**A**

**PBL Report**

on

**“SMILEFIE”**

In partial fulfilment for the award of the degree of

**Bachelor of Technology**

**in**

**Computer Science & Engineering**



**SUBMITTED BY**

Reeta Soni (170382)

Riya Sinha (170385)

**UNDER THE GUIDANCE OF**

**Mr. Robin Tyagi**

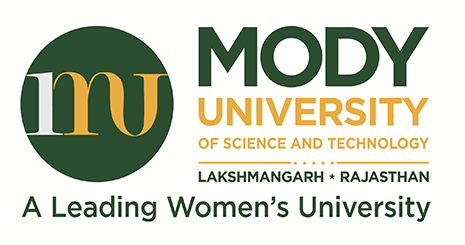
***Computer Science & Engineering***

**SCHOOL OF ENGINEERING & TECHNOLOGY**

MODY UNIVERSITY OF SCIENCE & TECHNOLOGY

LAKSHMANGARH, SIKAR – 332311 (RAJASTHAN)

**NOVEMBER, 2019**

****

**School of Engineering and Technology**

**Mody University of Science and Technology,**

**Lakshmangarh**

Date :

**Certificate**

This is to certify that the PBL Report entitled “SMILEFIE” submitted by Ms. Riya Sinha and Ms. Reeta Soni, as a partial fulfillment for the requirement of B.Tech V Semester Examination of the School of Engineering and Technology, Mody University of Science and Technology, Lakshmangarh for the academic session 2019-2020 is an original project work carried out under the supervision and guidance of Mr. Robin Tyagi.

PROJECT GUIDE: HEAD OF DEPARTMENT

Signature: Signature:

Name: Mr. Robin Tyagi Name:

Date: Date:

EXAMINER-1 EXAMINER-2

Signature: Signature:

Name: Name:

Date: Date:

i

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many of the individuals. We would like to extend our sincere gratitude to all of them.

We are highly obliged to Dr. V.K. Jain, Dean CET, Mody University of science and technology, for the amenities provided to accomplish this main project.

We would like to thank our Head of Department, Dr. A. Senthil for his constructive criticism all the way through our project.

We are indebted to Mr. ROBIN TYAGI for his guidance and constant supervision as well as providing necessary information regarding the project and also for their support in finishing the project.

We would like to express our gratefulness towards our parents and members of Mody University for their kind co-operation and encouragement which helped us in completion of this project.

Our thanks and appreciation goes to all those people who have keenly helped us with their abilities.

TABLE OF CONTENTS

Certificate i

Acknowledgment ii

Abstract 1

Chapter :1 Introduction 2

Chapter :2

2.1) MACHINE LEARNING

2.2) FACIAL FEATURE DETECTION

2.1) SMILE DETECTION

Chapter :3

3.1) SOURCE CODE & EXPLANATION

3.2) FULL CODE

3.3) OUTPUT

Chapter :4 ADVANTAGES OF SMILE DETECTION

Chapter :5 INDIVISUAL CONTRIBUTION

Chapter :6 CONCLUSION AND REFERENCE

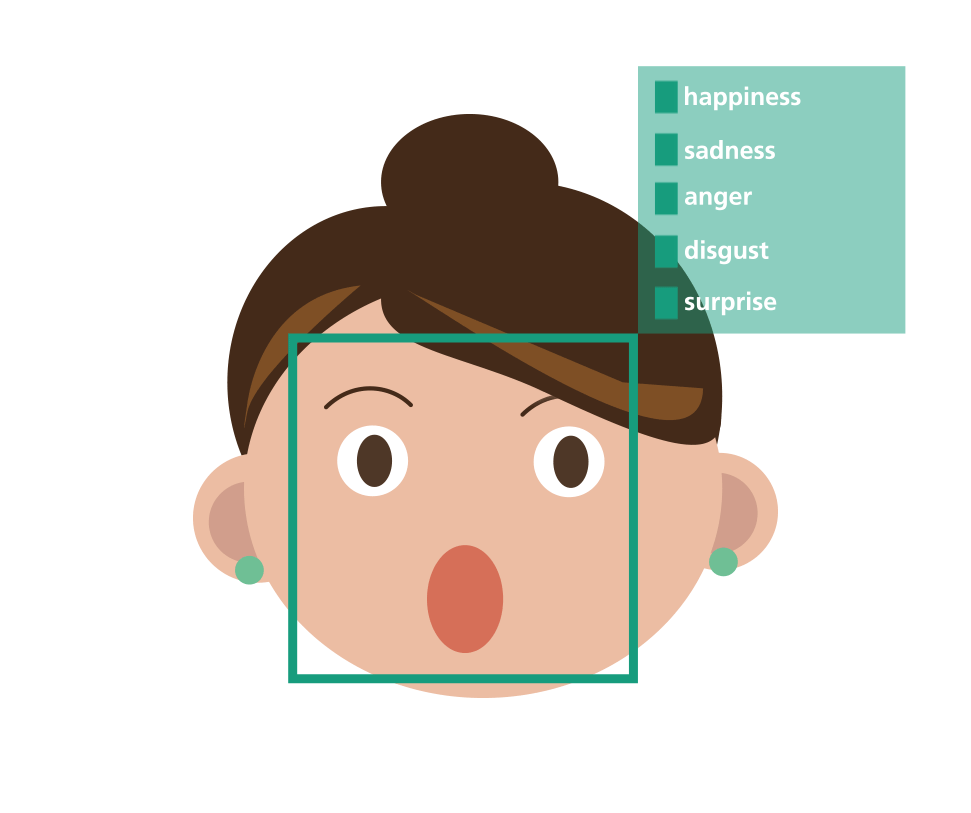
Chapter :7 PLAGIARISM REPORT

ABSTRACT

*SMILE detectors are used in many industries, one being the media industry where it is important for the companies to determine the public reaction to their products. In this Project, we are going to build a smile detector using OpenCV which takes in live feed from webcam. The smile/happiness detector that we are going to implement would be a raw one, there exist many better ways to implement it. As soon as subject smile image is being captured and saved in the desired path (location). Haarcascades are classifiers that are used to detect features (of face in this case) by superimposing predefined patterns over face segments and are used as XML files. In our model, we shall use face, eye and smile haarcascades*.

CHAPTER: 1 INTRODUCTION

Businesses strive to deliver the most important product of all: happiness.



Why? Happiness might just be more than a chemical reaction. A happy customer is more likely to walk through the door again, and data on happiness can help businesses understand which products would do better and have a higher retention rate. Machines can learn to recognize happiness and in this tutorial, I’m going to show you how to create a facial recognition model that can do so. Humans are hardwired to read emotions off faces. This ability has been ingrained in us as a result of the enormous evolutionary advantage it provides. A smiling face implies agreeableness and from an angry face, we infer danger! Thus, our minds immediately evaluate the underlying emotion of a person by merely looking at their face. The Smile Detector illustrates how facial expression analysis solutions can be put to use in a real-life environment. As the software measures mood in real-time, people can see the result of their actions instantaneously (when your smile is big enough, your picture appears on the (big) screen). Face analysis technology hence opens up a whole area of interactive application opportunities with for instance smart TV’s, mobile devices with frontal facing cameras, or robots with a webcam serving elderly people in assisted living environments. Not only smile levels can be used for controls, but also for instance eye movement or head movement.

