Image Capturing and Automatic Face Recognition

Momtahina
Student Id: 011142103
Rakibul Hossain
Student Id: 011142104
Md. Mushfiqur Rahman
Student Id: 011142084
Onul Ashrafi Tania
Student Id: 011142006

A project in the Department of Computer Science and Engineering presented in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science and Engineering

United International University
Dhaka, Bangladesh

January 2019

Declaration

We, Momtahina, Rakibul Hossain, Md. Mushfiqur Rahman and Onul Ashrafi declare that this project titled, **"Image Capturing and Automatic Face Recognition**" and the work presented in it are our own. I confirm that:

- This work was done wholly or mainly while in candidature for a BSc degree at United International University.
- Where any part of this project has previously been submitted for a degree or any other qualification at United International University or any other institution, this has been clearly stated.
- Where we have consulted the published work of others, this is always clearly attributed.
- Where we have quoted from the work of others, the source is always given. With the exception of such quotations, this project is entirely our own work.
- We have acknowledged all the main sources of help.
- Where the project is based on work done by ourselves jointly with others, we have made clear exactly what was done by others and what we have contributed ourselves.

_____

Momtahina
ID: 011142103
Department of CSE
United International University
Dhaka-1212, Bangladesh

_____

Rakibul Hossain
ID: 011142104
Department of CSE
United International University
Dhaka-1212, Bangladesh

_____

Md. Mushfiqur Rahman
ID: 011142084
Department of CSE
United International University
Dhaka-1212, Bangladesh

_____

Onul Ashrafi Tania
ID: 011142006
Department of CSE
United International University
Dhaka-1212, Bangladesh

Certificate

We do hereby declare that the research works embodied in this project entitled "**Image Capturing and Automatic Face Recognition**" is the outcome of an original work carried out by Momtahina, Rakibul Hossain, Md. Mushfiqur Rahman and Onul Ashrafi Tania under my supervision.

I further certify that the dissertation meets the requirements and the standard for the degree of BSc in Computer Science and Engineering.

_____

Dr. Mohammad Nurul Huda
Professor & MSCSE Director
Department of CSE
United International University
Dhaka-1212, Bangladesh

Abstract

As we know, the face is the easiest way to distinguish someone's individual identity of each other. Face recognition is someone's personal identification system where it uses personal characteristics to identify his/her individuality.

A human face is a three-dimensional object that consists of illumination, pose, expression and these can be identified from a two-dimensional image. A human face recognition procedure has two phases. The first phase is face detection, finding where the face is located at a short distance away.
The next phase is an introduction which is recognizing a face as individuals.

There are different techniques for face recognition. For our project, we used different python libraries built with deep learning to make it simple in term of calculation and faster response.

Acknowledgment

First, we like to thank almighty for giving us enough courage and knowledge to complete our final project on time.

We extend our sincere thanks to our supervisor Prof. Dr. Mohammad Nurul Huda for providing guidance at every stage of this project work. We are profoundly grateful towards the unmatched services rendered by him.

We would like to thank all the people including our classmates for their inspiration throughout the work.

Last but not least, we would like to express our deep sense of gratitude and earnest thanks for giving to our dear parents for their moral support.

Table of Contents

LIST OF FIGURES

Chapter 1

Introduction

Face Recognition is a system to identify a “face” of a person from an image or a video frame. There are many methods or ways to recognize a face. Usually, we compare all facial features in an image or frame to recognize.
Generally, a Face Recognition application can detect or identify a person uniquely by analyzing facial features like shapes, textures or patterns. To recognize a face we have to go through two different processes. Firstly we have to detect a face in a frame, then we can recognize it. These two terms are different.

1.1 Face Detection

To recognize a face, we need to detect one first. For that, there are many algorithms, built in libraries etc. We’ve used HOG (Histogram of Oriented Gradients) as face detection algorithm.

- Model-based Face Tracking
    - Real-Time Face Detection Using Edge-Orientation Matching
    - Robust Face Detection Using the Hausdorff Distance
        - Genetic Model Optimization for Hausdorff Distance-Based Face Localization
- Weak classifier cascades
- HOGs and Deep Learning

Above mentioned algorithms are mostly used for face detection. We have used the HOG algorithm for face detection. Python’s ‘face_recog’ library provides the HOG algorithm.

1.2 Face Recognition

After being scaled or resized, cropped etc, and converted into grayscale, then we use face recognition algorithms to identify the face. There are several algorithms for face recognition. Some of them are very popular.

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

Each algorithm has a different approach to recognize. Some algorithm may look for size, shape, patterns, eye, ears, jaws every feature of a face. The other will not work the same way. Many algorithms get important data which will recognize a face, and then, compresses the face data. Then compare it with other face data.

With these two processes, we can recognize a face uniquely.

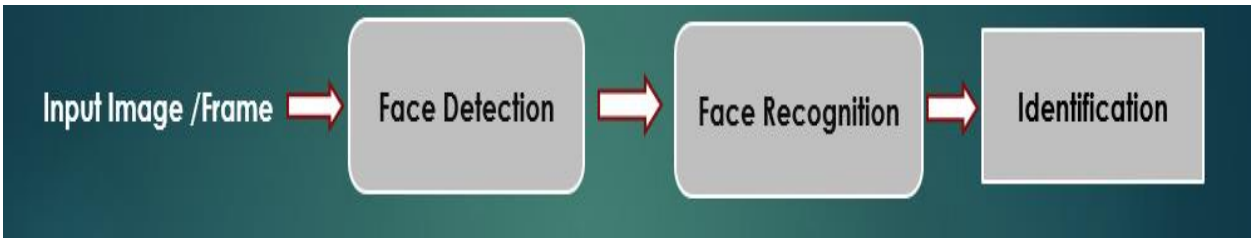


Fig 1.1: Face Recognition

Chapter 2

Machine Learning on Face Recognition

For analyzing visual imagery, Convolutional Neural Network or CNN which is a class of feedforward artificial neural network has been successfully used. It can be used also in video analysis, natural language processing etc. Here we'll talk about Machine Learning on Face Recognition.

Our goal is to identify a person's face through a deep neural network's output. So, we have to train our model to identify features, patterns from a face. If we pass different images of a person, neural network should give output similar. But, if we pass totally different image or new person's image it will give "Unknown" as output.

2.1 Facial Recognition

To apply machine learning on face recognition we can use a library named "face_recog", which is mostly used for face recognition. It is python's library, which uses dlib within itself. There are three major steps:

- Detection: Detect faces in frame/image
- Landmark: Finding features/patterns in frame/image
- Comparison: Identification of faces in pictures through comparison

2.2 Steps of recognition

The first step is to prepare the image. We've to prepare or process the image first. Then we can get all the necessary features from it.

