Criminal Face Identification System

A

Project Report Submitted for the partial fulfilment of

B.tech Degree in

INFORMATION TECHNOLOGY

By

Anjani Kumar Mall (1705213901)

Rahul Majhi (1705213908)

Sunil Kumar Bajpai (1705213911)

Under the supervision of

Ms. Deepa Verma

Department of Computer Science & Engineering

Institute of Engineering and Technology

Dr. A.P.J. Abdul Kalam Technical University, Lucknow, Uttar Pradesh.

July, 2020

Contents

Declaration

We hereby declare that this submission is our own work and that, to the best of our belief and knowledge, it contains no material previously published or written by another person or material which to a substantial error has been accepted for the award of any degree or diploma of university or other institute of higher learning, except where the acknowledgement has been made in the text. The project has not been submitted by us at any other institute for requirement of any other degree.

Submitted by: - Date: 25 July, 2020

(1) Name: Anjani Kumar Mall

Roll no: 1705213901

Branch: Information Technology

Signature: Anjani kumar mall

(2) Name: Rahul Majhi

Roll no: 1705213908

Branch: Information Technology

Signature: RahulMajhi

(3) Name: Sunil Kumar Bajpai

Roll no: 1705213911

Branch: Information Technology

Signature: SunilBajpai

Certificate

This is to certify that the project report entitled “Criminal Face Identification System” presented by Anjani Kumar Mall, Rahul Majhi and Sunil Kumar Bajpai in the partial fulfilment for the award of Bachelor of Technology in Information Technology, is a record of work carried out by them under my supervision and guidance at the Department of Computer Science and Engineering at Institute of Engineering and Technology, Lucknow.

It is also certified that this project has not been submitted at any other Institute for the award of any other degrees to the best of my knowledge.

Ms. Deepa Verma

Department of Computer Science and Engineering

Institute of Engineering and Technology, Lucknow

Acknowledgement

We would like to express our deepest gratitude and thanks to our supervisor, Ms Deepa Verma, who gave guidance in the project “Criminal Face Identification System”. Also, we would like to thank her for being supportive, and for her advice throughout the semester that helped in realizing this project. We are really grateful for her supervising methodology that made the project easier.

We would also like to express our gratitude towards, Prof. D.S. Yadav and Prof. Pawan Kumar Tiwari for giving their consent for the project and giving advice for improvement and future works. We would also like to thank other supervisors for their guidance through semester.

<u>Abstract</u>

Crime preventions and criminal identification are the primary issues for the police personnel, since property and life protection are the basic concerns of the police but to combat the crime, the availability of police personnel is limited.

The goal of this project is to identify face of previously convicted persons and provide a solution with higher accuracy, better response rate and an initial step for video surveillance. Solution is proposed based on nature of criminal psychic of repeating crime or involvement in it. This system is used to track history sheeters and recognize them before and after any mischief or any unlawful activity.

In the system we are storing the image of criminal in the database along with its other detail to provide ease in data retrieval and ensuring fast deployment of results in real world. The project is built on python 3.5 with the use of OpenCV along with the algorithms like Haar cascade classifier, LBPH and face_recognition etc. to store the detail of person we have used SQLite.

List of Figures and Tables

Chapter 1. Introduction

Criminal record contains details about a particular person along with photograph and personal information. To identify any history sheeter we need identification regarding that person. One of the ways is face identification. The face is our primary focus of attention in social intercourse, playing a major role in conveying identity and emotions. Human ability to remember and recognize faces is remarkable. This system aims to provide a copy of human trait of identification along with the details of person in real time for efficient tracking of habitual criminals. Criminal face identification system creates database of criminal and recognize the person if one's image matches with an existing one in distributed environment. The project will be a milestone for video based face identification and for surveillance.

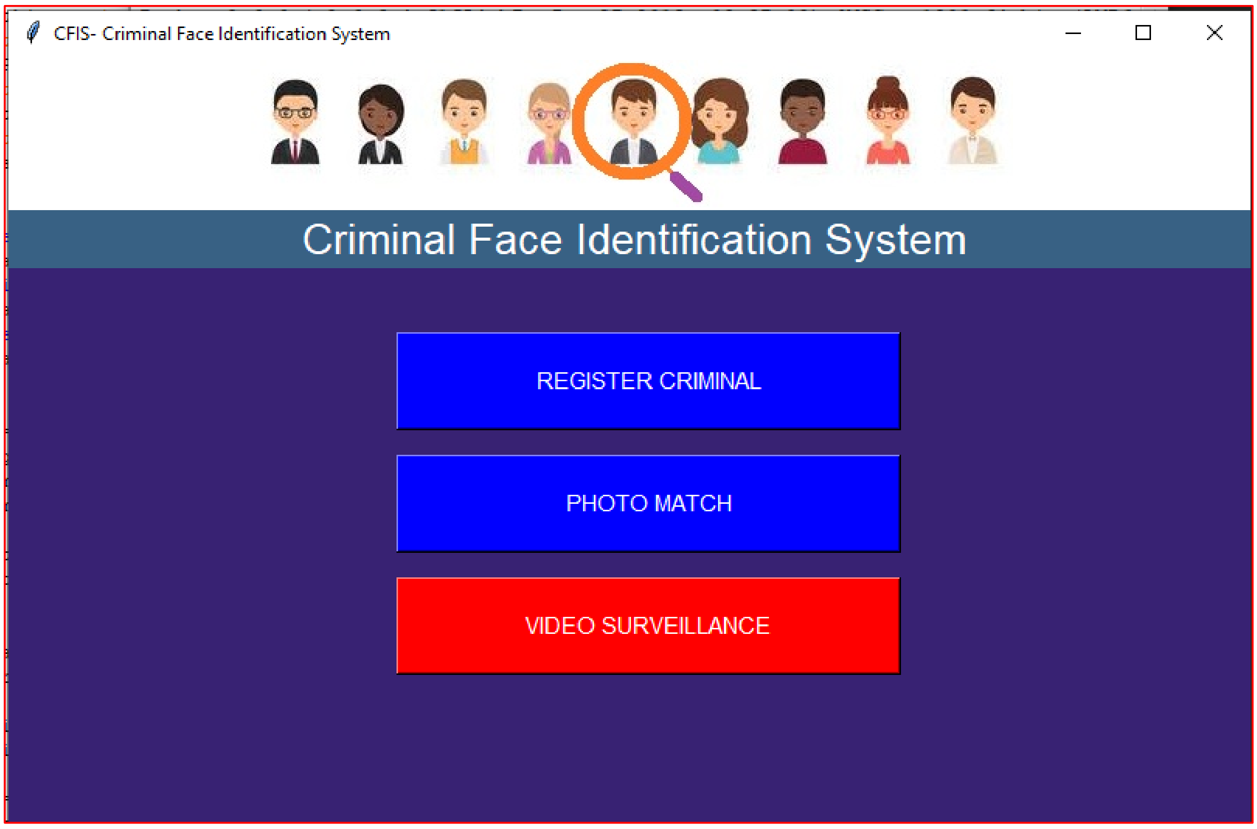

CFIS- Criminal Face Identification System
Criminal Face Identification System
REGISTER CRIMINAL
PHOTO MATCH
VIDEO SURVEILLANCE

Fig 1.1 Home page

Chapter 2. Literature Review

2.1 Face detection

We have used OpenCV which presents a Haar cascade classifier, which is used for face detection. The Haar cascade classifier uses the AdaBoost algorithm to detect multiple facial features. First, it reads the image to be detected and converts it into the gray image, then loads Haar cascade classifier to decide whether it contains a human face. If so, it proceeds to examine the face features and draw a rectangular frame on the detected face. Otherwise, it continues to test the next picture [3]**.**

2.1.1 Haar features

A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. This modified feature set is called 2rectangle feature. Faces are scanned and searched for Haar features of the current stage. The weight and size of each feature and the features themselves are generated using a machine learning algorithm from AdaBoost.