CHAPTER: 2.1 MACHINE LEARNING

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs. Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers. Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes. In this tutorial, we’ll look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbour algorithm, decision tree learning, and deep learning. We’ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we’ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms.

## Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed. Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

### Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional un-labelled data. For example, with supervised learning, an algorithm may be fed data with images of sharks labelled as fish and images of oceans labelled as water. By being trained on this data, the supervised learning algorithm should be able to later identify un-labeled shark images as fish and unlabelled ocean images as water.

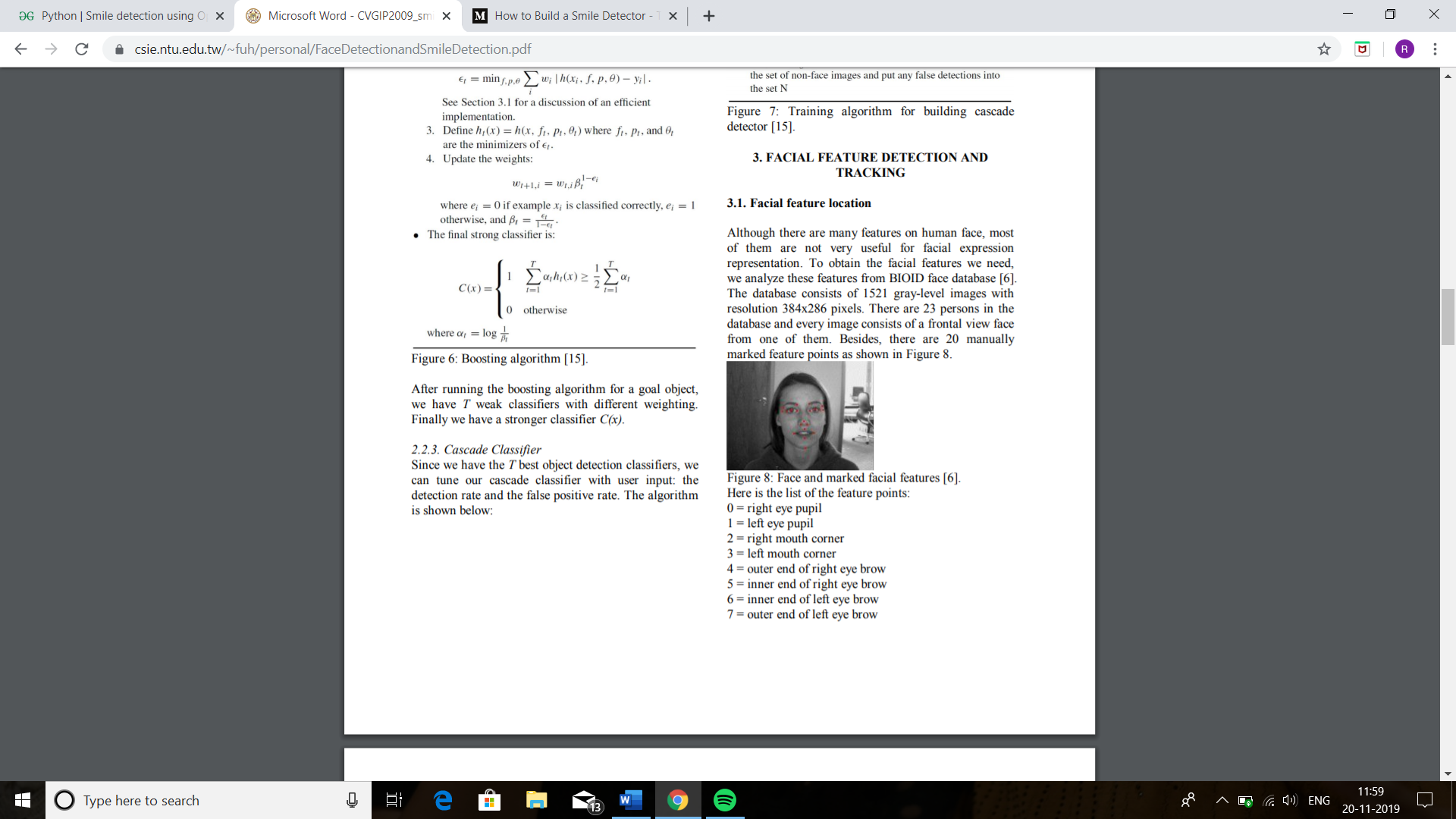
A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

### Unsupervised Learning

In unsupervised learning, data is unlabeled, so the learning algorithm is left to find commonalities among its input data. As unlabeled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data. Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases. Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

2.2) FACIAL FEATURE DETECTION AND TRACKING

Although there are many features on human face, most of them are not very useful for facial expression representation. To obtain the facial features we need, we analyze these features from BIOID face database [6]. The database consists of 1521 gray-level images with resolution 384x286 pixels. There are 23 persons in the database and every image consists of a frontal view face from one of them. Besides, there are 20 manually marked feature points as shown in Figure (i).

 Fig(i)

Here is the list of the feature points:

0 = right eye pupil

1 = left eye pupil

2 = right mouth corner

3 = left mouth corner

4 = outer end of right eye brow

5 = inner end of right eye brow

6 = inner end of left eye brow

7 = outer end of left eye brow

8 = right temple

9 = outer corner of right eye

10 = inner corner of right eye

11 = inner corner of left eye

12 = outer corner of left eye

13 = left temple

14 = tip of nose

15 = right nostril

16 = left nostril

17 = centre point on outer edge of upper lip

18 = centre point on outer edge of lower lip

19 = tip of chin

We first use different algorithm to detect the face region in the image with scale factor 1.05 to get as precise position as possible, and then normalize the face size and calculate the feature relative positions and their standard deviation.

