FacePlay-Face,Gender and Emotion detection using OpenCV,dlib and FisherFace Recognizer

A Report Submitted
in Partial Fulfillment of the Requirements
for the Degree of
Bachelor of Technology
in
Computer Science & Engineering

by

Deepak Bulani(20153026)
Gaurav Agarwal(20154097)
Bonthu Harsha Vardhan Reddy(20154148)
Avichal Verma(20154085)
Amit Ranjan(20154101)

to the
COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
MOTILAL NEHRU NATIONAL INSTITUTE OF TECHNOLOGY
ALLAHABAD
November, 2018

UNDERTAKING

I declare that the work presented in this report titled "*FacePlay-Face,Gender and Emotion detection using OpenCV,dlib and FisherFace Recognizer*", submitted to the Computer Science and Engineering Department, Motilal Nehru National Institute of Technology, Allahabad, for the award of the **Bachelor of Technology** degree in **Computer Science & Engineering**, is my original work. I have not plagiarized or submitted the same work for the award of any other degree. In case this undertaking is found incorrect, I accept that my degree may be unconditionally withdrawn.

November, 2018
Allahabad

(Deepak Bulani, 20153026)

(Gaurav Agarwal, 20154097)

(Bonthu Harsha Vardhan Reddy, 20154148)

(Avichal Verma, 20154085)

(Amit Ranjan, 20154101)

‘

CERTIFICATE

Certified that the work contained in the report titled "*FacePlay-Face,Gender and Emotion detection using OpenCV,dlib and FisherFace Recognizer*", by
Deepak Bulani(20153026)
Gaurav Agarwal(20154097)
Bonthu Harsha Vardhan Reddy(20154148)
Avichal Verma(20154085)
Amit Ranjan(20154101)
has been carried out under my supervision and that this work has not been submitted elsewhere for a degree.

Dr. Dinesh Singh
(Assistant Professor)
Computer Science and Engineering Dept.
M.N.N.I.T, Allahabad

November, 2018

Preface

Face recognition is the process of identifying people through facial images, has numerous practical applications in the area of bio-metrics, information security, access control, law enforcement, smart cards and surveillance system.We aim to augment the utility of face recognition with emotion as well as gender detection for the full purview of our project. Emotion detection and computing aims to enable machines to recognize and synthesize human emotions. As we all know, a change of user's emotion is one of the foundation of communication. Emotional states can motivate human's actions, and can also supplement the meaning of communication. Thus,in this project we first went through OpenCV library which is aimed at real time computer vision and includes more than 2500 optimized algorithms,which includes a comprehensive set of both classic and state of the art computer vision and machine learning algorithms. OpenCV helped us to detect faces and various facial regions like eyes,nose,lips and jaw and dlib was an improvement over OpenCV. We have used dlib library for mapping exact facial characteristics to obtain a higher accuracy during training and testing. Here, Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software to solve real world problems. For gender and emotion detection we make use of FisherFace Recognizer included under OpenCV which we train using dataset from KDEF(Karolinska Directed Emotional Faces) and IMDB-WIKI.FisherFace uses Linear Discriminant Analysis (LDA) to determine the vector representation. It produces float value in the prediction. This also means that the result is better compared to Eigenface, that is considered first successful method of face recognition.

Acknowledgements

We would like to express our sincere gratitude towards Mr. Dinesh Singh, for his valuable guidance and advice. Without his co-operation and support, this project could not have been completed. From aiding us with purposeful software and also patiently resolving our doubts, Dinesh Sir played a significant role in making the project happen.
His dedication and keen interest above all his overwhelmingly friendly attitude to help his students had been solely and mainly responsible for completing this work. His timely advice, meticulous scrutiny, scholarly advice and scientific approach have helped us to a very great extent to accomplish this task. From the start, by allowing us to choose the project of our choice to sharing his pearls of wisdom on ways to enhance the quality of our project, we all are greatly indebted to Dinesh Sir.
In addition,we also extend our thanks to our colleagues who provided valuable insight and expertise that greatly assisted in making this research work comprehensive and precise. We would also like to expand our deepest gratitude to all those who have directly and indirectly guided us in writing this assignment.

Contents

Chapter 1

Introduction

Face recognition is the process of identifying people through facial images. It has numerous practical applications in the area of biometrics, information security, access control, law enforcement, smart cards and surveillance system.Face recognition system identifies a face by matching it with the facial database. It has gained great progress in the recent years due to improvement in design and learning of features and face recognition models. Face recognition plays a crucial role in applications such as security system, credit card verification, identifying criminals in airport, railway stations etc. Although many methods have been proposed to detect and recognize human face developing a computational model for a large data base is still a challenging task. That is why face recognition is considered as high level computer vision task in which techniques can be developed to achieve accurate results.

Gender recognition methods often rely on face recognition. There are many gender classification approaches and methods. The gender recognition might be one of the factors that fastens the person authentication because you limit the search for a person to 50 percent of the population.

We have also performed emotion detection in our project as human emotions can have great impact on the way of using software and in various other dimensions of our life. Therefore, robust and accurate recognition of human emotional states can play an important role in many software systems in different application fields.

Most facial recognition systems function based on the different nodal points on a human face. The values measured against the variable associated with points of a persons face help in uniquely identifying or verifying the person. With this technique, applications can

use data captured from faces and can accurately and quickly identify target individuals. We can perform fast, accurate face detection with OpenCV using a pre-trained deep learning face detector model shipped with the library. OpenCV ships out-of-the-box with pre-trained Haar cascades that can be used for face detection.
To build our face recognition system, we will first perform face detection, extract face embeddings from each face using deep learning, train a face recognition model on the embeddings, and then finally recognize faces in images with OpenCV.
A computer program that decides whether an image is a positive image or negative image is called a classifier. A classifier is trained on hundreds of thousands of face and non-face images to learn how to classify a new image correctly. OpenCV provides us with two pre-trained and ready to be used for face detection classifiers:

- Haar Classifier
- LBP Classifier

We have used Haar Classifier with OpenCV for face detection in this project. The Haar Classifier is a machine learning based approach, an algorithm created by Paul Viola and Michael Jones; which are trained from many many positive images (with faces) and negatives images (without faces).For improvement of Face detection we make use of Dlib Library.
Fisherface is one of the popular algorithms used in face recognition, and is widely believed to be superior to other techniques, such as eigenface because of the effort to maximize the separation between classes in the training process. And therefore we will use FisherFaces for emotion and gender detection.We train Fisherface Recognizer using images from KDEF(Karolinska Directed Emotional Faces) which contains images of 70 individuals each displaying 7 different emotions,each emotion being photographed(twice) from 5 different angles. Also images from IMDB-WIKI dataset have been taken which is largest publicly available dataset of face images with gender and age labels for training.It contains 4,60,723 face images from 20,284 celebrities from IMDB and 62,328 from wikipedia thus 5,23,051 in total.

