GALGOTIAS UNIVERSITY

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Face Recognition System

A Report for the Evaluation 3 of Project 2

Submitted by

Mohd Saim Hashmi

(1613105065)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING WITH SPECIALIZATION OF CLOUD COMPUTING AND VIRTUALIZATION

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

Under the Supervision of

Dr. SPS Chauhan, Assoc. Professor

April/May - 2020

TABLE OF CONTENTS

1. Abstract

The goal of this project is to build a face recognition system – from either a single photograph or from a set of faces tracked in a video or from a live webcam footage. Face recognition is a computationally challenging task that humans perform effortlessly. Nonetheless, this remarkable ability is better for familiar faces than unfamiliar faces. To account for humans' superior ability to recognize familiar faces, current theories suggest that different features are used for the representation of familiar and unfamiliar faces.

In this we use technologies like openCV, face_recognition to recognize people in a photograph, a video or a live video footage.

2. Introduction

Facial recognition is the task of making a positive identification of a face in a photo or video image against a pre-existing database of faces. It begins with detection - distinguishing human faces from other objects in the image - and then works on identification of those detected faces.

Face recognition is the task of identifying an already detected object as a known or unknown face. Face recognition is a computationally challenging task that humans perform effortlessly. Nonetheless, this remarkable ability is better for familiar faces than unfamiliar faces. To account for humans' superior ability to recognize familiar faces, current theories suggest that different features are used for the representation of familiar and unfamiliar faces.

Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do.

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc.

Facial tasks in machine learning operate based on images or video frames (or other datasets) focussed on human faces.

In this project we convert a photograph or a frame of video in black and white which makes it easier for the algorithm to recognize face.

We use a labeled datasets (separate photographs of individuals that might appear in input data) to train the model i.e to recognize the people and build a list of names. Using that list we loop a photograph or a from to detect and recognize individuals in that photograph.

2.1. Overall Description

This project is carried out to fulfill the following tasks.

- To build a model to recognize individuals in a photograph or video.
- To effectively recognize find people in a footage.

2.2. Purpose

- To build a model to recognize people in photo or a video using the face_detection and openCV and numpy libraries.

2.3. Promising Applications

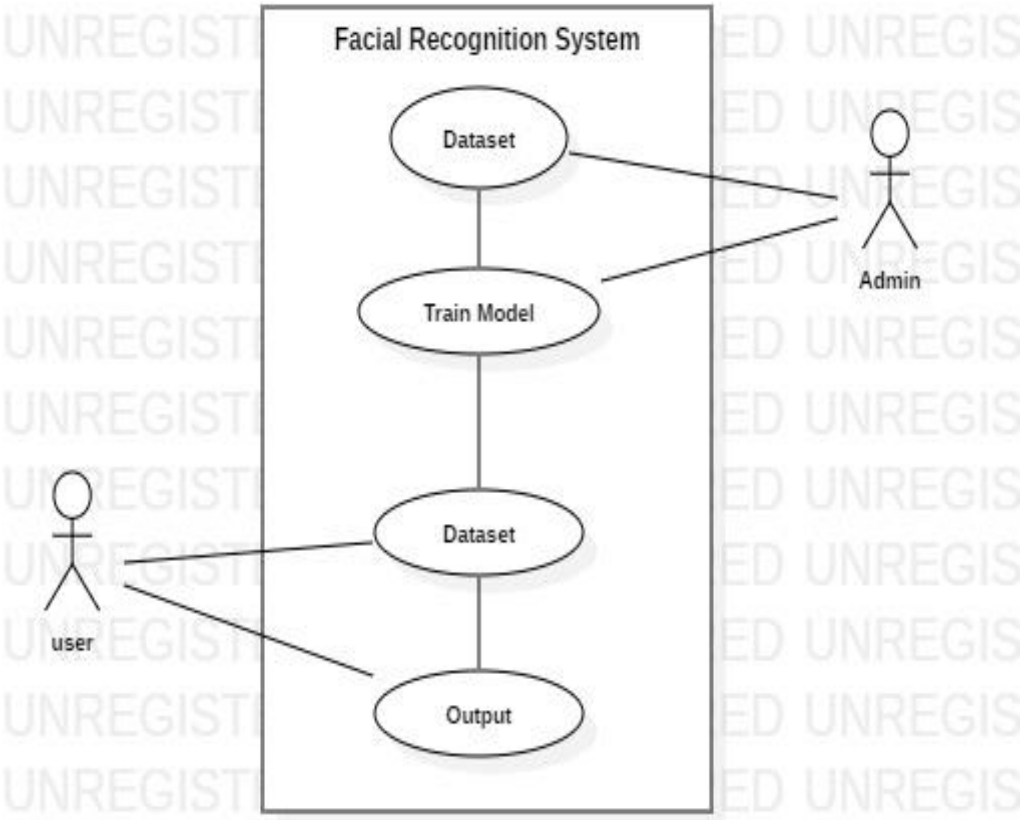
- Facebook replaced manual image tagging with automatically generated tag suggestions for each picture that was uploaded to the platform.
- Face detection technique is being used to maintain the security of personal devices such as mobile phones.
- larger scale implementation such as enabling cameras to capture images and detect faces.

