A

Project Report

on

Recognition of Face Emotion in Real Time using MATLAB

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2022-23

in

Electronics & Communication Engineering

by:

Tanya Varshney (1900290310180)

Samreen Siddiqui (1900290310135)

Satwik Pal (1900290310139)

Amit Kumar Yadav (1900290310019)

Under the supervision of

Dr. Hunny Pahuja

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

May, 2023

DECLARATION

We hereby declare that this submission is our own work and that, to the best of our knowledge and belief, it contains no material previously published or written by another person nor material which to a substantial extent has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Samreen Siddiqui	Tanya Varshney
1900290310135	1900290310180
Date:	Date:
Satwik Pal	Amit Kumar Yadav
1900290310139	1900290310019
Date:	Date:

CERTIFICATE

This is to certify that Project Report entitled **"Recognition of Face Emotion in Real Time using MATLAB"** which is submitted by Tanya Varshney, Samreen Siddiqui, Satwik Pal and Amit Kumar Yadav in partial fulfilment of the requirement for the award of degree B. Tech. in Department of Electronics and Communication Engineering of Dr. A.P.J. Abdul Kalam Technical University, Lucknow is a record of the candidates own work carried out by them under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

Date:

Dr. Hunny Pahuja

Assistant Professor,

ECE Department

ACKNOWLEDGEMENT

It gives us a great sense of pleasure to present the report of the B. Tech Project undertaken during B. Tech. Final Year. We owe special debt of gratitude to **Dr. Hunny Pahuja**, Assistant Professor, Department of Electronics and Communication Engineering, KIET Group of Institutions, Ghaziabad, for his constant support and guidance throughout the course of our work. His sincerity, thoroughness and perseverance has been a constant source of inspiration for us. It is only his cognizant efforts that our endeavours have seen light of the day.

We also take the opportunity to acknowledge the contribution of **Dr. Vibhav Kumar Sachan**, HOD, Electronics and Communication Engineering Department, KIET Group of Institutions, Ghaziabad, for his full support and assistance during the development of the project. We also do not like to miss the opportunity to acknowledge the contribution of all the faculty members of the department for their kind assistance and cooperation during the development of our project. Last but not the least, we acknowledge our friends for their contribution in the completion of the project.

Samreen Siddiqui	Tanya Varshney
1900290310135	1900290310180
Date:	Date:
Satwik Pal	Amit Kumar Yadav
1900290310139	1900290310019
Date:	Date:

ABSTRACT

The difficult topic of automatic face expression recognition has recently attracted a lot of interest because of the numerous applications it offers in a wide range of industries. This study aims to identify the emotion that a person's face communicates from a grayscale image of their face. The percentage of photos correctly labelled for each emotion will serve as our evaluation criterion and show which emotions are more accurately recognized than others. We can help those who require medication by using this technology. In the subject of automation, this technology can be utilized to understand human emotions. With the use of this technology, vehicles may alert the driver if they are feeling tired, track students who are participating in virtual learning, and more.

Real-time facial emotion recognition is a method for determining a person's feelings. The person's image is taken, and it is then compared to the images that are already in our database, to complete the task. This system is based on Support Vector Machines (SVM) and Linear Binary Patterns (LBP). Local Binary Pattern, often known as LBP, is a type of texture descriptor that takes the features from the image and extracts them. By comparing each pixel with its surrounding cells, it provides a local representation of the image and is composed of relative values. It is utilised in a range of recognition software applications, including face, gesture, and object recognition among others. One of the most well-liked supervised learning algorithms, Support Vector Machine or SVM, is used to solve Classification and Regression problems. By creating a hyperplane that divides these data, it organises observations. It is its potential to extract inherent qualities from datasets with greater dimensions. We took references from the Kaggle 2013 FER database to create our own database. Kaggle dataset is made up of 48x48 pixel grayscale portraits of people. We have 28,709 examples for training and 3,589 examples for testing. For creating our own dataset, we took 150 photos for each emotion and trained them further using a training model. Our system is based on a training set and a testing set. To create a training set, an image is fed into the machine, a face is detected, and the image is cropped. Using LBP, its features are extracted and then classified using the SVM algorithm. It is then trained using a learning model and stored on the device. Now for the testing set, whenever an image is taken from a camera, it uses the same steps for extracting its features and classifying it. The results are now obtained by comparing this image to the images kept in our training set.

Keywords:

Facial Expression Recognition (FER), Local Binary Pattern (LBP), Support Vector Machine (SVM)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

The process of identifying human emotions (such as happiness) on the face is known as facial emotion identification. Happiness is the most significant emotion that each person expresses. MATLAB is the best tool for identifying facial expressions of emotion. Face emotion recognition, or FER, is a well-known abbreviation. The quality of human machine connections will increase if computers and other everyday technology devices can accurately read facial expressions. Facial expressions can be changed, thus in order to achieve high accuracy, we will be looking for micro expressions. Micro expressions are ones that only last for a very little period of time. They are undetectable to the observers in the area. This happens every single half-second. The same emotion is consistently shown in micro expressions. However, they are related because they make use of emotions to make someone appear to be lying or concealing something, which reveals their true nature (a feeling). The micro expressions are challenging to identify and comprehend. Learning about micro expressions will help you understand the emotion of recognizing others more deeply. Human faces have traits that enable us to distinguish between various expressions. Human faces have a range of logical features, such as eyes, brows, mouths, noses, and so forth, as you are undoubtedly already aware. These traits are used by the facial expression recognition system to precisely identify the emotion that a person has shown. A statement's emotional impact might be either positive or negative. Finding the right emotion through facial expressions is crucial. Using computer vision and machine learning techniques, the facial recognition system models and categorizes facial traits acquired from images and videos. Face identification algorithms map and extract facial characteristics, then compare them to a database of recognized faces to determine the best match. This technology is applicable in a wide range of circumstances. By identifying and minimizing fake insurance claims, identifying thieves, spotting politically astute arrogances, spotting low-energy drivers, and installing fraud prevention systems, we can assist this technology. We can also use it for public safety purposes, such as checking crime scene footage for criminal intent or looking for terrorist activity in populated regions. It can also be used to track student engagement in online courses, keep tabs on patients' health, and assess consumers' moods.