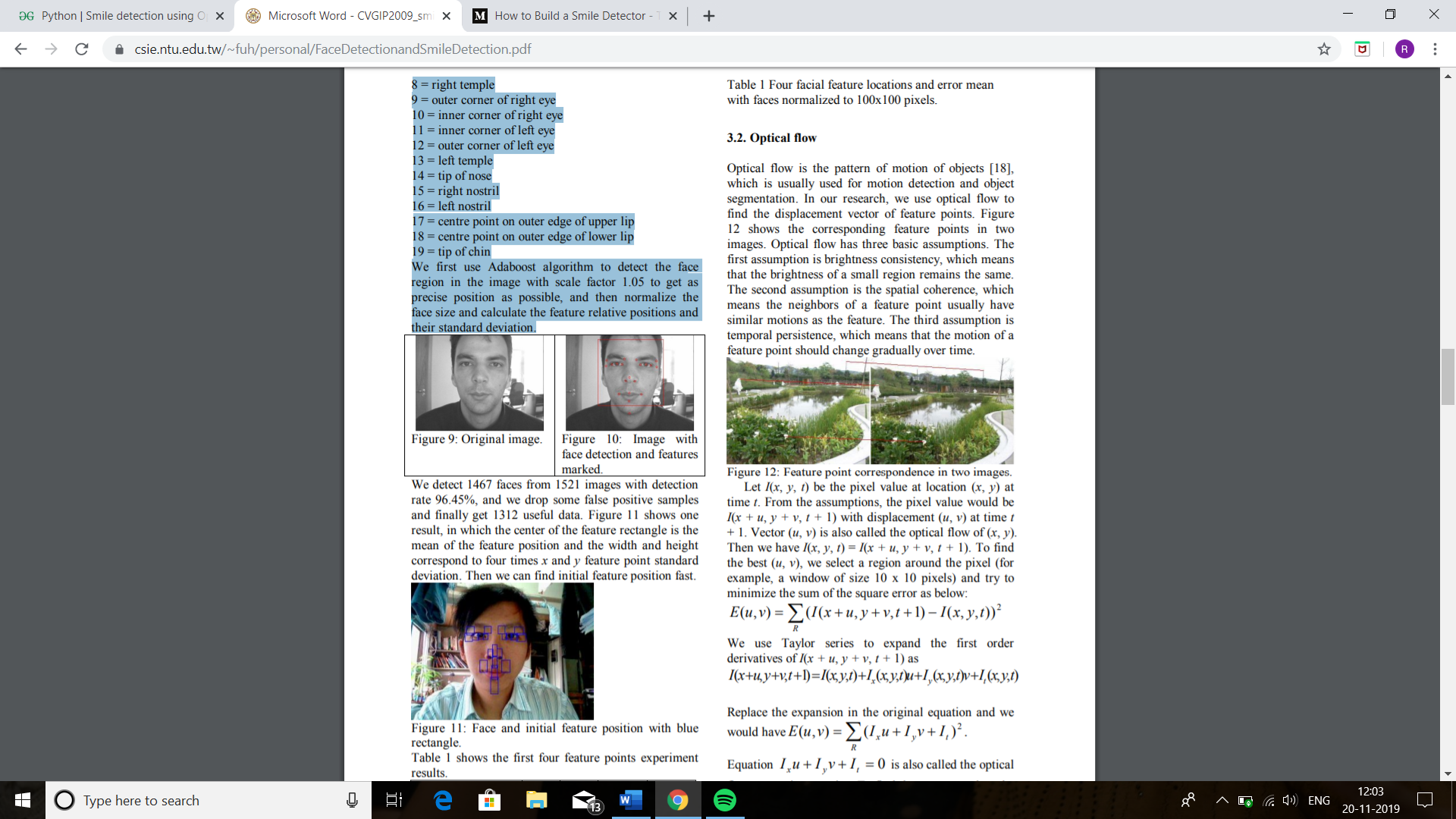


Fig (ii) – Original Image Fig (iii) - Image with face detection and features marked.

In the past few years, face recognition owned significant consideration and appreciated as one of the most promising applications in the field of image analysis. Face detection can consider a substantial part of face recognition operations. According to its strength to focus computational resources on the section of an image holding a face. The method of face detection in pictures is complicated because of variability present across human faces such as pose, expression, position and orientation, skin colour, the presence of glasses or facial hair, differences in camera gain, lighting conditions, and image resolution. Object detection is one of the computer technologies, which connected to the image processing and computer vision and it interacts with detecting instances of an object such as human faces, building, tree, car, etc. The primary aim of face detection algorithms is to determine whether there is any face in an image or not.

In recent times, a lot of study work proposed in the field of Face Recognition and Face Detection to make it more advanced and accurate, but it makes a revolution in this field when Viola-Jones comes with its Real-Time Face Detector, which is capable of detecting the faces in real-time with high accuracy.

Face Detection is the first and essential step for face recognition, and it is used to detect faces in the images. It is a part of object detection and can use in many areas such as security, bio-metrics, law enforcement, entertainment, personal safety, etc.

It is used to detect faces in real time for surveillance and tracking of person or objects. It is widely used in cameras to identify multiple appearances in the frame Ex- Mobile cameras and DSLR’s. Facebook is also using face detection algorithm to detect faces in the images and recognise them.

2.3) SMILE DETECTION: -

The new Sony Cyber-shot DSC T-200 digital camera has an ingenious “smile shutter” mode. Using proprietary algorithms, the camera automatically detects the smiling face and closes the shutter. To detect the diﬀerent degrees of smiles by the subject, smile detection sensitivity can be set to high, medium or low. Some reviews argue that: the technology is not still so much sensitive that it can capture minor facial changes. Your facial expression has to change considerably for the camera to realize that”, or “The camera’s smile detection – which is one of its more novel features – is reported to be inaccurate and touchy”. Whatever the case, detection rates or details of the algorithm are not available, and so it is diﬃcult to compare the system. Canon also has a similar smile detection system. Sensing component company Omron has recently developed a “smile measurement software”, which measures the amount of happiness that human subject of a photo are exhibiting. The software uses a proprietary 3D model ﬁtting technique to detect and analyse faces. This smile checking software rates how much a subject is smiling and gives a ’smile factor’ on a scale of 0 to 100%.



Fig. (iv) Examples of positive images used for training the smile detector

This analysis only takes about 44 milliseconds using a PIV at 3.2Ghz and can be performed on images of faces as small as 60 pixels wide. Omron claims that this device is more than 90% accurate .On a more scientiﬁc level, there are a signiﬁcant number of papers that have tackled facial expression recognition, see the surveys. Few systems, however, have been speciﬁcally designed for smile detection. The smile detector of used a vector of lip measures (extracted from an edge image) and a perceptron classiﬁer. Edge features, however, may no be robust enough for practical use. More elaborated is the method of, which used HLAC (Higher-order Local Autocorrelation) along with Fisher weight maps, achieving recognition rates of 97.9%. The BROAFERENCE system was developed to assess TV or multimedia content through smile measurement. In this case, 8 mouth points are tracked, feeding a neural network classiﬁer with the 16 feature vector. Unfortunately the authors do not give precise ﬁgures for its performance, although they claim that it achieves a 90% detection rate. The smile detection system proposed in this paper is based on a Viola-Jones cascade classiﬁer. Training was carried out using 2436 positive images and 3376 negative images. The images were ﬁrst extracted from Internet, then detected and normalized by the face detection system described above. Figure shows some examples of the positive images used for training. When the cascade detector is searching over the image, it may produce multiple positives around the positive region (the smile). Those detected rectangles largely overlap. Usually, isolated detections are false detections and they should be discarded. The number of neighbour detections is normally used as a conﬁdence threshold. For smile detection, the number of neighbour detections can also be considered as a conﬁdence measure. The more neighbours detected around an image region, the more conﬁdence that the region contains a smile. If the negative images of the training set contain mostly neutral faces then the number of neighbours can be considered as a measure of smile intensity. It used in the negative set.

CHAPTER: 3.1 SOURCE CODE AND EXPLAINATION

LIBRARIES USED:

**OpenCV**:

OpenCV (*Open source computer vision*) is a [library of programming functions](https://en.wikipedia.org/wiki/Library_(computing)) mainly aimed at real-time [computer vision](https://en.wikipedia.org/wiki/Computer_vision). Originally developed by [Intel](https://en.wikipedia.org/wiki/Intel_Corporation), it was later supported by [Willow Garage](https://en.wikipedia.org/wiki/Willow_Garage) then Itzel (which was later acquired by Intel). The library is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) and free for use under the [open-source](https://en.wikipedia.org/wiki/Open-source_software) [BSD license](https://en.wikipedia.org/wiki/BSD_license). Emotion detectors are used in many industries, one being the media industry where it is important for the companies to determine the public reaction to their products. In this article, we are going to build a smile detector using OpenCV which takes in live feed from webcam. The smile/happiness detector that we are going to implement would be a raw one, there exist many better ways to implement it.