Chapter 2

Related Work

Li Cuimei[1] has proposed solution in Eyes Detection and Mouth Detection with in a Human Face Candidate.A new human face detection algorithm is proposed to implement a stronger whole detection system by appending three weak classifiers, i.e., a classifier based on skin tone histogram matching, a classifier based on eyes detection, and third, a classifier based on mouth detection.

Chung-Hua Chu, Yu-Kai Feng [2] proposed a method that detects the movements of eyeball and the number of eye blinking to improve face recognition for screen unlock on the mobile devices. In some cases, people can use the photos and face masks to hack mobile security systems, so they proposed an eye blinking detection, which finds eyes through the proportion of human face.

Bruce Poon, Hong Yan [3] proposed a PCA Based Human Face Recognition with Improved Method for Distorted Images due to Facial Makeup. Facial makeup may change the appearance of a face which can degrade the accuracy of an automatic face recognition system. Experiment results show that by applying the Gradientfaces technique at the preprocessing stage which computes the orientation of the image gradients in each pixel of the face images the recognition rates can be improved for a mixture of facial images with makeup and nonmakeup.

H. Salih and L. Kulkarni [4] proposed to present the methodologies in terms of feature extraction and classification used in facial expression and/or emotion recognition methods with their comparative study. The comparative study is done based on accuracy, implementation tool, advantages and disadvantages.

Chapter 3

Proposed Work

Our entire project is divided into the following parts :

1. Visualization of facial landmarks and implementing facial recognition on the frames obtained by the web-cam or on sample images. We have two approaches for it :
   - OpenCV library
   - Haar Classifier
   - Dlib Library
2. Gender and Emotion Detection to myriad of emotions ranging from happy to sad, disgusted, surprised,afraid,angry and neutral. We have used the following technique to obtain a high degree of accuracy :
   - FisherFace, OpenCV and Dlib

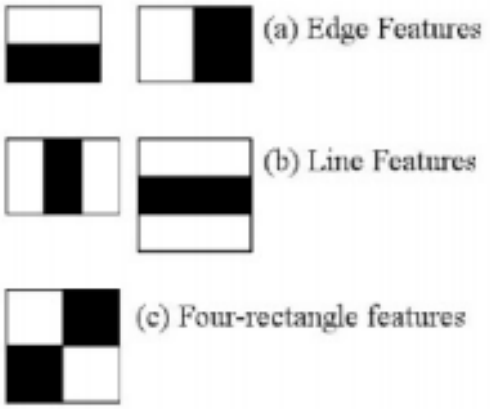
3.1 OpenCV

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code. The library has

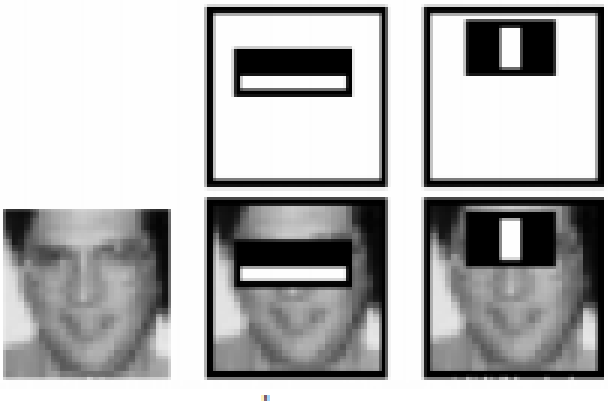
more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving 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.
The library is used extensively in companies, research groups and by governmental bodies.OpenCV was designed for computational efficiency with a strong focus on real-time applications.The library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous computing platform.
To be able to recognize emotions on images we will use OpenCV. OpenCV has a few face recognizer classes that we can also use for emotion recognition. They use different techniques, of which we will mostly use the Fisher Face one.

3.2 Haar Classifier

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images. Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



For example consider the image below.The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose. But the same windows applied to cheeks or any other place is irrelevant. So how do we select the best features out of 160000+ features? It is achieved by Adaboost.

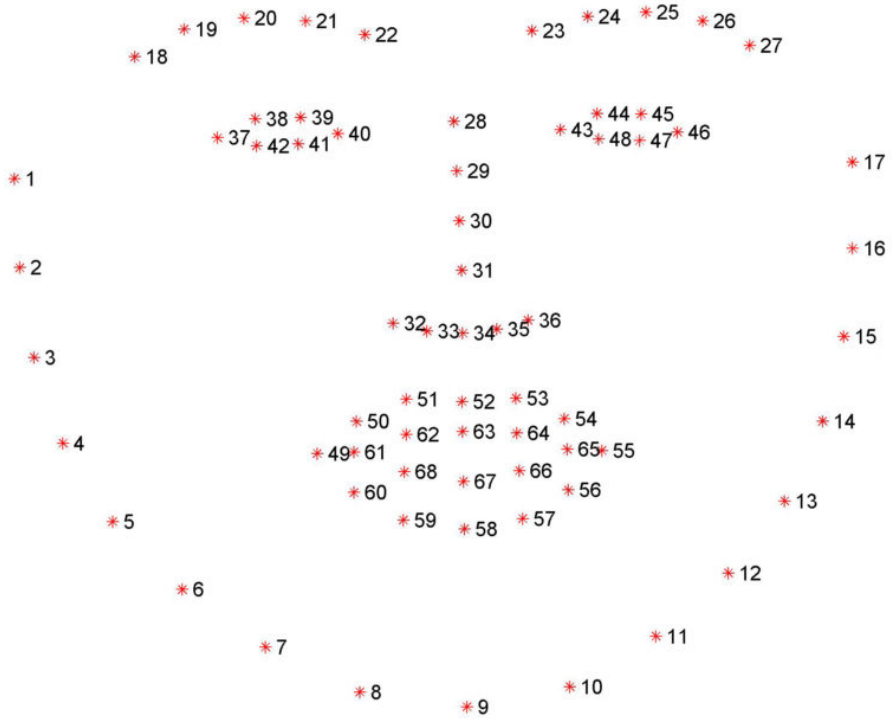


For this, we apply each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or mis-classifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong

classifier. The paper says even 200 features provide detection with 95 percent accuracy. Their final setup had around 6000 features. In an image, most of the image is non-face region. So it is a better idea to have a simple method to check if a window is not a face region. If it is not, discard it in a single shot, and don't process it again. Instead, focus on regions where there can be a face. This way, we spend more time checking possible face regions. For this they introduced the concept of Cascade of Classifiers. Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and applied one-by-one. If a window fails the first stage, discard it. We don't consider the remaining features on it. If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region. This is the plan which we are trying to achieve.