1.1 Project Description

The latest version of MATLAB, R2022b win64, is what we are utilizing for this. The following list includes the main MATLAB toolboxes for recognizing the emotions seen on people's faces:

Toolboxes for "Deep Learning," "Machine Learning," "Image Processing," and "Image Acquisition." Using computer vision and machine learning techniques, the facial recognition system models and categorizes facial traits acquired from images and videos. Face identification algorithms map and extract facial characteristics, then compare them to a database of recognized faces to determine the best match. This technology is applicable in a wide range of circumstances. By identifying and minimizing fake insurance claims, identifying thieves, spotting politically astute arrogances, spotting low-energy drivers, and installing fraud prevention systems, we can assist this technology. We can also use it for public safety purposes, such as checking crime scene footage for criminal intent or looking for terrorist activity in populated regions. It can also be used to track student engagement in online courses, keep tabs on patients' health, and assess consumers' moods.

1.2 Motivation

The need to protect information or physical property is growing both more crucial and more challenging in today's networked society. Within nation. The number of crimes is rising daily, much like in India. There are no automated procedures in place to monitor a person's activity. Since facial expressions change while engaging in various activities, if we were able to track people's expressions automatically, we could simply track down criminals. In light of this, we made the decision to develop a facial emotion recognition system. After reading a few publications in this field, we became interested in this topic. The papers were published in accordance with how the systems for accurate and reliable face expression recognition systems were created. We are therefore extremely driven to create a system that can track a single person's activity and detect face emotion.

1.3 Problem Statement

Human emotions and intents are expressed through facial expressions, and the key part of the facial expression system is the derivation of an effective and efficient characteristic. Facial recognition is crucial for applications that interpret facial expressions such as real-time animation from live motion images, intelligent visual surveillance, teleconferencing, and intelligent man-machine interaction and communication. Face expressions are helpful for effective communication. The majority of facial expression recognition research and systems only recognize six fundamental expressions (joy, sad, anger, disgust, fear, surprise). Facial expressions are categorised based on facial actions because it is observed that there is not enough space to describe all of them. When it's important to focus on crucial elements like: face configuration, orientation, and place where the face is set, it can be quite challenging to detect faces and identify facial expressions.

1.4 Applications

A. Crime Identification

- a. Spots out thieves
- b. Detects politically aware arrogances
- c. Detects driver’s low energy
- d. Fraud prevention systems
- e. Identifies & decreases forged insurance claims

B. Public Safety

- a. Inspects crime scene footages for crime motives
- b. Smart boundary controls & untruth detectors
- c. Screening of public places to detect terrorism threats

C. Education

- a. Identifies student’s engagement in virtual learning
- b. Models emotional teaching system
- c. Responses based learners learning track
- d. Observes students’ responsiveness

D. Employment

- a. Observes attention & mood swings of employees
- b. Assists to decision making
- c. Spots unresponsive applicants in interviews

E. Medicare

- a. Prevents suicides xx
- b. Patient condition monitoring

c. Aged-people’s depression level identification

d. Mental disorders prediction & assistance

e. Identifies neurodegenerative & autism syndromes

F. Personalized Service Provision

a. Individual reaction prediction in movies, shops & ads

b. Facial expression tracking for marketing motives

c. Analysing customers’ emotions in shopping

d. Personal recommendations in e-commerce

e. Analysing the state of mind to display personalized messages

CHAPTER 2

LITERATURE REVIEW

The difficult subject of automatic person recognition [1] has attracted a lot of interest lately since it has so many uses in so many different industries. Facial recognition is one of those challenging challenges, and as of now, there is no technology that can provide a trustworthy response in every situation. A novel method for recognizing human faces is presented in this research. This research presents a novel face recognition method that combines a SOM-based classifier with features extracted from DCT coefficients. A database of 25 face photographs with five subjects and five images of each subject's face in a variety of emotions was used to test the system in MATLAB. During around 850 epochs of training, the system achieved an 81.36% recognition rate for 10 consecutive trials. When compared to conventional DCT feature extraction methods, the method's computational requirements are drastically lowered by the reduced feature space mentioned for experiment. This method's primary benefits are its high-speed processing capacity and low computing demands, both in terms of speed and memory usage. Our technology is therefore highly adapted for real-time, low-cost hardware implementation. There aren't any commercial applications of this method at the moment. It is feasible, nevertheless, that a useful SOM based face recognition system will someday be developed. The self-organizing map also known as a Kohonen Map is a well-known artificial neural network. It is an unsupervised learning process, which learns the distribution of a set of patterns without any class information. It has the property of topology preservation. There is a competition among the neurons to be activated or fired. The result is that only one neuron that wins the competition is fired and is called the "winner". A SOM network identifies a winning neuron using the same procedure as employed by a competitive layer. However, instead of updating only the winning neuron, all neurons within a certain neighbourhood of the winning neuron are updated using the Kohonen Rule. The Kohonen rule allows the weights of a neuron to learn an input vector, and because of this it is useful in recognition applications. Hence, in this system, a SOM is employed to classify DCT-based vectors into groups to identify if the subject in the input image is "present" or "not present" in the image database. This paper has presented a novel face recognition technique that uses features derived from DCT coefficients, along with a SOM-based classifier. The system was evaluated in MATLAB using an image database of 25 face images, containing five subjects and each subject having 5 images with different facial expressions. A reduced feature space dramatically reduces the computational requirements of the method as compared with standard DCT feature extraction methods. This makes the system well suited for low-cost, real-time hardware implementation.