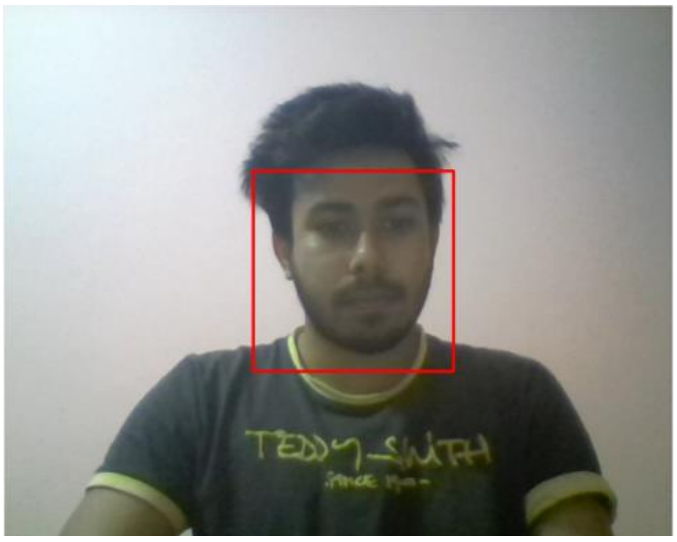
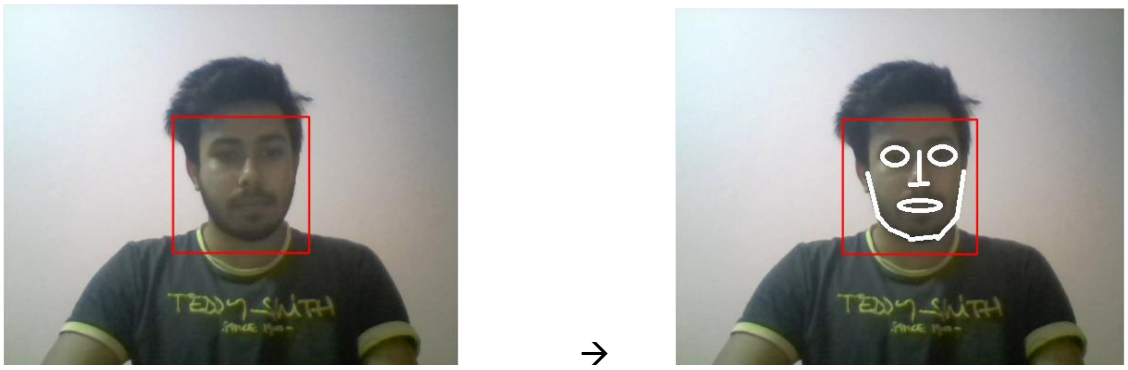


Fig 2.1: Preparing the image

After preparing the image, we'll manipulate the image to get the necessary information. For that, we need to get locations or outlines of a person's mouth, eye, ear etc.



Input → Output

Fig 2.2: Extracting important features from the image

After that, we analyze the features. Then we identify the image/face.

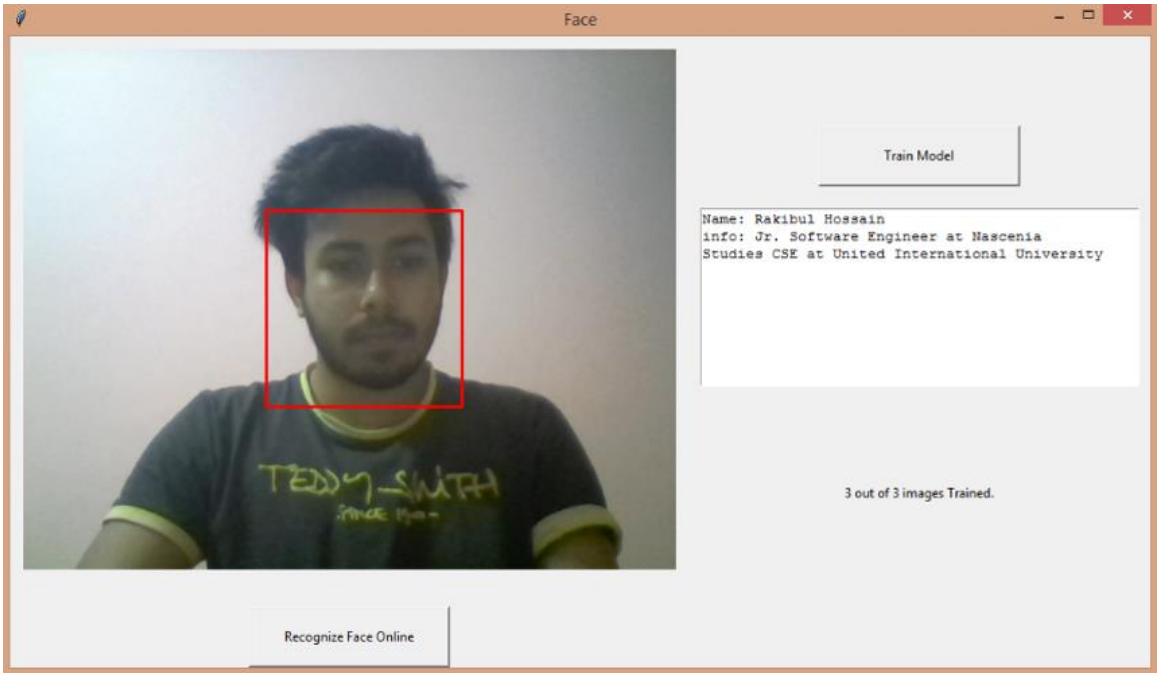


Fig 2.3: Successful Identification

This is how we identify a face. This process or technique can be used for multiple faces in a single frame/photo/image. For that, the list of inputs will be of N, and the outputs will be of N as well.

Basically, by doing a simple approach we can recognize who is in the image. Detection, Manipulation, Identification are the approaches. This clearly shows, how machine learning is taking over the world of artificial intelligence.

Chapter 3

Image Capturing & Automatic Face Recognition

3.1 Image Capturing & preprocessing

Capturing image is the first step for the face recognition process. As there is a various tool for image capturing, we used OpenCV (Open Source Computer Vision Library) which is a machine learning software library. It consists of more than 2500 optimized algorithms[3]. These algorithms are capable of detecting and recognizing faces from an image. As OpenCV was actually designed for real-time application and image processing, we used it to capture the photo in real time through the webcam. Some other advantages of OpenCV are:

- Open source and is available free of cost.
- Works fast as it is written in C/ C++.
- Portable and can run on any device that can run C.

