Facelock Algorithm

Submitted in partial fulfilment of requirements for the award of degree of

B. TECH

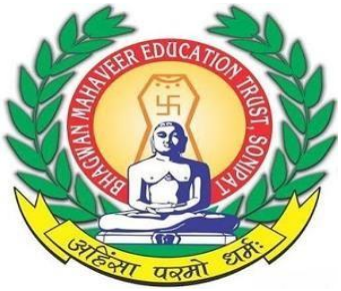
COMPUTER SCIENCE AND ENGINEERING

SUBMITTED BY

Aparna (CSE/17/119)

Under the Guidance
Of

Ms. Gurminder Kaur



Department of Computer Science and Engineering

B.M. Institute of Engineering and Technology
Sector-10, Sonipat
(Affiliated to GGSIP University, Delhi)

Certificate

It is to certify that the project has been carried out by the students of 7^{th} /8^{th} semester **Aparna(CSE/17/119)** under my guidance. The report covers all the aspects of the work done (including H/W & S/W, Coding etc.).

The project report is complete in all respects and I have understood the entire software.

Ms Gurminder Kaur

Guide name

Ms Gurminder Kaur

Incharge Name

Certificate

It is to certify that the project has been carried out by the student of 7th semester Aparna (CSE/17/119) under the guidance of Computer Science and Engineering Department. The report covers all the aspects of the work done (including H/W & S/W, Coding etc.)

Mr. Pardeep Tyagi

Name of incharge CSE Department
H.O.D. COMPUTER SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY, DELHI) DELHI – 110089

CANDIDATE'SDECLARATION

It is hereby certified that the work which is being presented in the B. Tech Minor Project Report entitled **"Facelock Algorithm"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** and submitted in the **Department of Computer science and engineering , B.M.I.E.T (Affiliated to Guru Gobind Singh Indraprastha University, Delhi)** is an authentic record of our own work carried out during a period from **AUGUST,2020 to DECEMBER,2020** under the guidance of **Guide name with designation**

The matter presented in the B. Tech Minor Project Report has not been submitted by me for the award of any other degree of this or any other Institute.

Student name	Aparna
Rollno	CSE/17/119

This is to certify that the above statement made by the candidate is correct to the best of my knowledge. They are permitted to appear in the External Major Project Examination

Guide Name – Ms Gurminder Kaur

The B. Tech Minor Project Viva-Voce Examination of **Aparna(CSE/17/119)** has been held on **…….**

Project Coordinator **(Signature of External Examiner)**

ACKNOWLEDGEMENT

We express our deep gratitude **to Ms Gurminder Kaur**, Department of computer science and Engineering for his valuable guidance and suggestion throughout my project work. We are thankful to **Mr Pardeep Tyagi,** Project Coordinators, for their valuable guidance.

We would like to extend my sincere thanks to HOD**,** for his time to time suggestions to complete my project work. I am also thankful to **Dr.HARISH MITTAL** for providing me the facilities to carry out my project work.

Aparna

CSE/17/119

List of Figures

ABSTRACT

This report describes my minor project that has been done in partial fulfilment of the requirements for the award of the degree of Bachelor of Technology. The title of this project is “Security using facial recognition”. This project is basically divided into 2 Modules namely Mod1 and Mod2.

This project report is divided into four different chapters

Chapter 1 includes the problem statement, nee of the study, introduction and the objective of the project. It also includes the theoretical explanation about the same.

Chapter2 includes all the necessary introduction to the projects with brief into about the functionality of Mod1 and Mod2. All the important data flow diagrams, flow hart related to project is covered in this chapter. Software and hardware required for this project is also included in this chapter.

Chapter 3 includes experimental result with all the necessary output and result with screenshots attached. This chapter also included the merit and demerit of the project and the output obtained from this project.

Chapter 4 includes conclusions and the future scope of this project. All the future idea that how can this project be extended will be cover in this chapter. The final conclusion and accuracy of this project is discussed in this chapter briefly.

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

The security of applications in mobile phones or website's is always a major issue, to ensure the security and privacy design a Machine Learning Algorithm which will only unlock the application or website when it will scan your face.

1.2 Need of the study

As we see in the digital era every personal or important thing is in our mobile phones or systems. Anyone can easily get access to the private documents or things in your mobile phones. To increase the privacy, we need a security system which will open when it will detect our face. So to increase the security and privacy of the digital systems we need an algorithm which will detect our face and get unlocked otherwise it will not open.

1.3 Introduction

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.