Commercial implementations of this technique do not currently exist. However, it is conceivable that a practical SOM-based face recognition system may be possible in the future.

We suggested a precise and quick emotion recognition algorithm in this paper [2]. To quickly identify skin colour, colour and feature-based detections were used together with carefully chosen candidate blocks. This paper's main contribution is that the suggested method can identify image edges, and from those edges, distances between different features are determined using Euclidean distance formulae. For each image portraying a distinct emotion, this distance varies. The classification of emotions is based on this gap. The suggested strategy can be used for hardware implementation in further development. The suggested method's straightforward structure makes it appropriate for hardware implementation in order to produce extremely high performance and low power systems. One of the most popular uses of machine vision in recent years is the identification of facial emotions. It can be used for machine human interaction, entertainment, and security (HMI). Emotion recognition frequently makes use of scientifically based image processing, audio processing, gesture signal processing, and physiological signal processing. Automatic facial expression recognition (FER) with a set of specific desired accuracy and performance requirements will help one to create human-like robots and machines that are expected to enjoy truly intelligent and transparent communications with humans. Facial expression recognition deals with the problem of classifying facial images into expression classes. Expression recognition involves a variety of subjects such as perceptual recognition, machine learning, affective computing etc. The classification based facial expression recognition method uses a bank of multilayer perceptron neural networks. Logarithmic Gabor filters are applied to extract the features of the Cohn –Kanade database and the feature size reduction is done by Principal Component Analysis. An input side pruning technique is also incorporated into constructive learning process to reduce the network size without sacrificing the performance of the resulting network. This technique constructs one-hidden-layer feed forward neural network with fewer number of hidden units and weights, while simultaneously providing improved generalization and recognition performance capabilities.

The automatic methods for recognizing face expressions and different research challenges [3] are described. In essence, these systems combine feature extraction, face recognition, and categorization. These processes are included in the architecture of a facial expression recognition system: input, face detection, normalization, face extraction, classification, and output. A higher recognition rate can be achieved using a variety of ways. Techniques with a higher rate of recognition perform better. Due to the confusing physical and psychological characteristics of emotions that are linked to the distinctive characteristics of each individual, emotion identification through facial expression is a problem that affects everyone and poses challenges. Facial expression recognition can be used to identify human emotions. With the aid of the captured series of photos, the proposed system recognized the object. In order to efficiently extract feature points, it recognizes faces in the collected photos. The algorithm successfully assigns each of the photos to one of the six universal emotions. The system's classifier correctly produces the desired results.

We present a system that uses facial expressions [4] to automatically identify human emotions. This research and the procedures employed are more appropriate because they currently employ the most successful facial recognition methods, which decrease response latency. Live streaming, Skin Colour Segmentation, Face Detection, Eye Detection, Lip Detection, Longest Binary Pattern, Bezier Curve Algorithm, Emotion detection, Database, and Output display are the ten fundamental components of emotion recognition that are presented. The Cubic Bezier Curve Implementation, which is more versatile and emerges as the most important in a number of diverse industries, including robotics, computer graphics, automation, and animation, also advances the emotion detection approach. The method recognizes the basic six emotional moods and is useful for faces of different shapes, complexions, and skin tones.

The suggested FEER-HRI system, [5] based on facial expression emotion recognition, is designed as a four-layer system framework. The FEER-HRI technology gives the robots the ability to create facial expressions in order to adapt to human emotions in addition to recognizing human emotions. Robotic facial expressions are exhibited on an LED screen that is built into the robots and are represented by straightforward cartoon symbols that are understandable to humans. The four scenarios in the experiment on human-robot interaction include scene simulation, home service, directing, and entertainment. In these situations, fluid communication is achieved via human facial expression recognition and robot face expression production within two seconds. The FEER-HRI system has potential applications in a variety of areas, including safe driving, smart homes, and home services.

CHAPTER 3

PROPOSED SYSTEM

3.1 Software and Hardware Requirement

3.1.1 Software Requirement

The software requirement for this project is MATLAB latest version i.e., Matlab_R2022b_win64 along with the mentioned major toolboxes: ‘Machine Learning Toolbox’, ‘Image Processing Toolbox', 'Image Acquisition Toolbox’, ‘Deep Learning Toolbox’.

3.1.2 Hardware Requirement

The hardware requirement for this project is a fluent working laptop with a minimum of 4Gb RAM and a web camera.

3.2 Features for Face Emotion Recognition

Expressions are identified by the features placed on the human faces. As you know that, human faces are featured with some sensible features such as eyes, eyebrows, mouth, nose, and so on. These are the features that are helping the face emotion recognition system to exactly identify the emotion expressed by an individual.