Steps for image preprocessing:

Fig 3.1: Image capture and pre-processing

3.2 Recognition Process

3.2.1 Face Detection

For the recognition process, we need to detect the face first. There are many detection algorithms and the popular ones are:

- HOG (Histogram of Oriented Gradients).
- CNN (Convolutional Neural Network).

Histogram of Oriented Gradients:

The first step is to convert the input image to black and white. It only looks at the changes between light and dark areas in an image. It doesn't need color information. It takes a pixel and compares how dark this pixel is compared to the next pixels around it and find out in which direction the biggest changes happened.

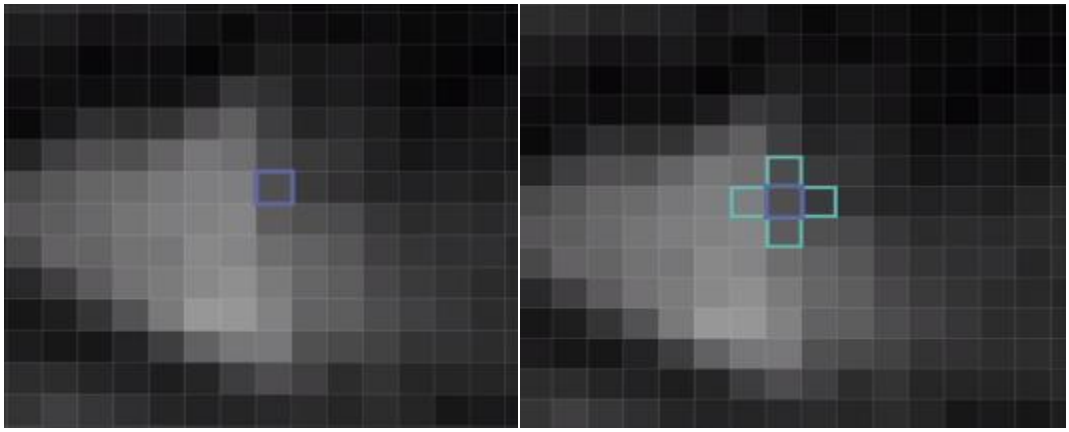


Fig 3.2: HOG dark pixel compare

In above the left pixel is lighter and the right pixel is darker. The direction for biggest change is left to right.

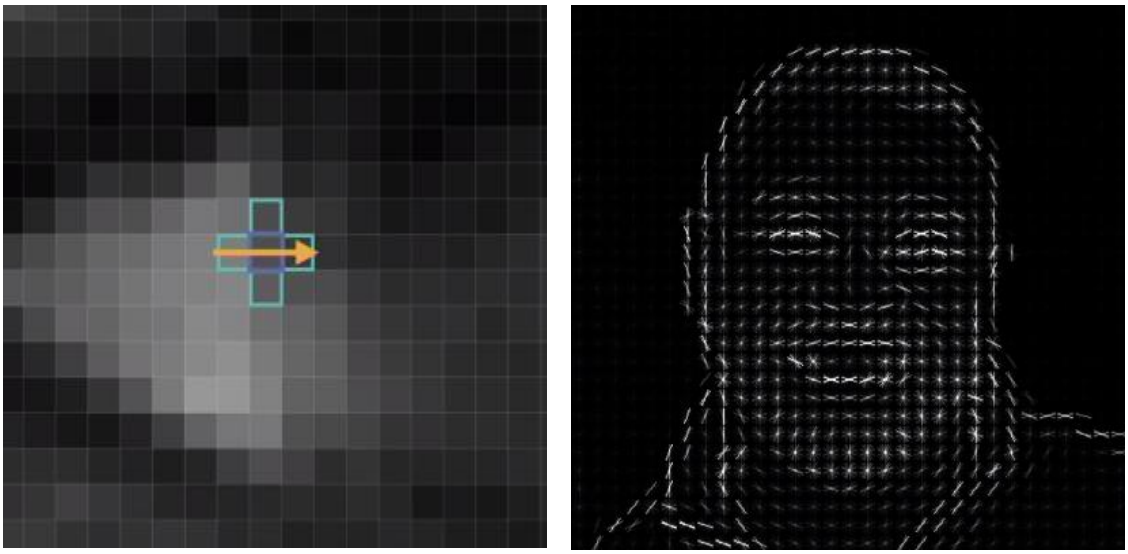


Fig 3.3: HOG gradient map

This shows the movements of the lighting of that exact point. If we repeat this process for every single pixel for an image then the image will turn into a map of transition from light to dark areas. After placing every pixel the full gradient map is the simplified version of the actual image. [4]

To detect a face all we need an overall structure of an image and that's what Hog does.

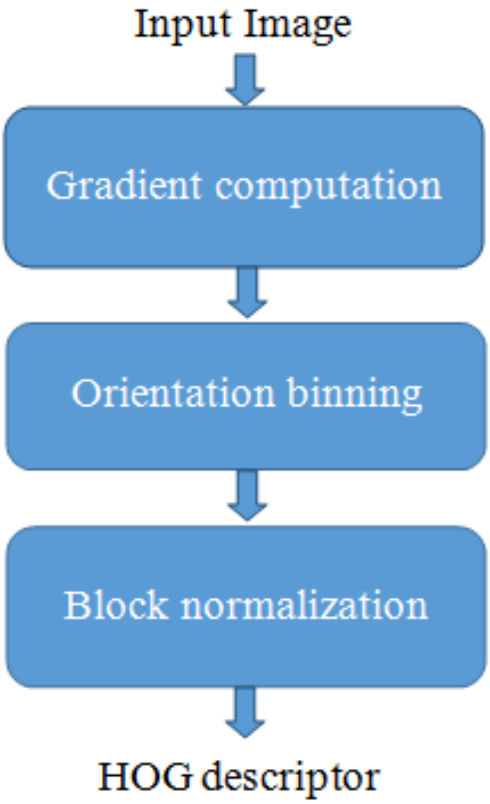


Fig 3.4: HOG process for image detection

Convolutional Neural Network:

A convolutional neural network (CNN) is a class of deep neural network. CNN is designed to recognize the variability of 2D shape.