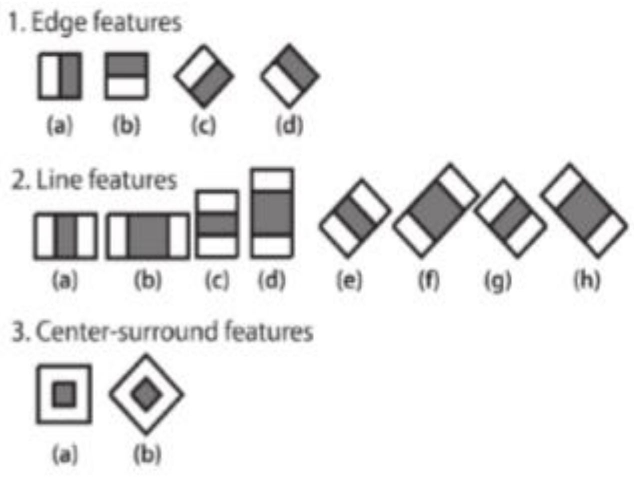


Fig 2.1   Some common Haar Feature (original paper)

2.2 Face extraction

The LBP operator is applied to describe the contrast information of a pixel to its neighborhood pixels. The original LBP operator is defined in the window of 3*3.Using the median pixel value as the threshold of the window, it compares with the gray value of the adjacent 8 pixels. If the neighborhood pixel value is larger or equal compare to the median pixel value, the value of pixel position is marked as 1, otherwise marked as 0. The function is defined as shown in equation 1. It can be illustrated in Figure 2.2.

$$N(x) = \begin{Bmatrix} 1, x \geq 0 \\ 0, x < 0 \end{Bmatrix} \qquad \text{eq. (1)}$$

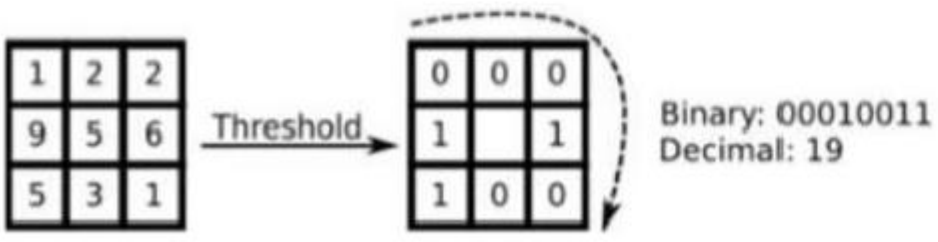


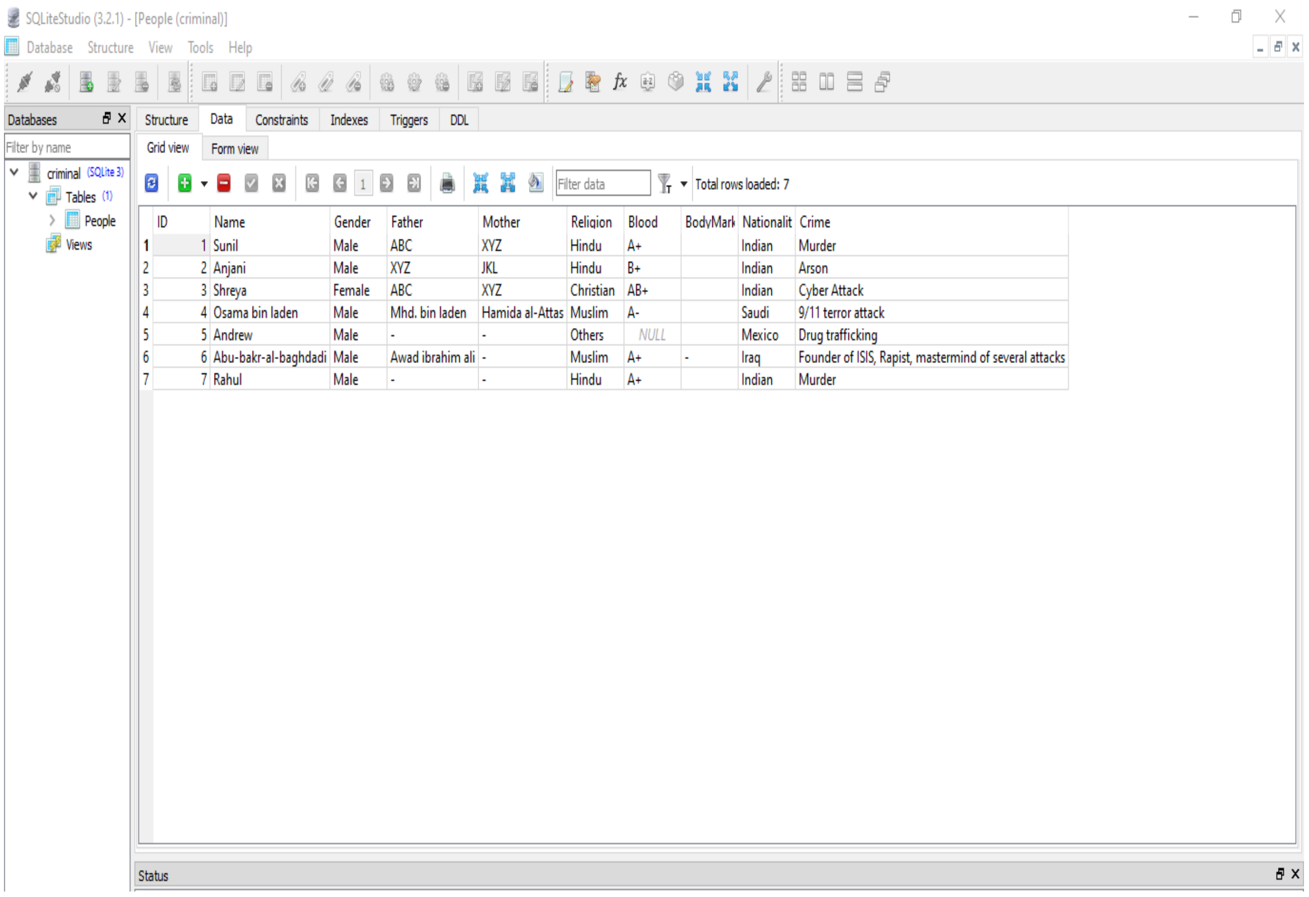
Fig 2.2 original LBP operator

2.3 SQLite

SQLite is a very popular database which has been successfully used with on disk file format for desktop applications like version control systems, financial analysis tools, media cataloging and editing suites, CAD packages, record keeping programs etc.

There are a lot of advantages to use SQLite for this application:

1) Lightweight: SQLite is a very light weighted database so, it is easy to use it as an embedded software with devices like televisions, Mobile phones, cameras, home electronic devices, etc.
2) Better Performance: Reading and writing operations are very fast for SQLite database. It is almost 35% faster than File system. It only loads the data which is needed, rather than reading the entire file and hold it in memory. If you edit small parts, it only overwrite the parts of the file which was changed.
3) No Installation Needed: SQLite is very easy to learn. You don't need to install and configure it. Just download SQLite libraries in your computer and it is ready for creating the database.
4) Reliable: It updates your content continuously so, little or no work is lost in a case of power failure or crash.
5) Portable: SQLite is portable across all 32-bit and 64-bit operating systems and big- and little-endian architectures.
7) Reduce Cost and Complexity: It reduces application cost because content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural queries.


SQLiteStudio (3.2.1) - [People (criminal)]
Database Structure View Tools Help
Databases
Filter by name
criminal (SQLite 3)
Tables (1)
People
Views
Structure Data Constraints Indexes Triggers DDL
Grid view Form view
Filter data
Status

Total rows loaded: 7

	ID	Name	Gender	Father	Mother	Religion	Blood	BodyMark	Nationalit	Crime
1	1	Sunil	Male	ABC	XYZ	Hindu	A+		Indian	Murder
2	2	Anjani	Male	XYZ	JKL	Hindu	B+		Indian	Arson
3	3	Shreya	Female	ABC	XYZ	Christian	AB+		Indian	Cyber Attack
4	4	Osama bin laden	Male	Mhd. bin laden	Hamida al-Attas	Muslim	A-		Saudi	9/11 terror attack
5	5	Andrew	Male	-	-	Others	NULL		Mexico	Drug trafficking
6	6	Abu-bakr-al-baghdadi	Male	Awad ibrahim ali	-	Muslim	A+	-	Iraq	Founder of ISIS, Rapist, mastermind of several attacks
7	7	Rahul	Male	-	-	Hindu	A+		Indian	Murder

Fig 2.3 Snapshot of SQLite Database

Chapter 3. Methodology

3.1. Process

A. Import the required modules

The Modules required to perform the facial recognition are cv2, os, image module and numpy. cv2 is the OpenCV module and contains the functions for face detection and recognition. OS will be used to maneuver with image and directory names. First, we use this module to extract the image names in the database directory and then from these names individual number is extracted, which is used as a label for the face in that image. Image module from PIL is used to read the image in grayscale format.

B. Load the face detection Cascade

To Load the face detection cascade the first step is to detect the face in each image. Once we get the region of interest containing the face in the image, we use it for training the recognizer. For the purpose of face detection, we will use the Haar Cascade provided by OpenCV. The Haar cascades that come with OpenCV are located in the directory of OpenCV installation. haarcascade frontalface default.xml is used for detecting the face. Cascade is loaded using the cv2 CascadeClassifier function which takes the path to the cascade .yml file.