3.3 Dlib Library

Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.Its design is heavily influenced by ideas from design by contract and component-based software engineering. Thus it is, first and foremost, a set of independent software components. It is open-source software released under a Boost Software License.Dlib contains software components for dealing with networking, threads, graphical user interfaces, data structures, linear algebra, machine learning, image processing, data mining, XML and text parsing, numerical optimization, Bayesian networks, and many other tasks. The facial landmark detector implemented inside dlib produces 68 (x, y)-coordinates that map to specific facial structures. These 68 point mappings were obtained by training a shape predictor on the labeled iBUG 300-W dataset.



Examining the image, we can see that facial regions can be accessed via simple Python indexing (assuming zero-indexing with Python since the image above is one-indexed):

- The mouth can be accessed through points [48, 68].
- The right eyebrow through points [17, 22].
- The left eyebrow through points [22, 27].
- The right eye using [36, 42].
- The left eye with [42, 48].
- The nose using [27, 35].
- And the jaw via [0, 17].

These mappings are encoded inside the FACIAL_LANDMARKS_IDXS dictionary inside face_utils of the imutils library.Using this dictionary we can easily extract the indexes into the facial landmarks array and extract various facial features simply by supplying a string as a key.
FACIAL_LANDMARKS_IDXS = OrderedDict([("mouth", (48, 68)), ("right_eyebrow",

(17, 22)), ("left_eyebrow", (22, 27)), ("right_eye", (36, 42)), ("left_eye", (42, 48)), ("nose", (27, 35)), ("jaw", (0, 17))])
We will need the visualize_facial_landmarks function, which is already included in the imutils library.Our visualize_facial_landmarks function requires two arguments, followed by two optional ones, each detailed below:

- image : The image that we are going to draw our facial landmark visualizations on.
- shape : The NumPy array that contains the 68 facial landmark coordinates that map to various facial parts.
- colors : A list of BGR tuples used to color-code each of the facial landmark regions.
- alpha : A parameter used to control the opacity of the overlay on the original image.

```
def visualize_facial_landmarks(image, shape, colors=None, alpha=0.75):
    #create two copies of the input image – one for the
    #overlay and one for the final output image
    overlay = image.copy()
    output = image.copy()
    #if the colors list is None, initialize it with a unique
    #color for each facial landmark region
    if colors is None:
        colors = [(19, 199, 109), (79, 76, 240), (230, 159, 23), (168, 100, 168),
        (158, 163, 32), (163, 38, 32), (180, 42, 220)])
```

Thus, We are now ready to visualize each of the individual facial regions via facial landmarks. Now we have to extract the parts of the face using dlib where we instantiate dlib's HOG-based face detector and load the facial landmark predictor.Then we will compute a bounding box over the ROI.This ROI is then resized to have a width of 250 pixels so we can better envision it. At last we are going to visualize the image using different transparent overlays with each facial landmark region highlighted with a different color.

3.4 FisherFace Recognizer

A key problem in computer vision, pattern recognition and machine learning is to define an appropriate data representation for the task at hand. One way to represent the input data is by finding a subspace which represents most of the data variance. This can be obtained with the use of Principal Components Analysis (PCA). When applied to face images, PCA yields a set of eigenfaces. These eigenfaces are the eigenvectors associated to the largest eigenvalues of the covariance matrix of the training data. The eigenvectors thus found correspond to the least-squares (LS) solution. This is indeed a powerful way to represent the data because it ensures the data variance is maintained while eliminating unnecessary existing correlations among the original features (dimensions) in the sample vectors. When the goal is classification rather than representation, the LS solution may not yield the most desirable results. In such cases, one wishes to find a subspace that maps the sample vectors of the same class in a single spot of the feature representation and those of different classes as far apart from each other as possible. The techniques derived to achieve this goal are known as Discriminant analysis (DA). The most known DA is Linear Discriminant Analysis (LDA), which can be derived from an idea suggested by R.A. Fisher in 1936. When LDA is used to find the subspace representation of a set of face images, the resulting basis vectors defining that space are known as Fisherfaces. Fisherface algorithm considers the ratio between the variation of one person and that of another person. That is to say, it maximizes the determinant of between-class scatter matrix simultaneously, minimizing the determinant of within-class scatter matrix. Fisherface procedure is as the following. Let there be total N images and total c persons. Suppose the number of images from one person is K. From PCA we can get N1 eigenfaces. To minimize the determinant of within-class scatter matrix and maximize that of between-class scatter matrix, we constitute the S_w^{-1} s_b matrix and get Fisherfaces.

The definition of S_w and s_b is as following:

$$S_w = \sum_{i=1}^{c} \sum_{j=1}^{K} (y_j - M_i)(y_j - M_i)^T$$

$$S_b = \sum_{i=1}^{c} (M_i - M)(M_i - M)^T$$

where Mi is the mean vector of ith class and M is the mean vector of all classes. The cl dimensional feature vectors are obtained by projecting feature vectors of PCA onto the Fisherface matrix. And then, we recognize face by using predetermined feature vectors.

Chapter 4

Experimental Setup and Results Analysis

We have used Haar Classifier provided in OpenCV for face detection and then Dlib which is improvement over OpenCV for face detection.
We have also used Fisher Face Recognizer provided in OpenCV for gender detection and emotion detection. The latest version of OpenCV we are using is 3.2.0.

4.1 Face detection using Haar Classifiers

The following dependencies are required for detection using haar classifiers-
1.OpenCV should be installed.
2.Python should be installed.
3.Matplotlib should be installed.

Let's import the necessary libraries first.

```
#import OpenCV library
import cv2
import matplotlib library
import matplotlib.pyplot as plt
#importing time library for speed comparisons of both classifiers
import time
```

To read/load our image and convert it to grayscale, I used OpenCVs built in function cv2.imread(img_path) and passed our image path as the input parameter.

test1 = **cv2.imread**('data/test1.jpg')
#convert the test image to gray image as opencv face detector expects gray images
gray_img = **cv2.cvtColor**(test1, cv2.COLOR_BGR2GRAY)

Before we can continue with face detection, we have to load our Haar cascade classifier.
OpenCV provides us with a class **cv2.CascadeClassifier** which takes as input the training file of the Haar classifier we want to load and loads it for us. The XML files of Haar Classifier are stored in the opencv/data/haarcascades/ folder.
Let's load up our classifier:
haar_face_cascade = cv2.CascadeClassifier('data/haarcascade_frontalface_alt.xml')

Now, how do we detect a face from an image using the CascadeClassifier we just loaded? Well, again OpenCV's CascadedClassifier has made it simple for us as it comes with the function detectMultiScale, which detects exactly that. Next are some details of its options/arguments:

- **detectMultiScale(image, scaleFactor, minNeighbors):** This is a general function to detect objects, in this case, it will detect faces since we called in the face cascade. If it finds a face, it returns a list of positions of said face in the form Rect(x,y,w,h)., if not, then returns None.
- **Image:** The first input is the grayscale image. So make sure the image is in grayscale.
- **scaleFactor:** This function compensates a false perception in size that occurs when one face appears to be bigger than the other simply because it is closer to the camera.
- **minNeighbors:**This is a detection algorithm that uses a moving window to detect objects, it does so by defining how many objects are found near the current one before it can declare the face found.