A. Mouth: Humans habitually hide other emotions by their fake smiles. This can be identified by an eye’s micro-expression which always shows genuine feelings. They consider some of the positions of the feature to exactly identify the state of feeling as mentioned below:

a. Raised mouth’s single side – disgust or dislike

b. Downed mouth curves – depressed & unhappiness

c. Upturned mouth corners – happy

d. Dilated mouth – anxiety & fear

e. Plunged jawlines – surprise & wonder

f. Casing mouth – concealing feelings

g. Biting lips – nervousness

h. Tightened lips – dislike & distaste

B. Eyebrows: Eyebrows of the human can reveal the personal emotions of the intended person. In addition, they are very important for emotion recognition. The state of eyebrows positions can be considered as mentioned below:

a. Drained inner bends – sadness

b. Raised curves – wonder

c. Joined & dropped curves – panic, anger, trouble

C. Eyes: Eyes are the illustrating feature of human beings who can see the state of others' feelings. Further, actions of the eyes can be well thought as:

a. Strong eyeing – thoughtfulness/hatred

b. Staring away – worry/diversion

c. Quick blinks – sorrow / distress

d. Short blinks – controlling eyes

e. Opened eyes – excitement / curiosity

The above listed are the major features that can convey the expressions of humans.

Table 3.2.1: Association between Facial Expressions and Emotions

Emotion	Happy	Anger	Surprise	Disgust	Sadness	Fear
Increase the Likelihood	Smile	Brow Furrow Lid Tight Eye Wide Chin Raise	Inner Brow Raise Brow Eye Furrow Jaw Drop	Nose Wrinkling Upper Lip Raise	Inner Brow Raise Brow Furrow Lip Corner Depressed	Inner Brow Raise Brow Furrow Eye Wide Lips Stretch
Decrease the Likelihood	Brow Furrow Brow Raise	Raise Brow Raise Smile Inner Brow	Brow Furrow	Smile Lips Stuck	Brow Raise Eye Wide Mouth Open Lip Press Lip Stuck Smile	Brow Raise Lip Corner Depressed Jaw Drop Smile

3.3 Data Collection

The data set, FER 2013, was downloaded from Kaggle. It is made up of 48x48 pixel grayscale portraits of people. The faces have been automatically registered so that each face nearly occupies the same amount of space in each image and is approximately in the centre. Each face is assigned to one of seven categories based on the emotion it conveys (0 = angry, 1 = disgust, 2 = fear, 3 = happy, 4 = sad, 5 = surprised, 6 = neutral). In training set, we have 28,709 examples and for testing set we have 3,589 examples.

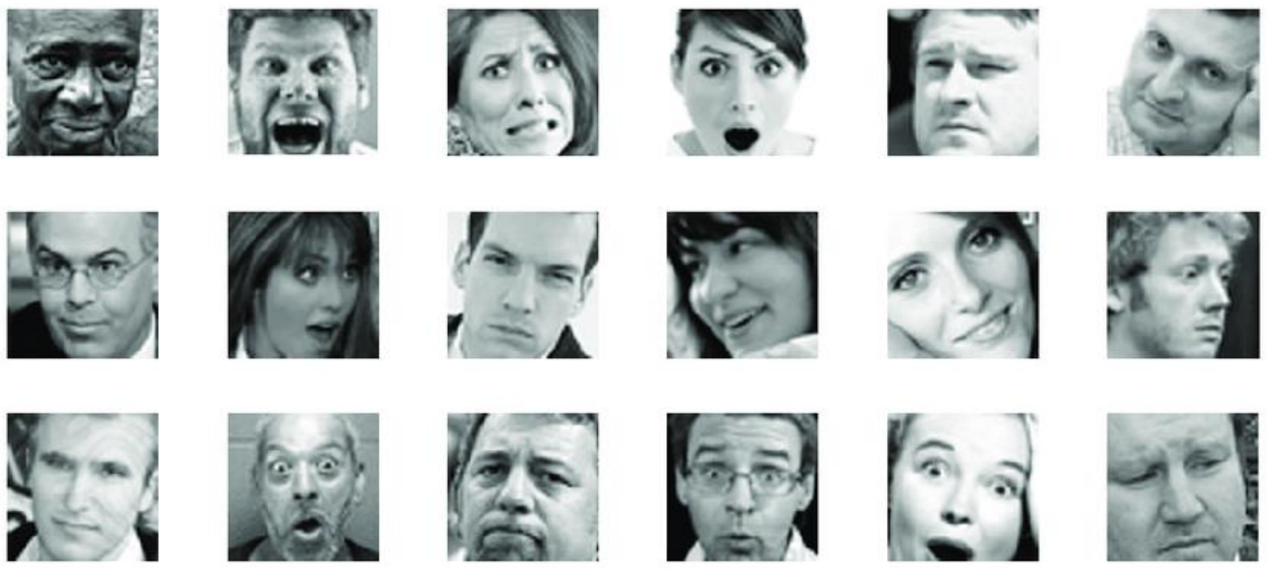


Figure 3.3.1: Images contained in FER-2013 Dataset

Taking an example from this dataset, we created our own dataset with 150 snaps for each emotion. Further training and testing of the model are done on our own newly created dataset.