C. Create the Face Recognizer Object

The next step involves creating the face recognizer object. The face recognizer object has functions like FaceRecognizer.train() to train the recognizer and FaceRecognizer.predict() to recognize a face. OpenCV currently provides Eigenface Recognizer, Fisherface Recognizer and Local Binary Patterns Histograms(LBPH) Face Recognizer. We have used LBPH recognizer because Real life isn’t perfect. We simply can’t guarantee perfect light settings in your images or 10 different images of a person. LBPH focus on extracting local features from images. The idea is to not look at the whole image as a high-dimensional vector but describe only local features of an object. The basic idea of Local Binary Patterns is to summarize the local structure in an image by comparing each pixel with its neighbourhood. LBP operator is robust against monotonic gray scale transformations.

D. Prepare the training set and Perform the training

To create the function to prepare the training set, we will define a function that takes the absolute path to the image database as input argument and returns tuple of 2 list, one containing the detected faces and the other containing the corresponding label for that face. For example, if the i^{th} index in the list of faces represents the 4th individual in the database, then the corresponding ith location in the list of labels has value equal to 4. Now to perform the training using the Face Recognizer. Train function. It requires 2 arguments, the features which in this case are the images of faces and the corresponding labels assigned to these faces which in this case are the individual number that we extracted from the image names.

DATASET:

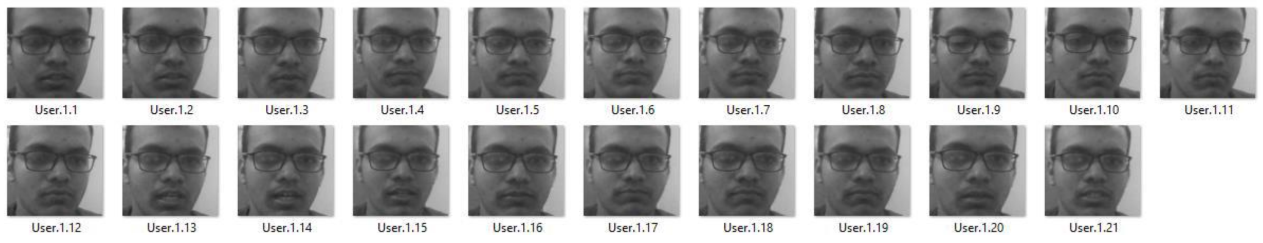
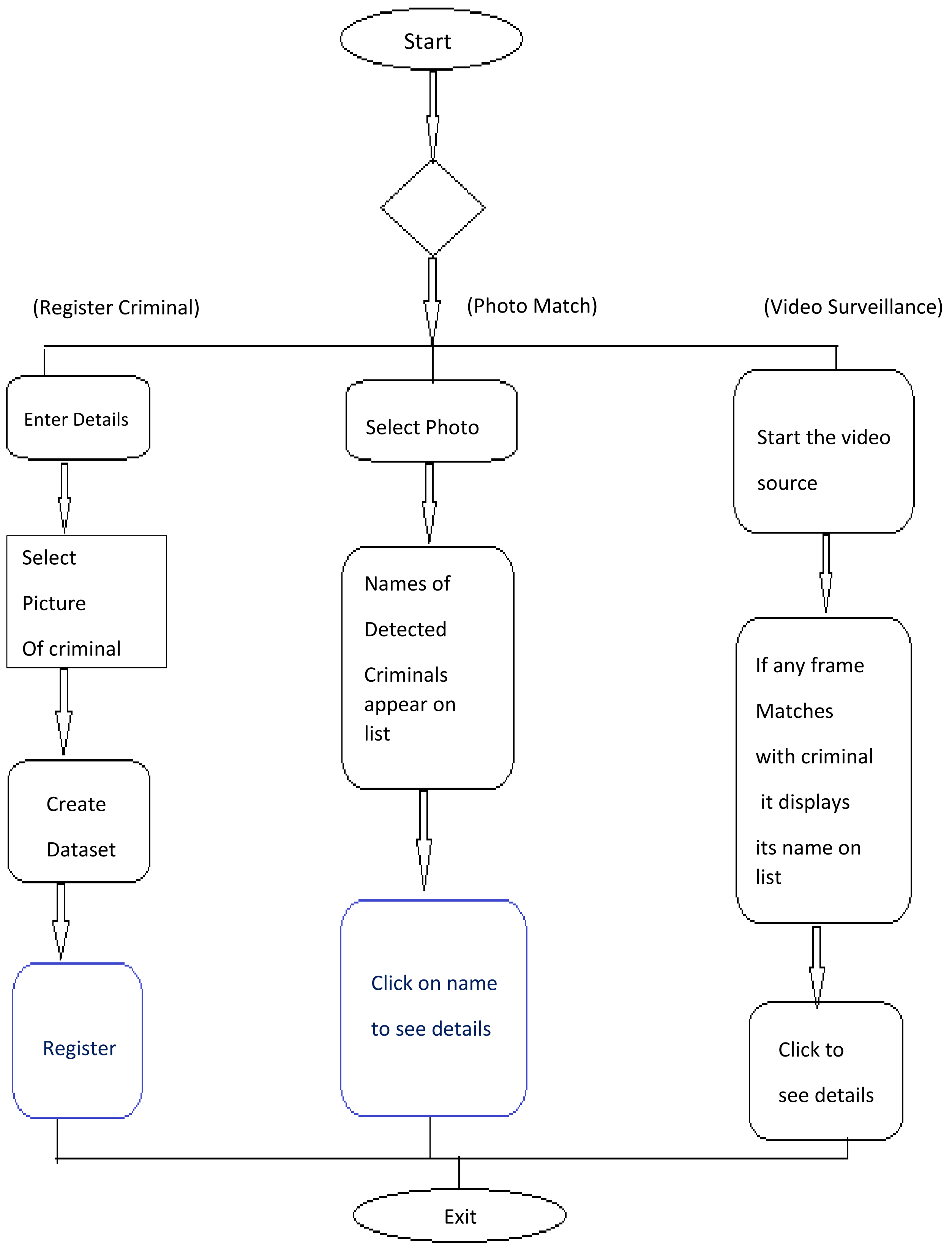


Fig 3.1 Dataset of a person generated by System

E. Testing

For testing the Face Recognizer, we check if the recognition was correct by seeing the predicted label when we bring the trained face in front of camera. The label is extracted using the os module and the string operations from the name of the sample images folder. Lower is the confidence score better is the prediction.

3.2 Process Flow Diagram



Start
(Register Criminal)
(Photo Match)
(Video Surveillance)
Enter Details
Select Picture Of criminal
Create Dataset
Register
Select Photo
Names of Detected Criminals appear on list
Click on name to see details
Start the video source
If any frame Matches with criminal it displays its name on list
Click to see details
Exit

Fig 3.2 Process flow diagram of CFIS.