The following code will detect a face from the image
#let's detect multiscale (some images may be closer to camera than others) images
faces = **haar_face_cascade.detectMultiScale**(gray_img, scaleFactor=1.1, minNeighbors=5)
Next, let's loop over the list of faces (rectangles) it returned and drew those rectangles using yet another built-in OpenCV rectangle function on our original colored image to see if it found the right faces:

for (x, y, w, h) in faces:
    cv2.rectangle(img_copy, (x, y), (x+w, y+h), (0, 255, 0), 2)
Now to finally display detected faces we use following functions:

- **plt.imshow(img,color_map):** This is a matplotlib function used to display an image. It takes two arguments;the first one is the image you want to plot and the second is the colormap (gray, RGB) in which the image is in.
- **cv2.imshow(window_name, image):** This is a cv2 function used to display the image. It also takes two arguments: the first one is the name of the window that will pop-up to show the picture and the second one is the image you want to display.
- **cv2.waitKey():**This is a keyboard binding function, which takes one argument: (x) time in milliseconds. The function delays for (x) milliseconds any keyboard event. If (0) is pressed, it waits indefinitely for a keystroke, if any other key is pressed the program continues.
- **cv2.destroyAllWindows():** This simply destroys all the windows we created using cv2.imshow(window_name, image)

cv2.imshow('Test Image', gray_img)
cv2.waitKey(0)
cv2.destroyAllWindows()

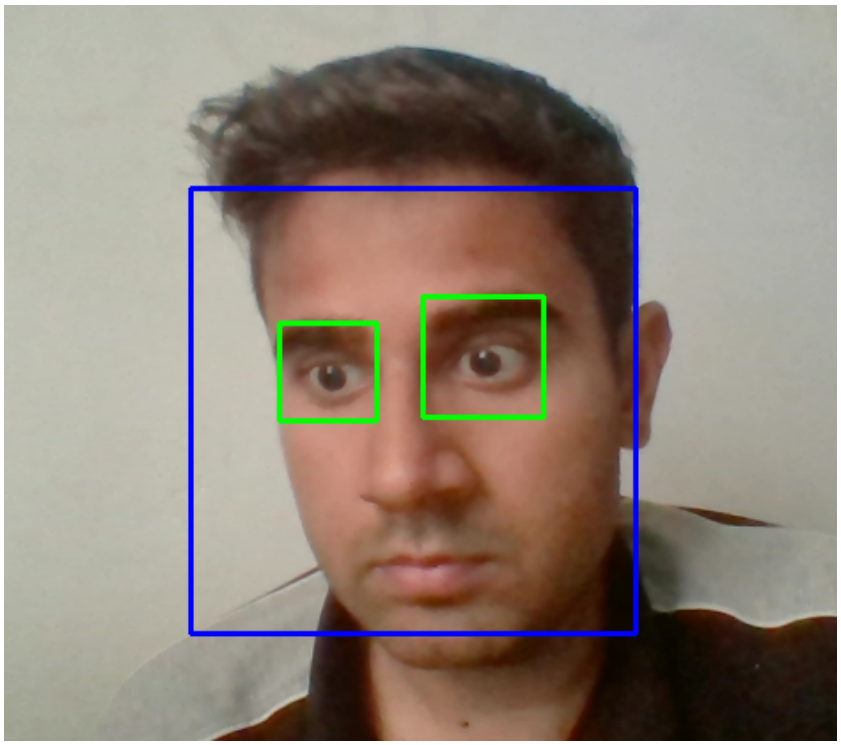


Figure 1: Face and eye detection using Haar Classifiers

4.2 Face detection using Dlib Library

4.2.1 Extracting parts of the face using dlib, OpenCV, and Python

Requirements

1. dlib library
2. imutils to the latest version, ensuring we have access to the face_utils submodule:
   pip install –upgrade imutils

Methodology:

We perform the following steps:

- Import our required Python packages.
- Parse our command line arguments.
- Instantiate dlibs HOG-based face detector and load the facial landmark predictor
- Load and pre-process our input image.

- Detect faces in our input image.

```
#import the necessary packages
from imutils import face_utils
import numpy as np
import argparse
import imutils
import dlib
import cv2
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-p", "–shape-predictor", required=True, help="path to facial landmark predictor")
ap.add_argument("-i", "–image", required=True, help="path to input image")
args = vars(ap.parse_args())
# initialize dlib's face detector (HOG-based) and then create
# the facial landmark predictor
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor(args["shape_predictor"])
#load the input image, resize it, and convert it to grayscale
image = cv2.imread(args["image"])
image = imutils.resize(image, width=500)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
# detect faces in the grayscale image
rects = detector(gray, 1)
```

Now that we have detected faces in the image, we can loop over each of the face ROIs individually:

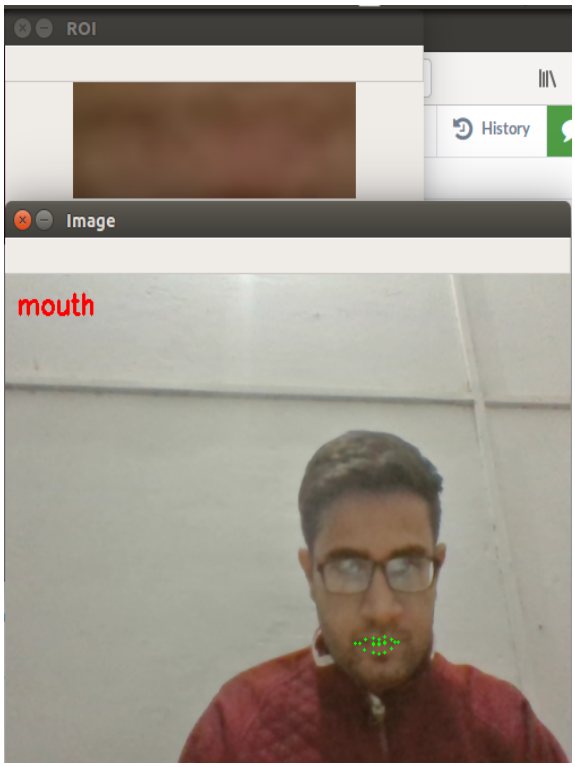
```
# loop over the face detections
for (i, rect) in enumerate(rects):
    # determine the facial landmarks for the face region, then
    # convert the landmark (x, y)-coordinates to a NumPy array
    shape = predictor(gray, rect)
```

```
shape = face_utils.shape_to_np(shape)
# loop over the face parts individually
for (name, (i, j)) in face_utils.FACIAL_LANDMARKS_IDXS.items():
    # clone the original image so we can draw on it, then
    # display the name of the face part on the image
    clone = image.copy()
    cv2.putText(clone, name, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.7,
    (0, 0, 255), 2)
    # loop over the subset of facial landmarks, drawing the
    # specific face part
    for (x, y) in shape[i:j]:
        cv2.circle(clone, (x, y), 1, (0, 0, 255), -1)
```