OpenCV's application areas include:

* 2D and 3D feature toolkits
* [Ego motion](https://en.wikipedia.org/wiki/Egomotion) estimation
* [Facial recognition system](https://en.wikipedia.org/wiki/Facial_recognition_system)
* [Gesture recognition](https://en.wikipedia.org/wiki/Gesture_recognition)
* [Human–computer interaction](https://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction) (HCI)
* [Mobile robotics](https://en.wikipedia.org/wiki/Mobile_robotics)
* Motion understanding
* Object identification
* [Segmentation](https://en.wikipedia.org/wiki/Segmentation_(image_processing)) and recognition
* [Stereopsis](https://en.wikipedia.org/wiki/Stereopsis) stereo vision: depth perception from 2 cameras
* [Structure from motion](https://en.wikipedia.org/wiki/Structure_from_motion) (SFM)
* [Motion tracking](https://en.wikipedia.org/wiki/Video_tracking)
* [Augmented reality](https://en.wikipedia.org/wiki/Augmented_reality)

To support some of the above areas, OpenCV includes a statistical [machine learning](https://en.wikipedia.org/wiki/Machine_learning) library that contains:

* [Boosting](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm))
* [Decision tree learning](https://en.wikipedia.org/wiki/Decision_tree_learning)
* [Gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting) trees
* [Expectation-maximization algorithm](https://en.wikipedia.org/wiki/Expectation-maximization_algorithm)
* [k-nearest neighbour algorithm](https://en.wikipedia.org/wiki/K-nearest_neighbor_algorithm)
* [Naive Bayes classifier](https://en.wikipedia.org/wiki/Naive_Bayes_classifier)
* [Artificial neural networks](https://en.wikipedia.org/wiki/Artificial_neural_network)
* [Random forest](https://en.wikipedia.org/wiki/Random_forest)
* [Support vector machine](https://en.wikipedia.org/wiki/Support_vector_machine) (SVM)
* [Deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_network) (DNN)
* Programming language

OpenCV is written in [C++](https://en.wikipedia.org/wiki/C%2B%2B) and its primary interface is in C++, but it still retains a less comprehensive though extensive older [C interface](https://en.wikipedia.org/wiki/C_(programming_language)). There are bindings in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) and [MATLAB](https://en.wikipedia.org/wiki/MATLAB)/[OCTAVE](https://en.wikipedia.org/wiki/GNU_Octave). The API for these interfaces can be found in the online documentation.[[12]](https://en.wikipedia.org/wiki/OpenCV#cite_note-Cdocs-12) Wrappers in other languages such as [C#](https://en.wikipedia.org/wiki/C_Sharp_(programming_language)), [Perl](https://en.wikipedia.org/wiki/Perl),[[13]](https://en.wikipedia.org/wiki/OpenCV#cite_note-13) [Ch](https://en.wikipedia.org/wiki/Ch_(computer_programming)),[[14]](https://en.wikipedia.org/wiki/OpenCV#cite_note-14) [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)),[[15]](https://en.wikipedia.org/wiki/OpenCV#cite_note-15) and [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)) have been developed to encourage adoption by a wider audience.

Since version 3.4, OpenCV.js is a [JavaScript](https://en.wikipedia.org/wiki/JavaScript) binding for selected subset of OpenCV functions for the web platform. [[16]](https://en.wikipedia.org/wiki/OpenCV#cite_note-16)

All of the new developments and algorithms in OpenCV are now developed in the C++ interface