Chapter 4: Experimental Results

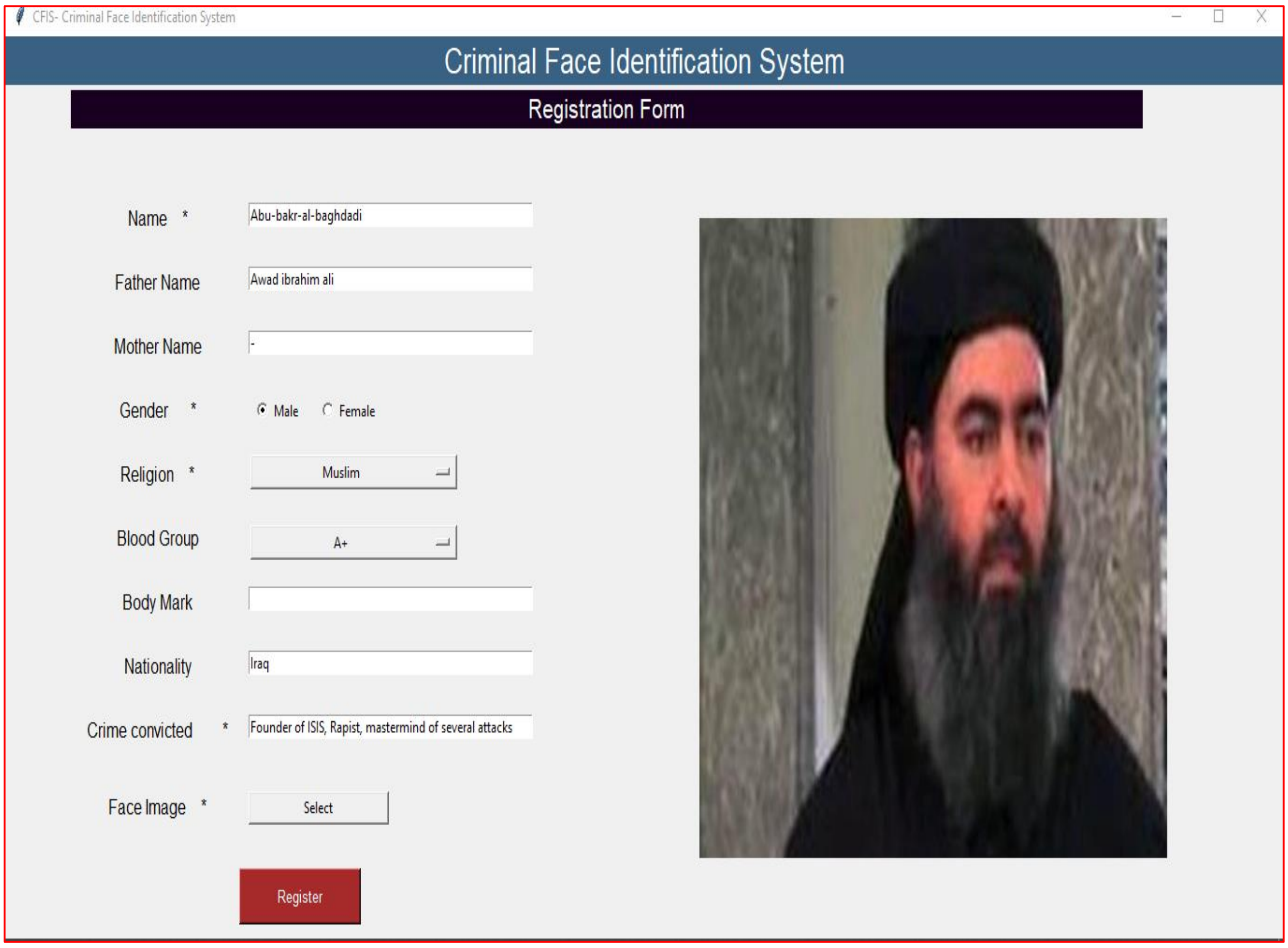
A. Homepage

Homepage is the main page of Criminal Identification System application. It contains three buttons for: Register Criminal, Photo match and Video Surveillance.

As shown in Fig 1.1

B. Criminal Registration

Criminal Registration page will take atleast 20 images of the criminal while creating dataset of criminal that needs to be registered and also provides input form for providing various details of the criminal like his Name, DOB, Identification mark, Profile picture etc. After selecting images and filling details, user will click register. The criminal will be successfully registered if any error doesn't occur.



CFIS- Criminal Face Identification System

Criminal Face Identification System

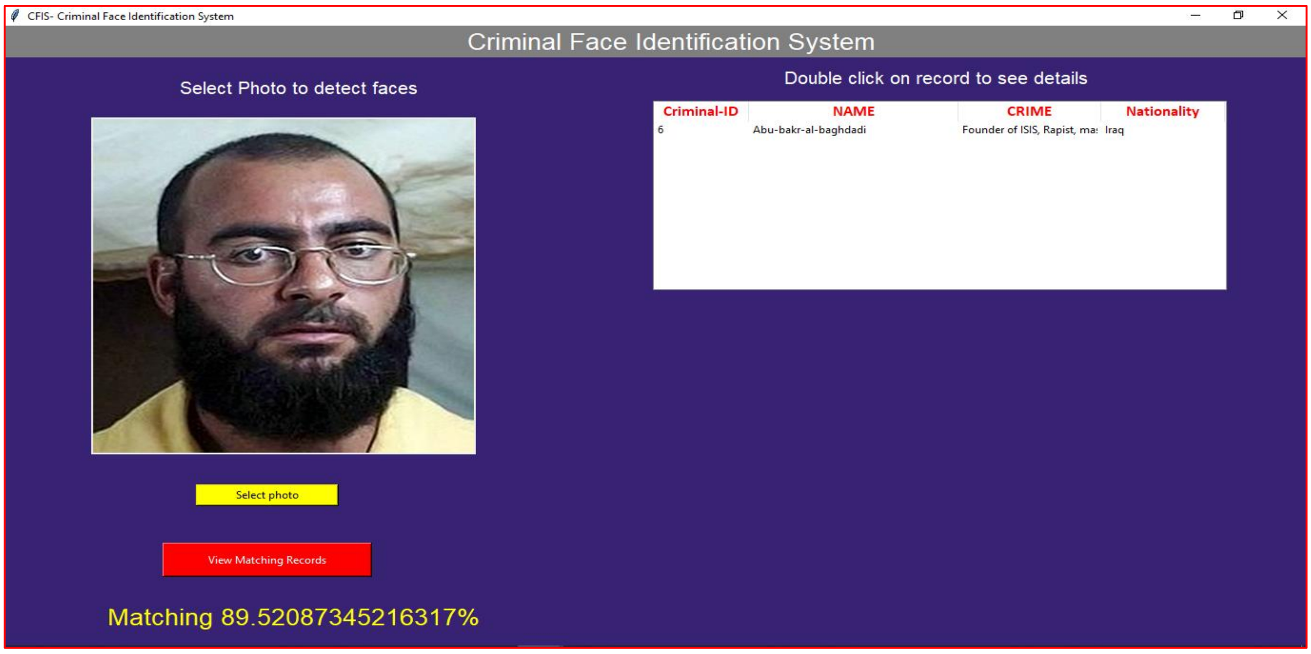
Registration Form

Name *	Abu-bakr-al-baghdadi
Father Name	Awad ibrahim ali
Mother Name	-
Gender *	[✓ Male] Female
Religion *	Muslim
Blood Group	A+
Body Mark	
Nationality	Iraq
Crime convicted *	Founder of ISIS, Rapist, mastermind of several attacks
Face Image *	Select

Fig 4.1 Registration page

C. Detect Criminal Page

This page allows the user to browse an image from the system and helps in detecting one or more criminals in it. User can also see the profile of the criminal by clicking on detected criminal names.


CFIS- Criminal Face Identification System

Criminal Face Identification System

Select Photo to detect faces

Double click on record to see details

Criminal-ID	NAME	CRIME	Nationality
6	Abu-bakr-al-baghdadi	Founder of ISIS, Rapist, ma:	Iraq

Select photo

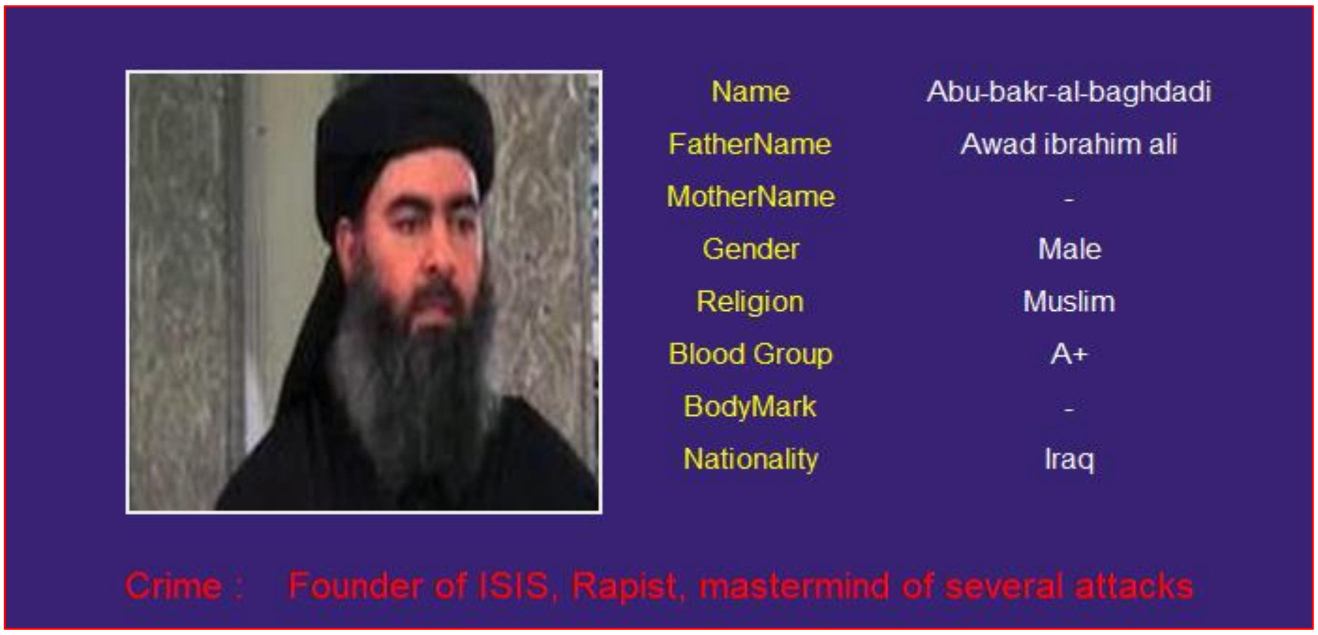
View Matching Records

Matching 89.52087345216317%

Fig 4.2 Detect face from image

D. Criminal Profile Page

This page will show criminal profile after clicking criminal name from detect criminal or video surveillance page.