2.4. Setup Required

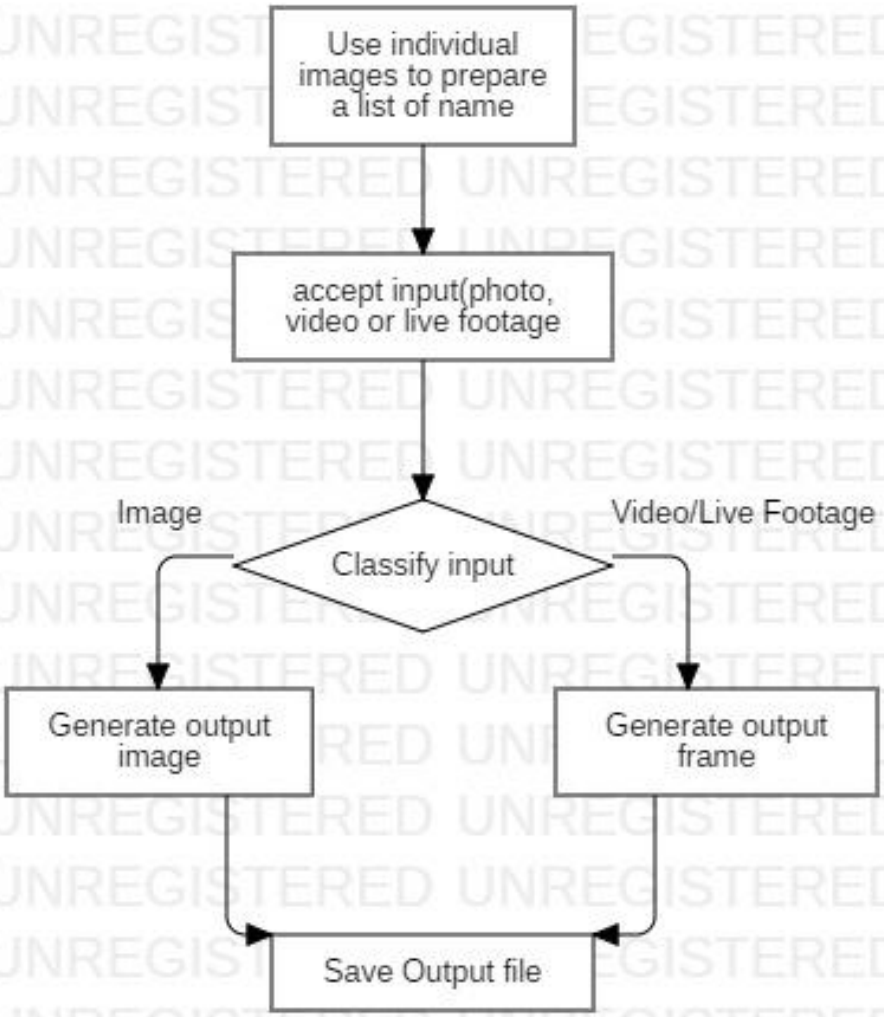
- python 3.6 or higher.
- openCV library
- face_detection API
- numpy
- PyCharm