To work on the project, we will use Facial Recognition Technique in the project

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:

1. Height/width of the face.
2. 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.
3. Color of the face.
4. 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:

1. Height of face (cm)
2. Width of face (cm)
3. Average color of face (R, G, B)
4. Width of lips (cm)
5. Height of nose (cm)

Essentially, given an image, we can map out various features and convert it into a feature vector like:

Height of face (cm)	Width of face (cm)	Average color of face (RGB)	Width of lips (cm)	Height of nose (cm)
23.1	15.8	(255, 224, 189)	5.2	4.4

So, our image is now a vector that could be represented as (23.1, 15.8, 255, 224, 189, 5.2, 4.4). Of course there could be countless other features that could be derived from the image (for instance, hair color, facial hair, spectacles, etc). However, for the example, let us consider just these 5 simple features.

Now, once we have encoded each image into a feature vector, the problem becomes much simpler. Clearly, when we have 2 faces (images) that represent the same person, the feature vectors derived will be quite similar. Put it the other way, the "distance" between the 2 feature vectors will be quite small.

Machine Learning can help us here with 2 things:

1. *Deriving the feature vector*: it is difficult to manually list down all of the features because there are just so many. A Machine Learning algorithm can intelligently label out many of such features. For instance, a complex features could be: ratio of height of nose and width of forehead. Now it will be quite difficult for a human to list down all such "second order" features.
2. *Matching algorithms*: Once the feature vectors have been obtained, a Machine Learning algorithm needs to match a new image with the set of feature vectors present in the corpus.

We will use OPENCV library in this project to detect the face of the person and to scan it.

The name OpenCV has become synonymous with computer vision, but what is OpenCV?

OpenCV is a collection of software algorithms put together in a library to be used by industry and academia for computer vision applications and research . OpenCV started at Intel in the mid 1990s as a method to demonstrate how to accelerate certain algorithms in hardware.

In 2000, Intel released OpenCV to the open source community as a beta version, followed by v1.0 in 2006. In 2008, Willow Garage took over support for OpenCV and immediately released v1.1.

Willow Garage dates from 2006. The company has been in the news a lot lately, subsequent to the unveiling of its PR2 robot . Gary Bradski began working on OpenCV when he was at Intel; as a senior scientist at Willow Garage he aggressively continues his work on the library.

OpenCV v2.0, released in 2009, contained many improvements and upgrades. Initially, OpenCV was primarily a C library. The majority of algorithms were written in C, and the primary method of using the library was via a C API. OpenCV v2.0 migrated towards C++ and a C++ API. Subsequent versions of OpenCV added Python support, along with Windows, Linux, iOS and Android OS support, transforming OpenCV (currently at v2.3) into a cross-platform tool. OpenCV v2.3 contains more than 2500 algorithms; the original OpenCV only had 500. And to assure quality, many of the algorithms provide their own unit tests.

1.4 Objective of the study

To develop a algorithm which provides access when face is scanned and matched.

CHAPTER 2: TECHNOLOGY USED

Python :- We have used python latest version " 3.8.5" in the project.

OpenCV :- OpenCV (OPEN SOUCE COMPUTER VISION) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. We will use many methods of opencv in this project.

Anaconda :- Anaconda Enterprise is an enterprise-ready, secure, and scalable data science platform that empowers teams to govern data science assets, collaborate, and deploy data science projects.

CHAPTER 3: IMPLEMENTATION

3.1 Methodology

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

Mod1 :-

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.

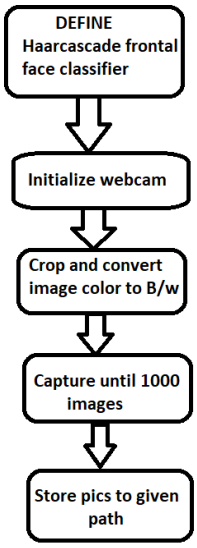
2. We will initialize webcam using " cv2.VideoCapture(0)".

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.



Mod1 flow of work diagram

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().