Name	Abu-bakr-al-baghdadi
FatherName	Awad ibrahim ali
MotherName	-
Gender	Male
Religion	Muslim
Blood Group	A+
BodyMark	-
Nationality	Iraq

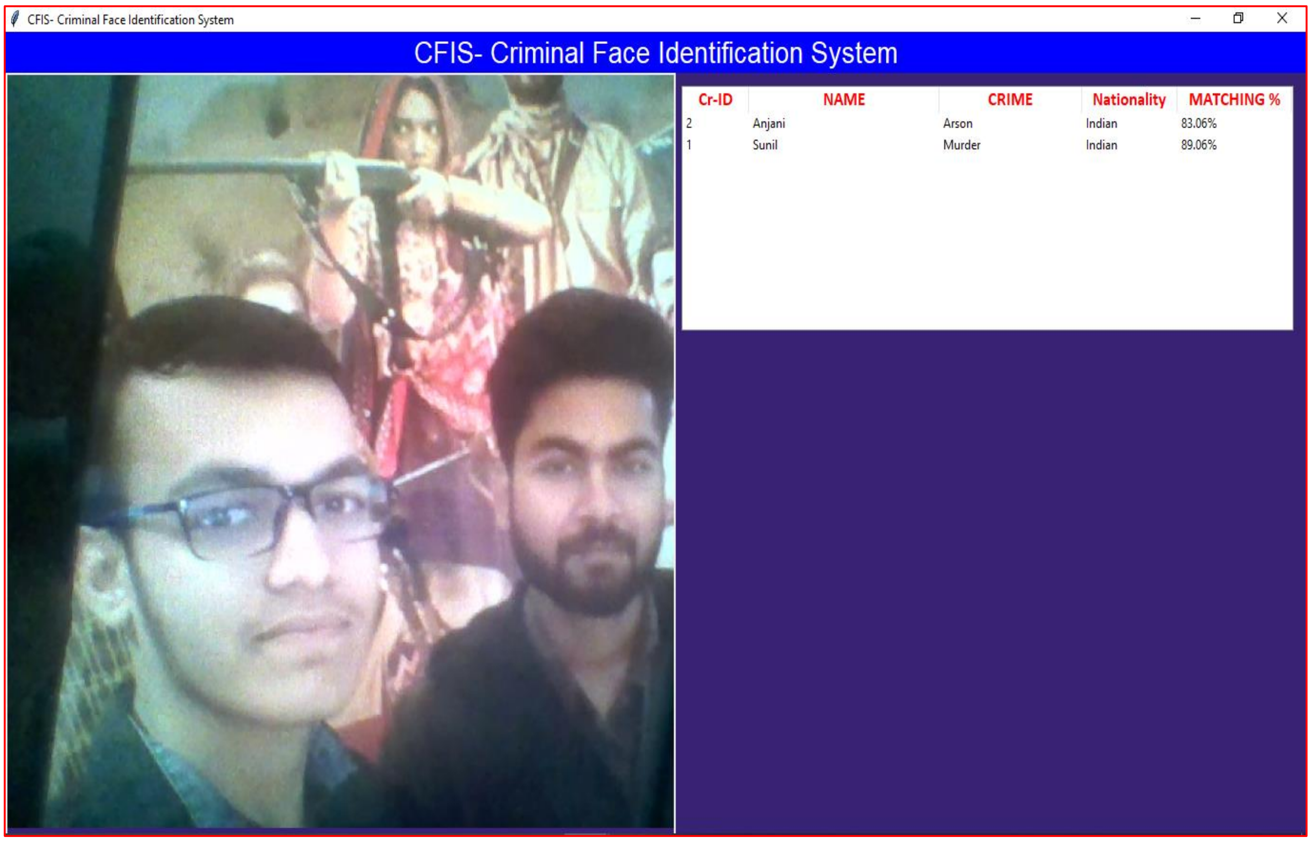
Crime : Founder of ISIS, Rapist, mastermind of several attacks

Fig 4.3 Details of Criminal

E. Detecting Unknown Criminal face

Our system is able to identify a non-criminal face.

F. Video Surveillance


CFIS- Criminal Face Identification System
CFIS- Criminal Face Identification System

Cr-ID	NAME	CRIME	Nationality	MATCHING %
2	Anjani	Arson	Indian	83.06%
1	Sunil	Murder	Indian	89.06%

Fig 4.4 Video Surveillance

This page will use the pc webcam to capture the video frames in real time. After this it will use face detection module on each frame to detect and recognize criminals in the video in real time. User can also see the profile of the criminal by clicking on detected criminal names.

Chapter 5: Conclusion

5.1 Conclusion

In this project, we are able to detect and recognize faces of the criminals in an image and in a video stream obtained from a camera in real time. We have used Haar feature-based cascade classifiers in OpenCV approach for face detection. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. Also, we have used Local Binary Patterns Histograms (LBPH) for face recognition. The project will be a milestone for video based face identification and for surveillance systems.

Several advantages of this algorithm are: Efficient feature selection, Scale and location invariant detector, instead of scaling the image itself, we scale the features such a generic detection scheme can be trained for detection of other types of objects (e.g. cars, sign boards, number plates, etc). LBPH recognizer can recognize faces in different lighting conditions with high accuracy. Also, LBPH can recognize efficiently even if single training image is used for each person. Our application has some disadvantages like: Detector is most effective only on frontal images of faces, it can hardly cope with 45° face rotation both around the vertical and horizontal axis.

5.2 Future Works:

- Light normalization may allow the threshold value to increase.
- Future work may include improvement of face recognition using specific character in the face (distance b/w eyes) and also analyse the face in 3D by using more than one camera. Using these two method the probability of error will decrease and system will be more accurate.

References

[1] LBPH Based Improved Face Recognition at Low Resolution

Aftab Ahmed, Jiandong Guo, Fayaz Ali, Farha Deeba and Awais Ahmed

2018 International Conference of Artificial Intelligence and Big Data, China

[2] Criminal Identification System Using Face Detection and Recognition

Piyush Kakkar and Vibhor Sharma

International Journal of Advance Research in Computer and Communication Technology
Vol 7 issue 3 march 2018

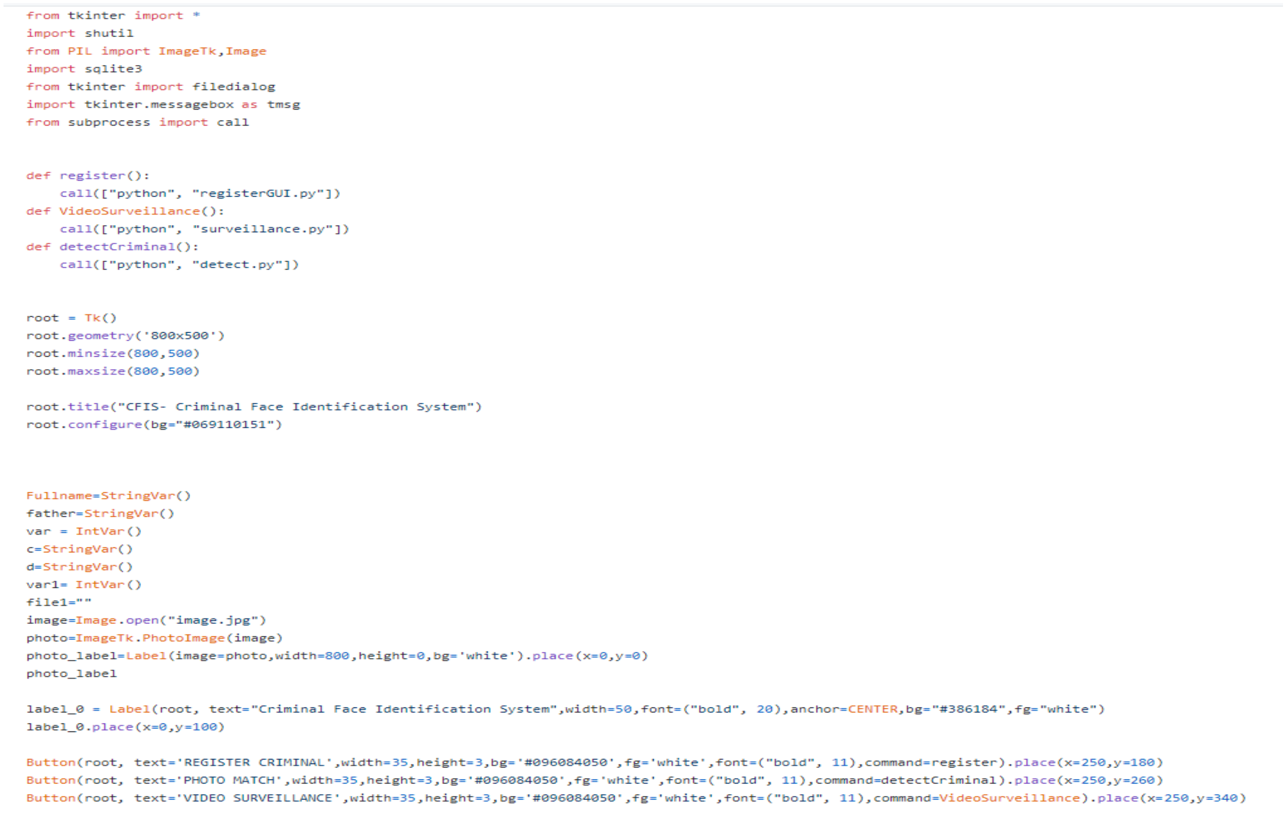
[3] Criminal Face Recognition System.

