M. Tech. Dissertation Report

Simultaneous Face Detection, Pose Estimation, Gaze Estimation, Face Recognition in gathering using ERT

IS SUBMITTED IN PARTIAL FULFILLMENT OF THE MASTER OF TECHNOLOGY IN EMBEDDED SYSTEMS

Submitted By

SARANSH BHANDARI

2017PEB5421

Supervisor

Mr. RAKESH BAIRATHI

Associate Professor, ECE

Department of Electronics and Communication Engineering

Malaviya National Institute of Technology, Jaipur, India

June, 2019

CERTIFICATE

Department of Electronics and Communication

Malaviya National Institute of Technology, Jaipur Rajasthan – 302017

This is to certify that the Dissertation Report on *"Simultaneous Face Detection and Pose Estimation"* by Saransh Bhandari (2017PEB5421) is bonafide work completed under my supervision, hence approved for submission in partial fulfillment for the Master of Technology in EMBEDDED SYSTEMS, Malaviya National Institute of Technology, Jaipur during academic session 2018-2019 for the full time post graduation program of session 2017-2019. The work has been approved after plagiarism check as per institute rule.

Date:

Mr. Rakesh Bairathi

Associate Professor, E.C.E Dept.

MNIT, Jaipur

DECLARATION

This is to certify that the dissertation report entitled **“Simultaneous Face Detection and Pose Estimation”** is being submitted by me in partial fulfillment of degree of Master of Technology in Embedded Systems during 2018-2019 in a research work carried out by me under supervision of **Mr. Rakesh Bairathi** and content of this dissertation work in full or in parts have not been submitted to any other institute or university for award of any degree or diploma. I am fully responsible for the material used in this report in case if any discrepancies arises. Wherever I have consulted the published work(in any form) of others, it is clearly attributed. Where I have quoted from the work of others, the source is always given. I have acknowledged all the relevant (to best of my knowledge) sources in the report. With the exception of above , this report is entirely my own work.

Saransh Bhandari

(2017PEB5421)

ACKNOWLEDGMENT

I take this opportunity to express my immense respect and gratitude towards my Supervisor (Guide), **Mr. Rakesh Bairathi** , Associate Professor, Department of Electronics & Communication Engineering, Malaviya National Institute of Technology. I am humbled and awed by his patience, dedication and commitment as a teacher/mentor/guide during the entire process of completion of my project as well as my studies. Having received his advice and guidance has sincerely convinced me and motivated me to complete this project during tough and hard times. Without his support and timely guidance, the completion of my project would have been a tough ask. I sincerely thank his candid approach towards tough subjects that he has discussed about and shared his opinion and knowledge with me.

I would like to thank **Dr. Dharmendra Boolchandani**, Head Of Department, Department of Electronics & Communication Engineering and **Dr. Vineet Sahula,** Professor, Department of Electronics & Communication Engineering for their co-operation and help rendered in numerous ways for the successful completion of this work.

I would also like to thank **Mr. Nirdosh Kumar** and many of my friends who have helped along the way of completion of this work.

I take this opportunity to express my regards and obligation to my family and mother whose belief in me has motivated and convinced me to follow my passion in life as well as studies.

Lastly, I am thankful to all those who have supported me directly or indirectly during the dissertation work.

Saransh Bhandari

(2017PEB5421)

Table of Contents

CERTIFICATE **2**
DECLARATION **3**
ACKNOWLEDGMENT **4**
List of Figures **8**
List of Tables **10**
Abbreviations **11**
Abstract **12**
1 Chapter1 - Introduction **13**
1.1 Motivation 13
1.2 Background 13
1.3 Thesis Organisation 14
2 Chapter 2- Literature survey **15**
2.1 Literature Survey 15
3 Chapter 3 – Hardware, Software, Language and Packages Used **19**
3.1 Hardware Used 19
3.1.1 System Specifications 19
3.1.2 CPU Specification 19
3.2 Software Used 19
3.3 Libraries Used 20
3.3.1 OpenCV 20
3.3.2 DLib 21
3.3.3 Numpy 21
3.3.4 Pandas 21
3.3.5 Math 21
3.3.6 Scikit-Learn 21
4 Chapter 4 – Techniques and Methodology **22**
4.1 Algorithms and techniques in image processing 22
4.1.1 Viola Jones Object Detection Algorithm 22
4.1.2 k-Nearest Neighbors (k-NN) Algorithm 23
4.1.3 CNN 23
4.1.4 Face Embedding 24
4.2 Ensemble Regression Tree used as ML Algorithm 26
4.2.1 Simple Ensemble Techniques 26
4.2.1.1 Max Voting 26

4.2.1.2 Averaging 27
4.2.1.3 Weighted Averaging 27
4.2.2 Advanced Ensemble Techniques 27
4.2.2.1 Stacking 27
4.2.2.2 Blending 27
4.2.3 Bagging 27
4.2.4 Boosting 28
4.2.5 Bagging Meta-estimator 29
4.2.6 Random Forest 29
4.3 ERT Model 29
4.3.1 Flowchart 29
4.3.2 Human Head Pose Detection 31
4.3.3 ERT Model Results 31

5 Chapter 5 – Proposed Work and Results 34

5.1 Dataset 34
5.2 Steps and Results 34
5.2.1 Step1: Getting Started 34
5.2.2 Step2 40
5.2.3 Step3 Final Results 42
5.2.3.1 Bounding box and Front of the face 42
5.2.3.2 Bounding Box along with x, y and z axis from nose. 43
5.3 Result and Analysis 44

6 Chapter 6 – Implementation of Gaze Estimation and Audio caption generation. 45

6.1 Gaze Estimation Implementation 45
6.2 Audio generation 46
6.3 Applications 47

7 Chapter 7 - Conclusion and Future Work 48

7.1 Conclusion 48
7.2 Future Works 48
7.3 References 49

List of Figures

Figure 3.1 This is the license for Pycharm Software Platform. 20
Figure 4.1 A classic image of challenges in Image Classification [7] 22
Figure 4.2 Haar cascades [9] 23
Figure 4.3 Haar cascade for face detection [9] 23
Figure 4.4 Artificial Neural network [10] 24
Figure 4.5 Model of CNN[11] 24
Figure 4.6 Flow Diagram for face encoding 25
Figure 4.7 Training resultfor face encoding 25
Figure 4.8 Bootstraping Flowchart 28
Figure 4.9 Flowchart of landmark predictor with bounding box using ML algorithm 30
Figure 4.10 2D mask for structure landmarks 30
Figure 4.11 3D mask for the structure landmarks 30
Figure 4.12 Flow Diagram 31
Figure 4.13 This is proposed logic image to detect the gaze of the human eye. 32
Figure 4.14 Face detection in dogs 32
Figure 4.15 output of face detection of dog 32
Figure 4.16 Training and Testing accuracy 32
Figure 4.17 Total face detected 33
Figure 4.18 Output of face detection 33
Figure 4.19 Total face detected 33
Figure 4.20 Output of face detection 33
Figure 4.21 Total face detection along with the position of boundary box 33
Figure 4.22 Output of face detection 33
Figure 5.1 This is the output of Region of Interest when looking center. 35
Figure 5.2 This is the output of Region of Interest when looking right. 35
Figure 5.3 This is the output of landmark object files 36
Figure 5.4 This is the output of landmark 36 36
Figure 5.5 This is the output of Horizontal length of the eye. 37
Figure 5.6 This is the output of the ratio of vertical to horizontal length. 37
Figure 5.7 This is the output of the left eye region 37
Figure 5.8 This is the output of rectangle on the detected structure 38
Figure 5.9 This is the output of circle on right eye corner. 38
Figure 5.10 This is the output caputured when blinking 38
Figure 5.11 This is output of identifying the structure of eyes in the structure identified. 38
Figure 5.12 This is the ouptut of required region extracted outside in new frame 38
Figure 5.13 This is the output of threshold at 70 and generating new frame of the structure. 38
Figure 5.14 This image has generated a mask for the identified structure in our region of interest. 39
Figure 5.15 This is the image of mask after bitwise_and with the original image. 39
Figure 5.16 This image is output of bitwise_and structure. 40
Figure 5.17 This is the output of an image with split region of interest. 40
Figure 5.18 This is the first output for looking left. 41
Figure 5.19 This is the first output for looking center. 41
Figure 5.20 This is the first output for looking right. 41

Figure 5.21 This is the output for ratio while the direction is left. 41
Figure 5.22 This is the output for ratio while the direction is center. 41
Figure 5.23 This is the output for ratio while the direction is right. 41
Figure 5.24 The image has boundary box on a class of students...42
Figure 5.25 The image shows the region of interest. 42
Figure 5.26 This is class of student along with the direction of pose in 1 axis only. 42
Figure 5.27 This is the output of Bounding box with x, y and z axis......................................43
Figure 6.1 Flow diagram for gaze estimation 45
Figure 6.2 The output displays the resultant cube for some of the faces. 46
Figure 6.3 Audio result of looking left. 46
Figure 6.4 Audio result of looking right 46

List of Tables

Table 3.1 Table containing the detailed specification of Workstation 19
Table 3.2 Table containing detailed specifications of CPU..19
Table 4.1 Core problems in Vision..22
Table 4.2 Face Recognition Dataset 26
Table 4.3 Boosting Steps 28
Table 4.4 Parameters utilized in the calculation of Bagging Meta-estimator..........................29
Table 4.5 Steps in a Random Forest model 29
Table 5.1 Table of Dataset used..34
Table 5.2 Result and Analysis 44
Table 6.1 Accuracy of Face Encoding...47

Abbreviations

AFW	Annotated Facial Landmark
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DFM	Dynamic Feature Matching
ERT	Ensemble Regression Technique
Ibug	Intelligent behaviour understanding group
KNN	K Nearest Neighbours System
LFLW	Labelled Face parts in the Wild
ML	ML Machine Learning
PFR	Partial Face Recognition
SRC	Sparse representation Classification
SRDT	Super resolution using Decision Tree
SVM	Support Vector Machine

Abstract

The concept of pattern analysis in humans is now, very actively used in various problems, from automatically identifying and tagging your friend in your Facebook post. The recent field of autonomous cars has to deal with humans and pattern analysis of the human pose, eye gaze, and pattern of body pose whether walking, running. There are challenges like ImageNet challenge, ibug challenge for face detection. Another challenge which should be addressed is attention span of the human during speech, examinations, classroom lectures.

The dissertation work looks into the subject of real-time human eye tracking for gaze estimation and pose estimation of head with respect to the 2D plane. In this work, we have used the landmarks needed for identifying the structure in the 2D image plane. The dissertation work is able to calculate blinking by using the delta (change) in the ratio between the horizontal and vertical distance between the identified pixels of our structure. Furthermore, gaze tracking is using the ratio of sclera and pupil to detect the direction of the gaze of the structure. The solution is running on video played on low specification laptop. The solution is able to recognise the detected face. The solution is able to generate an audio for the recognised face along with the direction of gaze.

Keyword: Face Detection, Pose Estimation, Gaze Estimation, Face Recognition, Ensemble Regression Tree.

1 Chapter1 - Introduction

1.1 Motivation

The dissertation work started to target the concept of easy, accessible, feasible and portable technology in the hands of all the people irrespective of rich and poor.

In present times, we consume the world with our eyes. With eyes closed, a human is literally disabled person. Now, if we can use this vital process of consumption for human beings to gain more information, better understanding in behaviour pattern of human then it will be good.

In product like remote controlled wheel chair, prosthetics, robots, there is a possibility to control them through eyes. Playing games, virtual reality as a concept also used Eye Tracking. The applications of this technique is in tracking better drivers in Ola and Uber, how long a person look at objects on a website before clicking away. Thus giving the website developers insight to change and update the website.

Thus, the idea of Pose Estimation in human beings with respect to head or in simple term Eye tracking as a subject is decided. This objective was to be able to track human eyes in real time.

1.2 Background

I attended faculty development workshops on Artificial Intelligence, machine learning and on deep learning and convolution neural networks. There was a subject computer vision in our 2nd semester, which further increased my curiosity towards the subject. Lately, there has been a boom in use of camera in mobile phone with increasing high megapixel camera added to the mobile phones. There are filters on Facebook, Instagram, snapchat and what not, that gives you interesting solutions to computer vision problems. I believe image detection is the solution to a lot of challenges and problems in our society. China, as a country, at government level, has developed algorithm based on image detection and is one of the pioneers in this field with cameras all over the country. They have systems that used face as a currency to pay for food, clothes etc. along with fingerprint and body pose. The recent use of image detection by Indian Armed Forces of UAV for reconnaissance has used image detection for identifying civilians & armed hostile forces. Hence, efficient and real time face detection and head pose estimation is proposed as the subject of this thesis.

The task of region detection in an image is one of the most challenging tasks in the field of Computer Vision. The region can be a human face, human head, watch on the wrist, or something more small like ring on the finger or any other accessories on human body. Some conventional techniques like applying a transform, detecting edges and shapes are used to detect some objects in an image. For example, body pose estimation will define some simple joints the human body like head, neck, chest, shoulder, elbow, fingertips, thighs, knees, toes. Now, the approach is to localize them in the image and then do the pattern analysis. The conventional techniques is to detect the shapes and edges and label them as part of the pattern.

1.3 Thesis Organisation

- Chapter 1

The thesis starts out with chapter 1 that briefs about the motivation for this work. The 2nd section of this chapter sheds some light on the background.

- Chapter 2

Chapter 2, as Literature survey, throws some light on the research work carried out by the others in recent years as well as on the problems.

- Chapter 3

Chapter 3 gives the details about the hardware, software, which we have used for work. In later part of the chapter, it talks about the failure with some of the software and why the Pycharm is used for the solution. This chapter talks about the language, which is used. It further talks about the open source libraries that we have used in the project.

- Chapter 4

Chapter 4 starts with the brief introduction of some of the techniques used in the past for computer vision problems. Later in the chapter, techniques we have extensively described the ERT model. The test results of the model is discussed in this chapter.
.

- Chapter 5

Chapter 5 discusses the dataset. The chapter further explains all the steps in the completion of the project. The explanation of the results at each steps is given along with it.

- Chapter 6

Chapter 6 covers the implementation of the improvement added to this work. The later part of the chapter explains the results and application of the results.

- Chapter 7

Chapter 7 covers the conclusion and the future scope along with enhancement which can be further be added to this work.

2 Chapter 2- Literature survey

2.1 Literature Survey

1. H. Wu, K. Zhang and G. Tian, "Simultaneous Face Detection and Pose Estimation Using Convolutional Neural Network Cascade," in *IEEE Access*, vol. 6, pp. 49563-49575, 2018. The paper does face detection. Face detection network detect face then do pose estimation. Region proposal network is used to identify possible region that may have face and do single deep multi task CNN to process these region for face detection and pose estimation. The network can't achieve real time thus need improvement. FDDB benchmark for face detection is used. AFW benchmark for pose estimation is used. Application: The head pose provides clues for judging people's motivation, intention and gaze. Recent studies show that convolutional neural networks (CNNs) has made a series of breakthroughs in the two tasks of face detection and pose estimation, respectively. There are two CNN frameworks for solving these two integrated tasks simultaneously. One is to use face detection network to detect faces firstly, and then use pose estimation network to estimate each face's pose; the other is to use region proposal algorithm to generate many candidate regions that may contain faces, and then use a single deep multi-task CNN to process these regions for simultaneous face detection and pose estimation. The former's problem is pose estimation's performance is affected by face detection network because two networks are separate. The latter generates lots of candidate regions, which will bring huge computation cost to CNN and can't achieve real-time.

2. L. He, H. Li, Q. Zhang and Z. Sun, "Dynamic Feature Matching for Partial Face Recognition," in *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 791-802, Feb. 2019. Partial face recognition (PFR) in an unconstrained environment is a very important task, especially in situations where partial face images are likely to be captured due to occlusions, out-of-view, and large viewing angle, e.g., video surveillance and mobile devices. However, little attention has been paid to PFR so far and thus, the problem of recognizing an arbitrary patch of a face image remains largely unsolved. This paper proposes a novel partial face recognition approach, called dynamic feature matching (DFM), which combines fully convolutional networks and sparse representation classification (SRC) to address partial face recognition problem regardless of various face sizes. DFM does not require prior position information of partial faces against a holistic face.

3. A.S. Dhavalikar and R. K. Kulkarni, "Face detection and facial expression recognition system," 2014 International Conference on Electronics and Communication Systems (ICECS), Coimbatore, 2014, pp. 1-7. A human-computer interaction system for an automatic face recognition or facial expression recognition has attracted increasing

attention from researchers in psychology, computer science, linguistics, neuroscience, and related disciplines. In this paper, an Automatic Facial Expression Recognition System (AFERS) has been proposed. The proposed method has three stages: (a) face detection, (b) feature extraction and (c) facial expression recognition. The first phase of face detection involves skin color detection using YCbCr color model, lighting compensation for getting uniformity on face and morphological operations for retaining the required face portion. The output of the first phase is used for extracting facial features like eyes, nose, and mouth using AAM (Active Appearance Model) method. The third stage, automatic facial expression recognition, involves simple Euclidean Distance method. In this method, the Euclidean distance between the feature points of the training images and that of the query image is compared. Based on minimum Euclidean distance, output image expression is decided. True recognition rate for this method is around 90% - 95%. Further modification of this method is done using Artificial Neuro-Fuzzy Inference System (ANFIS). This non-linear recognition system gives recognition rate of around 100% which is acceptable compared to other methods.