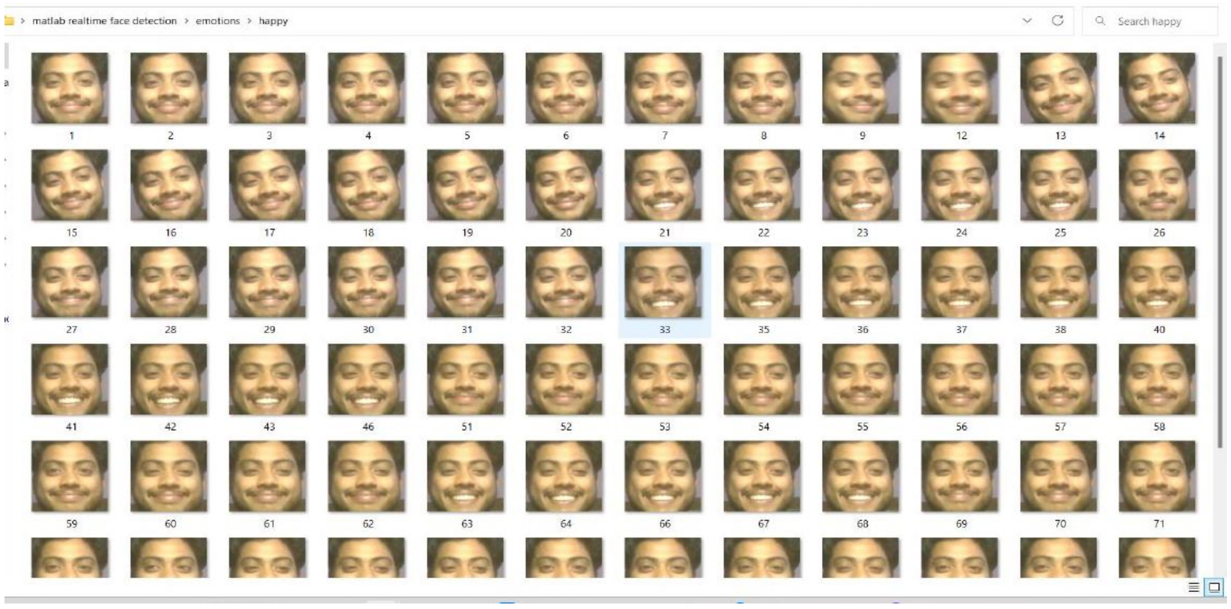


Figure 3.3.2: Images in our Dataset for Happy Emotion

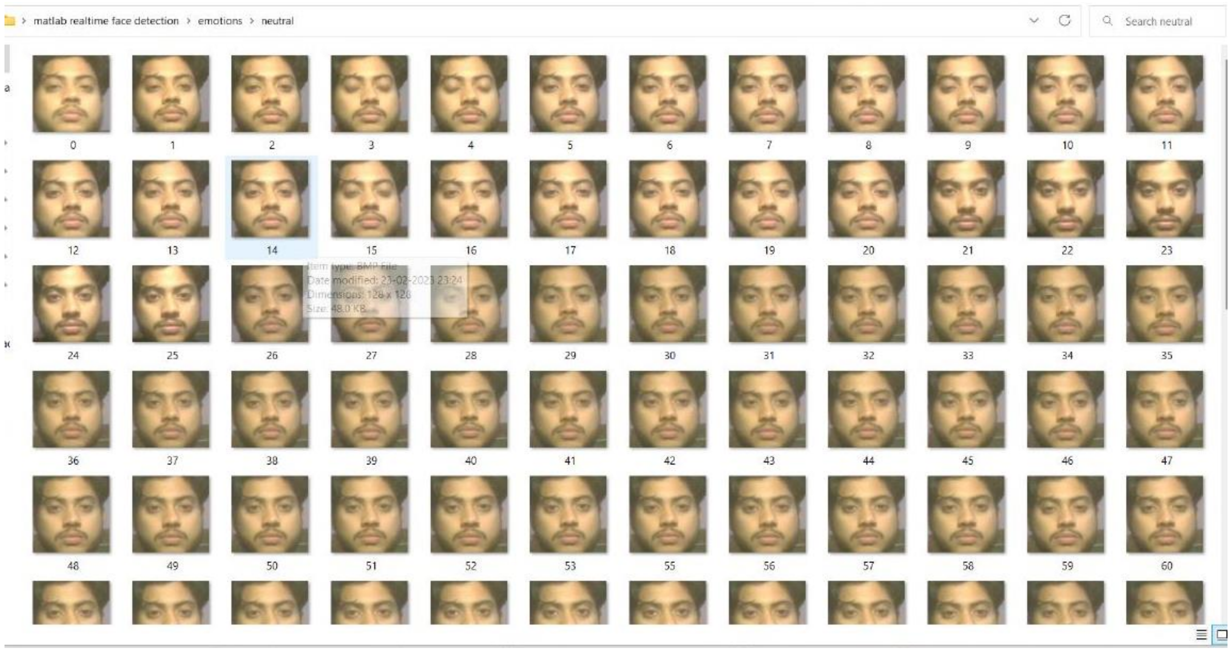


Figure 3.3.3: Images in our Dataset for Neutral Emotion

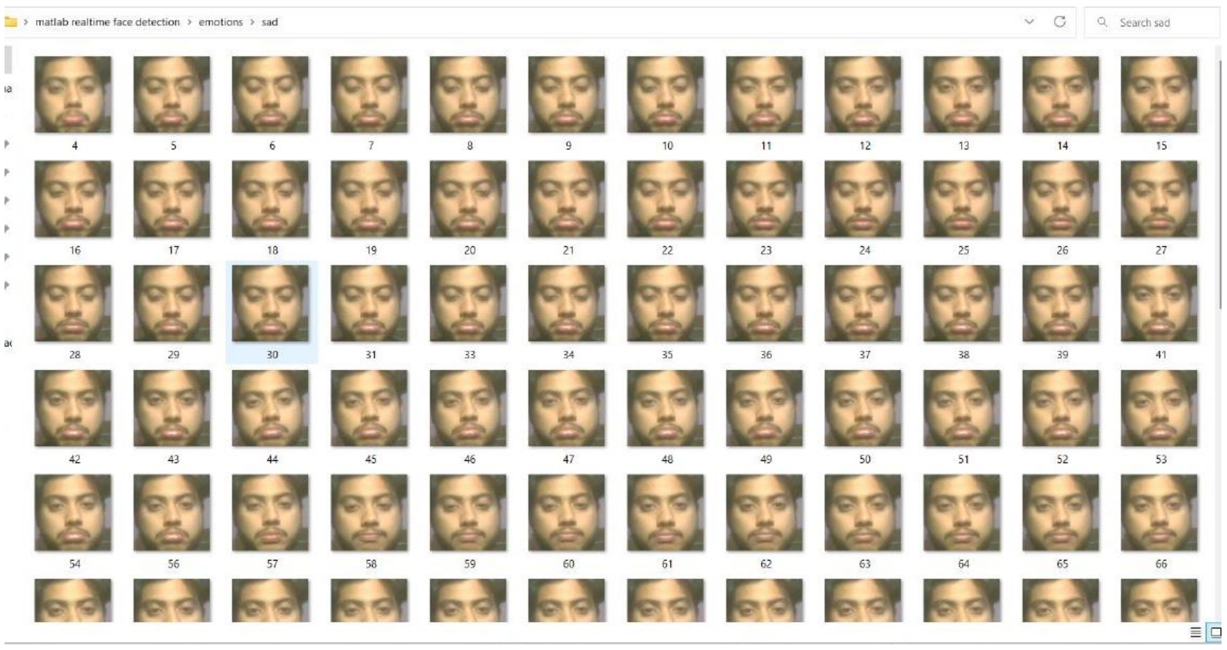