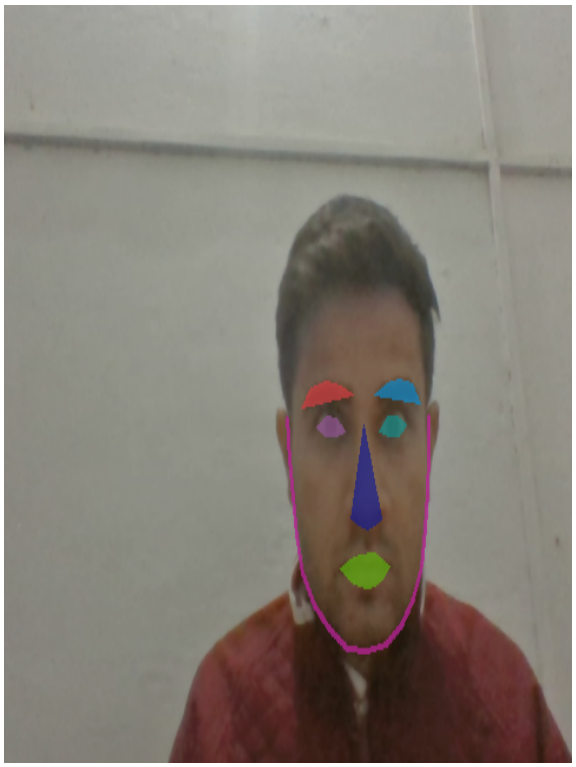
For each face region, we determine the facial landmarks of the ROI and convert the 68 points into a NumPy array. Then, for each of the face parts, we loop over them. We draw the name/label of the face region then draw each of the individual facial landmarks as circles. To actually extract each of the facial regions we simply need to compute the bounding box of the (x, y)-coordinates associated with the specific region and use NumPy array slicing to extract it:

```
    # extract the ROI of the face region as a separate image
    (x, y, w, h) = cv2.boundingRect(np.array([shape[i:j]]))
    roi = image[y:y + h, x:x + w]
    roi = imutils.resize(roi, width=250, inter=cv2.INTER_CUBIC)
    #show the particular face part
    cv2.imshow("ROI", roi)
    cv2.imshow("Image", clone)
    cv2.waitKey(0)
# visualize all facial landmarks with a transparent overlay
output = face_utils.visualize_facial_landmarks(image, shape)
cv2.imshow("Image", output)
cv2.waitKey(0)
```

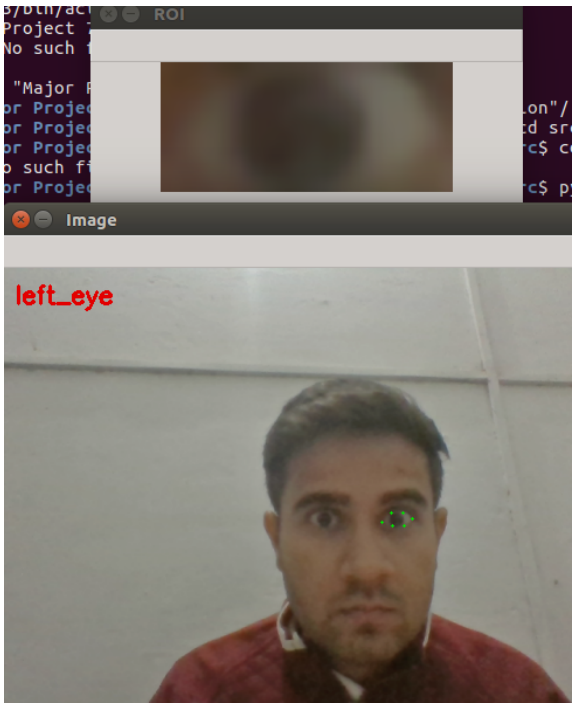
Computing the bounding box of the region is handled via cv2.boundingRect(). Using NumPy array slicing we can extract the ROI. This ROI is then resized to have a width of 250 pixels so we can better visualize it. Then we display the individual face region to our screen. We then apply the visualize_facial_landmarks function to create a transparent overlay for each facial part.