4. S. Kherchaoui and A. Houacine, "Facial expression identification system with Euclidean distance of facial edges," *2014 6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, Tunis, 2014, pp. 6-10. This paper present facial expression recognition system. Identification and classification is performed on the seven basic expressions: happy, surprise, fear, disgust, sad, anger and a neutral state. This system consists of three main parts. The first step is the detection of the face and facial features to extract the face centered region. Next step consists of a normalization of this interest region and edge extraction. At this step we have a face edge image that we use to calculate the Euclidean distance of all pixels that constitute edges. The third step is the classification of different emotional state by the SVM method.

5. V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1867-1874. This paper addresses the problem of Face Alignment for a single image. We show how an ensemble of regression trees can be used to estimate the face's landmark positions directly from a sparse subset of pixel intensities, achieving super-realtime performance with high quality predictions.

6. R. Ranjan, V. M. Patel and R. Chellappa, "HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 1, pp. 121-135, 1 Jan. 2019. This paper present an algorithm for simultaneous face detection, landmarks localization, pose estimation and gender recognition using deep convolutional neural networks (CNN). The proposed method called, HyperFace, fuses the intermediate layers of a deep CNN using a separate CNN

followed by a multi-task learning algorithm that operates on the fused features. It exploits the synergy among the tasks which boosts up their individual performances.

7. R. Ranjan, S. Sankaranarayanan, C. D. Castillo and R. Chellappa, "An All-In-One Convolutional Neural Network for Face Analysis," 2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017), Washington, DC, 2017, pp. 17-24. We present a multi-purpose algorithm for simultaneous face detection, face alignment, pose estimation, gender recognition, smile detection, age estimation and face recognition using a single deep convolutional neural network (CNN). The proposed method employs a multi-task learning framework that regularizes the shared parameters of CNN and builds a synergy among different domains and tasks.

8. S. Lawrence, C. L. Giles, Ah Chung Tsoi and A. D. Back, "Face recognition: a convolutional neural-network approach," in IEEE Transactions on Neural Networks, vol. 8, no. 1, pp. 98-113, Jan. 1997. The paper present a hybrid neural-network for human face recognition which compares favourably with other methods. The system combines local image sampling, a self-organizing map (SOM) neural network, and a convolutional neural network. The SOM provides a quantization of the image samples into a topological space where inputs that are nearby in the original space are also nearby in the output space, thereby providing dimensionality reduction and invariance to minor changes in the image sample, and the convolutional neural network provides partial invariance to translation, rotation, scale, and deformation.

9. Y. Taigman, M. Yang, M. Ranzato and L. Wolf, "DeepFace: Closing the Gap to Human-Level Performance in Face Verification," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1701-1708. In modern face recognition, the conventional pipeline consists of four stages: detect => align => represent => classify. The paper revisit both the alignment step and the representation step by employing explicit 3D face modeling in order to apply a piecewise affine transformation, and derive a face representation from a nine-layer deep neural network.

10. K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," in *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499-1503, Oct. 2016. Face detection and alignment in unconstrained environment are challenging due to various poses, illuminations, and occlusions. In this paper, a deep cascaded multitask framework that exploits the inherent correlation between detection and alignment to boost up their performance. The framework leverages a cascaded architecture with three stages of carefully designed deep convolutional networks to predict face and landmark location in a coarse-to-fine manner.

11. A. Kumar, A. Alavi and R. Chellappa, "KEPLER: Keypoint and Pose Estimation of Unconstrained Faces by Learning Efficient H-CNN Regressors," *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, Washington, DC, 2017, pp. 258-265. Keypoint detection is one of the most important pre-processing steps in tasks such as face modeling, recognition and verification. In this paper, we present an iterative method for Keypoint Estimation and Pose prediction of unconstrained faces by Learning Efficient H-CNN Regressors (KEPLER) for addressing the face alignment problem. The paper present a novel architecture called H-CNN (Heatmap-CNN) which captures structured global and local features and thus favors accurate keypoint detection. H-CNN is jointly trained on the visibility, fiducials and 3D-pose of the face.

12. J. Huang and W. Siu, "Learning Hierarchical Decision Trees for Single-Image Super-Resolution," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 27, no. 5, pp. 937-950, May 2017. Sparse representation has been extensively studied for image super-resolution (SR), and it achieved great improvement. Deep-learning-based SR methods have also emerged in the literature to pursue better SR results. The paper, use a set of decision tree strategies for fast and high-quality image SR. Our proposed SR using decision tree (SRDT) method takes the divide-and-conquer strategy, which performs a few simple binary tests to classify an input low-resolution (LR) patch into one of the leaf nodes and directly multiplies this LR patch with the regression model at that leaf node for regression. Both the classification process and the regression process take an extremely small amount of computation.

13. P. Sermanet, K. Kavukcuoglu, S. Chintala and Y. Lecun, "Pedestrian Detection with Unsupervised Multi-stage Feature Learning," *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, 2013, pp. 3626-3633. Pedestrian detection is a problem of considerable practical interest. Adding to the list of successful applications of deep learning methods to vision, we report state-of-the-art and competitive results on all major pedestrian datasets with a convolutional network model. The model uses a few new twists, such as multi-stage features, connections that skip layers to integrate global shape information with local distinctive motif information, and an unsupervised method based on convolutional sparse coding to pre-train the filters at each stage

3 Chapter 3 – Hardware, Software, Language and Packages Used

3.1 Hardware Used

3.1.1 System Specifications

The full specifications of the system used in our work is given in Table 3.1

Table 3.1 Table containing the detailed specification of Workstation

Workstation Model	Dell Inspiron 7530, HP Z620 Workstation
CPU	Intel® Core™ i5-3210M CPU @2.50GHz
RAM	4 GB DDR3, Speed 1600MHz
Hard Drive	1 TB
Operating System	Windows 10 Pro, Microsoft Corporation 64-bit operating system, x64 based processor
Chipset	Intel(R) 7 series chipset family

3.1.2 CPU Specification

The system has the Intel® Core™ i5-3210M processor. The detailed specifications of the CPU is shown in the Table 3.2

Table 3.2 Table containing detailed specifications of CPU

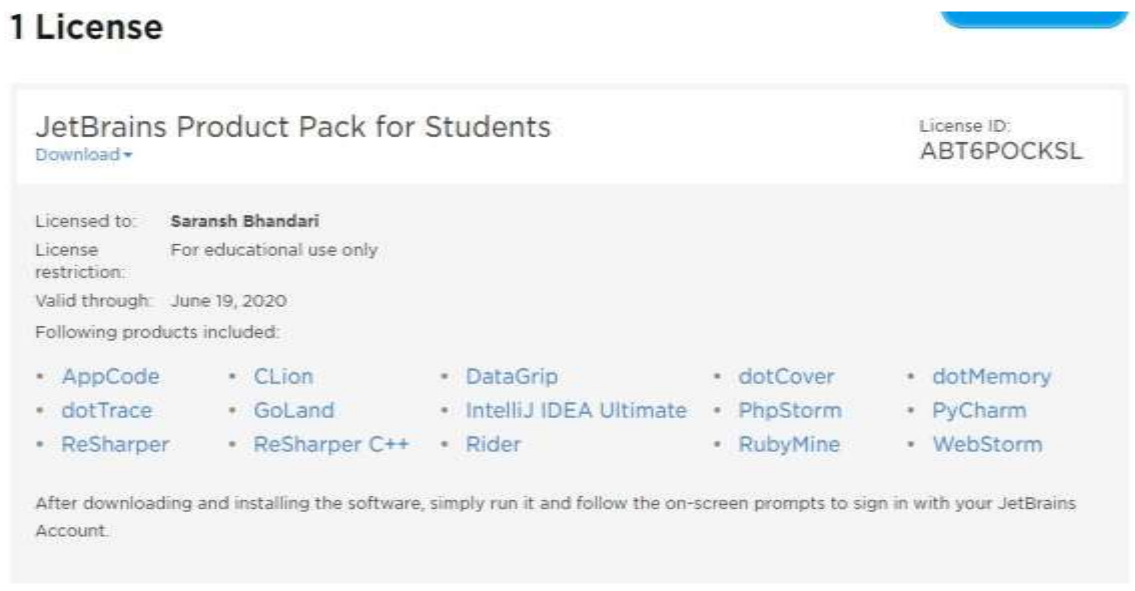
Base speed	2.50 GHz
Sockets	1
Cores:	2
Logical Processors	4
Virtualization	Enabled
L1 cache	128 KB
L2 cache	512 KB
L3 cache	3.0 MB

3.2 Software Used

The process of finding the right software was through trial and error. The first attempt made was on Anaconda. The language of choice was Python. I attempted to use this software because it is easy to integrate the coding as well presentation of data on this platform. The platform can connect to R Studio, which is great tool for numerical method as well as data analysis. Computer vision is also a part of data analysis. We are able to connect this platform to Google Colab. The Jupyter Notebooks developed in Anaconda are portable. We can use them in Google Colab.

The drawback, which I later found was that the Google Colab is new and does not support lot of libraries. It was started in 2017 itself.

The software finally decided upon is PyCharm, as shown in Figure 3.1. It is brainchild of JetBrain.



1 License

JetBrains Product Pack for Students
Download

License ID:
ABT6POCKSL

Licensed to: Saransh Bhandari
License restriction: For educational use only
Valid through: June 19, 2020
Following products included:

- AppCode
- CLion
- DataGrip
- dotCover
- dotMemory
- dotTrace
- GoLand
- IntelliJ IDEA Ultimate
- PhpStorm
- PyCharm
- ReSharper
- ReSharper C++
- Rider
- RubyMine
- WebStorm

After downloading and installing the software, simply run it and follow the on-screen prompts to sign in with your JetBrains Account.

Figure 3.1 This is the license for Pycharm Software Platform.
The license is free for students. The license is also valid for other platforms by JetBrain.

3.3 Libraries Used

Libraries: OpenCV, DLIB, Numpy, Math, Scikit Learn

3.3.1 OpenCV

Open Computer Vision, abbreviated as OpenCV, developed by Intel. This is open source library now. The library is structured and coded for real time performance. The structure of the library makes it easy & optimal to use memory of the system as well as the processing cost and speed. It is highly recommended for computer vision problems as there are people updating the library frequently.

Functions in OpenCV:

Cv2.line draws a line on image between two pixel coordinates.

Cv2.imread used to read input data

Cv2.imwrite used t write the output file on the OS.

Using OpenCV we can use rotation, translation, resize with respect to reference coordinates. It is not used for ploting graph. This can be done using matplotlib.

3.3.2 DLib

Dlib library is open source library. This library is a toolkit for solving real world problems. We have extensively used this library in this project work. This library is coded in C++. The library is used for machine learning algorithms, computer vision problems and algorithms are run on this framework.

3.3.3 Numpy

Numerical Python, abbreviated as NumPy, developed in 2000, is a library of python for scientific computation. It is commonly used for multi-dimensional array processing. Along with it poses some other features such as: i) Broadcasting (sophisticated) functions ii) It can be used in Fourier transform, linear algebra etc. iii) It includes tools which can integrate C/C++ and Fortan code.

3.3.4 Pandas

Panda (derived from “panel data”) is a python open source library released in 2008. It is employed for data analysis and manipulation. Specifically, it provides data structure and operations to perform complicated task efficiently with a small piece of code. It’s built over Numpy itself.

3.3.5 Math

It is open source library for most of the mathematics function required. We have used hypot function which is used to calculate the Euclidean Distance between two points.

3.3.6 Scikit-Learn

It is open source library built on top of Scipy, Numpy and matplotlib. It includes various classification, clustering, regression and other algorithms which are useful in data analysis and mining

4 Chapter 4 – Techniques and Methodology

4.1 Algorithms and techniques in image processing

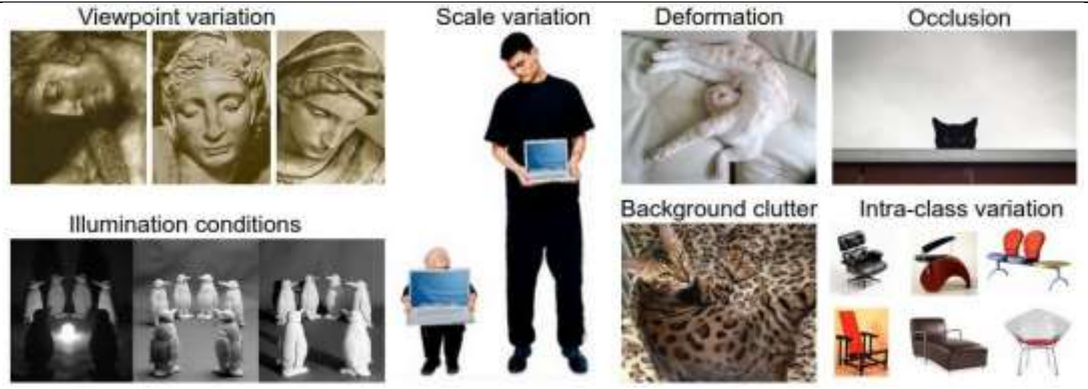


Figure 4.1 A classic image of challenges in Image Classification [7]

Figure 4.1, shows the core problems in the field of Computer Vision, Image Classification which is the task of assigning one label from the fixed set of labels to the given input image. This problem despite its simplicity is not so easy to solve and a vast variety of practical applications in various fields. Moreover, other problems like object detection and segmentation can be reduced to image classification, as shown in Table 4.1.

Table 4.1 Core problems in Vision

Viewpoint variation	An object of interest may be oriented in many different ways with respect to the viewer or the camera
Deformation	Many objects of interest are rigid bodies of a fixed shape and size and can be deformed in many different ways.
Occlusion	The object of interest can also be found occluded. Sometimes only a tiny portion of the object (as small as few pixels) could be visible
Illumination conditions	The effects of illumination are drastic which can many times hide most compelling features of the object
Background clutter	The object may blend into their environment making them hard to recognize.
Intra-class variation	The classes of interest can many times be relatively broad, such as a chair. The object belonging to that class can come in many different shapes, sizes, and colour.

4.1.1 Viola Jones Object Detection Algorithm

The Viola-Jones object detection method is one of the first methods to provide reasonably competitive object detection accuracy in real-time given by Paul Viola and Michael Jones [8]. The method was for face detection in real-time with reasonable accuracy. The features like robustness, speed, and requirement of small samples are the characteristics of this algorithm and the main reason for its acceptance. Intel developed Haar feature files for frontal face in xml. Intel also developed Haar files for eyes, nose, ears and other human head poses as shown in Figure 4.2 and Figure 4.3. The algorithm tries to match the image with the known Haar features of the image class. All the image of a particular class would share some characteristic properties. These regularities are matched using the relevant Haar features.

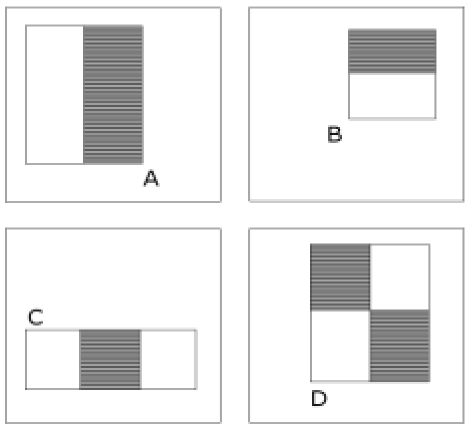


Figure 4.2 Haar cascades [9]

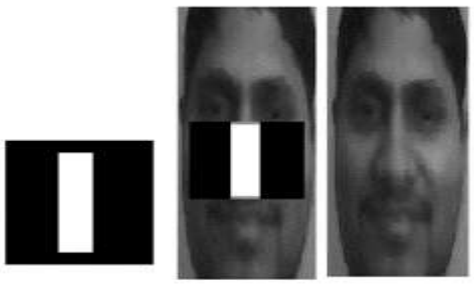


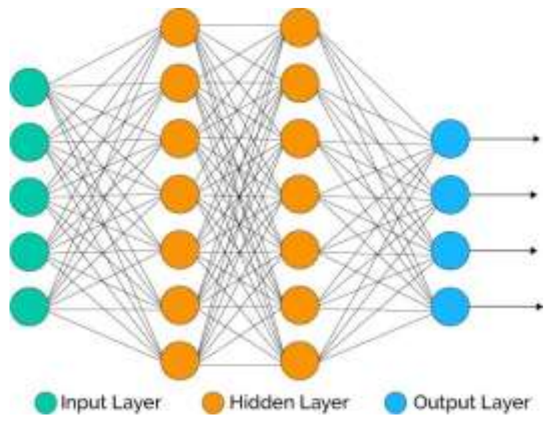
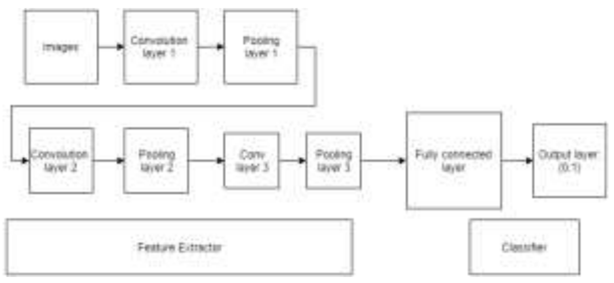
Figure 4.3 Haar cascade for face detection [9]

4.1.2 k-Nearest Neighbors (k-NN) Algorithm

This is an approach of unsupervised learning. In this algorithm clusters are created. This algorithm deals with classifying an object into one of the previously decided classes, or in another way it is the algorithm that assigns an unlabelled object, a label from a fixed set of labels. K number of nearest neighbors from the data point to be classified, are taken from the set of objects for which the classification is already known. The class which will have the maximum number of neighbors points will be assigned as the label to the object under classification. In kNN functions are approximated locally.