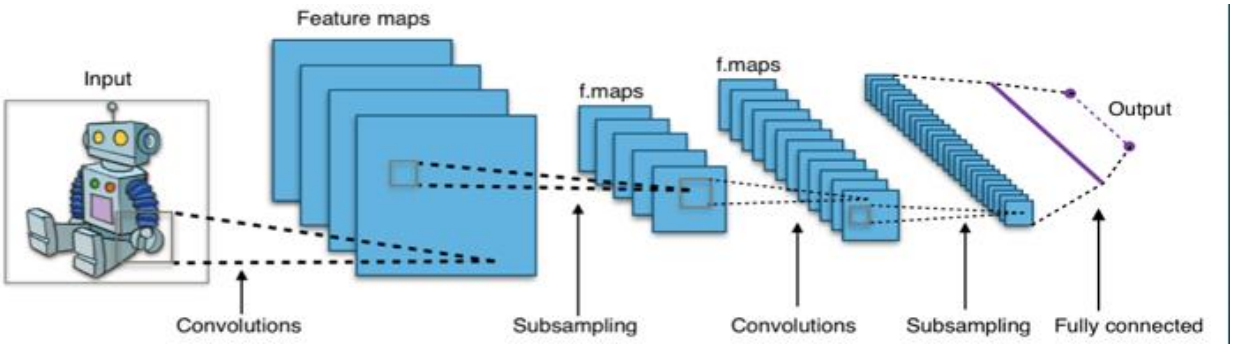


Fig 3.5: General Architecture of CNN

Advantages of the CNN model:

- Compared to other image classification algorithm CNN use little pre-processing.
- More accurate.

Detection method for our project:

Our project supports both HOG and CNN. For CNN, GPU acceleration (via NVidia's CUDA library) is required and has more accuracy. And for HOG, it has less accuracy than CNN but works faster for CPU.

3.2.2 Face Encoding From Image

For the recognition process, we need to encode the face from the image. For this, we used a python library which is Face Recognition.

face_recognition (python library):

- Built using dlib.
- Has an accuracy of 99.38%.
- Need only one image to train the model.
- Uses numpy, scipy, scikit-image, pillow etc.
- Make 128 dimension face encode for each other.[2]

Dlib:

Dlib is the underlying library of 'face_recognition' library. It's a C++ library and contains machine learning algorithms and tools and can solve a real-world problem like face recognition. Dlib model has an accuracy of 99.38% on the standard Labeled Faces in the wild benchmark[1].

The process of recognition by Dlib:

It actually compares two images and correctly predicts if both images are of the same person. It can predict 99.38% of the time which is pretty high compared to any recognition method. For our project, we used the ResNet model under the dlib library.

ResNet:

Dlib model is a ResNet network with 29 convolutional layers. It's a version of the ResNet-34 network and used for face recognition from Deep Residual Learning paper by He, Zhang, Ren, and Sun with a few layers removed and every layer was reduced by half by the number of filters.[1]

Face encoding from the database:

For our project, we used images from the database to compare with real-time image and that's why we used Face Recognition (python library) as it needs only one image to train the model and gives more accuracy than other face recognition method. And from Face Recognition library we used ResNet model from dlib to predict if the given image belongs to a person from our database.

3.3 Block Diagram

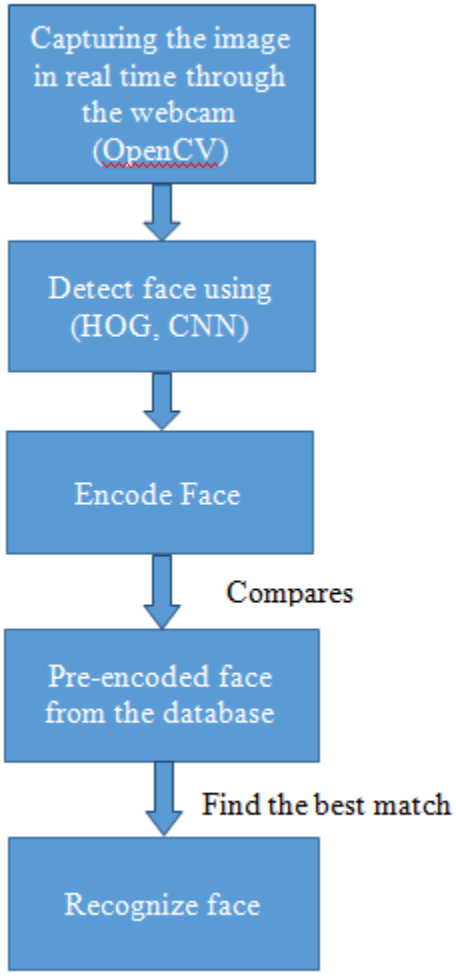


Fig 3.6: Face Recognition Process

3.4 Example

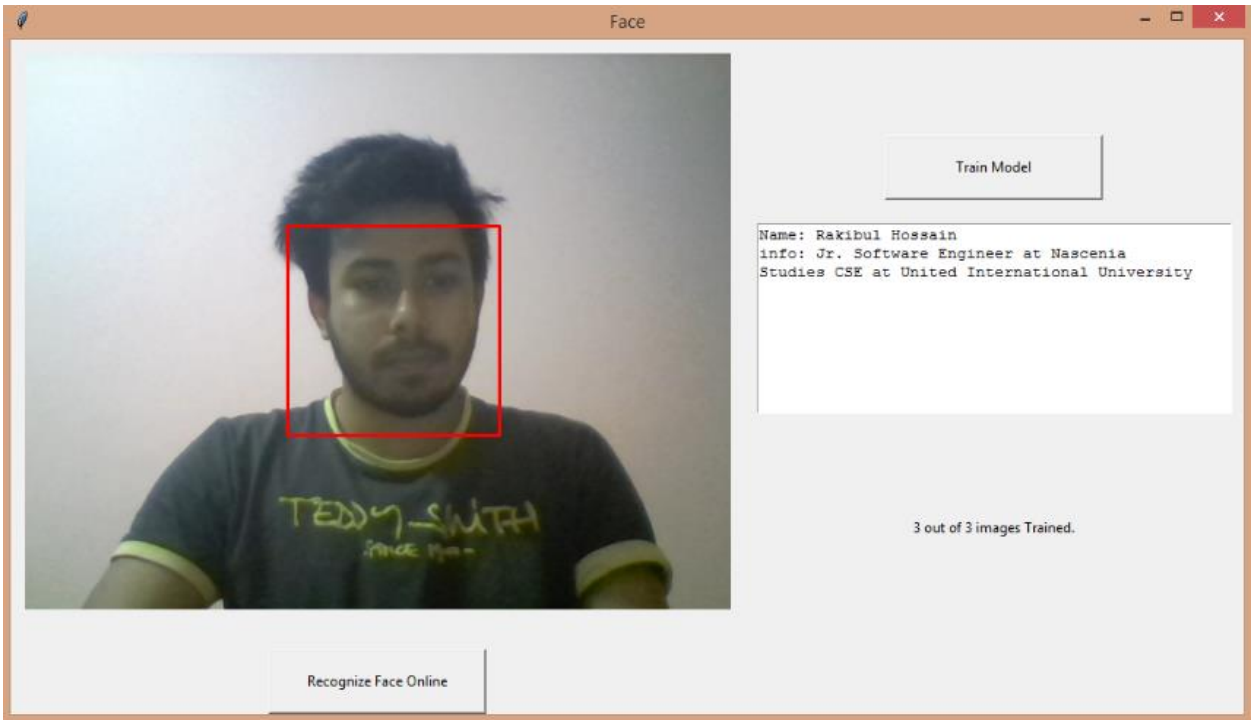


Fig 3.7: Successfully recognizing a face

After recognizing a face we will simply fetch information for the person and show it in the window.