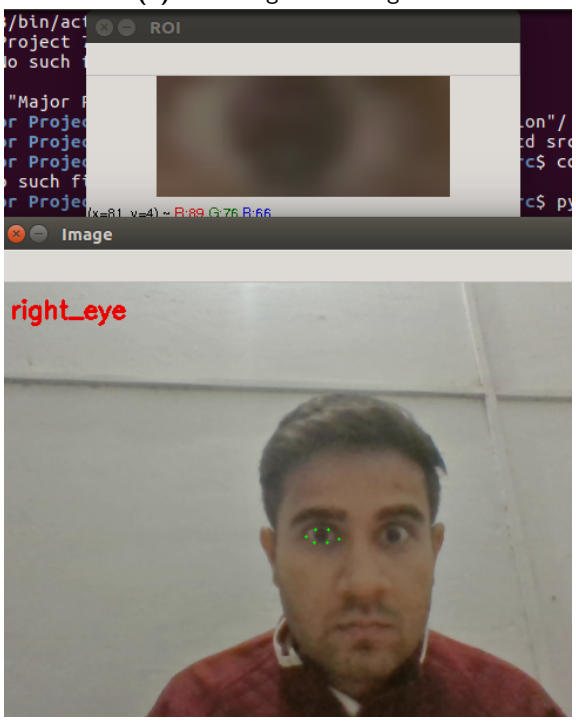
(a) Detecting mouth



(b) Detecting all face regions

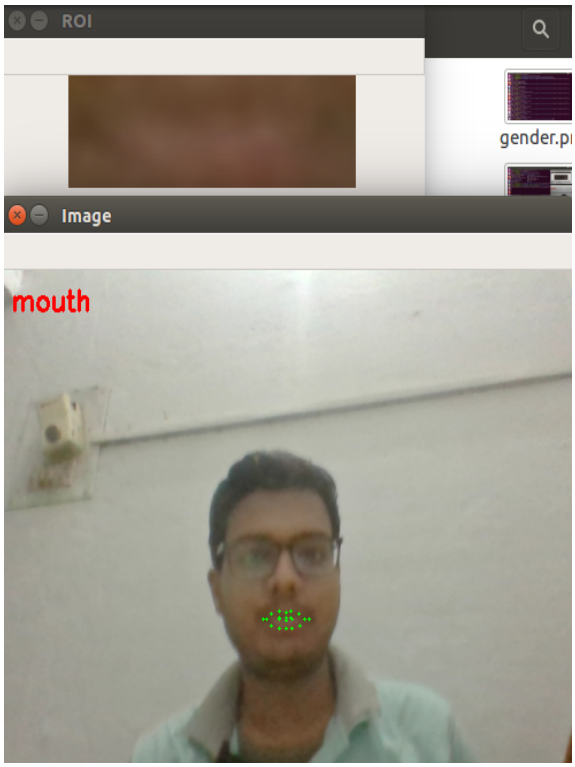


(c) Detecting left eye

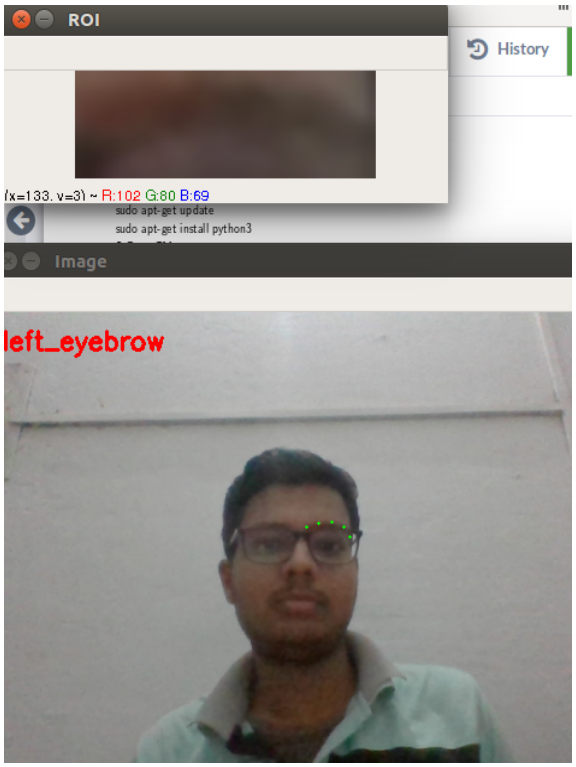


(d) Detecting right eye

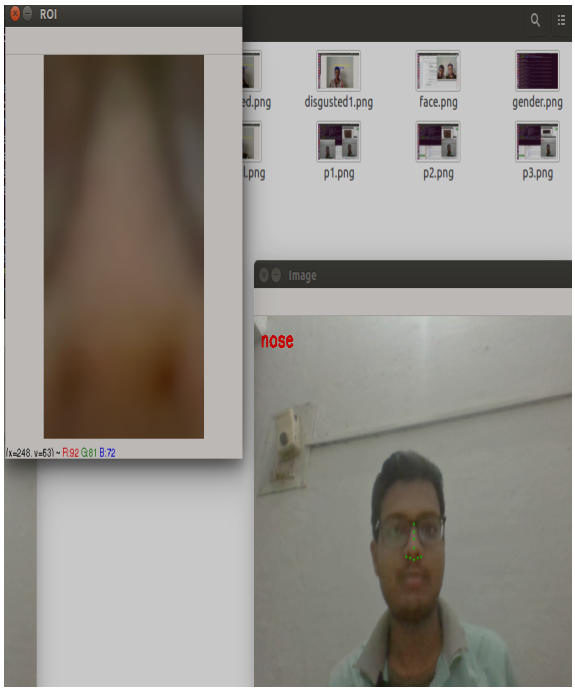
Figure 2: Detecting facial regions using Dlib(person 1)



(a) Detecting mouth



(b) Detecting left eyebrow



(c) Detecting nose

(d) Detecting all face regions

Figure 3: Detecting facial regions using Dlib(person 2)

4.3 Gender and Emotion detection using FisherFace Recognizer

4.3.1 Requirements

1.Python3:

- **On linux open terminal and type:**

sudo apt-get update
sudo apt-get install python3

2.OpenCV
Once you have installed python3 type the following commands:
pip3 install opencv-python
pip3 install opencv-contrib-python

4.3.2 Running the program

Run the file facifier.py with the command **python** facifier.py
The program would ask you to use your webcam. **Type y for yes and n for no.**

Webcam Mode
If you choose y, the program would continuously process the images that is recorded through your webcam.

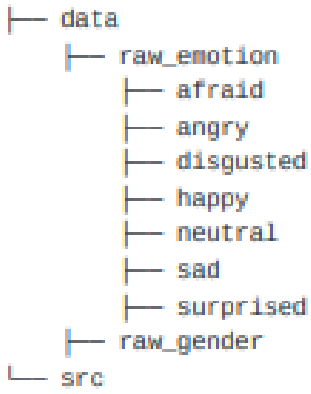
Static Image Mode
Otherwise, if you chose n, the program would ask for an image file to be processed. Put the image you want to process, preferably in jpg format, to the folder data/sample and simply type the file name to process it.

4.3.3 Data Preprocessing

Fisherface recognizer requires every training data to have the same pixel count. This raises a problem because the dataset from KDEF and IMDB does not have uniform size and thus produces error during training. To address this problem, emotion_data_prep.py and gender_data_prep.py are created. Both of them use face detection algorithm from

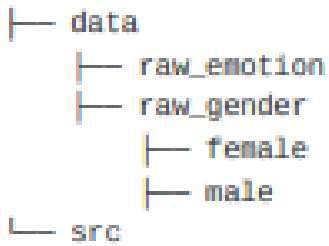
face_detection.py to detect faces in photos. Then, the picture would be normalized to uniform size (350px x 350px) and saved in grayscale to speed up the training process.

KDEF



Before running emotion_data_prep.py , ensure that your file structure is as pictured above. You would need to extract the KDEF images and put them in the respective directories that resembles the appropriate emotion. While the KDEF database has the image shot from multiple angles, I only use one angle, where the person is staring straight to the camera.

IMDB-Wiki



Before running gender_data_prep.py , ensure that your file structure is as pictured above.

You would need to extract the IMDB-Wiki images and put them in the respective directories that resembles the appropriate gender. You could also use images from KDEF database, just put the males in data/raw_gender/male and the females in data/raw_gender/female.

We do not have to include all images from the IMDB-Wiki. A personal computer with 8GB RAM could only handle at most around 2000 photos.

4.3.4 About Models

The included models are essential for the program to detect faces, emotions, and genders.