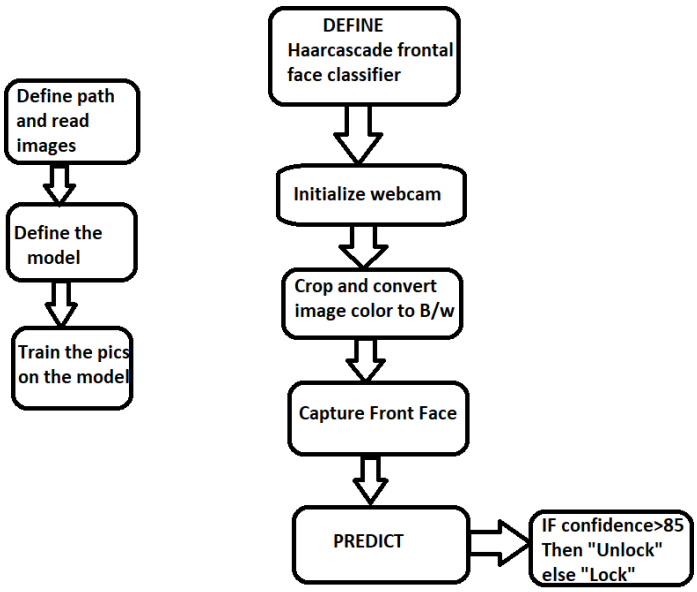
4. Initialize webcam again and again do same cropping and converting of image into black and white.

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.



mod2 flow of work diagram

3.2 SOFTWARE USED

Python :- We have used python latest version " 3.8.5" in the project.

OpenCV :- OpenCV (OPEN SOUCE COMPUTER VISION) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. We will use many methods of opencv in this project.

Anaconda :- Anaconda Enterprise is an enterprise-ready, secure, and scalable data science platform that empowers teams to govern data science assets, collaborate, and deploy data science projects.

3.3 HARDWARE USED

A windows system with webcam in it.

ABOUT HARCASCADE CLASSIFIER

This is basically a machine learning based approach where a cascade function is trained from a lot of images both positive and negative. Based on the training it is then used to detect the objects in the other images.

USECASES OF HARCASCADE CLASSIFIERS

- 1. FACE DETECTION using haarcascade_frontalface_default.xml
- 2. FACE AND EYE DETECTION using haarcascade_eye.xml
- 3. VEHICLE DETECTION FROM STREAMING VIDEO using haarcascade_car.xml
- 4. PEDESTRIAN DETECTION FROM STREAMING VIDEO using haarcascade_fullbody.xml

HARCASCADE FRONTAL FACE CLASSIFIER

It is used to detect only the front face of the user and eliminates all the background details


```
<opencv_storage>
<cascade type_id="opencv-cascade-classifier"><stageType>BOOST</stageType>
  <featureType>HAAR</featureType>
  <height>24</height>
  <width>24</width>
  <stageParams>
    <maxWeakCount>211</maxWeakCount></stageParams>
  <featureParams>
    <maxCatCount>0</maxCatCount></featureParams>
  <stageNum>25</stageNum>
  <stages>
    <_>
      <maxWeakCount>9</maxWeakCount>
      <stageThreshold>-5.0425500869750977e+00</stageThreshold>
      <weakClassifiers>
        <_>
          <internalNodes>
            0 -1 0 -3.1511999666690826e-02</internalNodes>
          <leafValues>
            2.0875380039215088e+00 -2.2172100543975830e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 1 1.2396000325679779e-02</internalNodes>
          <leafValues>
            -1.8633940219879150e+00 1.3272049427032471e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 2 2.1927999332547188e-02</internalNodes>
          <leafValues>
            -1.5105249881744385e+00 1.0625729560852051e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 3 5.7529998011887074e-03</internalNodes>
          <leafValues>
            -8.7463897466659546e-01 1.1760339736938477e+00</leafValues></_>
        <_>
          <internalNodes>
            0 -1 4 1.5014000236988068e-02</internalNodes>
          <leafValues>
            -7.7945697307586670e-01 1.2608419656753540e+00</leafValues></_>
        <_>
          <internalNodes>
```

ABOUT LBPH MODEL

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.

PARAMETERS OF LBPH MODEL

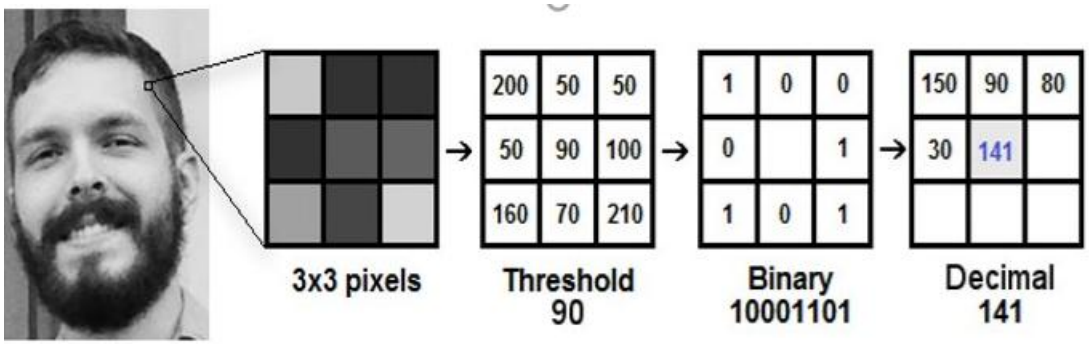
- Radius: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- Neighbors: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational 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.