3. **Proposed Model**

1. Use Case Diagram



2. Flow Chart Diagram



4. Implementation

faceRecSys_WebCam.py :

```
# import
import face_recognition
import cv2
import numpy as np

# Input from webcam
video_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.
saaim_image = face_recognition.load_image_file("saaim.jpg")
saaim_face_encoding = face_recognition.face_encodings(saaim_image)[0]

# Create arrays of known face encodings and their names
known_face_encodings = [
    saaim_face_encoding
]
known_face_names = [
    "Saaim Hashmi"
]

# Initialize some variables
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True

while True:
    # Grab a single frame of video
    ret, frame = video_capture.read()

    # Resize frame of video to 1/4 size for faster face recognition processing
    small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which
face_recognition uses)
    rgb_small_frame = small_frame[:, :, ::-1]

    # Only process every other frame of video to save time
    if process_this_frame:
        # Find all the faces and face encodings in the current frame of video
        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:
            # See if the face is a match for the known face(s)
            matches = face_recognition.compare_faces(known_face_encodings,
face_encoding)
```

```
            name = "Unknown"

            # # If a match was found in known_face_encodings, just use the first
one.
            # if True in matches:
            #     first_match_index = matches.index(True)
            #     name = known_face_names[first_match_index]

            # Or instead, use the known face with the smallest distance to the new
face
            face_distances = face_recognition.face_distance(known_face_encodings,
face_encoding)
            best_match_index = np.argmin(face_distances)
            if matches[best_match_index]:
                name = known_face_names[best_match_index]

            face_names.append(name)

    process_this_frame = not process_this_frame


    # Display the results
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        # Scale back up face locations since the frame we detected in was scaled
to 1/4 size
        top *= 4
        right *= 4
        bottom *= 4
        left *= 4

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom), (right, bottom), (0, 0, 255),
cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left, bottom), font, 1.0, (255, 255, 255), 1)

    # Display the resulting image
    cv2.imshow('Video', frame)

    # Hit 'q' on the keyboard to quit!
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release handle to the webcam
video_capture.release()
cv2.destroyAllWindows()
```

FaceRecSys_Video.py :

```
# Import
import face_recognition
import cv2
import numpy as np
import time
```

```
# Open the input video file
video = cv2.VideoCapture("hamilton_clip.mp4")

# Some variables
start = time.time()
length = int(video.get(cv2.CAP_PROP_FRAME_COUNT))

# Variables to match the input files resolution and frame rate in output file
fps = video.get(cv2.CAP_PROP_FPS)
reslen = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
reswid = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))

# Create an output video file
fourcc = cv2.VideoWriter_fourcc(*'XVID')
output_video = cv2.VideoWriter('output.avi', fourcc, fps, (reswid, reslen))

# Load some sample pictures and learn how to recognize them.
lmm_image = face_recognition.load_image_file("llm.png")
lmm_face_encoding = face_recognition.face_encodings(lmm_image)[0]

al_image = face_recognition.load_image_file("Alex.png")
al_face_encoding = face_recognition.face_encodings(al_image)[0]

known_faces_encodings = [
    lmm_face_encoding,
    al_face_encoding
]
known_face_names = [
    "Lin Manuel Miranda",
    "Alex lacamoire"
]
# Initialize some more variables
face_locations = []
face_encodings = []
face_names = []
frame_number = 0

while True:
    # Grab a single frame of video
    ret, frame = video.read()
    frame_number += 1

    # Quit when the input video file ends
    if not ret:
        break

    # Convert the image from BGR color (which OpenCV uses) to RGB color (which
face_recognition uses)
    rgb_frame = frame[:, :, ::-1]

    # Find all the faces and face encodings in the current frame of video
    face_locations = face_recognition.face_locations(rgb_frame)
    face_encodings = face_recognition.face_encodings(rgb_frame, face_locations)

    face_names = []
    for (top, right, bottom, left), face_encoding in zip(face_locations,
face_encodings):
        # See if the face is a match for the known face(s)
        matches = face_recognition.compare_faces(known_faces_encodings,
```

```
face_encoding)

        name = "unknown"

        # If you had more than 2 faces, you could make this logic a lot prettier
        # but I kept it simple for the demo
        face_distances = face_recognition.face_distance(known_faces_encodings,
face_encoding)
        best_match_index = np.argmin(face_distances)
        if matches[best_match_index]:
            name = known_face_names[best_match_index]

        face_names.append(name)

    # Label the results
    for (top, right, bottom, left), name in zip(face_locations, face_names):
        if not name:
            continue

        # Draw a box around the face
        cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

        # Draw a label with a name below the face
        cv2.rectangle(frame, (left, bottom ), (right, bottom), (0, 0, 255),
cv2.FILLED)
        font = cv2.FONT_HERSHEY_DUPLEX
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 0.5, (255, 255,
255), 1)

    # Write the resulting image to the output video file
    print("Writing frame {} / {}".format(frame_number, length))
    output_video.write(frame)

# All done!
video.release()
cv2.destroyAllWindows()
print("------------ {} seconds taken ---------------".format(time.time()-start))
```

