**Progress report of the semester-end project on Face Recognition system.**

**Introduction**

Facial recognition is a biometric software application capable of uniquely identifying or verifying a person by comparing and analyzing patterns based on the person's facial contours. Facial recognition is mostly used for security purposes, though there is increasing interest in other areas of use. In fact, facial recognition technology has received significant attention as it has potential for a wide range of application related to law enforcement as well as other enterprises.

**Modular Approach**

1. **Face Recognition**

The module face\_recognition is used to recognize and manipulate faces from Python or from the command line with the world’s simplest face recognition library. Built using dlib’s state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the labeled faces in the wild benchmark. This also provides a simple face\_recognition command line tool that lets you do face recognition on a folder of images from the command line.

1. **OpenCV**

OpenCV is the most popular library for computer vision. Originally written in C/C++, it now provides bindings for Python. OpenCV uses machine learning algorithms to search for faces within a picture. Because faces are so complicated, there isn’t one simple test that will tell you if it found a face or not. Instead, there are thousands of small patterns and features that must be matched. The algorithms break the task of identifying the face into thousands of smaller, bite-sized tasks, each of which is easy to solve. These tasks are also called classifiers. For something like a face, you might have 6,000 or more classifiers, all of which must match for a face to be detected (within error limits, of course).

**Source Code:**

import face\_recognition

import cv2

#import xlwt

#wb=xlwt.Workbook()

#ws=wb.add\_sheet("omega")

video\_capture = cv2.VideoCapture(0)

# Load a sample picture and learn how to recognize it.

omega\_image = face\_recognition.load\_image\_file("current.jpg")

omega\_face\_encoding = face\_recognition.face\_encodings(omega\_image)[0]

# Create arrays of known face encodings and their names

known\_face\_encodings = [ omega\_face\_encoding ]

known\_face\_names =[ "Yadav" ]

# 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:

# ws.write(0,0,name)

first\_match\_index = matches.index(True)

name = known\_face\_names[first\_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 - 35), (right, bottom), (0, 0, 255), cv2.FILLED)

font = cv2.FONT\_HERSHEY\_DUPLEX

cv2.putText(frame, name, (left + 6, bottom - 6), 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

#wb.save("Excel.xls")

# Release handle to the webcam

video\_capture.release()

cv2.destroyAllWindows()