TRAINING LBPH MODEL

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.

STEPS TO APPLY LBPH 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.



- we need take the central value of the matrix of pixel values to be used as the threshold.

- 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).
- 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.
- 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 and create histograms.

FACE RECOGNITION BY LBPH

- 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.
- 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 used as a 'confidence' measurement.

SNAPSHOTS OF CODE

MODULE 1

```python
import cv2
import numpy as np

# Load HAAR face classifier
face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

# Load functions
def face_extractor(img):
    # Function detects faces and returns the cropped face
    # If no face detected, it returns the input image

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)

    if faces is ():
        return None

    # Crop all faces found
    for (x,y,w,h) in faces:
        cropped_face = img[y:y+h, x:x+w]

    return cropped_face

# Initialize Webcam
cap = cv2.VideoCapture(0)
count = 0

# Collect 1000 samples of your face from webcam input
while True:

    ret, frame = cap.read()
    if face_extractor(frame) is not None:
        count += 1
        face = cv2.resize(face_extractor(frame), (200, 200))
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        # Save file in specified directory with unique name
        file_name_path = 'E:/apa/pics/' + str(count) + '.jpg'
        cv2.imwrite(file_name_path, face)

        # Put count on images and display live count
```

MODULE 2

```python
import cv2
import numpy as np
from os import listdir
from os.path import isfile, join

# Get the training data we previously made
data_path = 'E:/apa/pics/'
onlyfiles = [f for f in listdir(data_path) if isfile(join(data_path, f))]

# Create arrays for training data and labels
Training_Data, Labels = [], []

# Open training images in our datapath
# Create a numpy array for training data
for i, files in enumerate(onlyfiles):
    image_path = data_path + onlyfiles[i]
    images = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    Training_Data.append(np.asarray(images, dtype=np.uint8))
    Labels.append(i)

# Create a numpy array for both training data and labels
Labels = np.asarray(Labels, dtype=np.int32)

# Initialize facial recognizer
model = cv2.face.LBPHFaceRecognizer_create()

# NOTE: For OpenCV 3.0 use cv2.face.createLBPHFaceRecognizer()

# Let's train our model
model.train(np.asarray(Training_Data), np.asarray(Labels))
print("Model trained sucessefully")

face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

def face_detector(img, size=0.5):

    # Convert image to grayscale
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    if faces is ():
        return img, []
```

```python
    # Convert image to grayscale
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)
    if faces is ():
        return img, []

    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(0,255,255),2)
        roi = img[y:y+h, x:x+w]
        roi = cv2.resize(roi, (200, 200))
    return img, roi


# Open Webcam
cap = cv2.VideoCapture(0)

while True:

    ret, frame = cap.read()

    image, face = face_detector(frame)

    try:
        face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        # Pass face to prediction model
        # "results" comprises of a tuple containing the label and the confidence value
        results = model.predict(face)

        if results[1] < 500:
            confidence = int( 100 * (1 - (results[1])/400) )
            display_string = str(confidence) + '% Confident it is User'

        cv2.putText(image, display_string, (100, 120), cv2.FONT_HERSHEY_COMPLEX, 1, (255,120,150), 2)

        if confidence > 85:
            cv2.putText(image, "Unlocked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,255,0), 2)
            cv2.imshow('Face Recognition', image )
        else:
            cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
            cv2.imshow('Face Recognition', image )
```

```
    except:
        cv2.putText(image, "No Face Found", (220, 120) , cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
        cv2.putText(image, "Locked", (250, 450), cv2.FONT_HERSHEY_COMPLEX, 1, (0,0,255), 2)
        cv2.imshow('Face Recognition', image )
        pass

    if cv2.waitKey(1) == 13: #13 is the Enter Key
        break

cap.release()
cv2.destroyAllWindows()


cv2.imshow()

import time

# Load HAAR face classifier
face_classifier = cv2.CascadeClassifier('Haarcascades/haarcascade_frontalface_default.xml')

# Load functions
def face_extractor(img):
    # Function detects faces and returns the cropped face
    # If no face detected, it returns the input image

    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray, 1.3, 5)

    if faces is ():
        return None
```

CHAPTER 4: EXPERIMENTAL RESULTS

4.1 Result and output

As discussed in the chapter 2 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.