Alireza Chevelwalla, Ajay Gurav, Sachin Desai and Prof. Sumitra Sadhukhan
International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 4 Issue 03, March-2015

Annexure

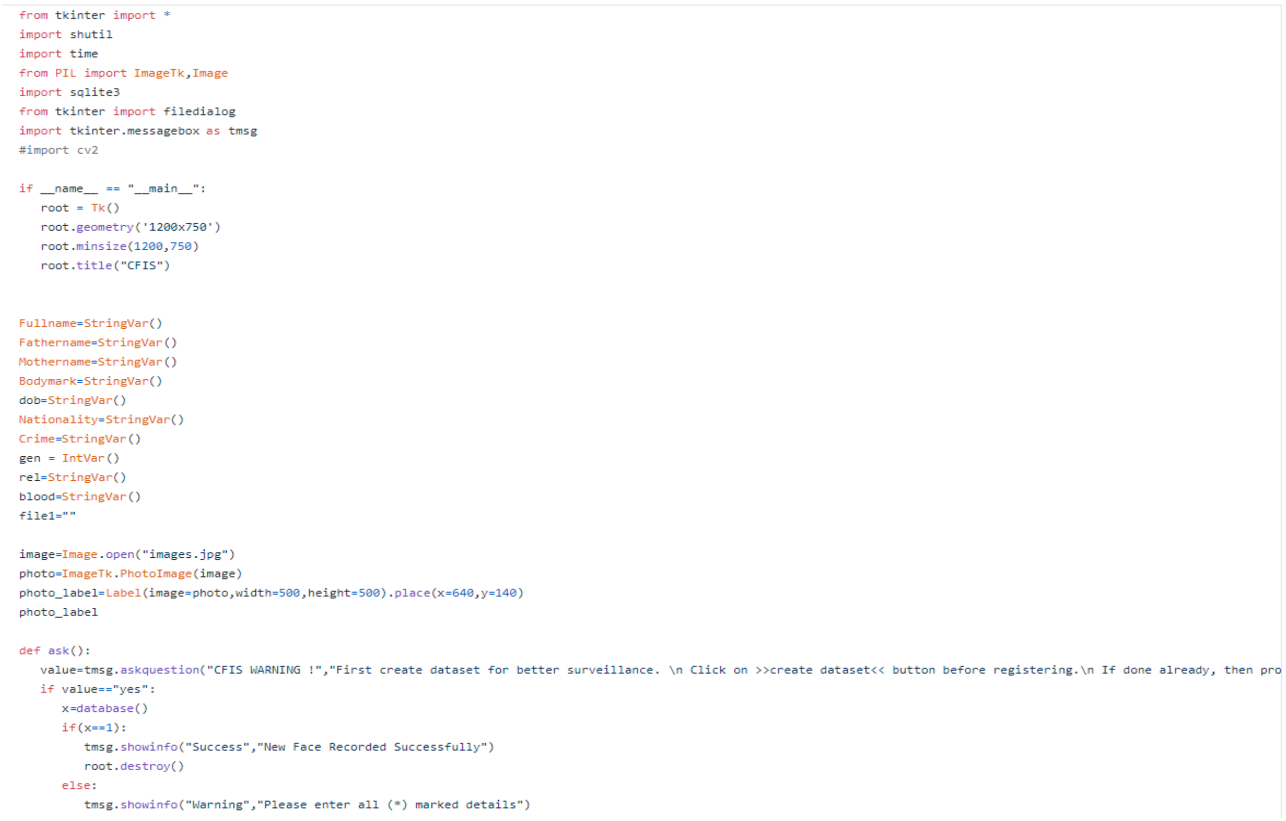
Code snippet

Start Page



```
from tkinter import *
import shutil
from PIL import ImageTk,Image
import sqlite3
from tkinter import filedialog
import tkinter.messagebox as tmsg
from subprocess import call


def register():
    call(["python", "registerGUI.py"])
def VideoSurveillance():
    call(["python", "surveillance.py"])
def detectCriminal():
    call(["python", "detect.py"])


root = Tk()
root.geometry('800x500')
root.minsize(800,500)
root.maxsize(800,500)

root.title("CFIS- Criminal Face Identification System")
root.configure(bg="#069110151")



Fullname=StringVar()
father=StringVar()
var = IntVar()
c=StringVar()
d=StringVar()
var1= IntVar()
file1=""
image=Image.open("image.jpg")
photo=ImageTk.PhotoImage(image)
photo_label=Label(image=photo,width=800,height=0,bg='white').place(x=0,y=0)
photo_label

label_0 = Label(root, text="Criminal Face Identification System",width=50,font=("bold", 20),anchor=CENTER,bg="#386184",fg="white")
label_0.place(x=0,y=100)

Button(root, text='REGISTER CRIMINAL',width=35,height=3,bg='#096084050',fg='white',font=("bold", 11),command=register).place(x=250,y=180)
Button(root, text='PHOTO MATCH',width=35,height=3,bg='#096084050',fg='white',font=("bold", 11),command=detectCriminal).place(x=250,y=260)
Button(root, text='VIDEO SURVEILLANCE',width=35,height=3,bg='#096084050',fg='white',font=("bold", 11),command=VideoSurveillance).place(x=250,y=340)
```

Register Page



```
from tkinter import *
import shutil
import time
from PIL import ImageTk,Image
import sqlite3
from tkinter import filedialog
import tkinter.messagebox as tmsg
#import cv2

if __name__ == "__main__":
   root = Tk()
   root.geometry('1200x750')
   root.minsize(1200,750)
   root.title("CFIS")


Fullname=StringVar()
Fathername=StringVar()
Mothername=StringVar()
Bodymark=StringVar()
dob=StringVar()
Nationality=StringVar()
Crime=StringVar()
gen = IntVar()
rel=StringVar()
blood=StringVar()
file1=""

image=Image.open("images.jpg")
photo=ImageTk.PhotoImage(image)
photo_label=Label(image=photo,width=500,height=500).place(x=640,y=140)
photo_label

def ask():
   value=tmsg.askquestion("CFIS WARNING !","First create dataset for better surveillance. \n Click on >>create dataset<< button before registering.\n If done already, then pro
   if value=="yes":
      x=database()
      if(x==1):
         tmsg.showinfo("Success","New Face Recorded Successfully")
         root.destroy()
      else:
         tmsg.showinfo("Warning","Please enter all (*) marked details")
```

```
############################################################
def dataset():
   database()

   detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
   cam = cv2.VideoCapture(0)

   x=getid()

   print(x)
   print(str(x))

   sampleNum=0
   time.sleep(2)
   while(True):
       ret, img = cam.read()
       gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
       faces = detector.detectMultiScale(gray, 1.3, 5)
       for (x,y,w,h) in faces:
           cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)
           #incrementing sample number
           sampleNum=sampleNum+1
           #saving the captured face in the dataset folder
           cv2.imwrite("dataSet/User."+str(x) +'.'+ str(sampleNum) + ".jpg", gray[y:y+h,x:x+w])
           cv2.waitKey(100)

       cv2.imshow('face',img)
       #wait for 100 miliseconds
       cv2.waitKey(1)
       # break if the sample number is morethan 20
       if(sampleNum>20):
           break

   ret, frame = cam.read()

   if ret:
      cv2.imwrite("images/user." + str(x) + ".png", cv2.cvtColor(frame, cv2.COLOR_RGB2BGR))

   cam.release()
   cv2.destroyAllWindows()
   tmsg.showinfo("Success","New Face Recorded Successfully")
   root.destroy()

############################################################
def getid():
   conn = sqlite3.connect('criminal.db')
   with conn:
      cursor=conn.cursor()
   cursor.execute('select max(ID) from People')
   conn.commit()
   for row in cursor:
    for elem in row:
        x = elem
   return x

def database():
   name=Fullname.get()
   father=Fathername.get()
   mother=Mothername.get()
   bl=blood.get()
   if(bl=="Select Blood Group"):
      bl=None
   body=Bodymark.get()
   nat=Nationality.get()
   crime=Crime.get()
   Dob=dob.get()
   gen1=""
   gender=gen.get()
   if(gender==1):
      gen1='Male'
   if(gender==2):
      gen1='Female'
   religion=rel.get()
   if(religion=="Select Religion"):
      religion=None

   if(name!="" and crime!="" and gen1!=""):
      conn = sqlite3.connect('criminal.db')
      with conn:
         cursor=conn.cursor()
      #cursor.execute('CREATE TABLE IF NOT EXISTS People (Fullname TEXT,Email TEXT,Gender TEXT,country TEXT,Programming TEXT)')
      cursor.execute('INSERT INTO People (Name,Gender,Father,Mother,Religion,Blood,Bodymark,Nationality,Crime) VALUES(?,?,?,?,?,?,?,?,?)',(name,ger
      conn.commit()
      x=getid()
      file="images/user."+str(x)+".png"
      newPath = shutil.copy('temp/1.png',file)
   else:
      return 0
   return 1
```