HaarCascade
These models are provided by OpenCV and allows the program to detect human faces. After some manual and automated testings, I decided to use the first alternate version. If for some reason you want to change the way this program detect human faces, open **face_detection.py**, search the following line:
faceCascade = **cv2.CascadeClassifier**('models/haarcascade_frontalface_alt.xml') and change the model path to the desired one.

Emotion Classifier
These models are created with **train_emotion_classifier.py**. Each model is trained with dataset from KDEF. There are 2 versions: normal and decent. The normal version is trained with all the data from KDEF, while the decent version is trained with modified data from KDEF.
Modified here means deleting obviously misleading emotions. For example, there was a picture labelled sad that shows the person smiling while having tears around the eyes. It is very unusual for people to smile while crying, but this one person does it. To achieve better result, the said picture is removed from dataset. Another example, a person shows no real emotion in a picture labelled angry. That particular picture is then re-labelled as neutral.
To switch versions, open **facifier.py** and search the following line:
fisher_face_emotion.read('models/emotion_classifier_model.xml') and change the model path to the desired one.

Gender Classifier

These models are created with **train_gender_classifier.py**. There are 3 versions: normal, KDEF, and IMDB. The normal version is trained with both KDEF and IMDB datasets. While KDEF and IMDB is trained with just KDEF or IMDB respectively.
Due to memory limitation only a handful of photos (2000+) from IMDB is used in building the normal and IMDB version. The best result is indeed achieved using the normal version which combined both KDEF and IMDB.
To switch versions, open **facifier.py** and search the following line:
fisher_face_gender.read('models/gender_classifier_model.xml') and change the model path to the desired one.

4.3.5 About FacePlay

Emotions under FacePlay

FacePlay can classify 7 basic emotions: afraid, angry, disgusted, happy, neutral, sad, and surprised.

Meaning of colored boxes around the faces

The box highlights any detected face whether from the webcam or in a static image. A blue box indicates that Facifier classified that person as a male and a red box indicates that Facifier classified that person as a female.

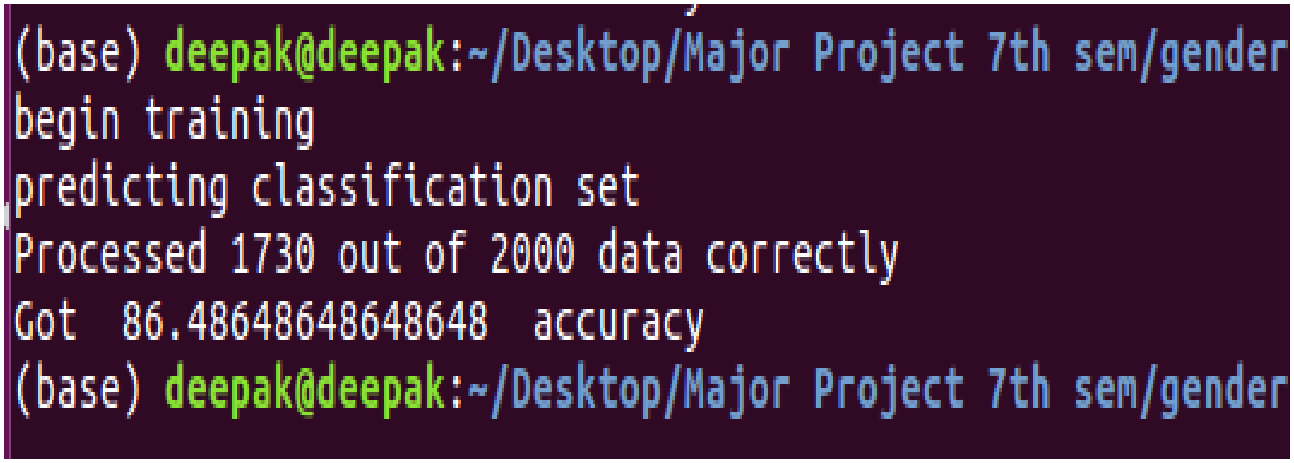
Training our model

To train a model for emotion classifier, put images of each of the 7 emotions mentioned above in **data/raw_emotion/** . For example, put the images that show happy emotion in **data/emotion/happy/** Then, run:
python3 emotion_data_prep.py
python3 train_emotion-classifier.py
However, training model for gender classifier is more difficult due to lack of datasets available. Put the images of each gender in **data/raw_gender/** .For example, put the images of females in data/raw_gender/female Then, run:
python3 gender_data_prep.py

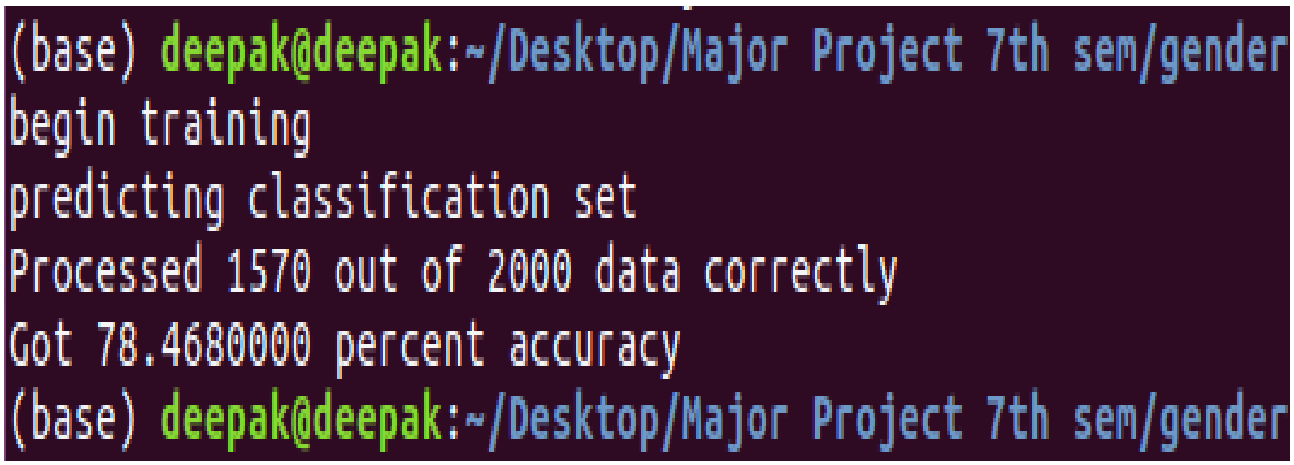
python3 train_gender_classifier.py

Why Fisherface?

Fisherface is not the only available recognizer in OpenCV, so why specifically choose it over the others? It is better than using Eigenface recognizer. Eigenfaces are the eigenvectors associated to the largest . It uses integer value in its prediction. This value is the corresponding eigenvalue of the eigenvector. Meanwhile, Fisherface uses Linear Discriminant Analysis (LDA) to determine the vector representation. It produces float value in the prediction. This also means that the result is better compared to Eigenface.