Figure 4.1 Capturing 1000 images of user

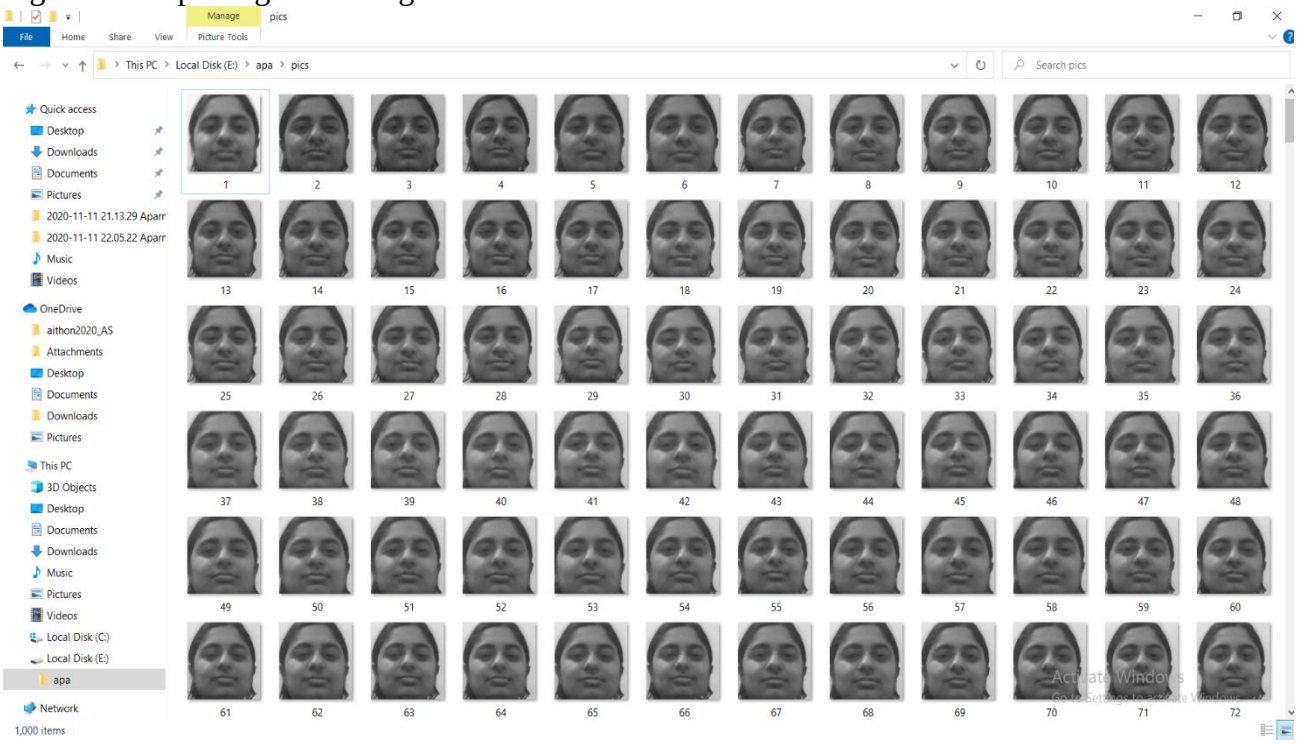
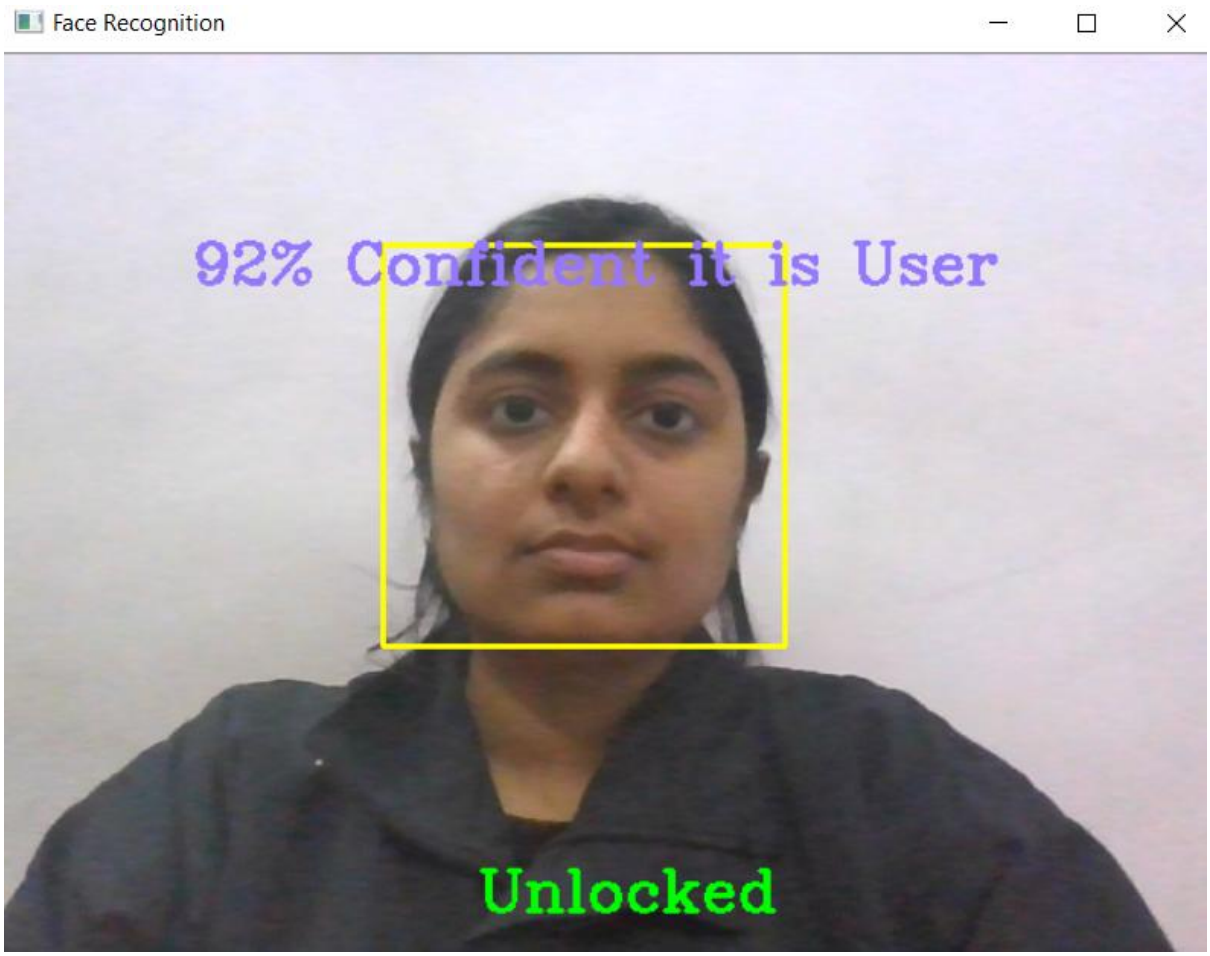