4.1.3 CNN

Convolutional Neural Network, abbreviated as CNN, is an modern approach which works like a neuron of human beings [12]. This is an exhaustive approach where eyery node of layer$_{n-1}$ is connected to all the node layer$_n$ as shown in Figure 4.4 and Figure 4.5 Simple and small patterns created in layers are merged to generated a final pattern in fully connected layer, which is the classifier with output [0,1]. The output can be set of classes.

 Figure 4.4 Artificial Neural network [10]	 Images Convolution layer 1 Pooling layer 1 Convolution layer 2 Pooling layer 2 Conv layer 3 Pooling layer 3 Fully connected layer Output layer (0,1) Feature Extractor Classifier Figure 4.5 Model of CNN[11] It has convolution layer, pooling layer for n-times. They are feature extractors. Classifier is fully connected layer with output [0,1]

The CNN trains on Images unlike SVM where it trains on features extracted from the region of interests. CNN requires very high amount of datasets as compared to SVM.

4.1.4 Face Embedding

Face embedding is another method introduced in FaceNet[13]. It is a Deep NN architecture based approach to use output vector at each stage of NN layer as feature label. The model generates a correlation between different output vectors of the NN layers. The approach of the model uses Euclidean distance for correlation. The faces with small distances are considered similar while with large distance are considered non-similar. The loss function reduces distance between output and positive where they have the same identity. It maximises the distance between output and negative where they do not have the same identity. On fully connected layer output , the loss function is trained to generate 1x1x128 vector output. Figure 4.6 shows the flow diagram for the face enconding,

The success rate is 98.87 % on LFW[21]. The result computed are on fixed center crop of the faces, which means the faces are aligned to the center of the image.

We created a new dataset for face recognition which has 7 different faces. We have trained the model on it. We have input 7 front face images as training images as (key, value) pairs. The key is name of the face while the value is 128 byte vector which is the embedding generated for the face.

The Table 4.2 shows the dataset images on which the face encoding is trained. The face names are [Abhaya Shahare, Deepeankar Sharma, Garima Sharma, Nirdosh Kumar, Saransh Bhandari, Rajveer Mali]. We are able to recognise all the image. We can classify the input images into known classes, face names, as well as unknown class if the image does not match the trained encodings. Figure 4.7 shows the training result

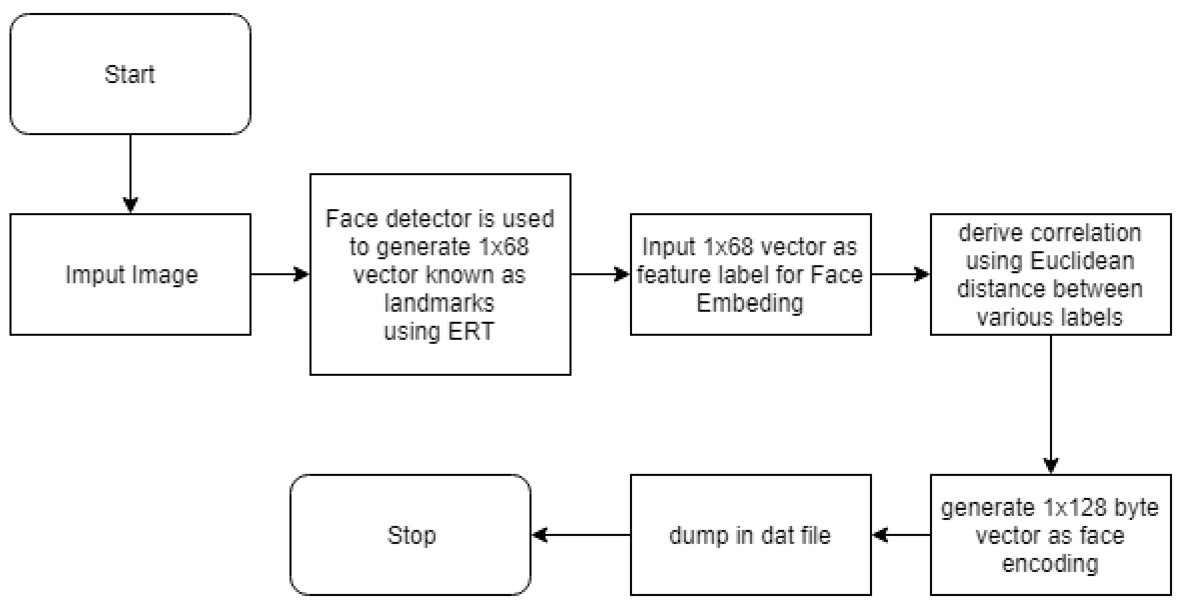
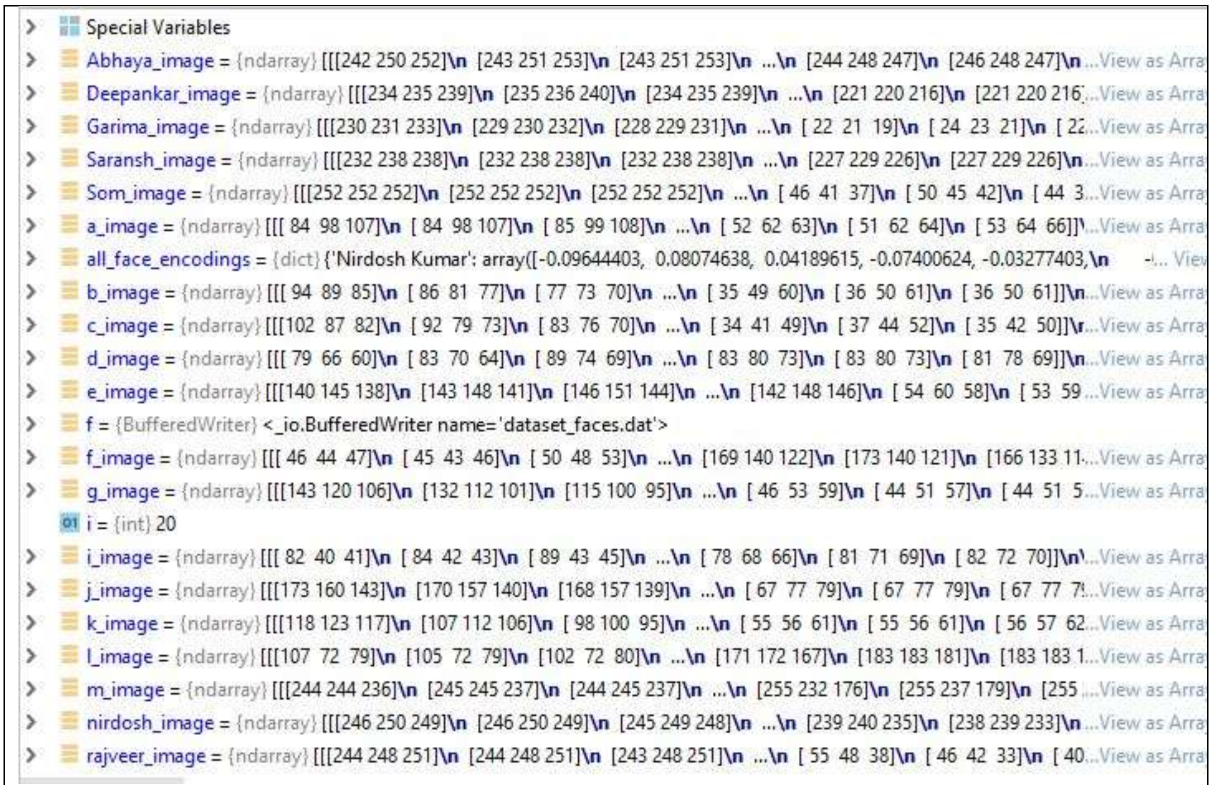

Start
Imput Image
Face detector is used to generate 1x68 vector known as landmarks using ERT
Input 1x68 vector as feature label for Face Embeding
derive correlation using Euclidean distance between various labels
generate 1x128 byte vector as face encoding
dump in dat file
Stop

Figure 4.6 Flow Diagram for face encoding



```
> Special Variables
> Abhaya_image = {ndarray} [[[242 250 252]\n  [243 251 253]\n  [243 251 253]\n  ...\n  [244 248 247]\n  [246 248 247]\n...View as Arra
> Deepankar_image = {ndarray} [[[234 235 239]\n  [235 236 240]\n  [234 235 239]\n  ...\n  [221 220 216]\n  [221 220 216]...View as Arra
> Garima_image = {ndarray} [[[230 231 233]\n  [229 230 232]\n  [228 229 231]\n  ...\n  [ 22  21  19]\n  [ 24  23  21]\n  [ 22...View as Arra
> Saransh_image = {ndarray} [[[232 238 238]\n  [232 238 238]\n  [232 238 238]\n  ...\n  [227 229 226]\n  [227 229 226]\n...View as Arra
> Som_image = {ndarray} [[[252 252 252]\n  [252 252 252]\n  [252 252 252]\n  ...\n  [ 46  41  37]\n  [ 50  45  42]\n  [ 44  3...View as Arra
> a_image = {ndarray} [[[ 84  98 107]\n  [ 84  98 107]\n  [ 85  99 108]\n  ...\n  [ 52  62  63]\n  [ 51  62  64]\n  [ 53  64  66]]\...View as Arra
> all_face_encodings = {dict} {'Nirdosh Kumar': array([-0.09644403,  0.08074638,  0.04189615, -0.07400624, -0.03277403,\n      -... View
> b_image = {ndarray} [[[ 94  89  85]\n  [ 86  81  77]\n  [ 77  73  70]\n  ...\n  [ 35  49  60]\n  [ 36  50  61]\n  [ 36  50  61]]\n...View as Arra
> c_image = {ndarray} [[[102  87  82]\n  [ 92  79  73]\n  [ 83  76  70]\n  ...\n  [ 34  41  49]\n  [ 37  44  52]\n  [ 35  42  50]]\r...View as Arra
> d_image = {ndarray} [[[ 79  66  60]\n  [ 83  70  64]\n  [ 89  74  69]\n  ...\n  [ 83  80  73]\n  [ 83  80  73]\n  [ 81  78  69]]\n...View as Arra
> e_image = {ndarray} [[[140 145 138]\n  [143 148 141]\n  [146 151 144]\n  ...\n  [142 148 146]\n  [ 54  60  58]\n  [ 53  59...View as Arra
> f = {BufferedWriter} <_io.BufferedWriter name='dataset_faces.dat'>
> f_image = {ndarray} [[[ 46  44  47]\n  [ 45  43  46]\n  [ 50  48  53]\n  ...\n  [169 140 122]\n  [173 140 121]\n  [166 133 11...View as Arra
> g_image = {ndarray} [[[143 120 106]\n  [132 112 101]\n  [115 100  95]\n  ...\n  [ 46  53  59]\n  [ 44  51  57]\n  [ 44  51  5...View as Arra
  i = {int} 20
> i_image = {ndarray} [[[ 82  40  41]\n  [ 84  42  43]\n  [ 89  43  45]\n  ...\n  [ 78  68  66]\n  [ 81  71  69]\n  [ 82  72  70]]\n\...View as Arra
> j_image = {ndarray} [[[173 160 143]\n  [170 157 140]\n  [168 157 139]\n  ...\n  [ 67  77  79]\n  [ 67  77  79]\n  [ 67  77  7...View as Arra
> k_image = {ndarray} [[[118 123 117]\n  [107 112 106]\n  [ 98 100  95]\n  ...\n  [ 55  56  61]\n  [ 55  56  61]\n  [ 56  57  62...View as Arra
> l_image = {ndarray} [[[107  72  79]\n  [105  72  79]\n  [102  72  80]\n  ...\n  [171 172 167]\n  [183 183 181]\n  [183 183 1...View as Arra
> m_image = {ndarray} [[[244 244 236]\n  [245 245 237]\n  [244 245 237]\n  ...\n  [255 232 176]\n  [255 237 179]\n  [255 ...View as Arra
> nirdosh_image = {ndarray} [[[246 250 249]\n  [246 250 249]\n  [245 249 248]\n  ...\n  [239 240 235]\n  [238 239 233]\n...View as Arra
> rajveer_image = {ndarray} [[[244 248 251]\n  [244 248 251]\n  [243 248 251]\n  ...\n  [ 55  48  38]\n  [ 46  42  33]\n  [ 40...View as Arra
```

Figure 4.7 Training resultfor face encoding

Table 4.2 Face Recognition Dataset

4.2 Ensemble Regression Tree used as ML Algorithm

When you need to buy another vehicle, will you approach the principal vehicle shop and buy at the advice to the seller? It is very impossible. You would likely see a couple of online websites, where individuals have posted their surveys and look at changed vehicle models, checking at their highlights and costs. You will likewise most likely approach your companions and associates for their assessment. To put it plainly you will not directly make a choice, however will rather settle on a choice thinking about the sentiments of other individuals also.

Group models in AI work on a comparative thought. They join the choices from different models to improve the general execution. This can be accomplished in different ways, which is explained in this chapter.

4.2.1 Simple Ensemble Techniques

4.2.1.1 Max Voting

The max vote is used for creating sets. In this strategy, various models are utilized to make expectations for every data point. The expectations by each model are considered as a 'vote'. The forecasts which we get from most of the models are utilized as the last expectation [17].

Example: If five people vote [3, 4, 5, 4, 4]. Then the final result used by the model will be 4.

Example2: suppose we run KNN, Logistic Regression, Decision tree Classifier then the final output used by the model will be by voting.

4.2.1.2 Averaging

Like the max voting method, numerous forecasts are made for every data point in averaging. In this technique, we take a normal of forecasts from every one of the models and use it to make the last expectation. Averaging can be utilized for making forecasts in regression issues or while computing probabilities for classification [18].

4.2.1.3 Weighted Averaging

This is an expansion of the averaging technique [18]. All models are allocated various weightage, characterizing the significance of each model for expectation. For example, if two of your associates are experts, while others have no related knowledge in this field, at that point the appropriate responses by these two companions are given more significance when contrasted with different individuals.

4.2.2 Advanced Ensemble Techniques

4.2.2.1 Stacking

Stacking is an ET that utilize forecasts from various models (for instance Decision tree, KNN or SVM) to assemble another model. This model is utilized for making forecasts on the test set [19].

Example predictions of model1, model2, model3 are used as feature vector for a complete new model generation which is the final model.

4.2.2.2 Blending

Blending pursues a similar methodology as stacking yet utilizes just a validation set from the train set to make expectations. As it were, not at all like stacking, the forecasts are made on the validation set as it were. The validation set and the predictions are utilized to manufacture a model which is kept running on the test set [20].

4.2.3 Bagging

The thought behind Bagging is joining the consequences of different models (for example, all choice trees) to get a summed up result. Here's an inquiry: If you make every one of the models on a similar arrangement of information and join it, will it be helpful? There is a high possibility that these models will give a similar outcome since they are getting a similar info. So how might we take care of this issue? One of the systems is bootstrapping [17] [20].

Bootstrapping, as shown in Figure 4.6, is an examining method where we make subsets of perceptions from the first dataset, with substitution. The size of the subsets is equivalent to the size of the first set.

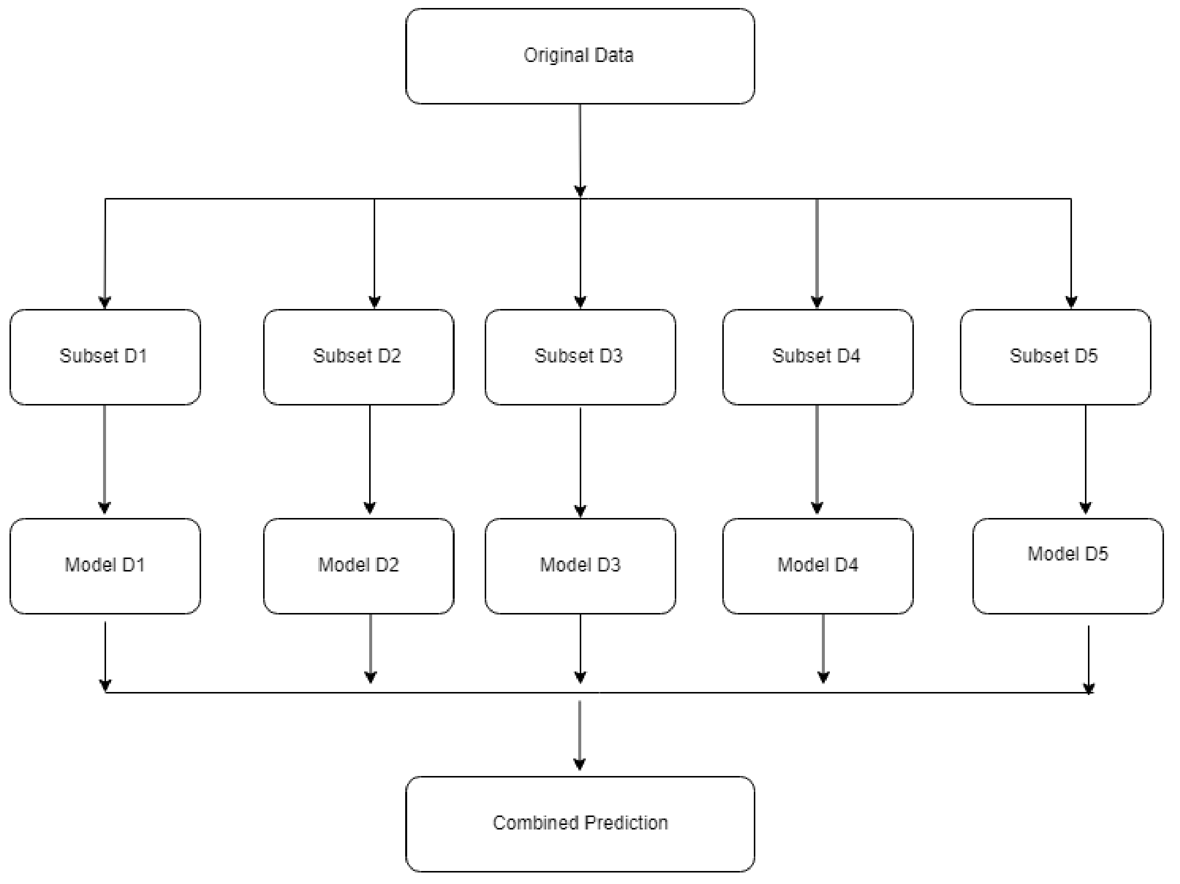


Figure 4.8 Bootstraping Flowchart

4.2.4 Boosting

In the event that an information point is mistakenly predicted by the principal model, and after that the following (most likely all models), will consolidating the predictions give better outcomes? Such circumstances are dealt with by boosting.