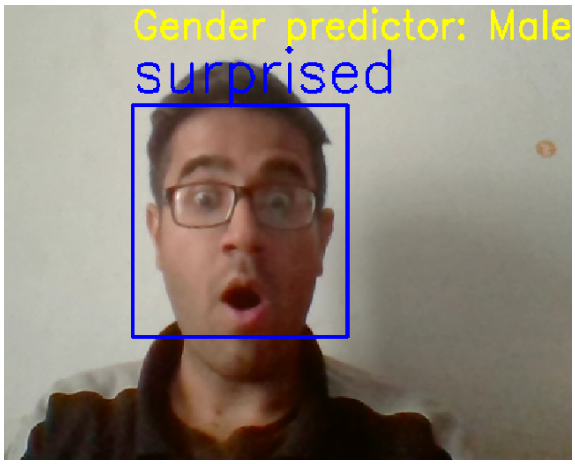


(a) Accuracy score on training FisherFace Recognizer for gender detection using 2000 images.

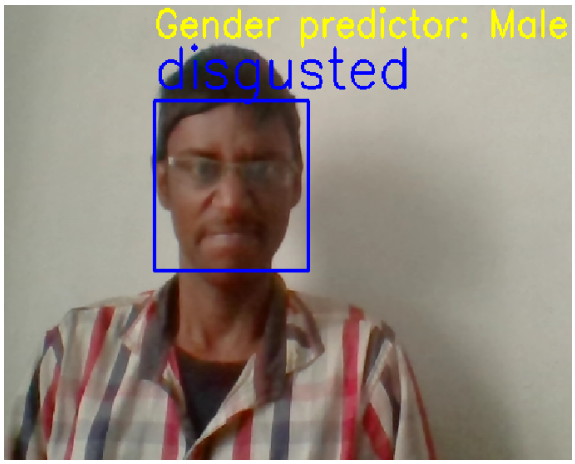


(b) Accuracy score on training FisherFace Recognizer for emotion detection using 2000 images.

Figure 4: Accuracy score

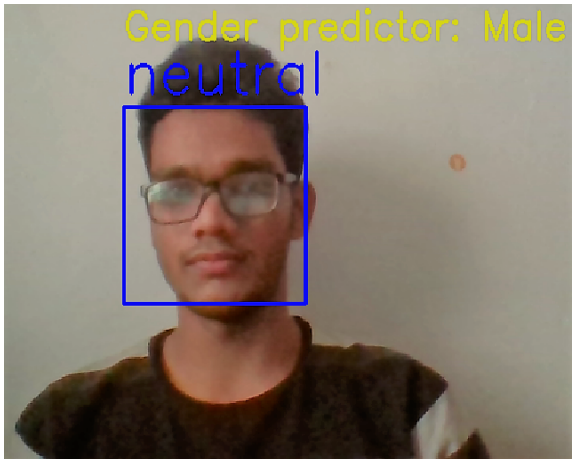


(a) gender: Male, emotion : surprised

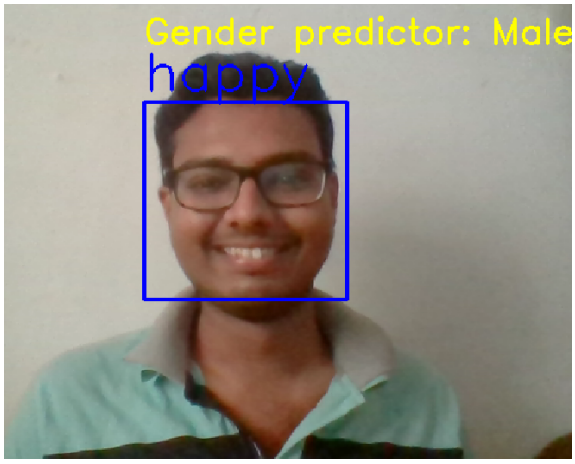


(b) gender : Male, emotion : disgusted

(c) gender: Male, emotion : sad



(d) gender: Male, emotion : neutral



(e) gender: Male, emotion : happy

(f) gender: Male, emotion : angry

Figure 5: Gender and Emotions detected using FisherFace Recognizer

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The facial expression recognition system presented in this research work contributes a resilient face recognition model based on the mapping of behavioural characteristics with the physiological biometric characteristics. The physiological characteristics of the human face with relevance to various expressions such as happiness, sadness, fear, anger, surprise and disgust are associated with geometrical structures which restored as base matching template for the recognition system.

5.2 Future Work

Facial recognition is becoming more prominent in our society. Over the past few years, we have seen major developments to facial recognition technology.We have established ground level work for gender as well as emotion detection. We wish to add "degree" or "value" to a certain emotion, like rating on scale of 0-1. For e.g. a 0.2 can indicate little happiness, 0.5 with normal happiness and 1.0 with levels of happiness similar to euphoria.This requires a very accurate dataset for training and thus will help us to understand how facial landmarks help us to mask a certain emotion on the face. Further, we can use the advantage of real time results and integrate the working structure into an Android App or an API which can be used as service. It will have the capability to send the emotions to other people at the same time so that they can know the emotions of people they are attached with at crucial times. An example would be the video call

between mother and son during hostel time, where the emotions can also be sent to the grandparents who are not present there but can now using our app perceive the nature of the video conversation. Further we can elevate our software to be used in wearable technology. Pivothead is a firm that employs use of various cognitive services API's in their eye-wear products which can benefit visually impaired people. Our project can be integrated with it and the person with visual impairment can utilize it to understand the emotions and gender of the person they are speaking to. Thus, our project can be wielded in various ways for benefits of customers using it.

References

[1] Li Cuimei,Qi Zhiliang,Jia Nan,Wu Jianhua , "Human face detection algorithm via Haar cascade classifier combined with three additional classifiers" *2017 13th IEEE International Conference on Electronic Measurement Instruments (ICEMI)*.

[2] Chung-Hua Chu and Yu-Kai Feng,Member of IEEE, "Study of Eye Blinking to Improve Face Recognition for Screen Unlock on Mobile Devices ," *J Electr Eng Technol.2017*.

[3] Bruce Poon, M. Ashraful Amin, and Hong Yan, "PCA Based Human Face Recognition with Improved Method for Distorted Images due to Facial Makeup," *International MultiConference of Engineers and Computer Scientists 2017 Vol I, IMECS 2017, March 15 - 17, 2017, Hong Kong* .

[4] H. Salih and L. Kulkarni, "Study of video based facial expression and emotions recognition methods," 2017 International Conference on I-SMAC *(IoT in Social, Mobile, Analytics and Cloud) (I-SMAC), Palladam, 2017, pp. 692-696.doi: 10.1109/I-SMAC.2017.8058267*.