Figure 3.3.4: Images in our Dataset for Sad Emotion

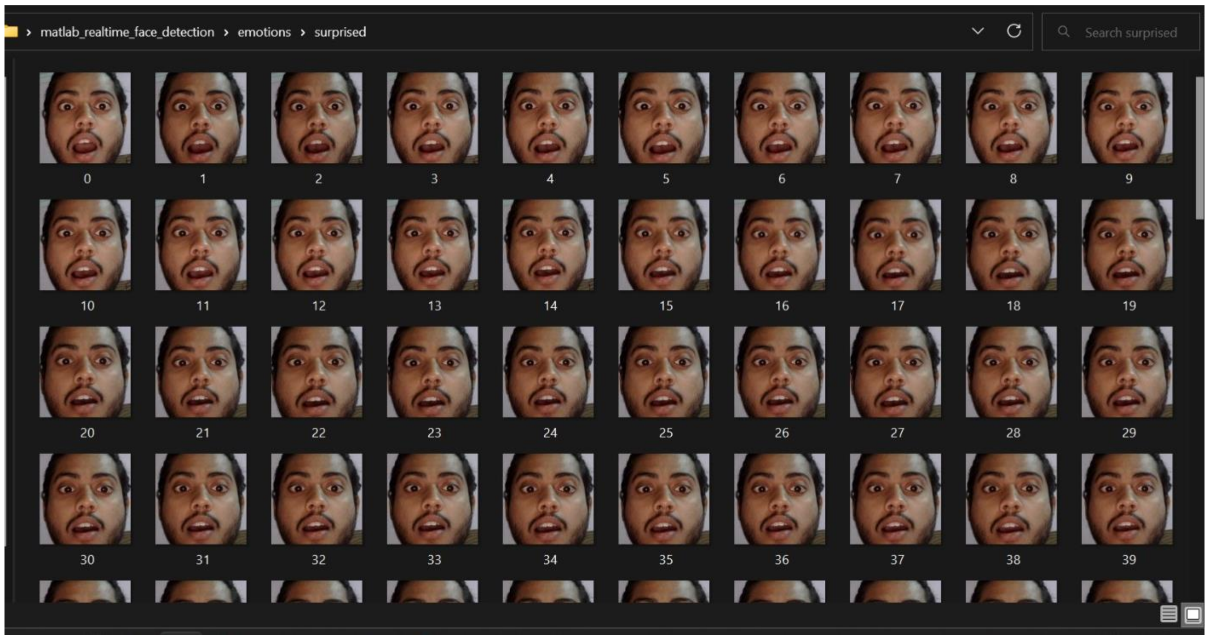


Figure 3.3.5: Images in our Dataset for Surprise Emotion

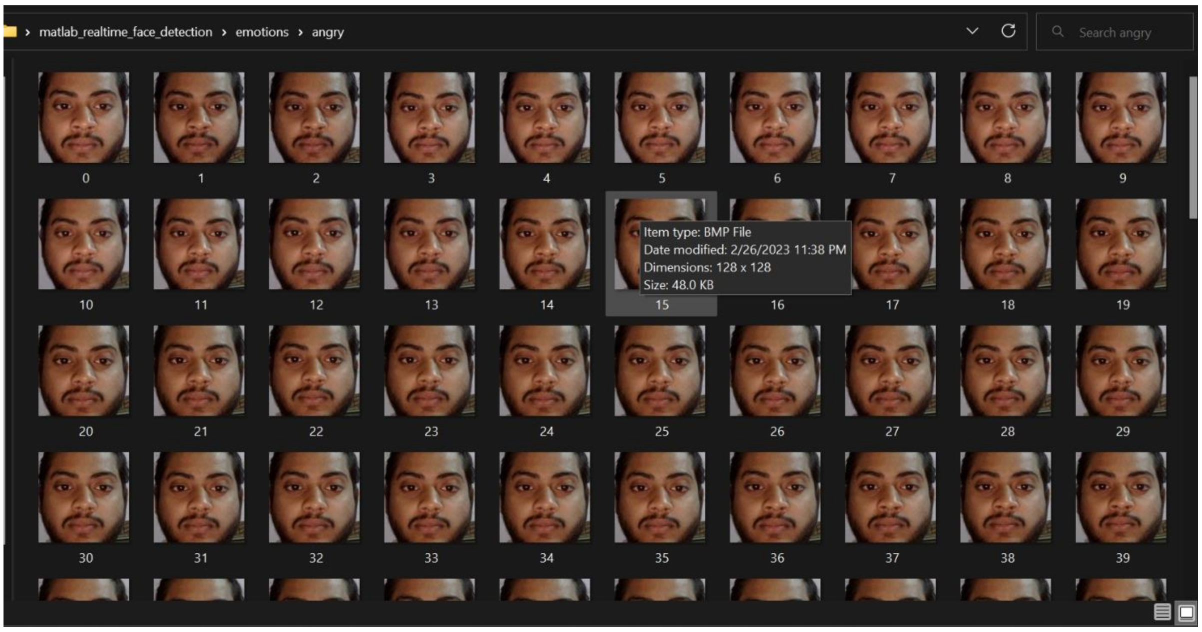

matlab_realtime_face_detection › emotions › angry