Boosting is a successive procedure, where each resulting model endeavors to address the mistakes of the past model, as shown in Table 4.3. The succeeding models are subject to the past model.

Table 4.3 Boosting Steps

1	A subset is made from the first dataset.
2	At first, all information focuses are given equivalent loads.
3	A base model is made on this subset.
4	This model is utilized to make forecasts in general dataset.
5	Errors are determined utilizing the real qualities and anticipated qualities.
6	The predictions which are inaccurately anticipated, are given higher weightage.
7	Another model is made and expectations are made on the dataset.
8	Likewise, different models are made, each adjusting the mistakes of the past model.

9	The last model (strong learner) is the weighted mean of the considerable number of models (weak learner).

4.2.5 Bagging Meta-estimator

The bagging meta-estimator process includes base_estimator, n_estimator, max samples, max features, n jobs, random state which is explained in Table 4.4

Table 4.4 Parameters utilized in the calculation of Bagging Meta-estimator

base_estimator: It characterizes the base estimator to fit on arbitrary subsets of the dataset. When nothing is determined, the base estimator is a decision tree.[17][20]
n_estimators: It is the quantity of base estimators to be made. The quantity of estimators ought to be painstakingly tuned as a huge number would set aside an exceptionally long effort to run, while a modest number probably won't give the best outcomes.
max_samples: This parameter controls the size of the subsets. It is the most extreme number of tests to prepare each base estimator
max_features: Controls the quantity of features to draw from the entire dataset. It characterizes the greatest number of features needed to train each base_estimator.
n_jobs: The quantity of thread to process parallel. The default, -1 , is set to systems quantity of core.
random_state: This parameter explains the process of random split. It indicates the technique for irregular split. At the point when arbitrary split value is same for two models, the irregular determination is same for the two models. This parameter is valuable when you need to compare about various models

4.2.6 Random Forest

It is another ML technique that pursues the bagging technique. It is derived of the Bagging Meta-estimator, as shown in Table 4.5. The base estimators are decision trees. Not at all like bagging meta estimator, random forest arbitrarily chooses a lot of features which are utilized to choose the best split at every node of the decison tree [17][20].

Table 4.5 Steps in a Random Forest model

Arbitrary subsets are made from the first dataset (bootstrapping).
At every node in the tree, only an arbitrary arrangements of features is taken into consideration to decide the best split.
A decision tree model is fitted on every one of the subsets.(weak learners)
The final prediction is calculated by averaging the forecasts from all weak learner trees.(Strong Learner)

4.3 ERT Model

4.3.1 Flowchart

Figure 4.7 shows the flowchart of the shape predictor model using supervised machine learning. In supervised machine learning algorithms, a machine is trained using labeled data, such as an input where the desired output is known and based on this new data is classified. On processing the training data, the algorithm generates a mapping function which predicts the output for the new data after adequate training. In our work, the data is labeled as either face=0 or face=1. There are different methods such as regression, classifcation, gradient boosting and others which are used for predicting the output in supervised machine learning. We have used Ensemble Regression Tree, abbreviated as ERT.

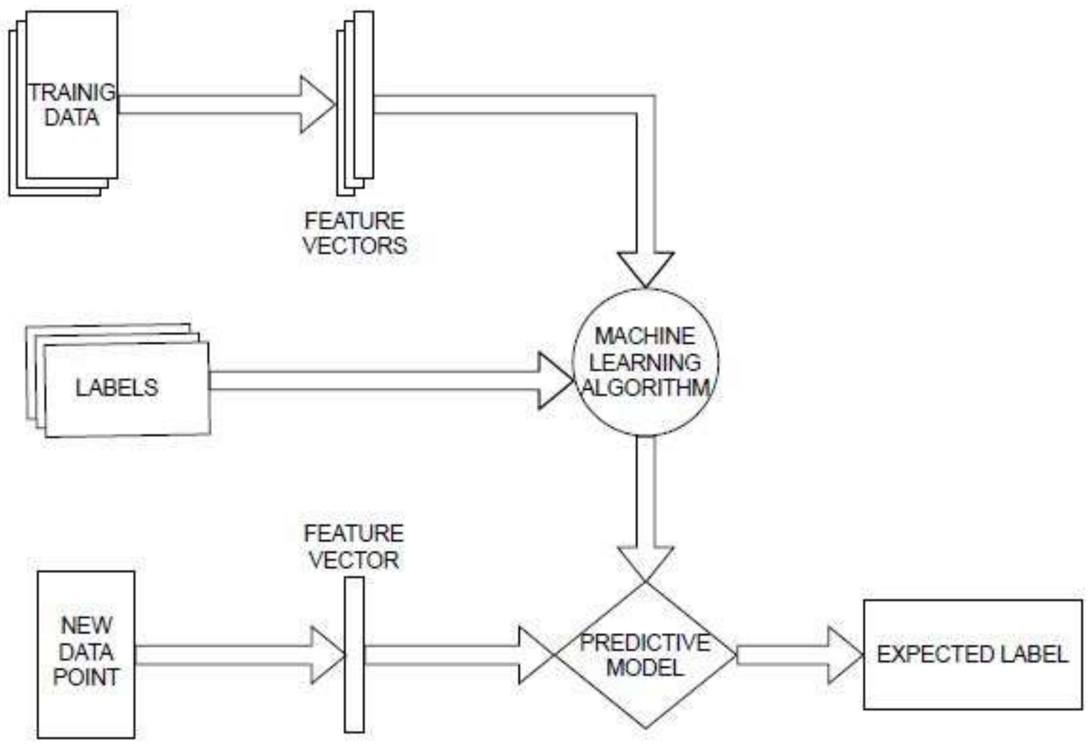


Figure 4.9 Flowchart of landmark predictor with bounding box using ML algorithm

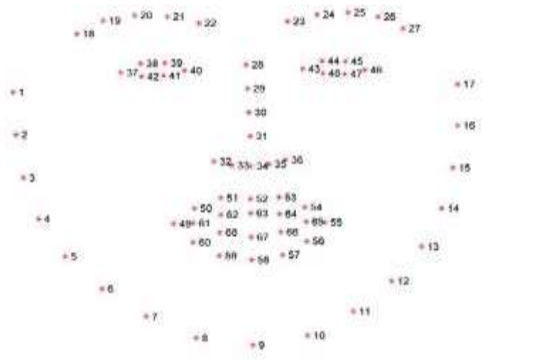
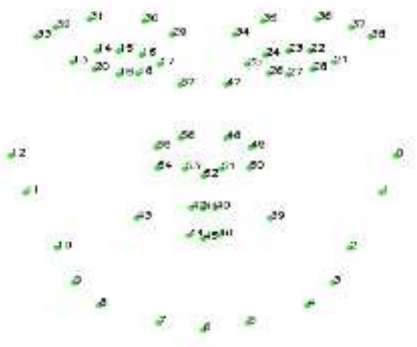
 Figure 4.10 2D mask for structure landmarks 2D facial landmark[5][6]	 Figure 4.11 3D mask for the structure landmarks 3D facial Landmark for reference. [5][6]

Figure 4.8 and Figure 4.9 are masks used in landmark detection. The mask here is 68 points divided into x parts: 0 to 17 is jaw line, 18 to 22 right eyebrow, 23 to 27 is left eyebrow, 28 to 31 is nose line, 32 to 36 is nostril structure, 37 to 42 is right eye, 43 to 48 is left eye. Each eye includes six data points, each data point is a pixel which is a matrix with RGB values. The eye will have iris, pupil and sclera (white part). Now, from 48 to 54 form outer upper lips, 55

to 59 is outer lower lips and finally 60 to 64 is inner part of upper lips and 65 to 67 is inner part of lower lips. This mask is used to train the model to detect face.

The face detector identifies the attributes as combination of number from this mask. The combination can result in eyes, eyebrows, jawline lips nose.

4.3.2 Human Head Pose Detection

Figure 4.10, explains the general outline of the Pseudo Code

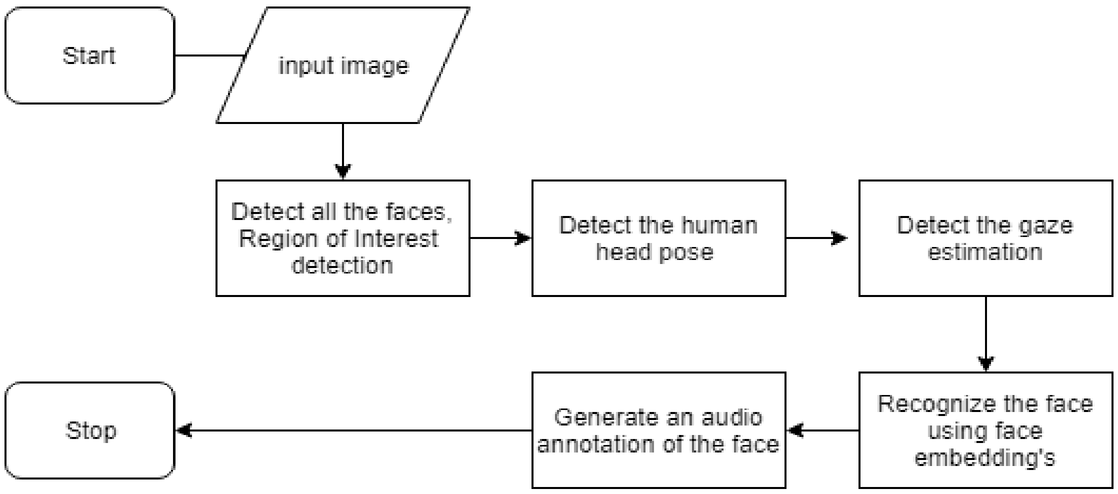

Start
input image
Detect all the faces, Region of Interest detection
Detect the human head pose
Detect the gaze estimation
Recognize the face using face embedding's
Generate an audio annotation of the face
Stop

Figure 4.12 Flow Diagram

The head pose estimation is done as follows

1. Load the trained model, say Model M
2. Run the detector on image with mask, as shown Figure 3.8 as reference
3. Get the model points corresponding to the mask
4. Generate the bounding box using the model points from top left to bottom right corner.
5. This gives the bounding box.
6. Now use the Figure 3.9, 3D mask as reference coordinates and generate the rotation and translation angle with respect to 6 coordinates defined in Figure 3.8, 3.9.
7. Use rotation and translation vector to create x , y and z axis with nose as center of the image.

4.3.3 ERT Model Results

Figure 4.11, the image has six circle known as landmarks of the structure. We calculate the horizontal length between the pixels as shown in the image. We also calculate the vertical length of the midpoint of the upper pixels as well as lower pixel. We use the structure to calculate the blinking as well as direction of the human eye.

Figure 4.12 to Figure 4.20 explains the result of face detection. The Detection percentage varied in images from 46% to 72 %. The reason for it are Occulsion, Illumination conditions and background deformation. There were some region of interest which should have been found, hence Detection % is between 46% to 73%. If the face is near the center of the image then it is easy to detect it. The faces away from center of image have high probability of not being detected.

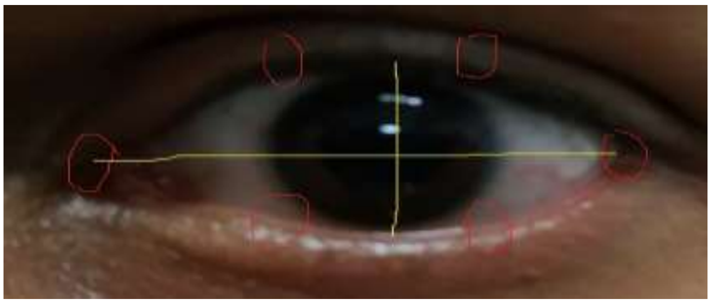
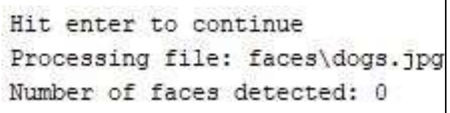


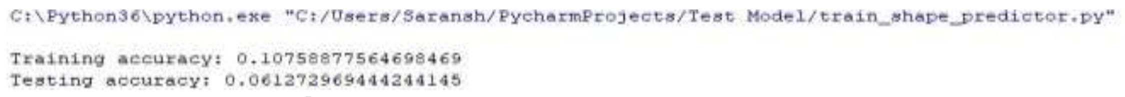
Figure 4.13 This is proposed logic image to detect the gaze of the human eye.



```
Hit enter to continue
Processing file: faces\dogs.jpg
Number of faces detected: 0
```

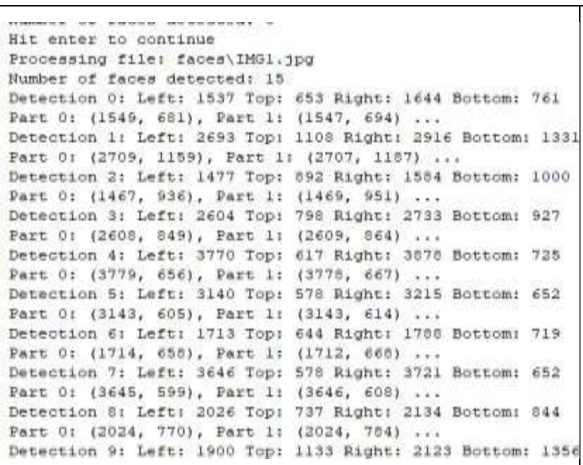
Figure 4.14 Face detection in dogs
Total number of dog detected : 0
Total number of dog : 4

Figure 4.15 output of face detection of dog



```
C:\Python36\python.exe "C:/Users/Saransh/PycharmProjects/Test Model/train_shape_predictor.py"

Training accuracy: 0.10758877564698469
Testing accuracy: 0.061272969444244145
```

Figure 4.16 Training and Testing accuracy



```
Hit enter to continue
Processing file: faces\IMG1.jpg
Number of faces detected: 15
Detection 0: Left: 1537 Top: 653 Right: 1644 Bottom: 761
Part 0: (1549, 681), Part 1: (1547, 694) ...
Detection 1: Left: 2693 Top: 1108 Right: 2916 Bottom: 1331
Part 0: (2709, 1159), Part 1: (2707, 1187) ...
Detection 2: Left: 1477 Top: 892 Right: 1584 Bottom: 1000
Part 0: (1467, 936), Part 1: (1469, 951) ...
Detection 3: Left: 2604 Top: 798 Right: 2733 Bottom: 927
Part 0: (2608, 849), Part 1: (2609, 864) ...
Detection 4: Left: 3770 Top: 617 Right: 3878 Bottom: 725
Part 0: (3779, 656), Part 1: (3778, 667) ...
Detection 5: Left: 3140 Top: 578 Right: 3215 Bottom: 652
Part 0: (3143, 605), Part 1: (3143, 614) ...
Detection 6: Left: 1713 Top: 644 Right: 1788 Bottom: 719
Part 0: (1714, 658), Part 1: (1712, 668) ...
Detection 7: Left: 3646 Top: 578 Right: 3721 Bottom: 652
Part 0: (3645, 599), Part 1: (3646, 608) ...
Detection 8: Left: 2026 Top: 737 Right: 2134 Bottom: 844
Part 0: (2024, 770), Part 1: (2024, 784) ...
Detection 9: Left: 1900 Top: 1133 Right: 2123 Bottom: 1356
```

Figure 4.17 Total face detected
Total number of face detected: 15.
Total number of faces are :32

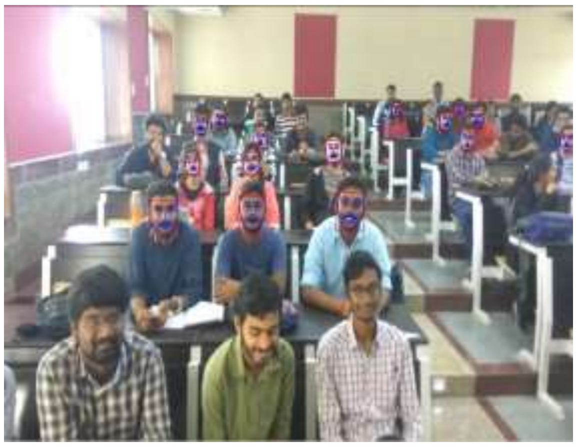
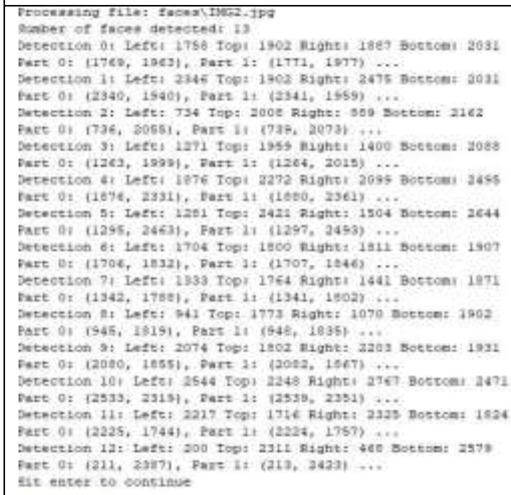