Trainer.py

```
import cv2,os
import numpy as np
from PIL import Image

recognizer = cv2.face.LBPHFaceRecognizer_create()
path="dataSet"

def getImgID(path):
    #get the path of all the files in the folder
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    #create empth face list
    faces=[]
    #create empty ID list
    Ids=[]
    #now looping through all the image paths and loading the Ids and the images
    for imagePath in imagePaths:
        #loading the image and converting it to gray scale
        faceImage=Image.open(imagePath).convert('L')
        #Now we are converting the PIL image into numpy array
        faceNp=np.array(faceImage,'uint8')
        #getting the Id from the image
        Id=int(os.path.split(imagePath)[-1].split(".")[1])
        faces.append(faceNp)
        Ids.append(Id)
        #cv2.imshow("training",faceNp)
        #cv2.waitKey(10)
        # extract the face from the training image sample
        #faces=detector.detectMultiScale(imageNp)
         #If a face is there then append that in the list as well as Id of it
    return Ids,faces


Ids,faces = getImgID(path)
#print(Ids,faces)
recognizer.train(faces, np.array(Ids))
recognizer.write('recognizer\\training_data.yml')
cv2.destroyAllWindows()
```

Surviellance.py

```
import cv2
import numpy as np
import sqlite3
import face_recognition as fr
import numpy as np
from tkinter import *
from tkinter import ttk
from PIL import Image,ImageTk
import os
import imutils
import math
import winsound

####################################################################################################

class App:
        def __init__(self,video_source=0):
                self.appname="CFIS- Criminal Face Identification System"
                self.window=Tk()
                self.window.title(self.appname)
                self.window.geometry('1350x720')
                self.window.state("zoomed")
                self.window["bg"]='#382273'
                self.video_source=video_source
                self.vid=myvideocapture(self.video_source)
                self.label=Label(self.window,text=self.appname,font=("bold",20),bg='blue',fg='white').pack(side=TOP,fill=BOTH)
                self.canvas=Canvas(self.window,height=700,width=700,bg='#382273')
                self.canvas.pack(side=LEFT,fill=BOTH)
                self.detectedPeople=[]
                self.images=self.load_images_from_folder("images")

                #get image names
                self.images_name=[]
                for img in self.images:
                        self.images_name.append(fr.load_image_file(os.path.join("images",img)))

                #get their encodings
                self.encodings=[]
                for img in self.images_name:
                        self.encodings.append(fr.face_encodings(img)[0])


                #get id from images
                self.known_face_names=[]
                for name in self.images:
                        self.known_face_names.append((os.path.splitext(name)[0]).split('.')[1])


                self.face_locations=[]
                self.face_encodings=[]
                self.face_names=[]
                self.process_this_frame=True



                print(self.known_face_names)
                self.faceDetect = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
                self.recognizer = cv2.face.LBPHFaceRecognizer_create()
                # self.recognizer.read("recognizer\\training_data.yml")
                self.Id=0


                #== showing treeview
                self.tree= ttk.Treeview(self.window, column=("column1", "column2", "column3","column4","column5"), show='headings')

                self.tree.heading("#1", text="Cr-ID")
                self.tree.column("#1", minwidth=0, width=70, stretch=NO)

                self.tree.heading("#2", text="NAME")
                self.tree.column("#2", minwidth=0, width=200, stretch=NO)

                self.tree.heading("#3", text="CRIME")
                self.tree.column("#3", minwidth=0, width=150, stretch=NO)

                self.tree.heading("#4", text="Nationality")
                self.tree.column("#4", minwidth=0, width=100, stretch=NO)

                self.tree.heading("#5", text="MATCHING %")
                self.tree.column("#5", minwidth=0, width=120, stretch=NO)

                ttk.Style().configure("Treeview.Heading",font=('Calibri', 13,'bold'), foreground="red", relief="flat")

                self.tree.place(x=710,y=50)

                self.update()
```

```
def load_images_from_folder(self,folder):
        images=[]
        for filename in os.listdir(folder):
                images.append(filename)
        return images

def doubleclick(self,event):
        item=self.tree.selection()
        itemid=self.tree.item(item,"values")
        ide=itemid[0]
        ide=(int(ide))
        self.viewdetail(ide)

def viewdetail(self,a):
        conn = sqlite3.connect("criminal.db")
        cur = conn.cursor()
        cur.execute("SELECT * FROM people where Id="+str(a))
        rows = cur.fetchall()
        print(rows)
        for row in rows:
                label_n = Label(self.window, text=row[1],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_n.place(x=1130,y=400)
                label_f = Label(self.window, text=row[3],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_f.place(x=1130,y=430)
                label_m = Label(self.window, text=row[4],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_m.place(x=1130,y=460)
                label_g = Label(self.window, text=row[2],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_g.place(x=1130,y=490)
                label_r = Label(self.window, text=row[5],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_r.place(x=1130,y=520)
                label_bl = Label(self.window, text=row[6],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_bl.place(x=1130,y=550)
                label_b = Label(self.window, text=row[7],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_b.place(x=1130,y=580)
                label_n = Label(self.window, text=row[8],bg="#382273",fg='white',width=20,font=("bold", 12))
                label_n.place(x=1130,y=610)
                label_c = Label(self.window, text=row[9],width=30,bg="#382273",font=("bold", 15),fg="red")
                label_c.place(x=1060,y=640)
        conn.close()
        label_name = Label(self.window, text="Name",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_name.place(x=930,y=400)
        label_father = Label(self.window, text="FatherName",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_father.place(x=930,y=430)
        label_mother = Label(self.window, text="MotherName",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_mother.place(x=930,y=460)
        label_gender = Label(self.window, text="Gender",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_gender.place(x=930,y=490)
        label_religion = Label(self.window, text="Religion",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_religion.place(x=930,y=520)
        label_bloodgroup = Label(self.window, text="Blood Group",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_bloodgroup.place(x=930,y=550)
        label_body = Label(self.window, text="BodyMark",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_body.place(x=930,y=580)
        label_nat = Label(self.window, text="Nationality",bg="#382273",fg='yellow',width=20,font=("bold", 12))
        label_nat.place(x=930,y=610)
        label_crime = Label(self.window, text="Crime",bg="#382273",width=23,font=("bold", 15),fg="red")
        label_crime.place(x=900,y=640)


        x='user.'+str(a)+".png"
        image=Image.open('images/'+x)
        image = image.resize((180,180), Image.ANTIALIAS)
        photo=ImageTk.PhotoImage(image)
        photo_l=Label(image=photo,width=180,height=180).place(x=750,y=450).pack()
```

```
    def update(self):
        isTrue,frame=self.vid.getframe()
        if isTrue:
            self.photo=ImageTk.PhotoImage(image=Image.fromarray(frame))
            self.canvas.create_image(0,0,image=self.photo,anchor=NW)

            #Resize the frame of video to 1/4 size for fast process
            small_frame=cv2.resize(frame,(0,0),fx=0.25,fy=0.25)

            #convert the image to BGR color(openCV) to RGB color(face_recognition)
            rgb_small_frame=small_frame[:,:,::-1]

            #Only process every other frame of video to save time
            if self.process_this_frame:
                #find all the faces and face encodings in the current frame of video
                self.face_locations=fr.face_locations(rgb_small_frame)
                self.face_encodings=fr.face_encodings(rgb_small_frame,self.face_locations)
                self.face_names=[]
                for face_encoding in self.face_encodings:
                    #See if the face is a match for known face(s)
                    matches=fr.compare_faces(self.encodings,face_encoding)
                    Id=0
                    face_distances=fr.face_distance(self.encodings,face_encoding)
                    best_match_index=np.argmin(face_distances)

                    percent=self.showPercentageMatch(face_distances[best_match_index])

                    #acc = accuracy_score(self.encodings[best_match_index], face_encoding)

                    if matches[best_match_index]:
                        Id=self.known_face_names[best_match_index]
                    self.face_names.append(Id)

            # self.gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            # faces=self.faceDetect.detectMultiScale(self.gray, 1.2, 5)
            # for(x,y,w,h) in faces:
            #       cv2.rectangle(frame,(x,y),(x+w,y+h),(225,0,0),2)
            #       Id, confidence = self.recognizer.predict(self.gray[y:y+h,x:x+w])

                    profile=self.getProfile(Id)
                    confidence=str(round(percent*100,2))+"%"

                    if profile not in self.detectedPeople and profile!=None:
                        self.detectedPeople.append(profile)
                        profilex=list(profile)
                        profilex.append(confidence)
                        profile=tuple(profilex)
                        self.tree.insert("", 'end', values=profile)
                        self.tree.bind("<Double-1>",self.doubleclick)
                        winsound.PlaySound("SystemExit", winsound.SND_ALIAS)

                    print(profile)
            self.process_this_frame=not self.process_this_frame
            # #display the result
            # for(top,right,bottom,left),name in zip(self.face_locations,self.face_names):
            #       top*=4
            #       right*=4
            #       bottom*=4
            #       left*=4
            #       cv2.rectangle(frame,(left,top),(right,bottom),(0,0,225),2)
            #       cv2.rectangle(frame,(left,bottom-35),(right,bottom),(0,0,225),cv2.FILLED)
            #       font=cv2.FONT_HERSHEY_DUPLEX
            #       cv2.putText(frame,name,(left+6,bottom-6),font,1.0,(225,225,225),1)

        self.window.after(15,self.update)

#########################################################################################################
class myvideocapture:
    def __init__(self,video_source=0):
        self.vid=cv2.VideoCapture(video_source)
        if not self.vid.isOpened():
            raise ValueError("unable to open",video_source)

        self.width=self.vid.get(cv2.CAP_PROP_FRAME_WIDTH)
        self.height=self.vid.get(cv2.CAP_PROP_FRAME_HEIGHT)

    def getframe(self):
        if self.vid.isOpened():
            ret, frame = self.vid.read()
            frame=imutils.resize(frame,height=700)
            if ret:
                return (ret, cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
            else:
                return (ret, None)
        else:
            return (ret, None)

    def __del__(self):
        if self.vid.isOpened():
            self.vid.release()

if __name__=="__main__":
    App()
```