Search angry
Item type: BMP File
Date modified: 2/26/2023 11:38 PM
Dimensions: 128 x 128
Size: 48.0 KB

Figure 3.3.6: Images in our Dataset for Angry Emotion

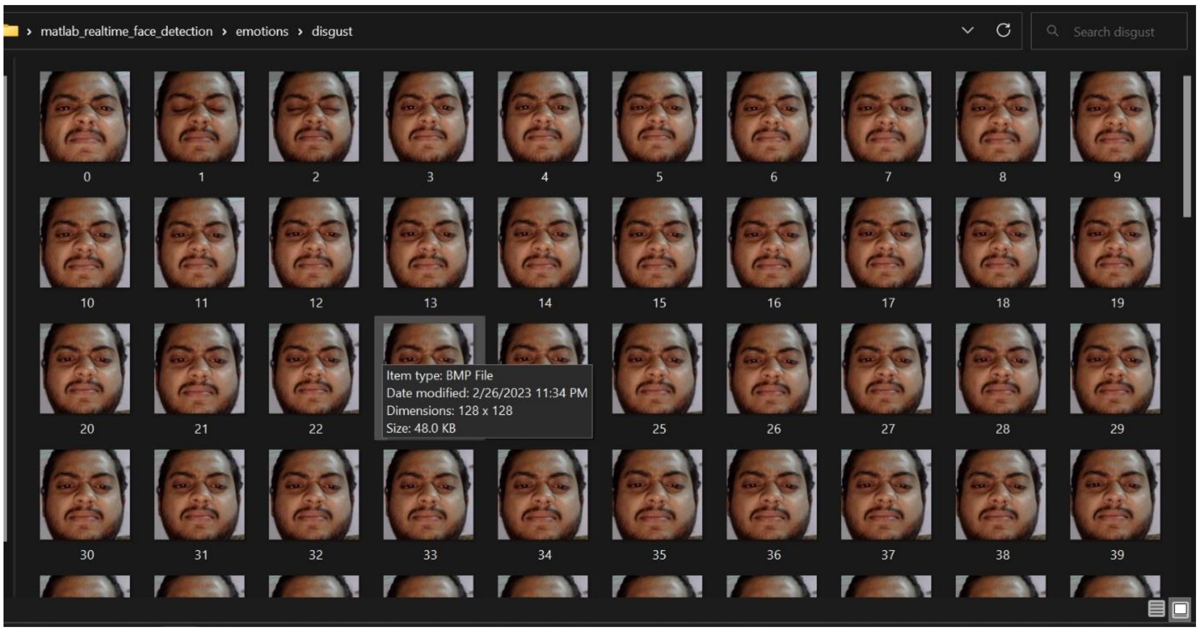
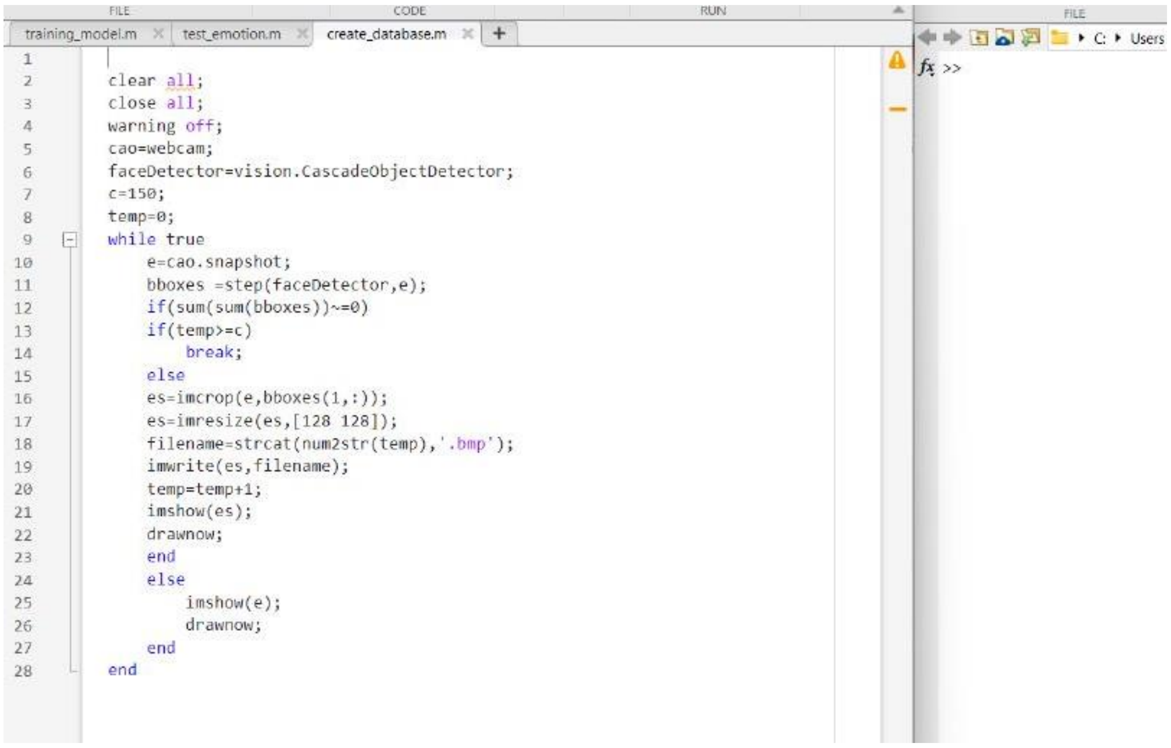