Figure 4.18 Output of face detection
Detection % = 46.87 %



```
Processing file: faces\IMG2.jpg
Number of faces detected: 13
Detection 0: Left: 1758 Top: 1902 Right: 1887 Bottom: 2031
Part 0: (1769, 1963), Part 1: (1771, 1977) ...
Detection 1: Left: 2346 Top: 1902 Right: 2475 Bottom: 2031
Part 0: (2340, 1940), Part 1: (2341, 1959) ...
Detection 2: Left: 734 Top: 2008 Right: 889 Bottom: 2162
Part 0: (736, 2058), Part 1: (739, 2073) ...
Detection 3: Left: 1271 Top: 1959 Right: 1400 Bottom: 2088
Part 0: (1263, 1999), Part 1: (1264, 2015) ...
Detection 4: Left: 1876 Top: 2272 Right: 2099 Bottom: 2495
Part 0: (1876, 2331), Part 1: (1880, 2361) ...
Detection 5: Left: 1281 Top: 2421 Right: 1504 Bottom: 2644
Part 0: (1295, 2463), Part 1: (1297, 2493) ...
Detection 6: Left: 1704 Top: 1800 Right: 1811 Bottom: 1907
Part 0: (1706, 1832), Part 1: (1707, 1846) ...
Detection 7: Left: 1333 Top: 1764 Right: 1441 Bottom: 1871
Part 0: (1342, 1788), Part 1: (1341, 1802) ...
Detection 8: Left: 941 Top: 1773 Right: 1070 Bottom: 1902
Part 0: (946, 1819), Part 1: (948, 1835) ...
Detection 9: Left: 2074 Top: 1802 Right: 2203 Bottom: 1931
Part 0: (2080, 1855), Part 1: (2082, 1867) ...
Detection 10: Left: 2544 Top: 2248 Right: 2767 Bottom: 2471
Part 0: (2533, 2319), Part 1: (2539, 2351) ...
Detection 11: Left: 2217 Top: 1716 Right: 2325 Bottom: 1824
Part 0: (2225, 1744), Part 1: (2224, 1757) ...
Detection 12: Left: 200 Top: 2311 Right: 468 Bottom: 2579
Part 0: (211, 2387), Part 1: (213, 2423) ...
Hit enter to continue
```

Figure 4.19 Total face detected
Total number of face detected: 13.
Total number of faces are :18

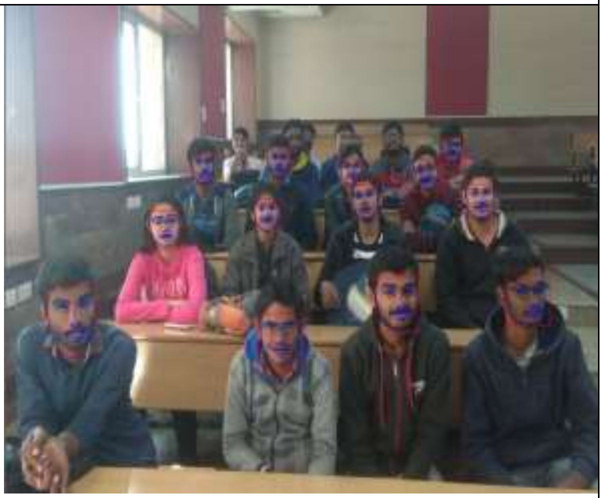
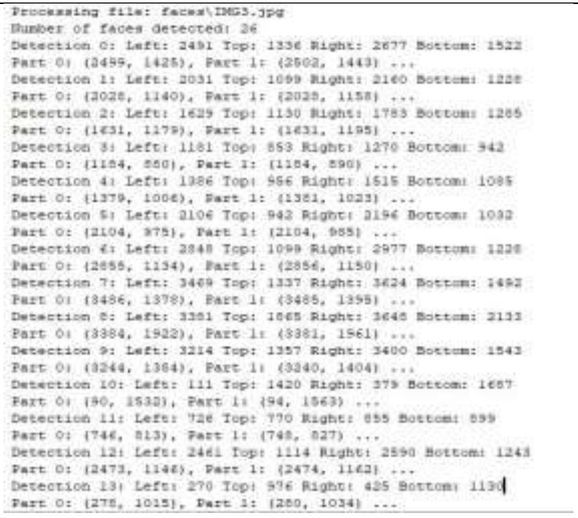


Figure 4.20 Output of face detection
Detection % = 72.22%



```
Processing file: faces\IMG3.jpg
Number of faces detected: 26
Detection 0: Left: 2491 Top: 1336 Right: 2677 Bottom: 1522
Part 0: (2499, 1425), Part 1: (2502, 1443) ...
Detection 1: Left: 2031 Top: 1099 Right: 2160 Bottom: 1228
Part 0: (2028, 1140), Part 1: (2028, 1158) ...
Detection 2: Left: 1629 Top: 1130 Right: 1783 Bottom: 1285
Part 0: (1631, 1179), Part 1: (1631, 1195) ...
Detection 3: Left: 1181 Top: 853 Right: 1270 Bottom: 942
Part 0: (1184, 880), Part 1: (1184, 890) ...
Detection 4: Left: 1386 Top: 956 Right: 1515 Bottom: 1085
Part 0: (1379, 1006), Part 1: (1381, 1023) ...
Detection 5: Left: 2106 Top: 942 Right: 2196 Bottom: 1032
Part 0: (2104, 975), Part 1: (2104, 985) ...
Detection 6: Left: 2848 Top: 1099 Right: 2977 Bottom: 1228
Part 0: (2855, 1134), Part 1: (2856, 1150) ...
Detection 7: Left: 3469 Top: 1337 Right: 3624 Bottom: 1492
Part 0: (3486, 1378), Part 1: (3485, 1395) ...
Detection 8: Left: 3381 Top: 1865 Right: 3648 Bottom: 2133
Part 0: (3384, 1922), Part 1: (3381, 1961) ...
Detection 9: Left: 3214 Top: 1357 Right: 3400 Bottom: 1543
Part 0: (3244, 1384), Part 1: (3240, 1404) ...
Detection 10: Left: 111 Top: 1420 Right: 379 Bottom: 1687
Part 0: (90, 1532), Part 1: (94, 1563) ...
Detection 11: Left: 726 Top: 770 Right: 855 Bottom: 899
Part 0: (746, 813), Part 1: (748, 827) ...
Detection 12: Left: 2461 Top: 1114 Right: 2590 Bottom: 1243
Part 0: (2473, 1146), Part 1: (2474, 1162) ...
Detection 13: Left: 270 Top: 976 Right: 425 Bottom: 1130
Part 0: (278, 1015), Part 1: (280, 1034) ...
```

Figure 4.21 Total face detection along with the position of boundary box
Total number of face detected: 26
Total number of faces are :45

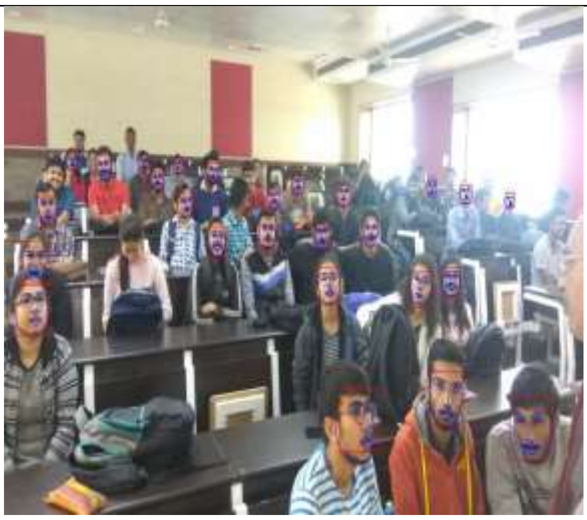


Figure 4.22 Output of face detection
Detection % = 58%

5 Chapter 5 – Proposed Work and Results

The process of detecting a face from an image has lots of similarity with detecting the head pose from an image. The process of feature extraction and feature mapping generated at different layers for detection and pose estimation can be used interchangeably. Thus merging the similar steps can reduce the time for face detection and pose estimation for the same image. The problem of simultaneous face detection and pose estimation is derived from the need for real time facial pattern analysis.

5.1 Dataset

We are using 300-W dataset downloaded from intelligent behaviour understanding group, abbreviated as ibug[1][2][3][4], as shown Table 5.1. The dataset consist of 300 faces. The faces are indoor as well as outdoor. The face captured have emotions, different alignment, illumination, identity. There is vast difference in appearance of the images. There are four parts in the dataset. All the images have mirror version of then to increase the dataset size.

Table 5.1 Table of Dataset used.

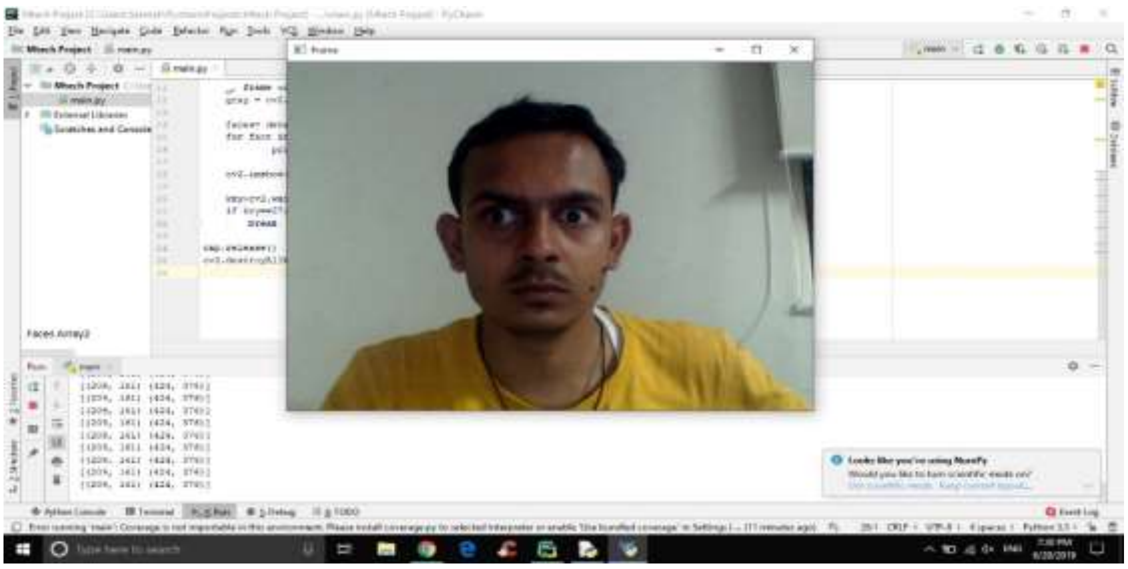
1	Labelled face parts in the wild	LFLW	There are total 1135 images.811 training images and 224 test images
2	Annotated facial landmark	AFW	There are 250 images. These images have 468 labelled human faces.
3	Helen	Helen	There are 2330 images. These are the most densely labelled images.
4	Intelligent behaviour understanding group	ibug	There are 135 images.

5.2 Steps and Results

The following are project steps along with their outputs at each stage.

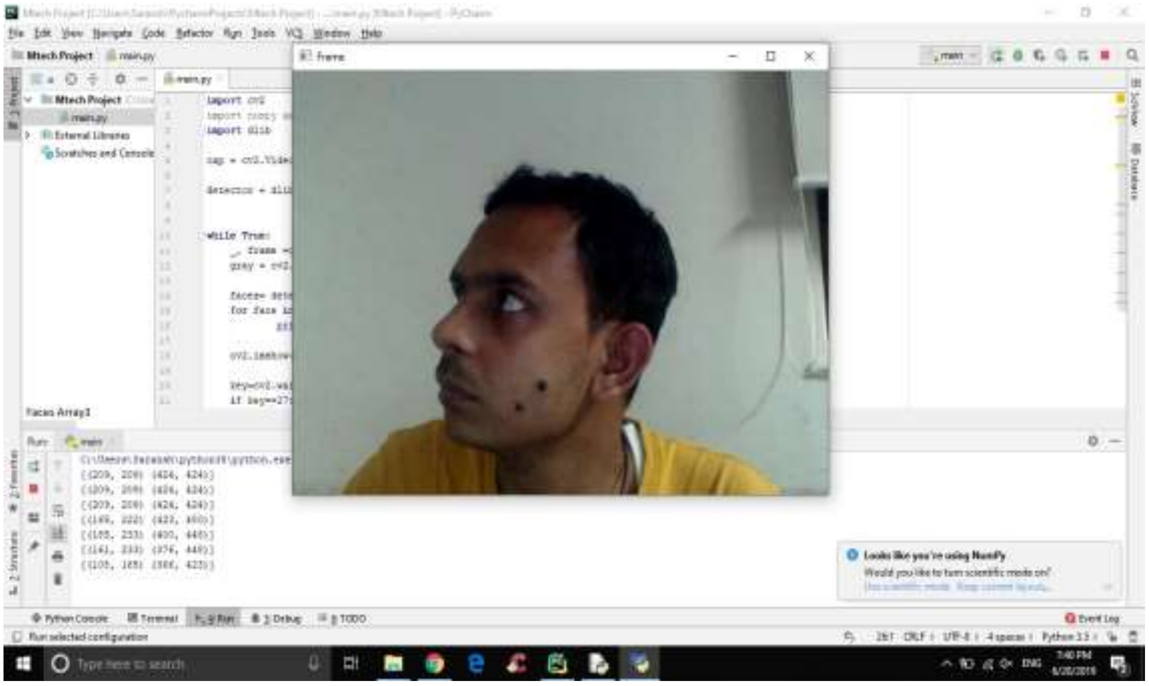
5.2.1 Step1: Getting Started

In Figure 5.1 to Figure 5.23, we printed the face coordinates in the image for left, right and center poses. We completed the landmark detection on the face and displayed the landmark object at each frame that is processed. The landmark 36 is displayed which is right eye corner. We calculated the aspect ratio of eye i.e. ratio of vertical length to horizontal length of the eye. We threshold (=70,42,125) the eye for black and white image. This output is used for gaze estimation by calculating the number of zeroes in the image. At the end of the process we expect to detect the face along with the head pose displayed as front of the face leading out of the bounding box surrounding the detected face.



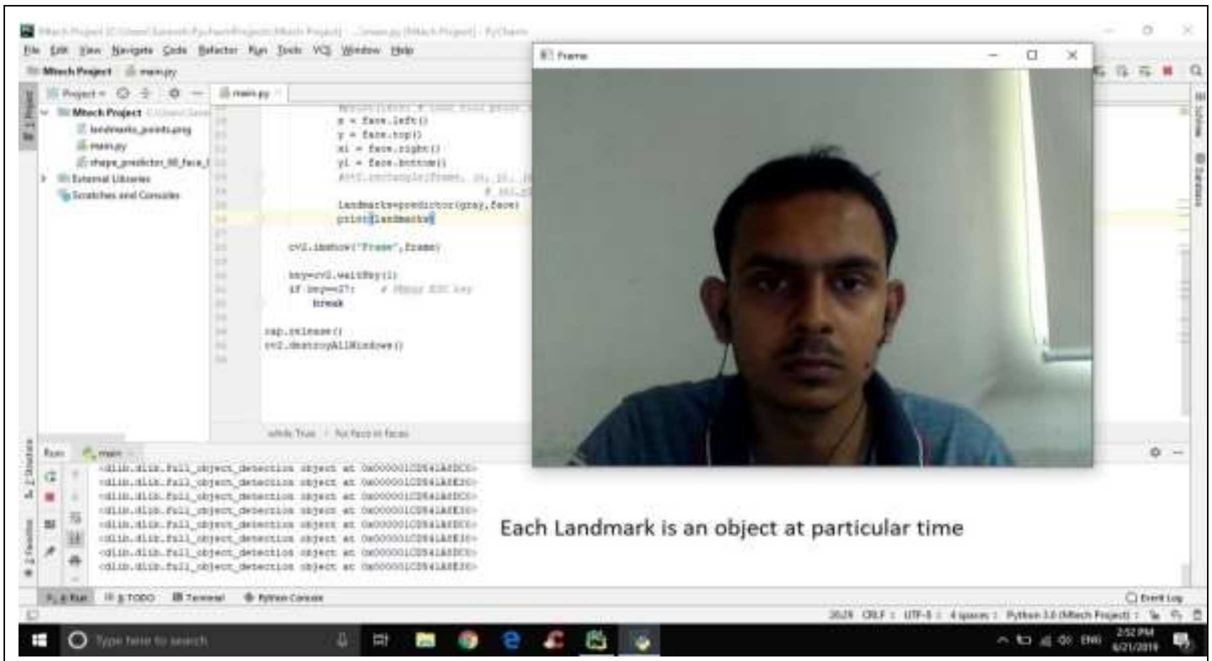
```
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
[(209, 141) (424, 376)]
```

Figure 5.1 This is the output of Region of Interest when looking center.
The output is an array 1x2 with top left corner (x, y) and bottom left corner (x, y)
The output is [(209,141), (424,376)]. The output is displayed at each frame processed.



```
[(209, 209) (424, 424)]
[(209, 209) (424, 424)]
[(209, 209) (424, 424)]
[(188, 222) (423, 460)]
[(185, 233) (400, 448)]
[(161, 233) (376, 448)]
[(108, 165) (366, 423)]
```

Figure 5.2 This is the output of Region of Interest when looking right.
The output is an array 1x2 with top left corner (x, y) and bottom left corner (x, y)
The output is [(108,165), (366,423)]. The output is displayed at each frame processed.