Now let’s understand the code written in python 5.8.2 using PyCharm IDE

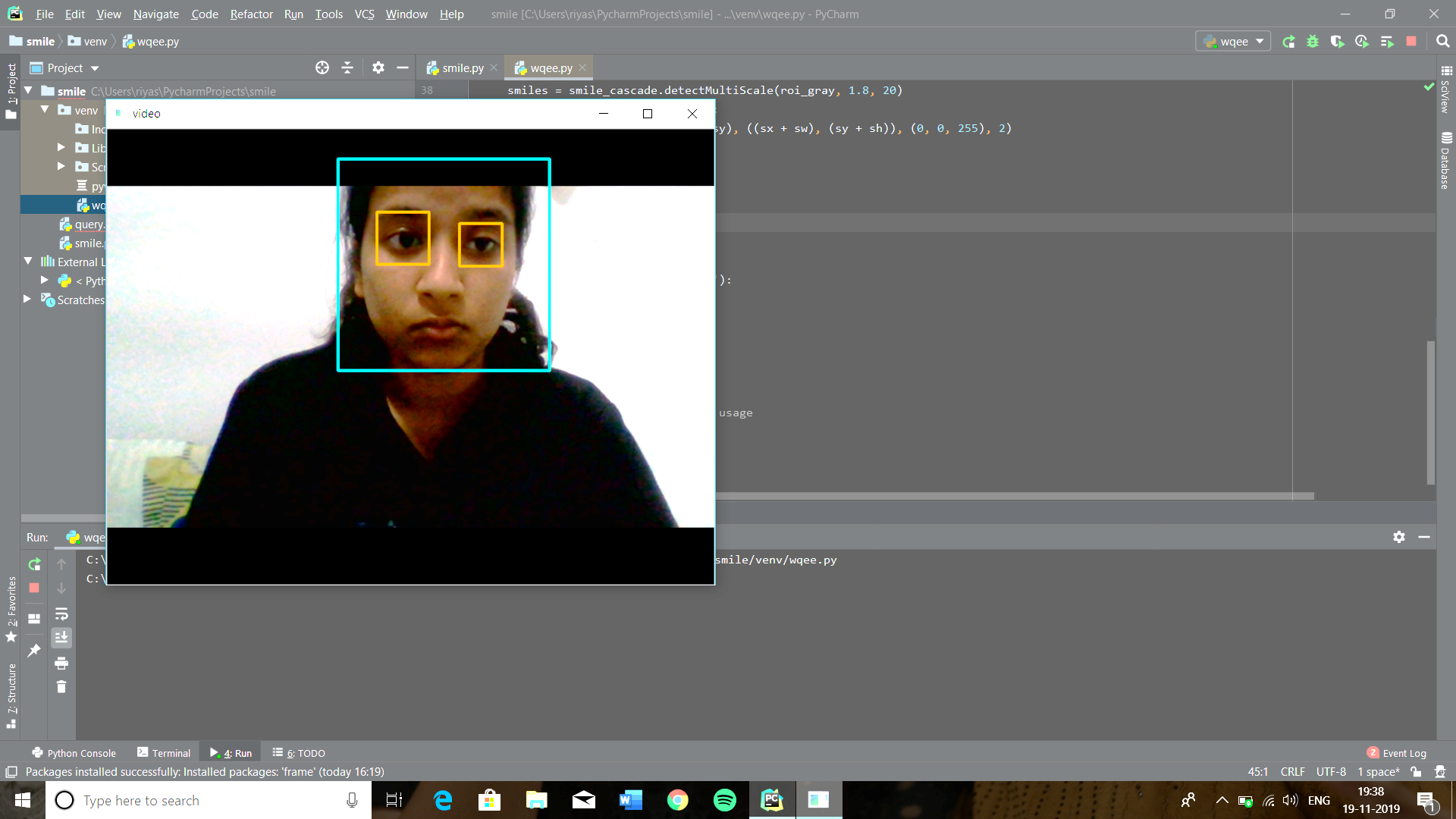
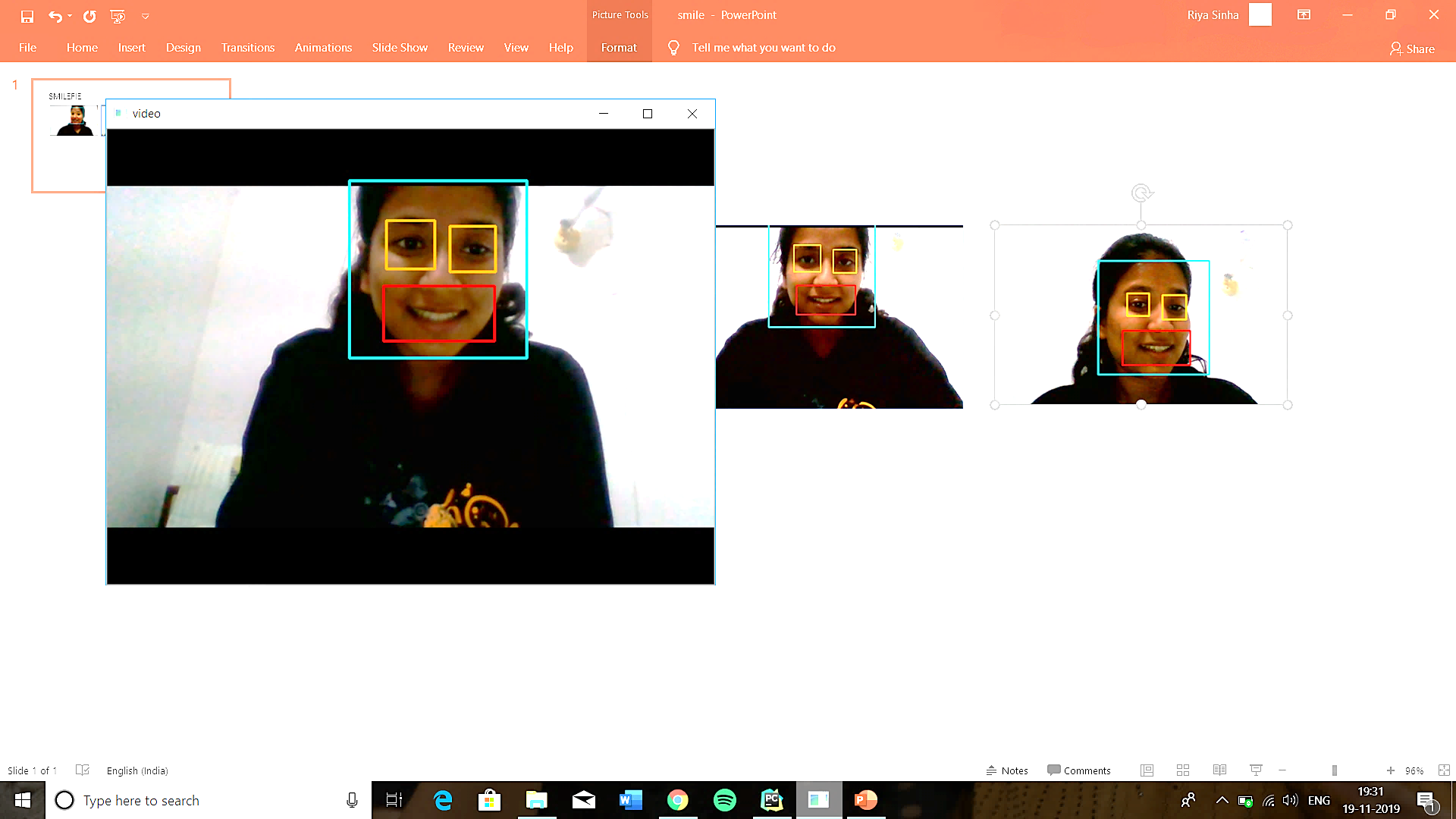
**Step # 1:** First of all, we need to import the OpenCV library

*import cv2*

**Step #2:** Include the desired haar-cascades.

Haar-cascades are classifiers that are used to detect features (of face in this case) by superimposing predefined patterns over face segments and are used as XML files. In our model, we shall use face, eye and smile haar-cascades, which after downloading need to be placed in the working directory