FaceRecSys_Image.py :

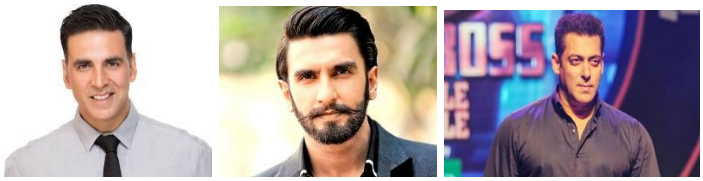
```
# import
import face_recognition
from PIL import Image, ImageDraw
import numpy as np

# Load a sample picture and learn how to recognize it.
salman_image = face_recognition.load_image_file("salman.jpg")
salman_face_encoding = face_recognition.face_encodings(salman_image)[0]

# Load a second sample picture and learn how to recognize it.
akshay_image = face_recognition.load_image_file("akshay.jpg")
akshay_face_encoding = face_recognition.face_encodings(akshay_image)[0]

# Load a third sample picture and learn how to recognize it.
ranveer_image = face_recognition.load_image_file("ranveer.jpg")
ranveer_face_encoding = face_recognition.face_encodings(ranveer_image)[0]

# Create arrays of known face encodings and their names
```

```
known_face_encodings = [
    salman_face_encoding,
    akshay_face_encoding,
    ranveer_face_encoding
]
known_face_names = [
    "Salman Khan",
    "Akshay Kumar",
    "Ranveer Singh"
]

# Load an image as an input
unknown_image = face_recognition.load_image_file("sample.jpg")

# Find all the faces and face encodings in the unknown image
face_locations = face_recognition.face_locations(unknown_image)
face_encodings = face_recognition.face_encodings(unknown_image, face_locations)

pil_image = Image.fromarray(unknown_image)
draw = ImageDraw.Draw(pil_image)

# Loop through each face found in the unknown image
for (top, right, bottom, left), face_encoding in zip(face_locations,
face_encodings):
    # See if the face is a match for the known face(s)
    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]

    face_distances = face_recognition.face_distance(known_face_encodings,
face_encoding)
    best_match_index = np.argmin(face_distances)
    if matches[best_match_index]:
        name = known_face_names[best_match_index]

    # Draw a box around the face
    draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))

    # Draw a label with a name below the face
    text_width, text_height = draw.textsize(name)
    draw.rectangle(((left, bottom - text_height), (right, bottom)), fill=(0, 0,
255), outline=(0, 0, 255))
    draw.text((left + 6, bottom - text_height), name, fill=(255, 255, 255, 255))


# Remove the drawing library from memory as per the Pillow docs
del draw

# Display the resulting image
pil_image.show()

# Save the resulting imgae
pil_image.save("output_image.jpg")
```