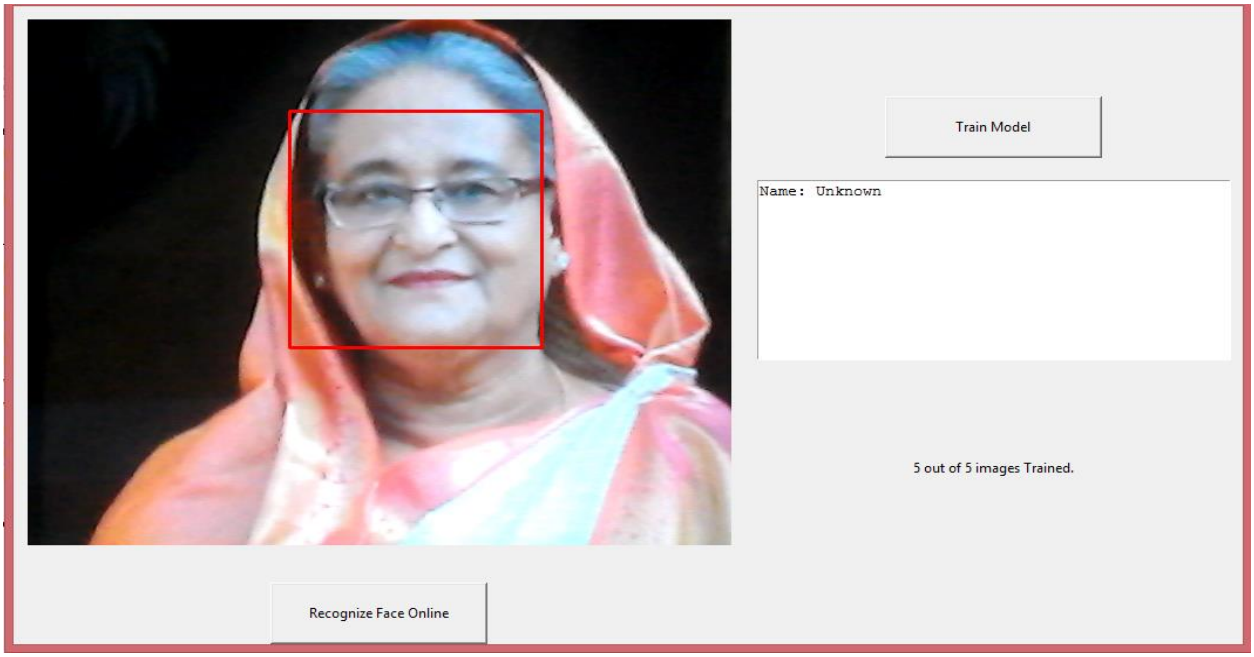


Fig 3.8: Unknown Face

Chapter 4

Information Extraction

4.1 Overview

After training the model we recognize the face, but some faces can remain unrecognized. Because those are new to our system. We label them as “Unknown”. Now, to recognize these Unknown faces, we can extract information about them from the “Internet”. Basically, we do a google image search from our system.

4.2 Image Search

Google has no API(Application Program Interface) regarding image search. So, we have done it manually.

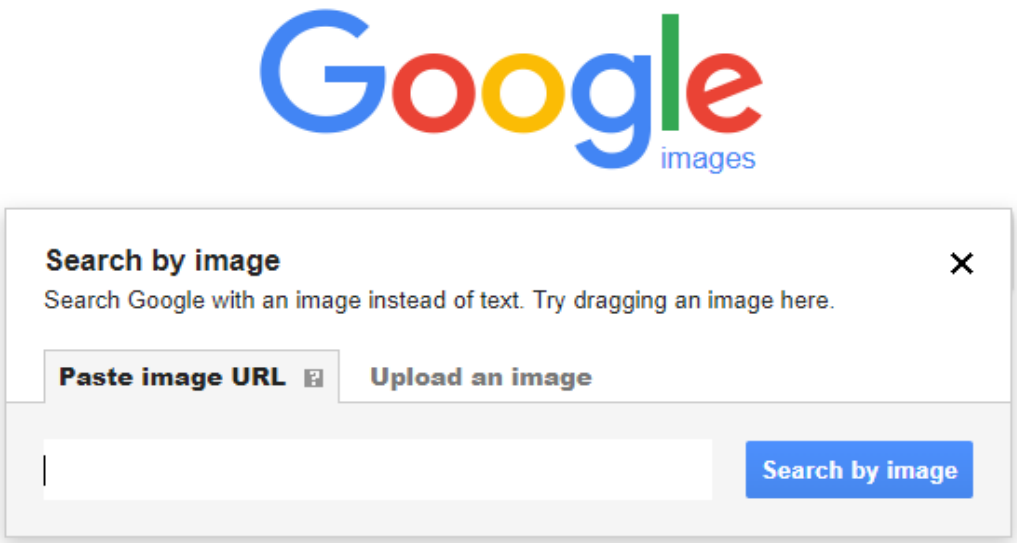


Fig 4.1: Google image search

4.2.1 Using URL to Search

We have used URL(Uniform Resource Locator) of the image, to search on google. For that, we have the option to capture still image, then use that image to search on google. We couldn't upload the image directly to google image search, because Google blocks it.

4.2.2 Upload Image to Online Site

To use the URL, we have to create one. So, we uploaded the still image to a site called "Imgur". It's an image sharing site, where we can upload the image. By this, we get a link or URL of the image we want to search.

4.2.3 Search on Google

We add two links together, the google image search link and the image link. Then we make a request call through python's 'urllib' module.

```
url = 'http://www.images.google.com/searchbyimage?image_url=' + imagepath
```

Now we take this link and send request.

```
req = urllib.request.Request(googlepath, headers=headers)
resp = urllib.request.urlopen(req)
respData = resp.read()
```

To read the response data, we used resp.read().