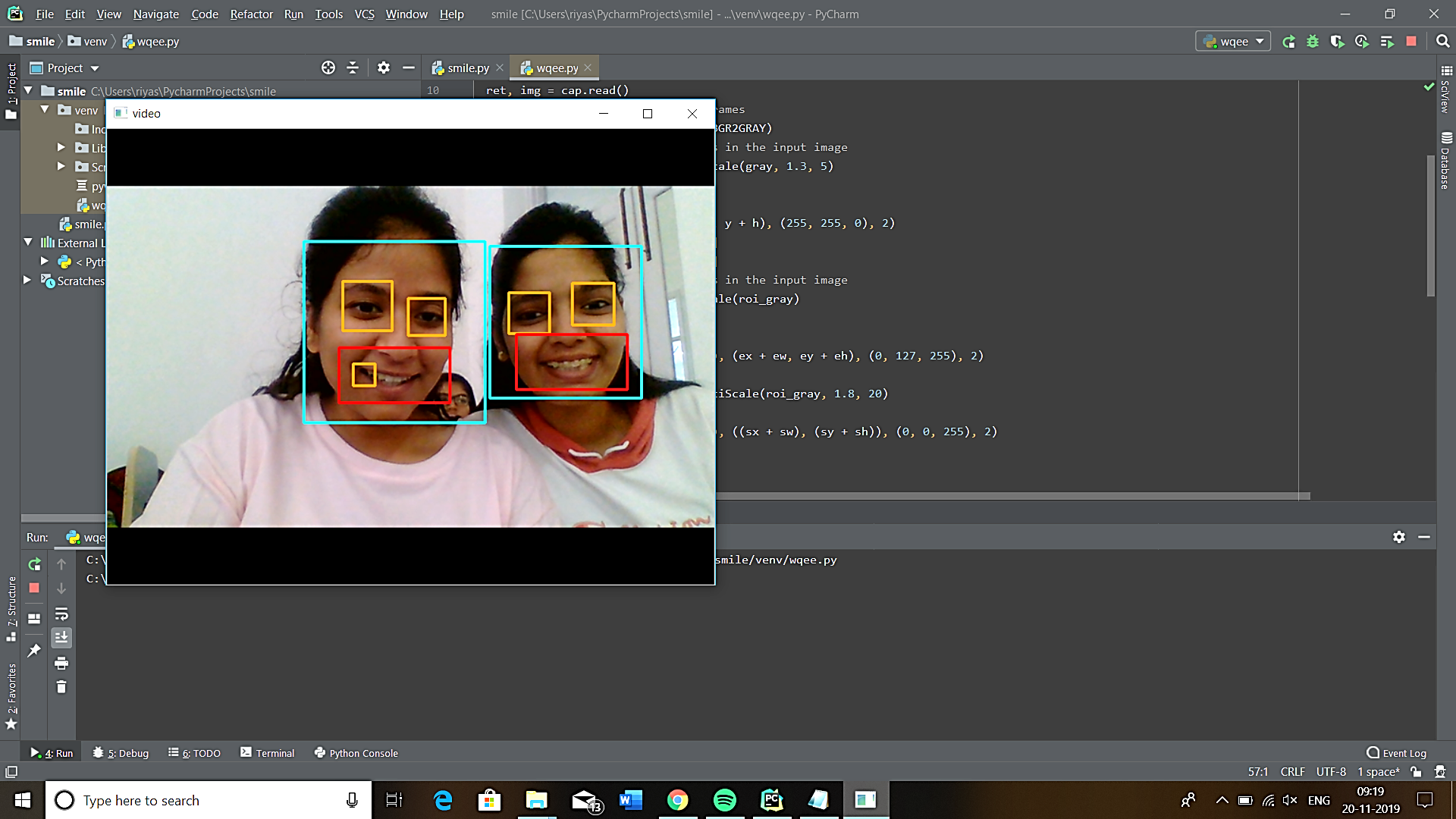
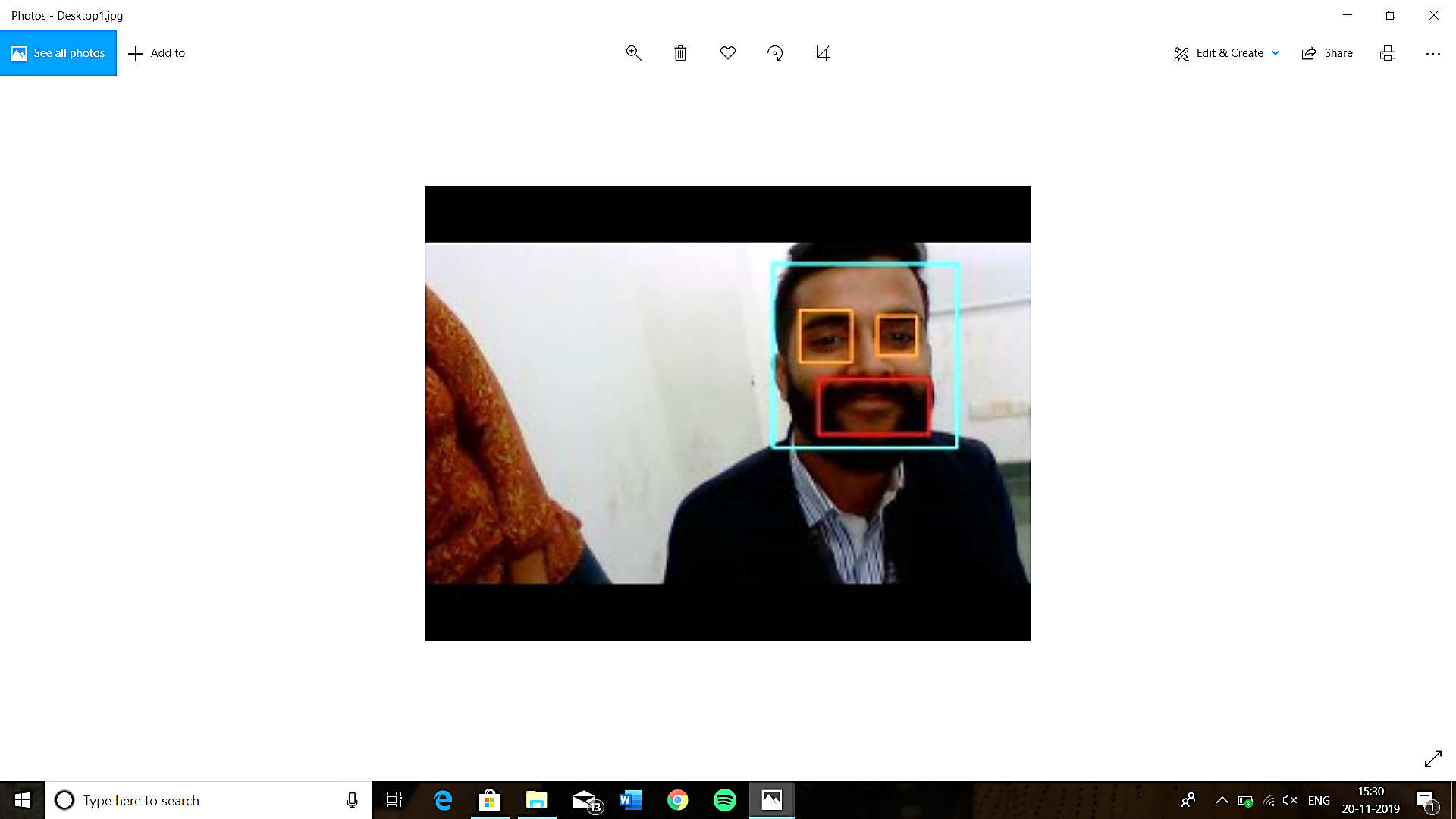
|  |
| --- |
| *face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')*  *eye\_cascade = cv2.CascadeClassifier('haarcascade\_eye.xml')*  *smile\_cascade = cv2.CascadeClassifier('haarcascade\_smile.xml')*  **Step #3:** In this step, we are going to build*main function* which would be performing the smile  detection. The live feed coming from the webcam/video device is processed frame by  frame. We process the gray scale image, as haar-cascades work better on them.  To detect the face, we use:  *faces  = face\_cascade.detectMultiScale(gray, 1.3, 5)*  where 1.3 is the scaling factor, and 5 is the number of nearest neighbours. We can adjust  these factors as per our convenience/results to improve our detector.  Now for each subsequent face detected, we need to check for smiles  *def detect(gray, frame):*  *faces = face\_cascade.detectMultiScale(gray, 1.3, 5)*  *for (x, y, w, h) in faces:*  *cv2.rectangle(frame, (x, y), ((x + w), (y + h)), (255, 0, 0), 2)*  *roi\_gray = gray[y:y + h, x:x + w]*  *roi\_color = frame[y:y + h, x:x + w]*  *smiles = smile\_cascade.detectMultiScale(roi\_gray, 1.8, 20)*    *for (sx, sy, sw, sh) in smiles:*  *cv2.rectangle(roi\_color, (sx, sy), ((sx + sw), (sy + sh)), (0, 0, 255), 2)*  *return frame*  **Explanations –**  The face data is stored as tuples of coordinates. Here, x and y define the coordinate of the  Upper left corner of the face frame, w and h define the width and height of the frame.  The cv2.rectangle function takes in the arguments frame, upper-left coordinates of the face,  lower right coordinates, the RGB code for the rectangle (that would contain within it  the detected face)  and the thickness of the rectangle.The roi\_gray defines the region of interest of the face  and roi\_color does the same for the original frame.  In line 7, we apply smile detection using the cascade.  **Step #4:** We define main function in this step. After execution, the function can be terminated by  pressing the “q” key.  *video\_capture = cv2.VideoCapture(0)*  *while True:*  *# Captures video\_capture frame by frame*  *\_, frame = video\_capture.read()*    *# To capture image in monochrome*  *gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)*    *# calls the detect() function*  *canvas = detect(gray, frame)*    *# Displays the result on camera feed*  *cv2.imshow('Video', canvas)*    *# The control breaks once q key is pressed*  *if cv2.waitKey(1) & 0xff == ord('q'):*  *break*    *# Release the capture once all the processing is done.*  *video\_capture.release()*  *cv2.destroyAllWindows()*    3.2) FULL CODE:  import cv2 as cv  import imutils  from imutils.video import VideoStream  print(cv.\_\_file\_\_)  face\_cascade = cv.CascadeClassifier('C:\\Users\\riyas\\PycharmProjects\\smile\\venv\\Lib  \\site-packages\\cv2\\data\\haarcascade\_frontalface\_default.xml')  eye\_cascade = cv.CascadeClassifier('C:\\Users\\riyas\\PycharmProjects\\smile\\venv\\Lib  \\site-packages\\cv2\\data\\haarcascade\_eye.xml')  smile\_cascade = cv.CascadeClassifier('C:\\Users\\riyas\\PycharmProjects\\smile\\venv\\Lib  \\site-packages\\cv2\\data\\haarcascade\_smile.xml')  #faces = face\_cascade.detectMultiScale(gray, 1.3, 5)  cap = cv.VideoCapture(0)  fourcc = cv.VideoWriter\_fourcc(\*'XVID')  out = cv.VideoWriter('output.avi',fourcc, 20.0, (640,480))  cap1 = 0  while 1:  # reads frames from a camera  ret, img = cap.read()  if ret:  # if video is still left continue creating images  # convert to gray scale of each frames  gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)  # Detects faces of different sizes in the input image  faces = face\_cascade.detectMultiScale(gray, 1.3, 5)  for (x, y, w, h) in faces:  # To draw a rectangle in a face  cv.rectangle(img, (x, y), (x + w, y + h), (255, 255, 0), 2)  roi\_gray = gray[y:y + h, x:x + w]  roi\_color = img[y:y + h, x:x + w]  # Detects eyes of different sizes in the input image  eyes = eye\_cascade.detectMultiScale(roi\_gray)  # To draw a rectangle in eyes  for (ex, ey, ew, eh) in eyes:  cv.rectangle(roi\_color, (ex, ey), (ex + ew, ey + eh), (0, 127, 255), 2)  # to detect smile  smiles = smile\_cascade.detectMultiScale(roi\_gray, 1.8, 20)  for (sx, sy, sw, sh) in smiles:  cap1 = cv.rectangle(roi\_color, (sx, sy), ((sx + sw), (sy + sh)), (0, 0, 255), 2)  # if video is still left continue creating images  for cap1 in range(1,5):  name = 'C:\\Users\\riyas\\Desktop' + str(cap1) + '.jpg'  # writing the extracted images  cv.imwrite(name, img)  print('Creating...' + name)  # increasing counter so that it will  # show how many frames are created  #cap1 += 1  if not condition(item):  cap1 = 0  break  else: break  # Display an image in a window  cv.imshow('video', img)  # Wait for Esc key to stop  if cv.waitKey(30) & 0xff == ord('a'):  break  # Close the window  cap.release()  # De-allocate any associated memory usage  cv.destroyAllWindows()  3.3) OUTPUT: |