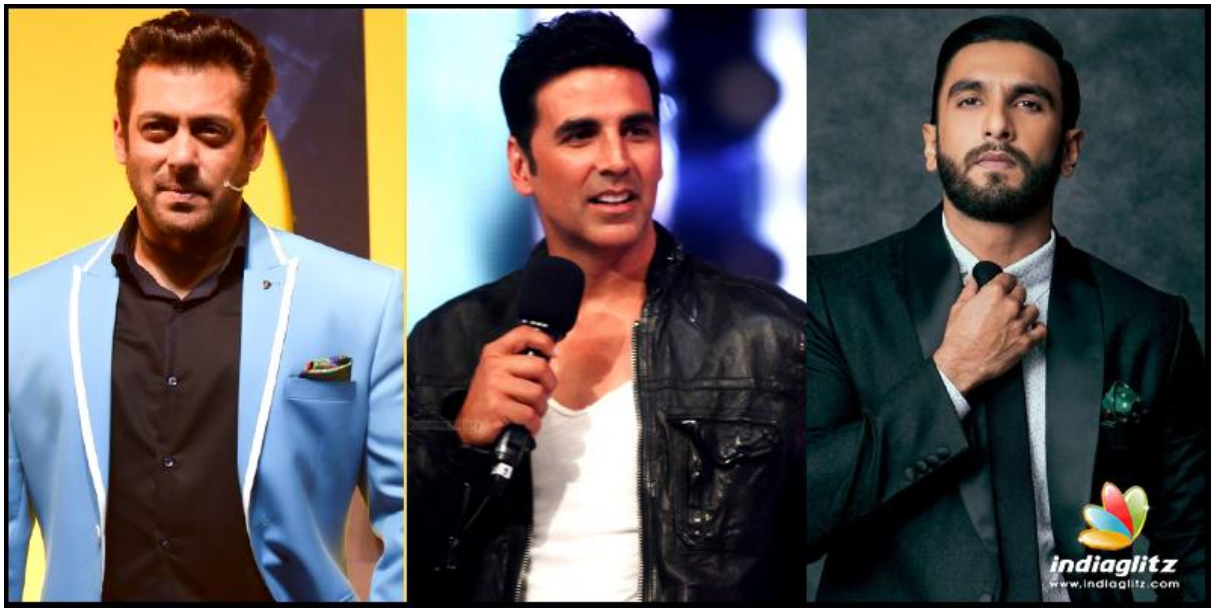
5. **Output**

- For image :

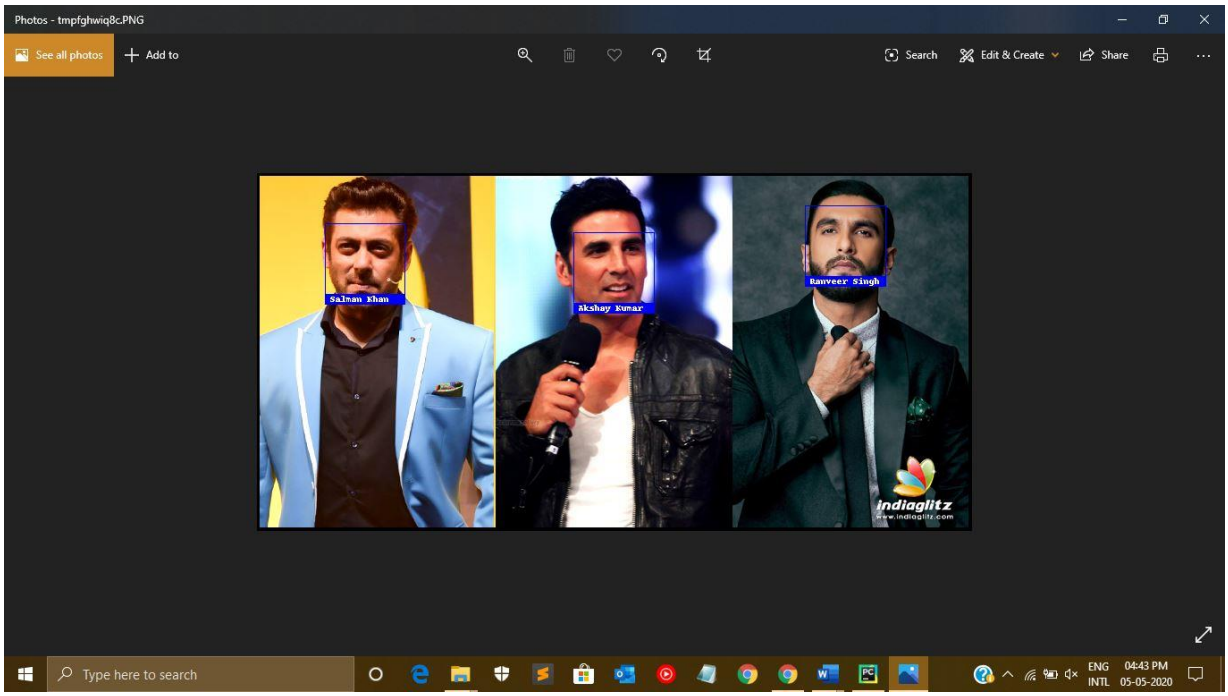
  Image database :



  Input :



  Output :