4.3 Data Extraction

We got the response data, now we all need to do is extract our useful information. To do that, we need Web scraping tool. We have used Beautiful Soup to extract our useful information.

4.3.1 Web Scraping

Web scraping is actually data extraction from websites. Web scraping software uses Hypertext transfer protocol to access world wide web. We've used Web scraping tool, Beautiful Soup which is a package of python.

4.3.2 Beautiful Soup

Beautiful soup is a package of python, which is used for parsing HTML and XML documents as we mentioned. From the parsed pages it creates a parse tree that extracts data from HTML. This is used for Web Scraping.

```
soup = BeautifulSoup(respData, "html.parser")
```

The response data we got previously is being parsed here by beautiful soup.

4.3.3 Information Extraction

Now that we've parsed the data, we can find our necessary information.

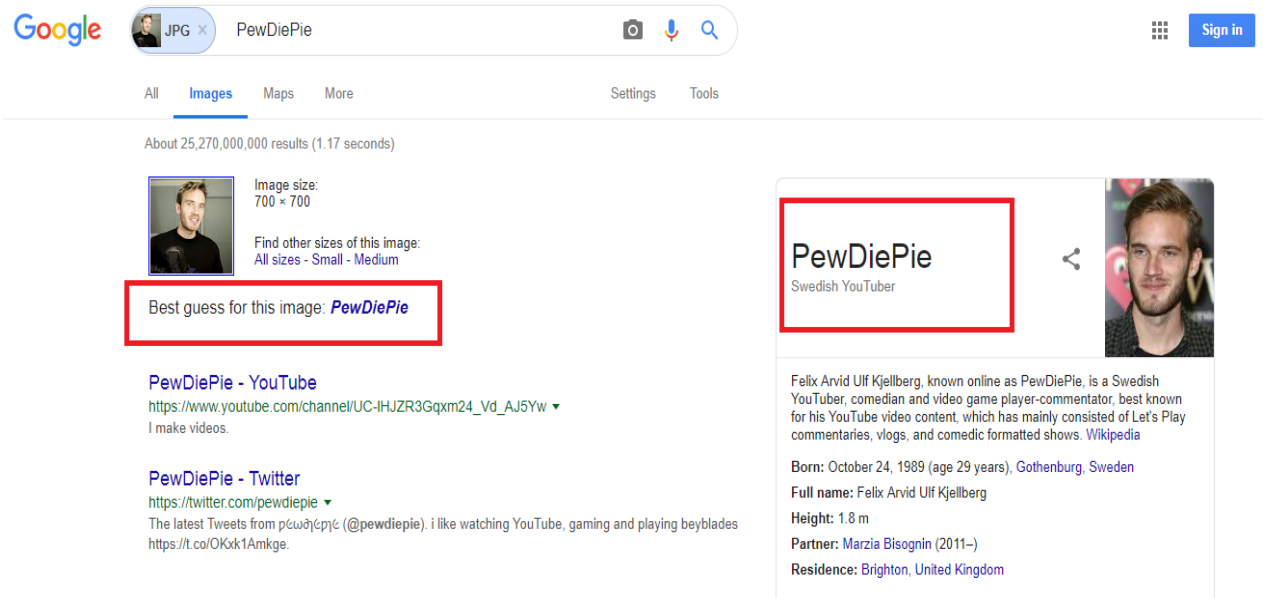
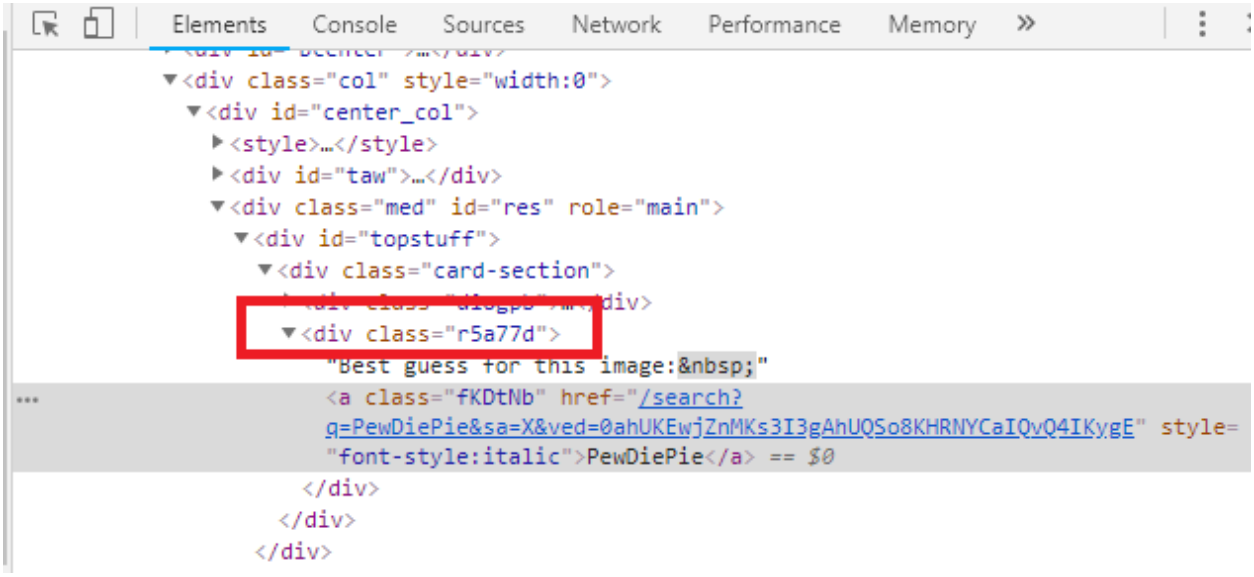


Fig 4.2: Google Image search result page

Here we've sent a request for a famous Youtuber called Pewdiepie. Google sends us a response, this page to be exact. So, from this page, we need to extract our important information. When we search in google, google shows us the best-guessed result. We need that information from there. To do that, we've to inspect that page to get our necessary information.



A division class named "r5a77d" has the information we need.

```
container = soup.find('div',class_='r5a77d')
```

We find that class through beautiful soup, and get our necessary information. For additional information, in similar way we find a division classed named “wwUB2c kno-fb-ctx” which has the information we need.

```
container = soup.find('div', class_='wwUB2c kno-fb-ctx')
```

This is how we get our necessary information through web scraping.