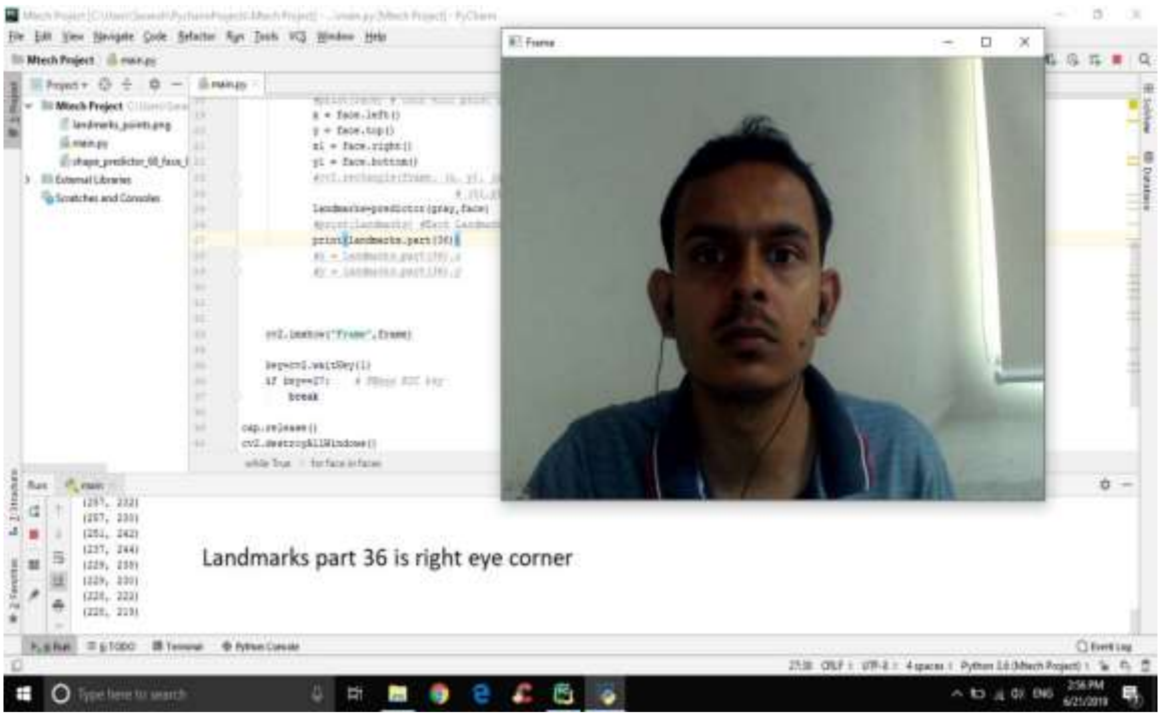
```
                x = face.left()
                y = face.top()
                x1 = face.right()
                y1 = face.bottom()
                #cv2.rectangle(frame, (x, y), (x1, y1), ...

                landmarks=predictor(gray,face)
                print(landmarks)

        cv2.imshow("Frame",frame)

        key=cv2.waitKey(1)
        if key==27:    # Press ESC key
            break

    cap.release()
    cv2.destroyAllWindows()
```

```
<dlib.dlib.full_object_detection object at 0x000001CD841A8DC0>
<dlib.dlib.full_object_detection object at 0x000001CD841A8E30>
<dlib.dlib.full_object_detection object at 0x000001CD841A8DC0>
<dlib.dlib.full_object_detection object at 0x000001CD841A8E30>
<dlib.dlib.full_object_detection object at 0x000001CD841A8DC0>
<dlib.dlib.full_object_detection object at 0x000001CD841A8E30>
<dlib.dlib.full_object_detection object at 0x000001CD841A8DC0>
<dlib.dlib.full_object_detection object at 0x000001CD841A8E30>
```

Each Landmark is an object at particular time

Figure 5.3 This is the output of landmark object files

The console displays object files at each frame that is processed from the video.



```
                x = face.left()
                y = face.top()
                x1 = face.right()
                y1 = face.bottom()
                #cv2.rectangle(frame, (x, y), (x1, y1), ...

                landmarks=predictor(gray,face)
                #print(landmarks) #Get Landmarks
                print(landmarks.part(36))
                #x = landmarks.part(36).x
                #y = landmarks.part(36).y



        cv2.imshow("Frame",frame)

        key=cv2.waitKey(1)
        if key==27:    # Press ESC key
            break

    cap.release()
    cv2.destroyAllWindows()
```

```
(257, 232)
(257, 230)
(251, 242)
(237, 244)
(229, 239)
(229, 230)
(228, 222)
(228, 219)
```

Landmarks part 36 is right eye corner

Figure 5.4 This is the output of landmark 36

The landmark 36 is a coordinate on 2d plane defines as (x, y). Here the output is (228, 219)

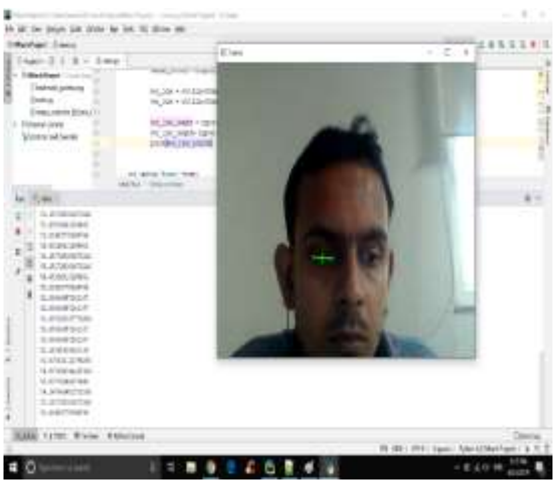


Figure 5.5 This is the output of Horizontal length of the eye.

The console displays the calculated value of the horizontal length of the structure, which is right eye. The output value is 53 to 54.

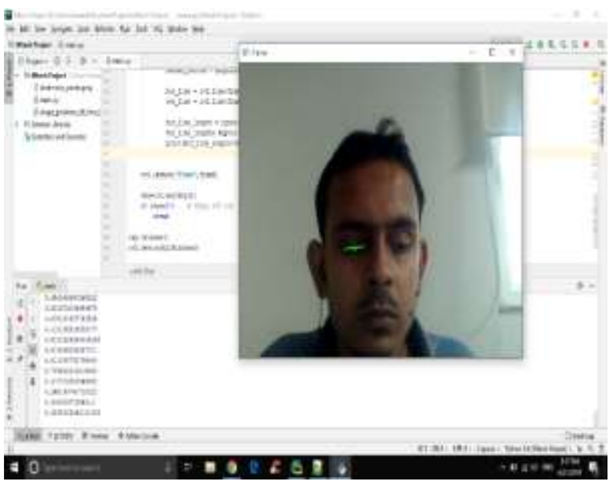
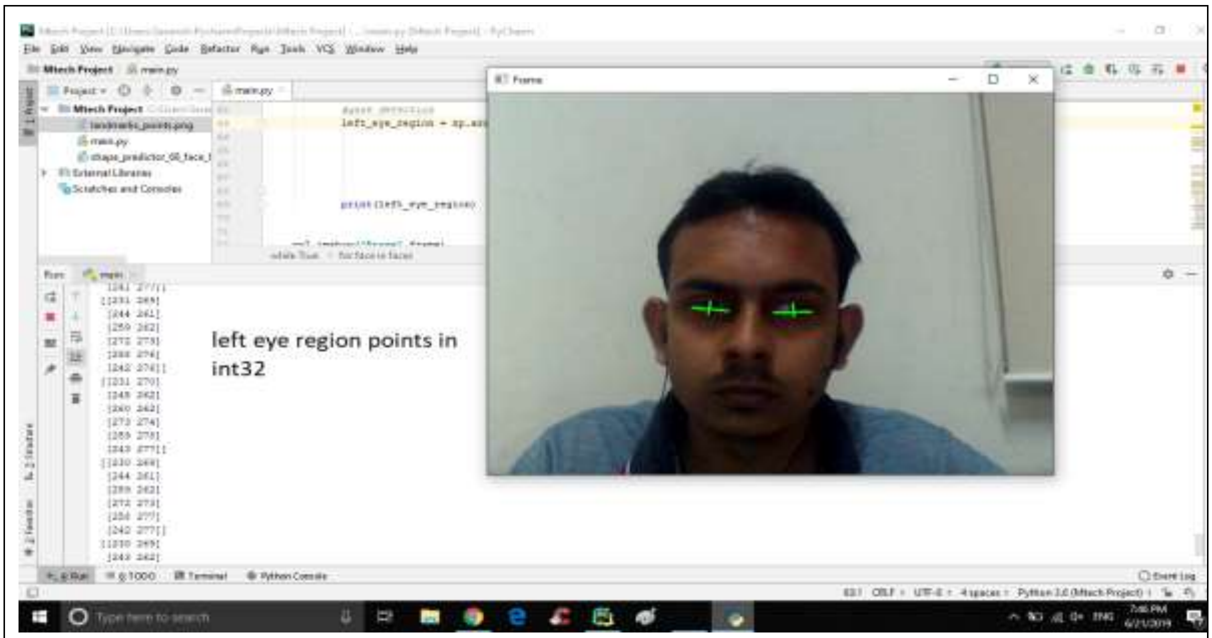


Figure 5.6 This is the output of the ratio of vertical to horizontal length.

The console displays the calculated value of the ratio of the structure, which is right eye. The output value is between 3 to 6.



```
 [241 277]]
[[231 269]
 [244 261]
 [259 262]
 [272 273]
 [258 274]
 [242 274]]
[[231 270]
 [245 262]
 [260 262]
 [273 274]
 [259 278]
 [243 277]]
[[230 268]
 [244 261]
 [259 262]
 [272 273]
 [258 277]
 [242 277]]
[[230 269]
 [243 262]
```

Figure 5.7 This is the output of the left eye region

The console displays the output in int32 format, which is an array of all the coordinates of the 6 structure points of the left eye. The processed output is for each frame. The output is [(231,270), (245,262),(260,262), (273,274), (259,278),(243,277)].
The console displays the output in int32 format, which is an array of all the coordinates of the 6 structure points of the left eye. The processed output is for each frame. The output is extracted to another frame for further processing.

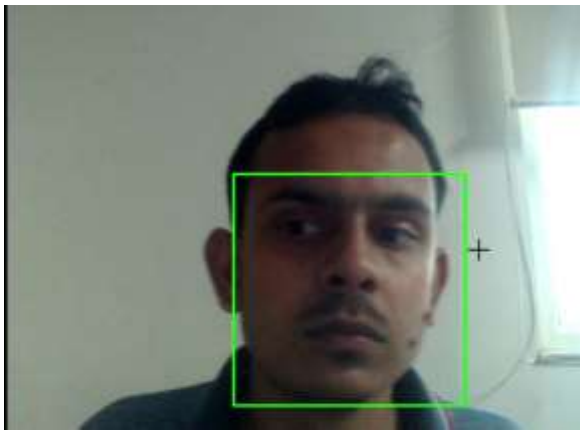


Figure 5.8 This is the output of rectangle on the detected structure

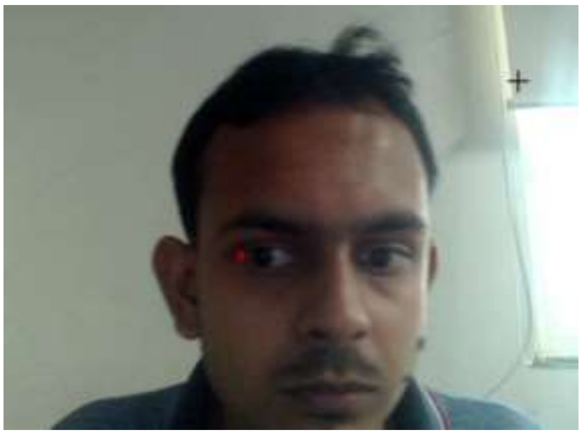


Figure 5.9 This is the output of circle on right eye corner.

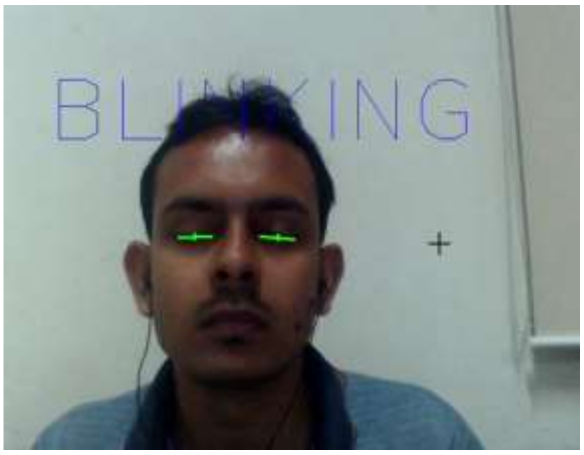


Figure 5.10 This is the output caputured when blinking

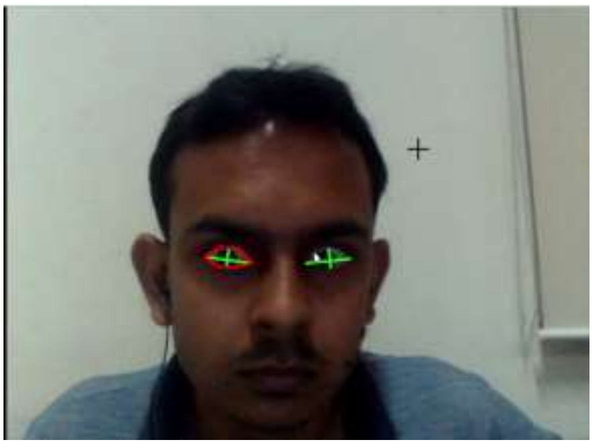


Figure 5.11 This is output of identifying the structure of eyes in the structure identified.

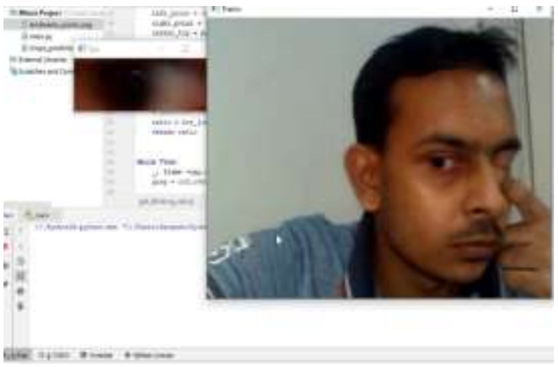


Figure 5.12 This is the ouptut of required region extracted outside in new frame .

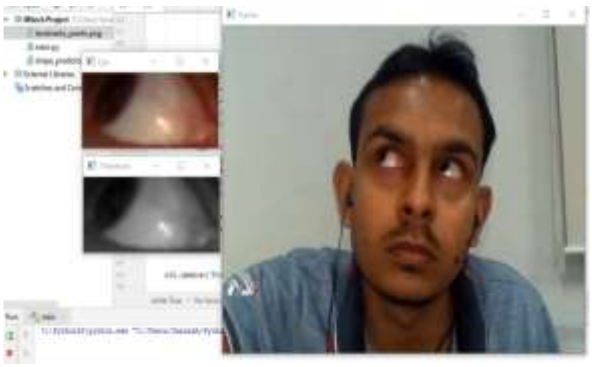


Figure 5.13 This is the output of threshold at 70 and generating new frame of the structure.

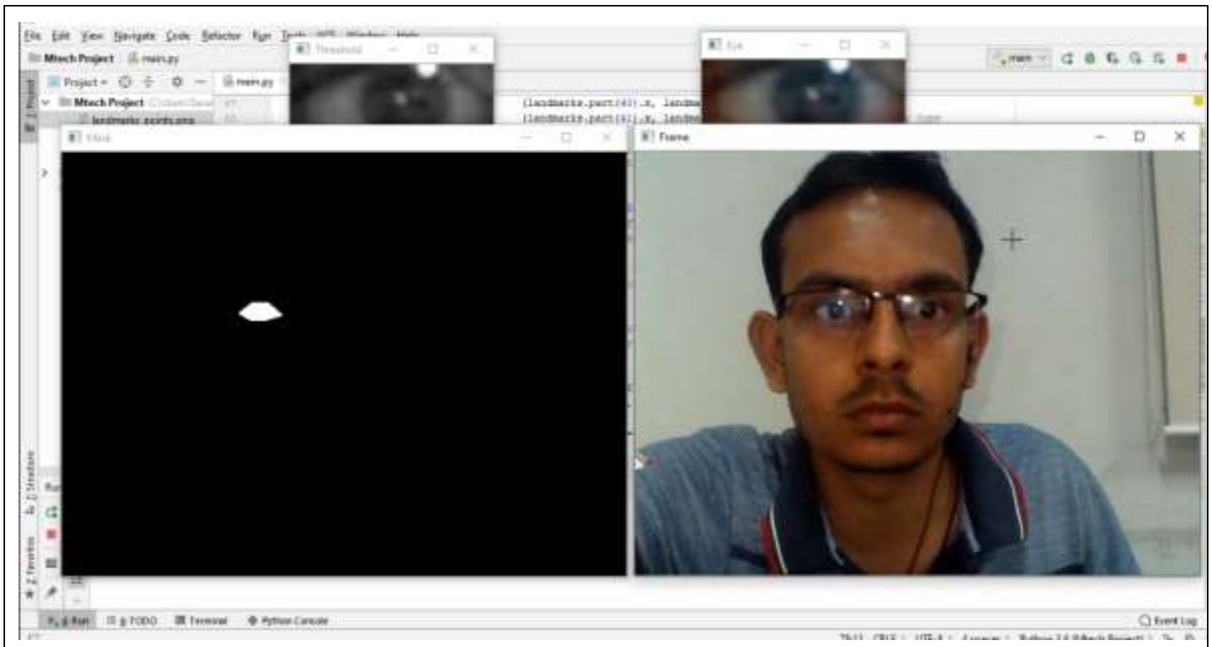


Figure 5.14 This image has generated a mask for the identified structure in our region of interest.
The mask generated on a video of which a screenshot is output. The image has a person on the right frame. We have found the region of interest in that frame and calculated another secondary region in the identified structure. We have created a mask for us on it for further processing.