Figure 4.2 1000 stored images

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.

3.1 Merits

The biggest of this project that we can use this security mechanism in websites, Apps and any other specific platform applications. As mobile phones or website's security is always a major issue, to ensure the security and privacy, this project will play an important role in the same field.

It will capture the data in real time which hardly takes 4-5 minutes. The training duration is also very short which completes in about few minutes.

This project is portable as well and can be used on any OS which has python installed in it.

3.2 Demerits

It requires proper lightning while capturing images.

CHAPTER 5: CONCLUSIONS AND FUTURE SCOPE

5.1 Conclusions

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.

Figure 5.1 Capturing image

Figure 5.2 Final output

The has been matched successfully with 92% accuracy.

5.2 Future Scope

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

https://docs.opencv.org/master/d6/d00/tutorial_py_root.html

https://docs.opencv.org/3.4/javadoc/org/opencv/face/LBPHFaceRecognizer.html

D. B. Desai and S. N. Kavitha, "Face Anti-spoofing Technique Using CNN and SVM," *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India, 2019, pp. 37-41, doi: 10.1109/ICCS45141.2019.9065873.

Jenkins R, McLachlan JL, Renaud K. 2014. Facelock: familiarity-based graphical authentication. *PeerJ* 2:e444 https://doi.org/10.7717/peerj.444

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

A short review paper on Face detection using Machine learning Farhad Navabifar(1), Mehran Emadi (2)Rubiyah Yusof(3) ,Marzuki Khalid(4)