Chapter 5

Experiments

At first, we had used the LBPH algorithm for our system. LBPH which stands for Local Binary Pattern, by combining with Histograms of oriented gradients it became Local Binary Pattern Histograms to improve the performance.[5]

5.1 LBPH (Local Binary Pattern Histograms)

LBPH algorithm has 4 parameters, which are, radius, neighbors, gridX, gridY. Radius is actually the radius of the central pixel. Usually set to 1. Neighbors are the number of sample points which is used to build the circular local binary pattern. So, the more sample points we add, the more computational cost. Set to 8. GridX is the number of cells in the Horizontal direction. GridY is the number of cells in the Vertical direction.

5.1.1 LBPH Steps

Convert the image/frame into grayscale, then we get the 2D array of pixels. Then we calculate the threshold which is the central value of the matrix. This value is used for the neighbor's value. We convert the threshold value to binary value and then convert it to decimal. Following diagram shows the steps:

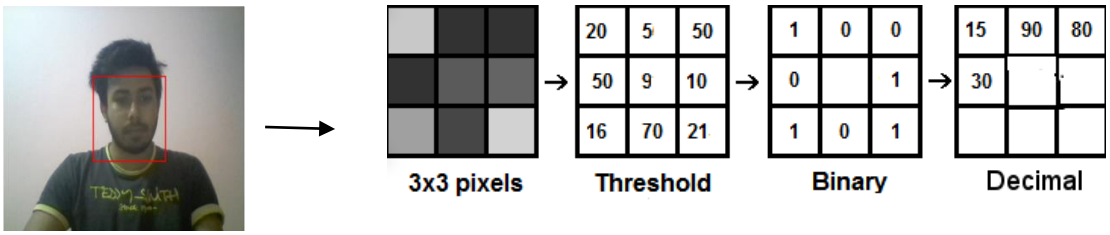


Fig 5.1: Calculating Information from an image

We extract histogram from it then. By then the model is trained, if new data comes, we calculate/predict the output. For comparing we can use Euclidian distance or absolute value or even chi-square.

5.1.2 LBPH Advantages

LBPH is one of the easiest algorithms used for face recognition. It can represent the local values of features in an image. It is provided by OpenCV library, so it's easy to use in code also.

5.1.3 Problems

Problems we faced were, LBPH requires more than 1 image to train, also less accurate.

5.2 face_recog (Library)

We've used the face_recog library provided by python for the solution of the previous problems. face_recog needs only 1 image for recognition, also it is more accurate than LBPH. Its accuracy is about 99.38%. So, we moved to the face_recog library that python provides rather than using LBPH algorithm that OpenCV provides.

Chapter 6

Conclusion & Future Works

6.1 Conclusion

The improvement of Image Capturing and Automatic face recognition is suggested by using the simple method without proposing any brand new algorithm but by using the already existing algorithm from a different library in an effective way. As our approach is very simple in terms of calculation, it improves the speed of recognition. It also requires only one scanning without any need to a complicated analysis.

We already showed our two process of face identification in a systematic way step by step.

The summary of two features of our proposed system is:

Face Identification:

Given a real-time captured photo that belongs to a person in our database and tells whose image it is with additional information of the person.

Face Identification from google:

Given a real-time captured photo that doesn't belong to our database, we will extract information from the internet by google search from our system.

6.2 Future Work

In future we will try to upgrade our system which will be able to do the task like followings:

1. For the unknown person, we will save the image and ask for information and update in our database.
2. Use the system for real-time face identification from security the camera.
3. Face identification for more than one person in an image.

References

[1] Junlin Hu, Jiwen Lu, Yap-Peng Tan, “Discriminative Deep Metric Learning for Face Verification in the Wild”, CVPR2014, IEEE Explore, 2014.

[2] Library: face_recog, available online at https://pypi.org/project/face_recognition

[3] OpenCV Library, available online at https://opencv.org

[4] O. Deniz, G. Bueno, J. Salido, F. De la Torre, “Face Recognition using Histograms of Oriented Gradients”, Pattern Recognition Letters, Vol. 32, Issue No.12, pp.1598-1603, Sep 2011.

[5] T. Ahonen, A. Hadid, M. Pietikainen, “Face Description with Local Binary Patterns: Application to Face Recognition”, Vol. 28, Issue No.12, pp. 2037-2041, Dec 2006.

Appendix A

Code file face_rec.py

```
import os
import threading

import face_recognition
import cv2
import datetime
import ui.ui as ui
import mysql.connector

NAME_COL=0
INFO_COL=1
IMAGE_COL=3

my_db = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="face_rec"
)


only_detect = True

BASE_DIR = os.path.dirname(os.path.abspath(__file__))
image_dir = os.path.join(BASE_DIR, "images")


def count_total_image_files():
    my_cursor = my_db.cursor()
    my_cursor.execute("select * from person")
    my_cursor.fetchall()
    row_number=my_cursor.rowcount
    my_cursor.close()
    return row_number
```

```
def load_and_encode_images():
    msg = ui.get_msg_setter()
    msg.set("Loading")
    known_face_encodings = []
    known_face_names = []
    known_face_info =[]

    total_image = str(count_total_image_files())
    msg.set(total_image + ' Images Found')
    cnt = 0

    my_cursor = my_db.cursor()
    my_cursor.execute("select * from person")
    record=my_cursor.fetchall()
    for row in record:
        image =
face_recognition.load_image_file(row[IMAGE_COL])
        a = datetime.datetime.now()
        face_encoding =
face_recognition.face_encodings(image)[0]
        print(datetime.datetime.now() - a)
        known_face_encodings.append(face_encoding)
        known_face_names.append(row[NAME_COL])
        if row[INFO_COL]:
            known_face_info.append(row[INFO_COL])
        else:
            known_face_info.append('No information
available.')
        cnt += 1


    msg.set(str(cnt) + " out of " + total_image + '
images Trained.')
    return known_face_encodings, known_face_names,
known_face_info


def recognize(known_face_encodings, known_face_names,
known_face_info, frame):
    if known_face_names is None:
```

```
        return None, None, None

    name = ""
    info = ""

    small_frame = cv2.resize(frame, (0, 0), fx=0.20,
fy=0.20)
    rgb_small_frame = small_frame[:, :, ::-1]



    face_locations =
face_recognition.face_locations(rgb_small_frame)
    face_encodings =
face_recognition.face_encodings(rgb_small_frame,
face_locations)

    face_names = []
    for face_encoding in face_encodings:
        matches =
face_recognition.compare_faces(known_face_encodings,
face_encoding)
        name = "Unknown"

        if True in matches:
            first_match_index = matches.index(True)
            name = known_face_names[first_match_index]
            info = known_face_info[first_match_index]

        face_names.append(name)

    # Display the results
    for (top, right, bottom, left), name in
zip(face_locations, face_names):
        top *= 5
        right *= 5
        bottom *= 5
        left *= 5

        cv2.rectangle(frame, (left, top), (right,
bottom), (0, 0, 255), 2)
```

```
    return frame, name, info
```