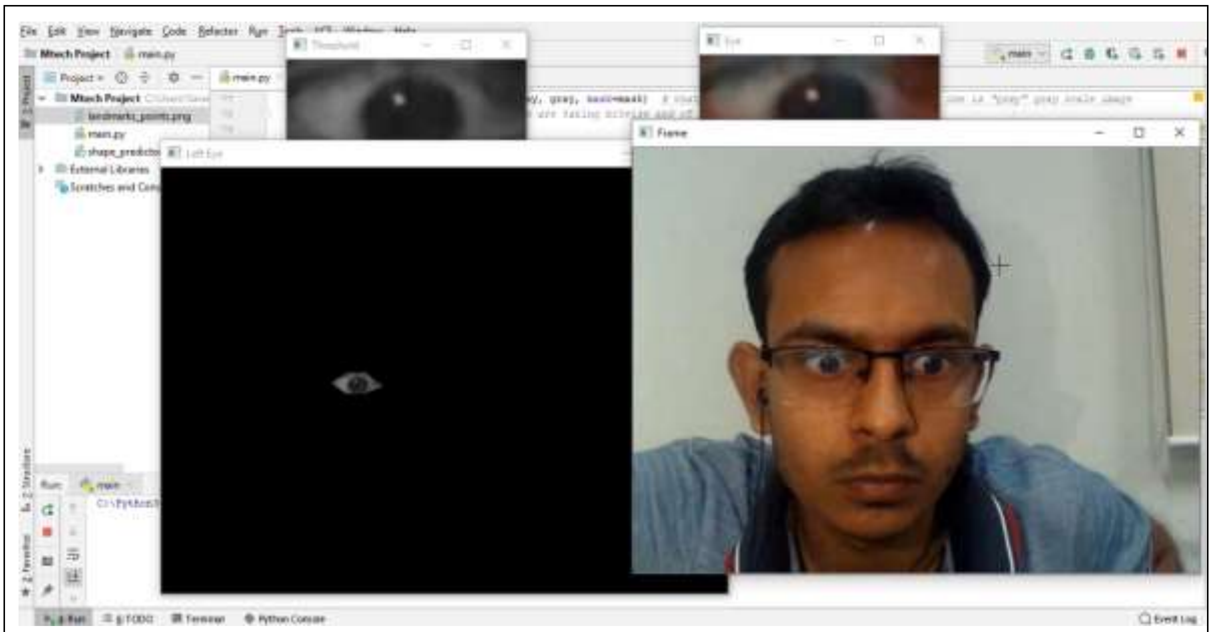


Figure 5.15 This is the image of mask after bitwise_and with the original image.
Thus, we are able to see the final structure image real time.

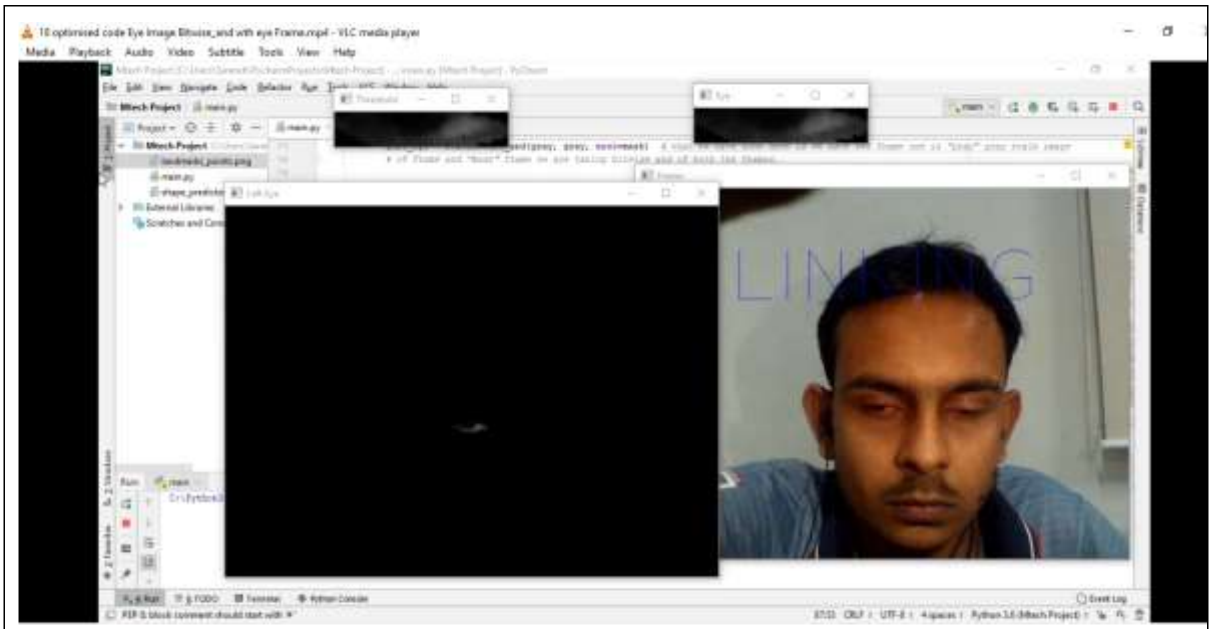
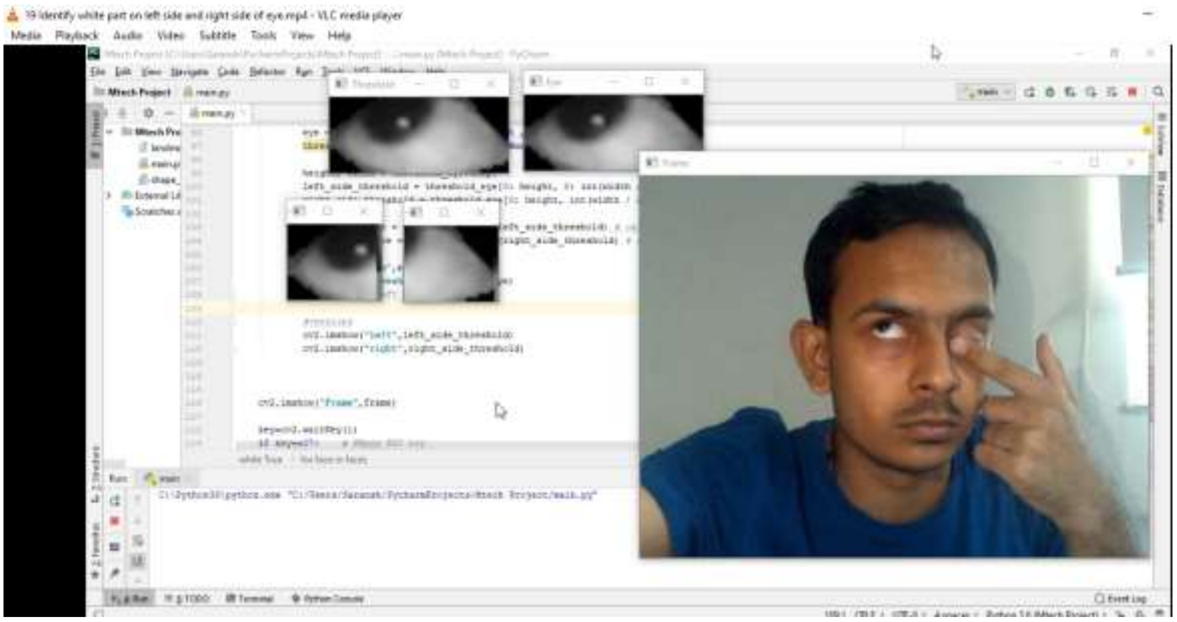

18 optimised code Eye image Bitwise_and with eye Frame.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Threshold
Eye
Left Eye
Frame
BLINKING

Figure 5.16 This image is output of bitwise_and structure.
Here we can see blinking, which is on the main frame while the mask show us the position of the structure that is analysed to conclude blinking.

5.2.2 Step2

Now we are moving into the part of report, which deals with identifying the direction of the identified structure.

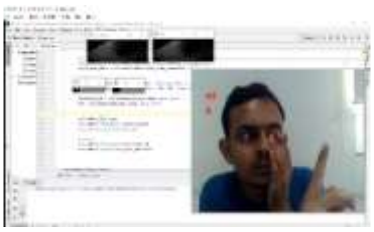
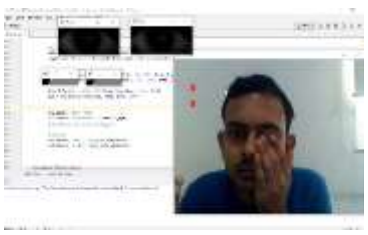
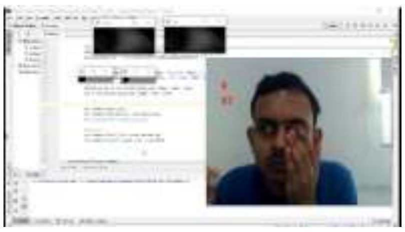

19 identify white part on left side and right side of eye.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Threshold
Eye
Frame

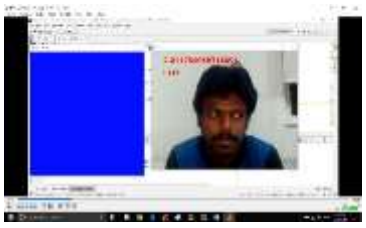
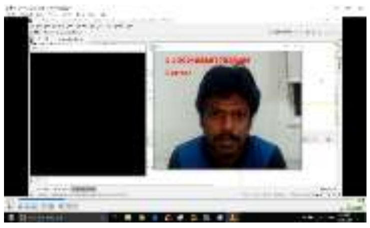
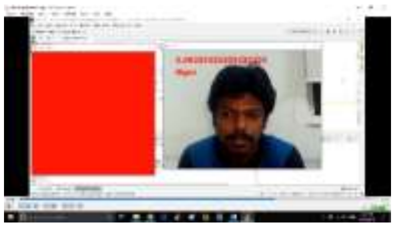
```
left_side_threshold = threshold_eye[0: height, 0: int(width / 2)]

cv2.imshow("left",left_side_threshold)
cv2.imshow("right",right_side_threshold)

cv2.imshow("Frame",frame)

key=cv2.waitKey(1)
```

Figure 5.17 This is the output of an image with split region of interest.
The right image is our main image frame. The frame is required to generate threshold image. We use the threshold image to create split of the threshold image. The important thing to note is we have removed the skin from the region of interest for calculating the pose of the identified structure.

.

The value, calculated by number of zeroes in the region of our structure that we have split. Figure 5.22, Figure 5.23 and Figure 5.24 are the output of the values.

 Figure 5.18 This is the first output for looking left. The output is 45.	 Figure 5.19 This is the first output for looking center. The output is 0.	 Figure 5.20 This is the first output for looking right. The output is 57.

 Figure 5.21 This is the output for ratio while the direction is left. The ratio is around 2.	 Figure 5.22 This is the output for ratio while the direction is center. The ratio is .42	 Figure 5.23 This is the output for ratio while the direction is right. The ratio is 0.083

5.2.3 Step3 Final Results

5.2.3.1 Bounding box and Front of the face

This is output for the direction of the pose. The pose is from nose using x, axis only. The output is shown in Figure 5.24, Figure 5.25, Figure 5.26.

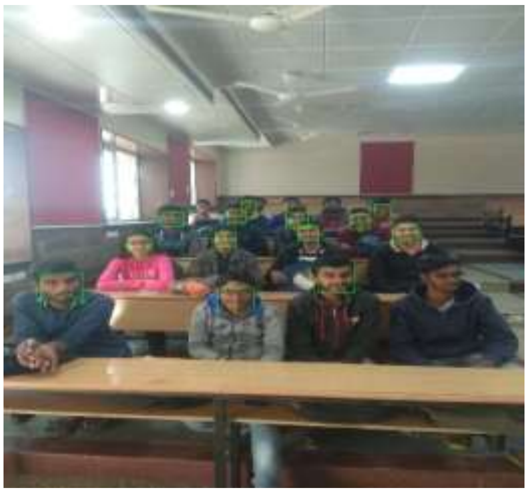


Figure 5.24 The image has boundary box on a class of students

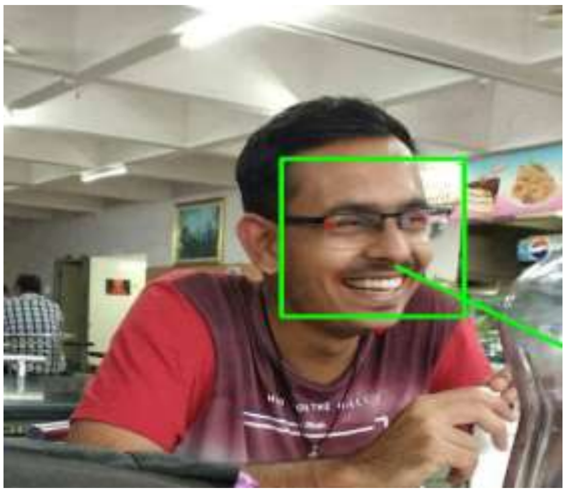


Figure 5.25 The image shows the region of interest.

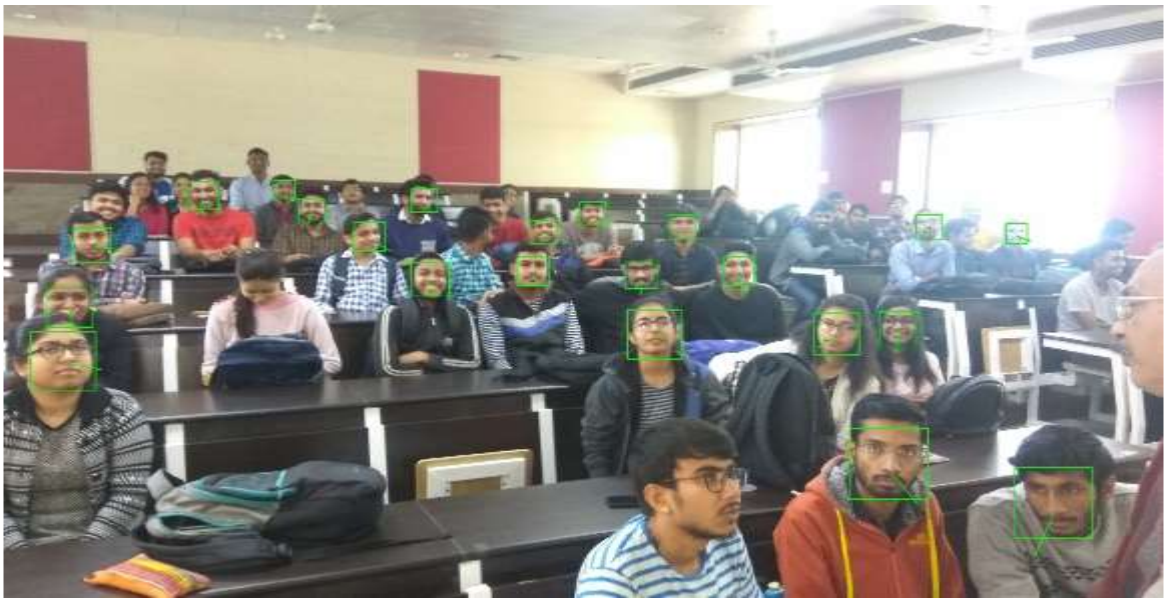


Figure 5.26 This is class of student along with the direction of pose in 1 axis only.

5.2.3.2 Bounding Box along with x, y and z axis from nose.

This is output for the direction of the pose. The pose is from nose using x, y and z axis. In Figure 5.27, the red colour line shows the front of the face while blue and green colour line are perpendicular to the red line.