****

Fig(v): When subject is not smiling Fig(vi): When subject is smiling the image

the mouth is not detected and image is gets captured automatically and the image

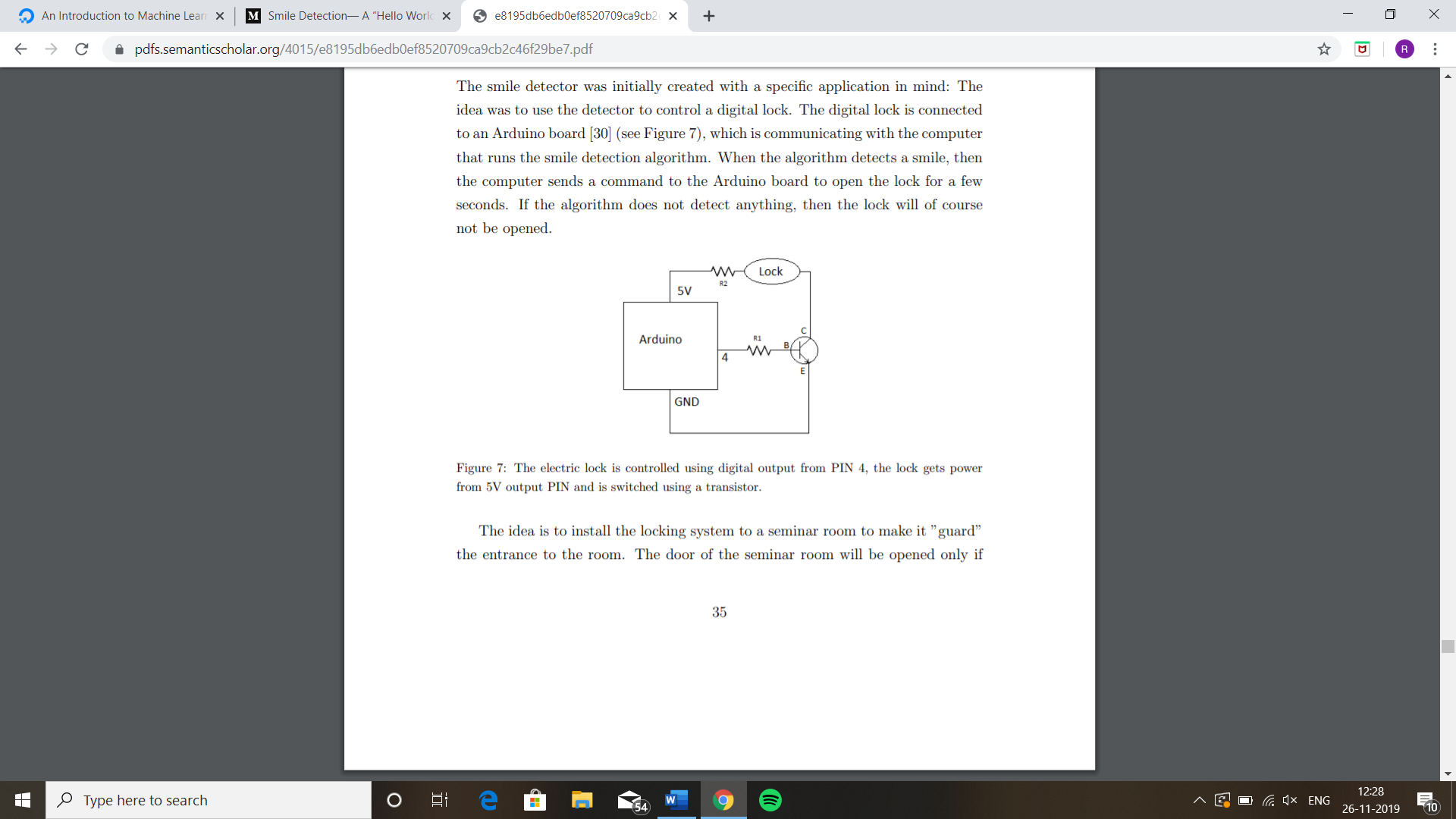
not captured. is saved on the provided location.