Code file ui.py

```
import threading
from tkinter import *
canvas = None
root = None
name = None
msg = None
v = None
popup_canvas = None
save_still_image = None
recognize_button = None
local_button = None
train_button = None
for_online_search=False

def dismiss():
    global for_online_search
    for_online_search=False

def recognize_from_web():
    global for_online_search
    for_online_search = True
    save_still_image()

def load_ui(train_callback, save_still_image_callback):
    print(threading.currentThread())
    global name, root, msg, canvas, v, save_still_image,
recognize_button, train_button, local_button
    save_still_image = save_still_image_callback
    root = Tk()
    root.title("Image Capturing & Automatic Face
Recognition")
    v = StringVar()
    view_frame = Frame(root, width=500, height=500)
    view_frame.grid(row=0, column=0)
```

```
    canvas = Canvas(view_frame, width=600, height=500)
    canvas.grid(row=0, column=0, padx=10, pady=10)

    recognize_button = Button(view_frame, text="Recognize
Face Online", width=25, height=3,
command=recognize_from_web)
    recognize_button.grid(row=1, column=0)

    local_button = Button(view_frame, text="Back",
width=25, height=3, command=dismiss)


    info_frame = Frame(root, width=500, height=500)
    info_frame.grid(row=0, column=1)

    train_button = Button(info_frame, text="Train Model",
command=train_callback
                          , width=25, height=3)
    train_button.grid(row=0, column=0, padx=10, pady=10)

    name = Text(info_frame, width=50, height=10)
    name.grid(row=1, column=0, padx=10, pady=10)

    msg = Label(info_frame, width=50, height=10,
textvariable=v)
    msg.grid(row=2, column=0, padx=10, pady=10)


def set_image(photo):
    global canvas
    canvas.create_image(0, 0, image=photo, anchor=NW)


def update():
    global root
    root.update_idletasks()
    root.update()


def set_name(n, info):
    global name
    name.delete(1.0, END)
```

```
    if info and info=='info':
        name.insert(INSERT, "Info: " + n)
    else:
        name.insert(INSERT, "Name: " + n)

def set_msg(text):
    global msg, v
    v.set(text)


def get_msg_setter():
    global v
    return v

def set_popup_image(photo):
    global popup_canvas
    popup_canvas.create_image(0, 0, image=photo,
anchor=NW)

def hide_buttons():
    global recognize_button, train_button, local_button
    recognize_button.grid_forget()
    train_button.grid_forget()
    local_button.grid(row=1, column=0)

def show_buttons():
    global recognize_button, train_button, local_button
    local_button.grid_forget()
    recognize_button.grid(row=1, column=0)
    train_button.grid(row=0, column=0, padx=10, pady=10)
```

Code file upload.py

```
import pyimgur

CLIENT_ID = 'b8de73761d59053'

def upload_file():
    import os

    PATH = os.path.abspath("tmp_image.png")
```

```
    im = pyimgur.Imgur(CLIENT_ID)
    uploaded_image = im.upload_image(PATH, title="tmp")
    print(uploaded_image.title)
    print(uploaded_image.link)
    print(uploaded_image.size)
    print(uploaded_image.type)
    return uploaded_image.link
```

Code file main.py

```
import threading

import train.face_rec as train
import cv2
import PIL.Image, PIL.ImageTk
import ui.ui as ui
import upload.upload as upload
import scrap.webScrap as scrap

info_string='Uploading and Fetching data from Web..
Please wait'
known_face_encodings = None
known_face_names = None
known_face_info = None

class ThreadHelper(threading.Thread):
    def run(self):
        global info_string
        uploaded_file_url = upload.upload_file()
        info = scrap.imageLookup(uploaded_file_url)
        print(info)
        info_string=info


def train_model():
    global known_face_encodings, known_face_names,
known_face_info
    known_face_encodings, known_face_names,
known_face_info = train.load_and_encode_images()
```

```
def save_still_image():
    frame = video_capture.read()[1]
    cv2.imwrite("tmp_image.png", frame)
    b, g, r = cv2.split(frame)
    frame = cv2.merge((r, g, b))

ui.set_image(PIL.ImageTk.PhotoImage(image=PIL.Image.froma
rray(frame)))
    ui.set_msg("Uploading and Fetching data from Web..
Please wait")
    ui.hide_buttons()
    ThreadHelper().start()
    while ui.for_online_search:
        ui.update()
        ui.set_name(info_string, 'info')
    ui.set_msg('')
    ui.show_buttons()


ui.load_ui(train_model, save_still_image)

video_capture = cv2.VideoCapture(0)


while True:
    ret, frame = video_capture.read()


    recognized_frame, name, info =
train.recognize(known_face_encodings, known_face_names,
known_face_info, frame)
    if recognized_frame is None:
        recognized_frame = frame
        ui.set_msg("Please Train the Model First")

    b, g, r = cv2.split(recognized_frame)
    recognized_frame = cv2.merge((r, g, b))
    photo =
PIL.ImageTk.PhotoImage(image=PIL.Image.fromarray(recogniz
ed_frame))
```

```
    ui.set_image(photo)
    if name is not None:
        ui.set_name(name+'\n'+'info: ' +info, None)
    ui.update()
```

Code file webScrap.py

```
import urllib.request
import urllib.parse


from bs4 import BeautifulSoup
from googletrans import Translator

def imageLookup(imagepath):

    try:
        googlepath =
'http://www.images.google.com/searchbyimage?image_url=' +
imagepath
        print(googlepath)
        headers = {}
        headers['User-Agent']= 'Mozilla/5.0 (Windows NT
6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/71.0.3578.98 Safari/537.36'
        req = urllib.request.Request(googlepath,
headers=headers)
        resp = urllib.request.urlopen(req)
        respData = resp.read()
        soup = BeautifulSoup(respData, "html.parser")

        container = soup.find('div',class_='r5a77d')
        translator = Translator()
        val = translator.translate(container.a.text)
        ret = val.text

        container = soup.find('div', class_='wwUB2c kno-
```

```
fb-ctx')
        try:
            x = str(container)
            val =
x.split('">')[2].replace("</span></div>","",-1)
            val = translator.translate(val)
            ret = ret +"\n"+ val.text
        except Exception:
            ret +"\n"+"no additional info found"
        return ret

    except Exception as e:
        print(str(e))
```