.

These 1000 images will be used in Mod2 as the training data. After completion of training, a window will pop up that try to match the face of user with the training result.

If our model finds the face and that particular face match with the trained data then it will show the Unlocked message else Locked.

Also, this module prints the accuracy of our model as well i.e., in this particular case our model is 91% sure that it's user.

Figure 6: Diagram showing the final result of the project with the accuracy ie., 91% in that particular case

V. CONCLUSION AND FUTURE SCOPE

1000 images that has been captured in Mod1 will be used in Mod2 as the training data. After completion of training, a window will pop up that try to match the face of user with the training result. After testing the project several times the accuracy was found to be 91% average which is excellent.

This project provides the vast opportunities of changes it. We can implement this security mechanism in website and apps. Extension and managing this project are quite easy as we can implement age, gender classification also.

We can also implement the face expression classification in this project which will provide more flexibility of our project as it will only unlock our application when normal expression found.

REFERENCES

[1]. Authentication Lock for Application Integration Face Recognition Security Muhammad Aliff Romi Bin Sharipudin, Firoz bin Yusuf Patel Dawoodi

[2]. Faclock.mobi, "Facelock", Facelock.mobi. [online] Available at http://www.facelock.mobi/ [Accessed: May 15,2017]

[3]. https://towardsdatascience.com/face-recognition-how-lbph-works-90ec258c3d6b