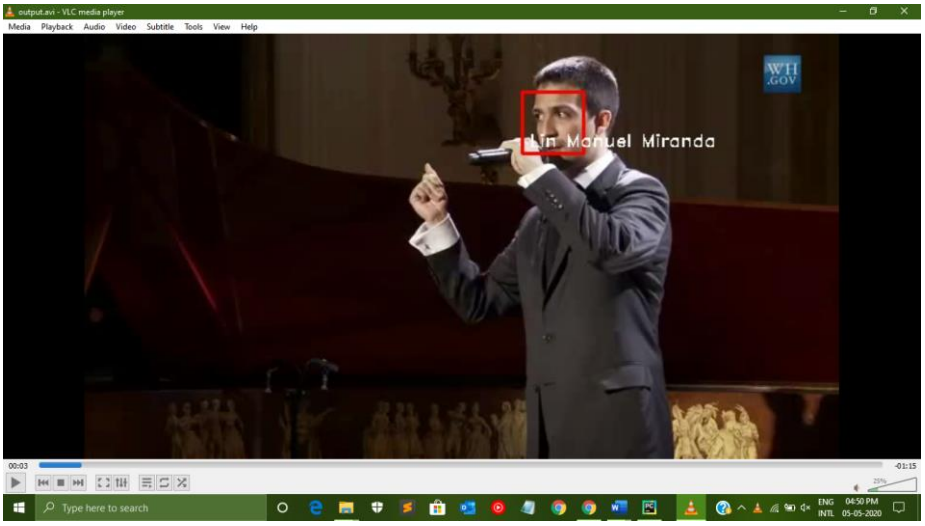


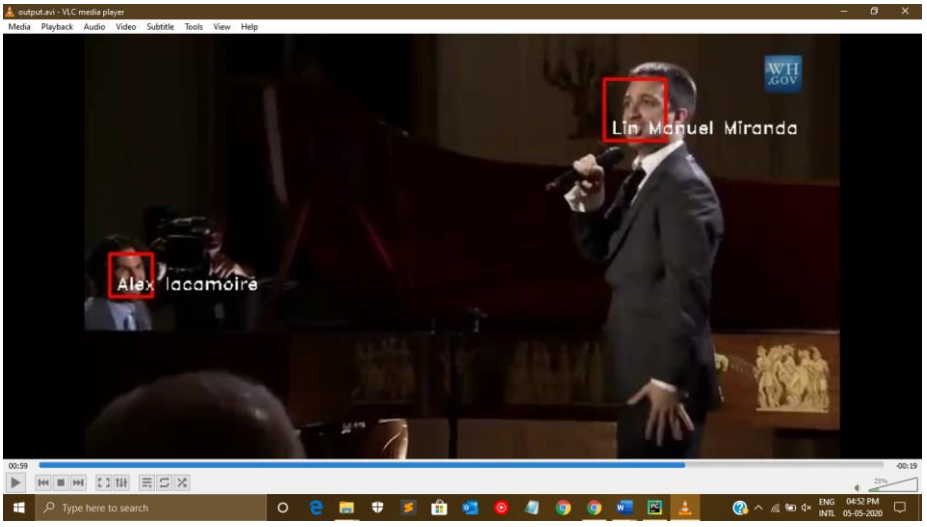
- For video :

  Image database :


  Input : A video.

  Output : Few frames from the output video.





6. Conclusion

An ideal face classifier would recognize faces in accuracy that is only matched by humans. The underlying face descriptor would need to be invariant to pose, illumination, expression, and image quality. It should also be general, in the sense that it could be applied to various populations with little modifications, if any at all. In addition, short descriptors are preferable, and if possible, sparse features. Certainly, rapid computation time is also a concern. We believe that this work, which departs from the recent trend of using more features and employing a more powerful metric learning technique, has addressed this challenge, closing the vast majority of this performance gap.

For future enhancements this model should work even faster than the current one.

7. References

1. "What is Facial Recognition? - Definition from Techopedia". Techopedia.com. Retrieved 2018-08-27.
2. ^ Andrew Heinzman. "How Does Facial Recognition Work?". How-To Geek. Retrieved 2020-02-28.
3. ^ "How does facial recognition work?". us.norton.com. Retrieved 2020-02-28.
4. ^ "Face Recognition Applications". Animetrics. Archived from the original on 2008-07-13. Retrieved 2008-06-04.
5. ^ Zhang, Jian, Yan, Ke, He, Zhen-Yu, and Xu, Yong (2014). "A Collaborative Linear Discriminative Representation Classification Method for Face Recognition. In 2014 International Conference on Artificial Intelligence and Software Engineering (AISE2014). Lancaster, PA: DEStech Publications, Inc. p.21 ISBN 9781605951508
6. ^ "Facial Recognition: Who's Tracking You in Public?". Consumer Reports. Retrieved 2016-04-05.
7. ^ Jump up to:[a] [b] Bramer, Max (2006). Artificial Intelligence in Theory and Practice: IFIP 19th World Computer Congress, TC 12: IFIP AI 2006 Stream, August 21-24, 2006, Santiago, Chile. Berlin: Springer Science+Business Media. p. 395. ISBN 9780387346540.