Detect.py

```
from tkinter import *
from tkinter import ttk
import shutil
from PIL import ImageTk,Image
import sqlite3
from tkinter import filedialog
import tkinter.messagebox as tmsg
import cv2,os


if __name__ == "__main__":
   root = Tk()
   root.geometry('1300x750')
   root.minsize(1300,750)
   root.title("CFIS")
image=Image.open("images.jpg")
photo=ImageTk.PhotoImage(image)
photo_label=Label(image=photo,width=400,height=400).place(x=90,y=110)
photo_label

label_1 = Label(root, text="Select Photo to detect faces",width=50,font=("bold", 15))
label_1.place(x=30,y=60)
label_177 = Label(root, text="Double click on record to see details",width=50,font=("bold", 15))
label_177.place(x=650,y=48)

label_0 = Label(root, text="Criminal Face Identification System",width=80,font=("bold", 20),anchor=CENTER,bg="grey",fg="white")
label_0.place(x=0,y=0)
'''
def View():
    conn = sqlite3.connect("TRIAL.db")
    cur = conn.cursor()
    cur.execute("SELECT * FROM profile")
    rows = cur.fetchall()
    for row in rows:
        print(row) # it print all records in the database
        tree.insert("", tkinter.END, values=row)
    conn.close()
'''
#####################################################################
###############  Get data ###########################################
cascadePath = "haarcascade_frontalface_default.xml"
faceDetect = cv2.CascadeClassifier(cascadePath)
recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read("recognizer\\training_data.yml")

##################################################################3###
def viewdetail(a):
   conn = sqlite3.connect("criminal.db")
   cur = conn.cursor()
   cur.execute("SELECT * FROM people where Id="+str(a))
   rows = cur.fetchall()
   print(rows)
   for row in rows:
      label_n = Label(root, text=row[1],width=20,font=("bold", 12))
      label_n.place(x=1000,y=400)
      label_f = Label(root, text=row[3],width=20,font=("bold", 12))
      label_f.place(x=1000,y=430)
      label_m = Label(root, text=row[4],width=20,font=("bold", 12))
      label_m.place(x=1000,y=460)
      label_g = Label(root, text=row[2],width=20,font=("bold", 12))
      label_g.place(x=1000,y=490)
      label_r = Label(root, text=row[5],width=20,font=("bold", 12))
      label_r.place(x=1000,y=520)
      label_bl = Label(root, text=row[6],width=20,font=("bold", 12))
      label_bl.place(x=1000,y=550)
      label_b = Label(root, text=row[7],width=20,font=("bold", 12))
      label_b.place(x=1000,y=580)
      label_n = Label(root, text=row[8],width=20,font=("bold", 12))
      label_n.place(x=1000,y=610)
      label_c = Label(root, text=row[9],width=30,font=("bold", 15),fg="red")
      label_c.place(x=930,y=640)
      #label_d = Label(root, text=row[10],width=20,font=("bold", 12))
      #label_d.place(x=1000,y=670)
      # it print all records in the database
   conn.close()
   ##############################################################################
   label_name = Label(root, text="Name",width=20,font=("bold", 12))
   label_name.place(x=830,y=400)
   label_father = Label(root, text="FatherName",width=20,font=("bold", 12))
   label_father.place(x=830,y=430)
   label_mother = Label(root, text="MotherName",width=20,font=("bold", 12))
   label_mother.place(x=830,y=460)
   label_gender = Label(root, text="Gender",width=20,font=("bold", 12))
   label_gender.place(x=830,y=490)
   label_religion = Label(root, text="Religion",width=20,font=("bold", 12))
   label_religion.place(x=830,y=520)
   label_bloodgroup = Label(root, text="Blood Group",width=20,font=("bold", 12))
   label_bloodgroup.place(x=830,y=550)
   label_body = Label(root, text="BodyMark",width=20,font=("bold", 12))
   label_body.place(x=830,y=580)
   label_nat = Label(root, text="Nationality",width=20,font=("bold", 12))
   label_nat.place(x=830,y=610)
   label_crime = Label(root, text="Crime",width=23,font=("bold", 15),fg="red")
   label_crime.place(x=780,y=640)
   ###########################################################################


   ###########################################################################
   x='user.'+str(a)+".png"
   image=Image.open('images/'+x)
   photo=ImageTk.PhotoImage(image)
   photo_1=Label(image=photo,width=250,height=250).place(x=590,y=400).pack()


def mfileopen():
   cleartree()
   file1=filedialog.askopenfilename()
   print(file1)
   newPath = shutil.copy(file1, 'temp/1.png')
   image=Image.open('temp/1.png')
   photo=ImageTk.PhotoImage(image)
   photolbl=Label(image=photo,width=400,height=400).place(x=90,y=110).pack()

def cleartree():
   records=tree.get_children()
   for el in records:
      tree.delete(el)

def doubleclick(event):
   item=tree.selection()
   itemid=tree.item(item,"values")
   ide=itemid[0]
   ide=(int(ide))
   viewdetail(ide)

def View():
    cleartree()
    im =cv2.imread("temp/1.png")
    gray=cv2.cvtColor(im,cv2.COLOR_BGR2GRAY)
    faces=faceDetect.detectMultiScale(gray, 1.2,5)
    Id, conf = recognizer.predict(gray)
    print(Id)
    conn = sqlite3.connect("criminal.db")
    cur = conn.cursor()
    cur.execute("SELECT ID,name,crime FROM people where ID="+str(Id))
    rows = cur.fetchall()
    print(conf)

    x=""
    if(conf>100):
       x="  No match!"
    elif(conf>80):
       x="  Low match!"
    elif(conf>50):
       x=" Good"
    else:
       x="  Best"
    for row in rows:
        print(row)
        # it print all records in the database
    conf=(100-float(conf))
    a="Matching "+str(conf)+"%" +x
    tree.insert("", 'end', values=row)
    tree.bind("<Double-1>",doubleclick)
    label_Match = Label(root, text=a,width=35,font=("bold", 20))
    label_Match.place(x=20,y=690)

    conn.close()

Fullname=StringVar()
father=StringVar()
var = IntVar()
c=StringVar()
d=StringVar()
var1= IntVar()
file1=""

btn=Button(text="Select photo",width=25,command=mfileopen).place(x=180,y=550)

#== showing treeview
tree= ttk.Treeview(root, column=("column1", "column2", "column3"), show='headings')
tree.heading("#1", text="Criminal-ID")
tree.heading("#2", text="NAME")
tree.heading("#3", text="CRIME")

tree.place(x=630,y=90)
```