CHAPTER: 4 ADVANTAGES OF SMILE DETECTOR

* Smile analysis software provides the capability to detect human faces with a simple camera or webcam and measure a number of features like mood (facial expressions), eye movement and head movement.
* Depending on the quality of the camera (zoom, resolution) human faces can be captured up to 15 meters distance. The Smile Detector detects faces of multiple people looking in the direction of the camera simultaneously. The software can distinguish which people are smiling and when a certain minimum ‘smile-threshold’ is triggered, it activates the camera to take a picture of the particular face. No pictures are stored to avoid privacy issues.
* The Smile Detector illustrates how facial expression analysis solutions can be put to use in a real-life environment. As the software measures mood in real-time, people can see the result of their actions instantaneously (when your smile is big enough, your picture appears on the (big) screen). Face analysis technology hence opens up a whole area of interactive application opportunities with for instance smart TV’s, mobile devices with frontal facing cameras, or robots with a webcam serving elderly people in assisted living environments. Not only smile levels can be used for controls, but also for instance eye movement or head movement.
* **Smile Detector Lock**

The smile detector was initially created with a specific application in mind: The idea was to use the detector to control a digital lock. The digital lock is connected to an Arduino board [30] (see Figure), which is communicating with the computer that runs the smile detection algorithm. When the algorithm detects a smile, then the computer sends a command to the Arduino board to open the lock for a few seconds. If the algorithm does not detect anything, then the lock will of course not be opened. The idea is to install the locking system to a seminar room to make it ”guard” the entrance to the room. The door of the seminar room will be opened only if 35 a smiling person is detected by the smile detection algorithm. The smile detection lock helps to increase the moral of the people by making them smile more when entering the room. The system is very suitable for a seminar room, because a seminar room generally does not need any real security. An interesting and entertaining application is needed instead.

 Fig (vii)

* **Plugin for an Instant Messenger**

People use instant messenger applications like Skype to communicate with friends and colleagues and it is common to use many different smileys. It is possible to apply the smile detection algorithm to automatically enter a smiley whenever a person smiles during an instant messaging conversation. This kind of system can potentially be implemented as a plugin for a messenger application.

* **Capturing a Photograph Automatically**

A common challenge when trying to take a good photo is to capture the moment when a person is smiling. The smile detector can make this process much more convenient, as it is possible to take the photo automatically at the exact moment when a person smiles. There are already cameras with this kind of functionality on the market, but it is likely that the accuracy can be improved.

* **Robot Interaction**

Robots can be successfully used in many different fields of life. It is very likely that in the future they will also be used for tasks that require interaction with people. A large amount of information in a human communication is delivered using speech, but a substantial part is also delivered using facial expressions. Hence it is necessary for a robot to understand them. The smile detection algorithm can be used to obtain a part of this information.

CHAPTER: 5 INDIVIDUAL CONTRIBUTION

Firstly, we learnt python, Machine learning and various libraries that are to be used and then we implemented that libraries into our code consisting face detection and smile detection. First of all we worked on facial detection module and then we moved towards eyes detection and smile detection, where we capture a person’s face image when he/she smile. Using the XML files present in OpenCV we get the data of various mouth aspect ratio (MAR) which helped us in getting a reference to how different people smile. At last we saved the captured image into desired locations (path).

CHAPTER: 6.1 CONCLUSION

IT and control systems manufacturers are seizing the opportunity of having new novel hardware devices as

the “Internet of Things” begins to scale up. As the number of devices continues to increase, more

automation will be required for both the consumer (e.g. home and car) and industrial environments. As

automation increases in IoT control systems, software and hardware vulnerabilities will also increase.

In the near term, data from IoT hardware sensors and devices will be handled by proxy network servers

(such as a cellphone) since current end devices and wearables have little or no built-in security. The

security of that proxy device will be critical if sensor information needs to be safeguarded. The number of

sensors per proxy will eventually become large enough so that it will be inconvenient for users to manage

using one separate app per sensor. This implies single appls with control many “things,” creating a data

management (and vendor collaboration) problem that may be difficult to resolve. An exponentially larger

volume of software will be needed to support the future IoT. The average number of software bugs per line

of code has not changed, which means there will also be an exponentially larger volume of exploitable bugs

for adversaries.

Until there are better standards for privacy protection of personal information and better security

guidelines on communication methods and data/cloud storage, security of wearable and other mobility

devices will remain poor. More work needs to be spent on designing IoT devices before too many devices

are built with default (little or no) security.

Physical security will change as well. As self-healing materials and 3D printers gain use in industry, supply-

chain attacks could introduce malicious effects, especially if new materials and parts are not inspected or

tested before use.

The main benefits of autonomous capabilities in the future IoT is to extend and complement human

performance. Robotic manufacturing and medical nanobots may be useful; however, devices (including

robots) run software created by human. The danger of the increased vulnerabilities is not being addressed

by security workers at the same rate that vendors are devoting time to innovation. Consider how one might

perform security monitoring of thousands of medical nanobots in a human body.

The ability to create secure IoT devices and services depends upon the definition of security standards and

agreements between vendors. ISPs and telecommunication companies will control access to sensor data “in

the cloud” and they cannot provide 100% protection against unauthorized access. IoT user data will be at

risk.

Diversity of the hardware and software in the future IoT provides strong market competition, but this

diversity is also a security issue in that there is no single security architect overseeing the entire “system” of

the IoT. The “mission” of the entire IoT “system” was not pre-defined; it is dynamically defined by the

demand of the consumer and the response of vendors. Little or no governance exists and current standards

are weak. Cooperation and collaboration between vendors is essential for a secure future IoT, and there is

no guarantee of success.

We implemented a smile detection algorithm. It works by locating a face of a person and calculating several of its reference points like the corners of the mouth, corners of the eyes, nostrils and chin. These reference points are tracked across several frames using optical flow. The algorithm detects when a person is smiling based on the motion of the reference points. The algorithm works in real-time speed. The detection rate depends on how to adjust the thresholds, it is possible to detect nearly all smiles, but this also results in a higher false positive rate. Reasonably chosen thresholds give a low misdetection rate while most of the smiles are detected. The algorithm was also used for an interesting application: a smile detector lock. It is a system where our smile detection algorithm controls an electric lock, the lock is opened when a person smile. The application is supposed to be installed to control access to a seminar room, as it does not need any security, but it is good to entertain people who enter the room.

6.2) REFERENCES

[1] R. Gonzales, R. Woods, ”Digital Image Processing”, 3rd Ed, Prentice Hall, 2007.

[2] G. Thomas, M. Weir, J. Hass, ”Thoma’s Calculus,” 12th Ed, Pearson, 2009.

[3] C. Darwin, ”The expression of the emotions in man and animals,” Oxford University Press, 1998. [4] P. Ekman and W. Friesen,”Constants across cultures in the face and emotion,” Journal of personality and social psychology, vol. 17, no. 2, 1971, pp. 124-129.

[5] M. Suwa, N. Sugie and K. Fujimora, ”A preliminary note on pattern recognition of human emotional expression,” International Joint Conference on Pattern Recognition, 1978, pp. 408-410. [6] P. Viola and M. Jones, ”Rapid object detection using a boosted cascade of simple features,” In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, 2001., vol. 1, pp. 511-518.

[7] R. Lienhart and J. Maydt, ”An extended set of haar-like features for rapid object detection,” In Proceedings IEEE Conf. on Image Processing, vol. 1, 2002., pp. 900-903.

[8] <https://medium.com/@reubensinha7/smile-detection-a-hello-world-for-computer-vision-1b70dc65b9dc>

[9] <https://towardsdatascience.com/facial-recognition-happiness-bbb3c4293d1d>

[10] <https://www.digitalocean.com/community/tutorials/an-introduction-to-machine-learning>

[11] <https://pdfs.semanticscholar.org/4015/e8195db6edb0ef8520709ca9cb2c46f29be7.pdf>

CHAPTER: 7 PLAGIARISM REPORT