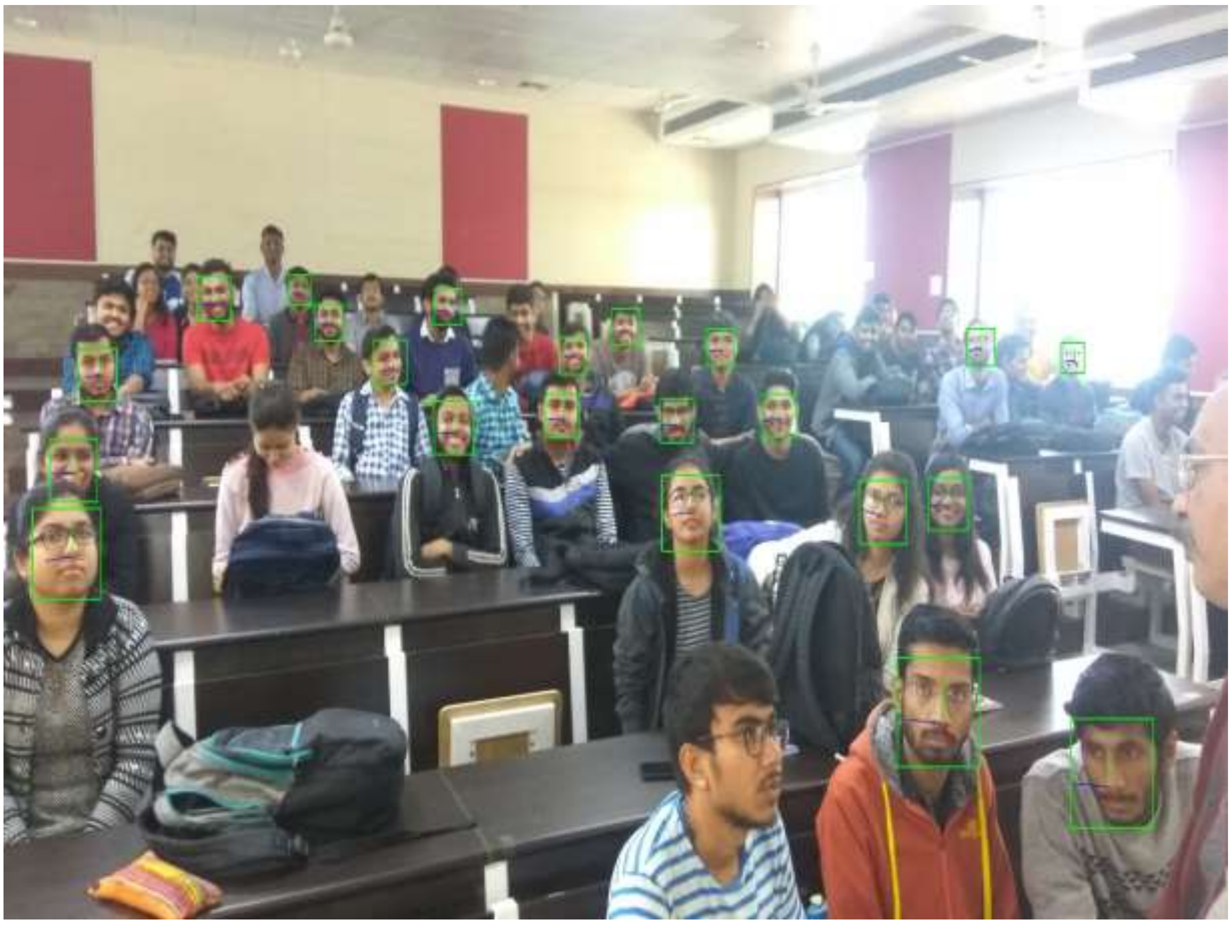


Figure 5.27 This is the output of Bounding box with x, y and z axis

5.3 Result and Analysis

The reference paper result[14] and my works result is compared in the Table 5.2. The reference work is able to detect the human face and human head pose on an image. The head pose detected only shows the front of the human face. In our work, we are able to detect he human face from an image as well as head pose of the human face. We are also able to display the human head pose with a cube on the human face. We have merged the head pose with gaze estimation. The back face of the cube shows the head pose in blue while the front face of the cube shows the gaze direction change in red colour.

The Simultaneous face and pose detection system is able to generate output for 30 fps. The output will only identify the region of interest and detect the pose of the front of the face. The output displays the direction of the face in x,y and z axis. Our work is able to replicate the result of the reference paper. The solution is able to detect face and pose of human head with 25 frames per second. The resultant average processing time is 0.03868949s with minimum time recorded 0.015s and maximum time recorded as 0.48s

Table 5.2 Result and Analysis

Model	Processing Time	Frames per Second
HyperFace[16]	3s	0.33fps
All in One[15]	3.5s	0.28fps
Simultaneous face and pose detection[14]	0.0333s	30fps
My work	0.03868949s	25fps

6 Chapter 6 – Implementation of Gaze Estimation and Audio caption generation.

6.1 Gaze Estimation Implementation

The process merging the gaze estimation and face detection is explained in Figure 6.1. It is achieved by changing the x, y and z-axis displayed in the solution as a cube which is derived from the structure landmarks as well as the pupil, eye movement.

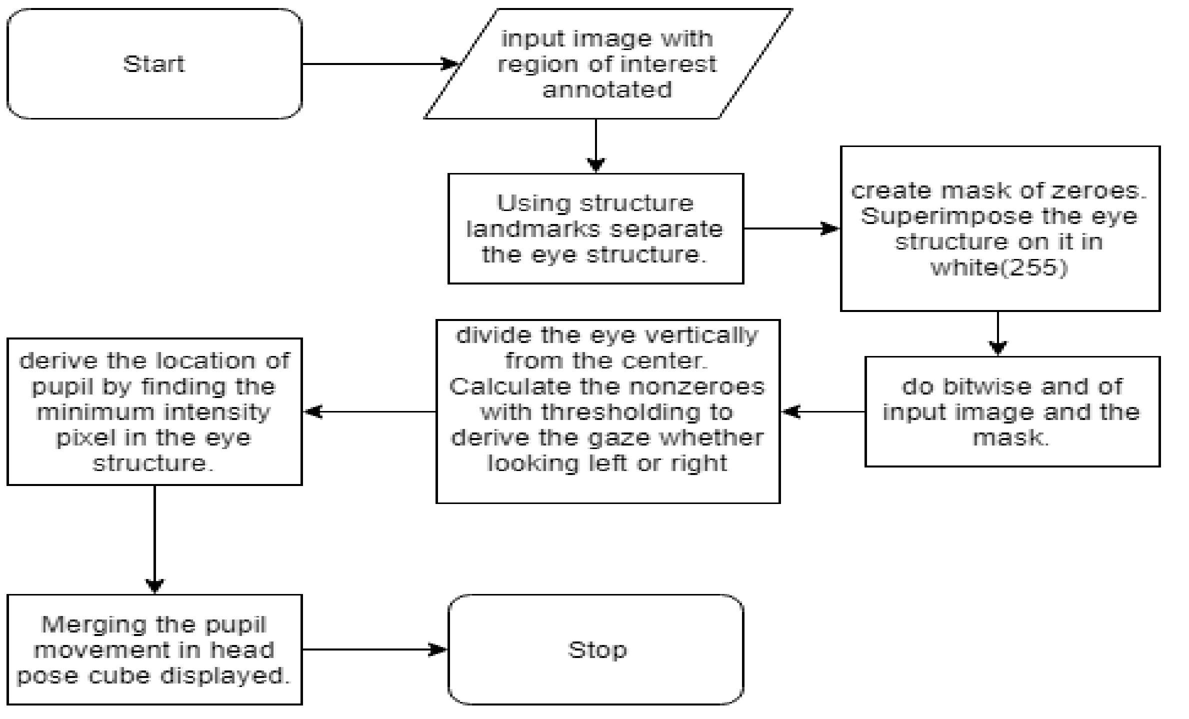

Start
input image with region of interest annotated
Using structure landmarks separate the eye structure.
create mask of zeroes. Superimpose the eye structure on it in white(255)
do bitwise and of input image and the mask.
divide the eye vertically from the center. Calculate the nonzeroes with thresholding to derive the gaze whether looking left or right
derive the location of pupil by finding the minimum intensity pixel in the eye structure.
Merging the pupil movement in head pose cube displayed.
Stop

Figure 6.1 Flow diagram for gaze estimation

6.2 Audio generation

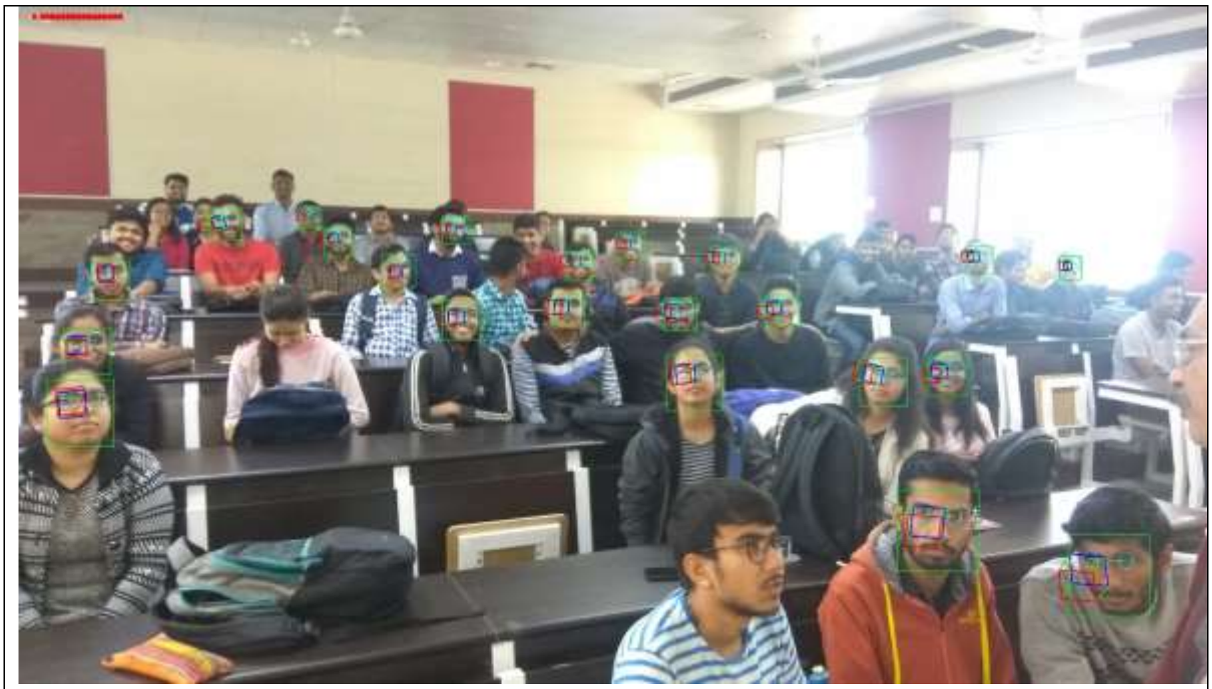
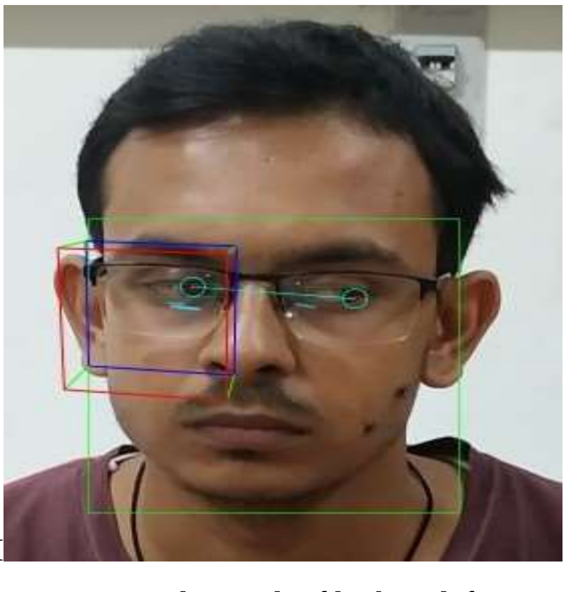
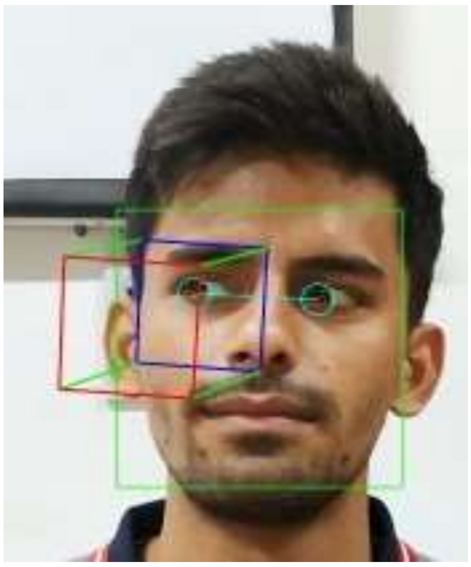


Figure 6.2 The output displays the resultant cube for some of the faces.

 I Figure 6.3 Audio result of looking left. An audio is generated with recognised name of person and with message “Saransh Bhandari looking Left”	 Figure 6.4 Audio result of looking right An audio is generated with recognised name of person and with message “Nirdosh Kumar looking Right”

The Figure 6.2 shows the image of classroom. The result image has 22 face cubes for the human face present in the image. The Figure 6.3 and Figure 6.4 are the output of input images on which we have created a bounding box to detect the face. The results display the cube. The origin if the cube is nose of the face. The head pose, shown with blue. The gaze estimation is display in red. The green lines joining the two structure are rotation angles with respect to world coordinates calculated using camera matrix. The solution generates a caption on the image if looking right or left. The accuracy of face recognition is in Table 6.1. The result recognizes the face in the bounding box and generates an audio caption “Saransh looking Left” and “Nirdosh Kumar Looking right”.

Table 6.1 Accuracy of Face Encoding

Positive Detection	54
Negative Detection	5
Accuracy	91

6.3 Applications

The applications of the solution is in various field like, classroom attention span of the person. The solution can identify the faces on the classroom. It can also estimate the head pose and eye gaze of all the identified faces. We can calculate the average attention span of the class during lecture. Similarly, if used in examination, the solution can process the frames of the real-time video and give the results.

Another interesting and morally controversial application is identifying the human in mass protest. For example, to identify some of the key protesters who indulge in vandalism, the solution is beneficiary.

At the same time, using face recognition by police, enforcement agency can curtail the right to peaceful protest. Recently, San Francisco, a city in USA, banned police of use of face recognition technology. At the same time, in India’s capital Delhi, police is advised by the High Court of Delhi to use face detection along with pose estimation and gaze detection to find the missing 5000 children in the NCT of Delhi, India.

The solution can be used to type on virtual keyboard using gaze estimation. The virtual reality field requires gaze analysis. The medical surgeries by robots is another field where the solution can be used to estimation of the gaze of the camera to conduct successful operations.

7 Chapter 7 - Conclusion and Future Work

7.1 Conclusion

In the work, the main objective was to find the region of human face in the image and complete the pose estimation on it. The main aim was to able to run the model on the real time video stream. The model is able to detect the human face and its landmark. The model is able to estimate the pose of the human head. The model is able to detect all 6 landmarks in the human face. We are able to extensively use the eye structure detected by our model. The model is able to detect the direction of human eye. We are able to recognise the detected face and generate an audio caption which speaks the name of the detected face along with the direction of the face. For example, "Ram looking right" where Ram is the name of recognised face while "looking right is the head pose". In conclusion, the task is able to detect face, estimate pose, detect gaze on real time video stream. We are able to recognise the human face and generate the audio caption appropriately.

7.2 Future Works

The future work includes applying the model on a mobile phone i.e. on an Android device. The process of capturing the frame can be optimised using multiple algorithm. We can merge few frames and then process them. As of now, we introduce delay which is the normal approach found in all the literature surveys. The logic for camera position if kept moving then how the model will be optimised to such situation. The field of computer vision is vast and we can think of more such practical problems faced by any camera/ portable device user.

The limitations of the system is that it expects to see the entire face in the 2D plane. Once the eyes are not available or there is partial face available the fails to give accurate pose estimations. The system also fails to identify the region of interest in that case.

7.3 References

1. https://ibug.doc.ic.ac.uk/resources/300-W/
2. 300 Faces In-The-Wild Challenge: database and results, Christos Sagonas a, *, Epameinondas Antonakosa, Georgios Tzimiropoulosb, Stefanos Zafeirioua, Maja Pantic a, c aImperial College London, Department of Computing, London, UK bUniversity of Nottingham, School of Computer Science, Nottingham, UK cFaculty of Electrical Engineering, Mathematics, and Computer Science, University of Twente, The Netherlands
3. 300 Faces in-the-Wild Challenge: The first facial landmark localization Challenge Christos Sagonas1 , Georgios Tzimiropoulos1,2 , Stefanos Zafeiriou1 and Maja Pantic1,3 1 Comp. Dept., Imperial College London, UK 2 School of Computer Science, University of Lincoln,U.K. 3 EEMCS, University of Twente, The Netherlands
4. A semi-automatic methodology for facial landmark annotation Christos Sagonas1 , Georgios Tzimiropoulos1,2 , Stefanos Zafeiriou1 and Maja Pantic1,3 1 Comp. Dept., Imperial College London, UK 2 School of Computer Science, University of Lincoln,U.K. 3 EEMCS, University of Twente, The Netherlands
5. http://dlib.net/webcam_face_pose_ex.cpp.html
6. http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2
7. Stanford University. Convolutional neural network - cs231n, stanford univeristy. http://cs231n.stanford.edu/, 2017.
8. Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features, 2001.
9. Wikipedia contributors. Viola jones object detection framework - wikipedia the free encyclopedia. https://en.wikipedia.org/wiki/Viola-Jones_object_detection_framework. [Online; accessed 3-July-2019].
10. Reinforcement learning lecture. https://cs.uwaterloo.ca/ ppoupart/teaching/cs885-spring18/slides/cs885-lecture4a.pdf.
11. https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/
12. https://en.wikipedia.org/wiki/Convolutional_neural_network
13. https://www.cv-foundation.org/openaccess/content_cvpr_2015/app/1A_089.pdf
14. H. Wu, K. Zhang and G. Tian, "Simultaneous Face Detection and Pose Estimation Using Convolutional Neural Network Cascade," in *IEEE Access*, vol. 6, pp. 49563-49575, 2018
15. V. Kazemi and J. Sullivan, "One millisecond face alignment with an ensemble of regression trees," 2014 IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, 2014, pp. 1867-1874.
16. R. Ranjan, V. M. Patel and R. Chellappa, "HyperFace: A Deep Multi-Task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 41, no. 1, pp. 121-135, 1 Jan. 2019
17. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/

18. https://en.wikipedia.org/wiki/Weighted_arithmetic_mean
19. https://en.wikipedia.org/wiki/Ensemble_learning#Stacking
20. https://medium.com/@rrfd/boosting-bagging-and-stacking-ensemble-methods-with-sklearn-and-mlens-a455c0c982de
21. Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In IEEE Conf. on CVPR, 2014. 1, 2, 5, 8