Report for Face Recognition with app lock

Python Programming (INT-213)

Name	: Kunal Tyagi
Reg no.	: 12013504
Name	: Sumit Kumar
Reg no.	:12012381
Program	: CSE B.tech
Semester	: Third
School	: School of Computer Science
Name of University	: Lovely Professional University
Date of submission	: 20-Nov-2021

Face Recognition with app lock

Abstract :-

The face is one of the easiest ways to distinguish the individual identity of each other. Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognize a face as individuals. Stage is then replicated and developed as a model for facial image recognition (face recognition) is one of the much-studied biometrics technology and developed by experts. There are two kinds of methods that are currently popular in developed face recognition pattern namely, Eigenface method and Fisherface method.

Facial image recognition Eigenface method is based on the reduction of facedimensional space using Principal Component Analysis (PCA) for facial features. The main purpose of the use of PCA on face recognition using Eigen faces was formed (face space) by finding the eigenvector corresponding to the largest eigenvalue of the face image. The area of this project face detection system with face recognition is Image processing. The software requirements for this project is PyCharm.

Table of Contents

Sr. No.	Topic	Pg no.
1.	Introduction	4
2.	Team Members	5
3.	Libraries	6-7
4.	Coding and Screenshot of project	8-12

Introduction :-

1.1 *Context* -

This project has been done as a part of my course for the CSE at Lovely Professionally University. Supervised by Ankita Wadhawan, I have to fulfill the requirements in order to succeed the module.

1.2 *Motivations* -

Because we are particularly interested in anything related to Machine Learning, the group project provided us with an excellent opportunity to study and reinforce our enthusiasm for this topic. In terms of application possibilities, the capacity to create estimates, forecasts, and offer machines the ability to learn on their own is both powerful and endless. Machine Learning may be used in finance, medicine, and practically any other field. As a result, I choose to focus my effort on Machine Learning.

1.3 *Idea* -

We intended to make project as didactic as possible as a first experience, so we approached each step of the machine learning process and tried to comprehend them thoroughly. We chose Face recognition with app lock as our strategy, which is known as a "toy problem," which are challenges that are not of immediate scientific relevance but are good to show and practice. The goal was to

Team Members :-

Team Leader :

Sumit Kumar:-

1. Contributions:-
2. Coding (joined)
3. machine learning(joined)
4. Reports(joined)
5. Datasets

Kunal Tyagi:-

1. Contributions:-
2. Coding (joined)
3. machine learning(joined)
4. Reports(joined)
5. Datasets

Libraries :-

Cv2:-

cv2 is the module import name for opencv-python, "Unofficial pre-built CPU-only OpenCV packages for Python". The traditional OpenCV has many complicated steps involving building the module from scratch, which is unnecessary. I would recommend remaining with the opencv-python library.

Numpy:-

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python. As the whole project is based on whole complex stats ,we will use this fast calculations and provide results.

Pandas:-

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

List Dir:-

listdir() method in python is used to get the list of all files and directories in the specified directory. If we don't specify any directory, then list of files and directories in the current working directory will be returned.

Matplotlib:-

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

Coding and ScreenShot of Project

Step 1 - Create Training Data

```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
```

```
count = 0

# Collect 100 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), (400, 400))
        # face = cv2.cvtColor(face, cv2.COLOR_BGR2GRAY)

        # Save file in specified directory with unique name
        file_name_path = 'D:\Python project\Pictures' + str(count) + '.jpg'
        cv2.imwrite(file_name_path, face)

        # Put count on images and display live count
        cv2.putText(face, str(count), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow('Face Cropper', face)

    else:
        print("Face not found")
        pass

    if cv2.waitKey(1) == 13 or count == 100:  # 13 is the Enter Key
        break

cap.release()
cv2.destroyAllWindows()
print("Collecting Samples Complete")
```

Step 2 - Train Model

```
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 = './faces/user/'
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
```

```
# 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")
```

Step 3 - Run Our Facial Recognition

```
import cv2
import numpy as np

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, []

    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 > 75:
            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
```

```
    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()
```

```
cap = cv2.VideoCapture(0)
count = 0
lst = []
start_time = time.time()
elapsed_time = time.time()
# Collect 100 samples of your face from webcam input
while True:

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

        # Put count on images and display live count
        cv2.putText(face, str(time.clock()), (50, 50), cv2.FONT_HERSHEY_COMPLEX, 1, (0, 255, 0), 2)
        cv2.imshow('Face Cropper', face)

    else:

        print("Face not found")
        elapsed_time = time.time() - start_time
        lst.append(elapsed_time)
        seconds = 0
        time.sleep(1)
        pass

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

```
        print("Face not found")
        elapsed_time = time.time() - start_time
        lst.append(elapsed_time)
        seconds = 0
        time.sleep(1)
        pass

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

        break

cap.release()
cv2.destroyAllWindows()
print("Collecting Samples Complete")
```