matlab_realtime_face_detection › emotions › disgust
Search disgust
Item type: BMP File
Date modified: 2/26/2023 11:34 PM
Dimensions: 128 x 128
Size: 48.0 KB

Figure 3.3.7: Images in our Dataset for Disgust Emotion

3.4 Database Creation

We need a database from which we can extract the features using local binary pattern. Database creation means taking snaps for each defined emotion (happy, sad, fear, angry, disgust, surprised and neutral). Further, this database is used to train the model. For this, we have taken 150 shots for each defined emotion, making a total of 1050 images for training. The required code is mentioned below:

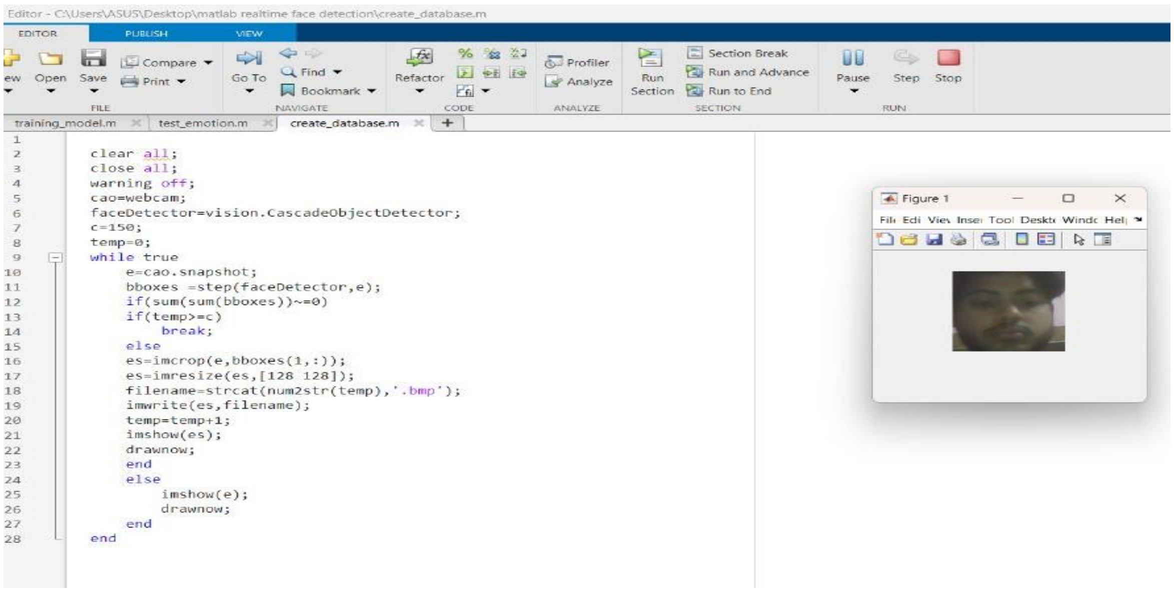
Command Window:



```
clear all;
close all;
warning off;
cao=webcam;
faceDetector=vision.CascadeObjectDetector;
c=150;
temp=0;
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(sum(sum(bboxes))~=0)
    if(temp>=c)
        break;
    else
    es=imcrop(e,bboxes(1,:));
    es=imresize(es,[128 128]);
    filename=strcat(num2str(temp),'.bmp');
    imwrite(es,filename);
    temp=temp+1;
    imshow(es);
    drawnow;
    end
    else
        imshow(e);
        drawnow;
    end
end
```

Figure 3.4.1: Database Creation Code

Result:



```
clear all;
close all;
warning off;
cao=webcam;
faceDetector=vision.CascadeObjectDetector;
c=150;
temp=0;
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(sum(sum(bboxes))~=0)
    if(temp>=c)
        break;
    else
    es=imcrop(e,bboxes(1,:));
    es=imresize(es,[128 128]);
    filename=strcat(num2str(temp),'.bmp');
    imwrite(es,filename);
    temp=temp+1;
    imshow(es);
    drawnow;
    end
    else
        imshow(e);
        drawnow;
    end
end
```

Figure 3.4.2: Execution of Database Creation

3.5 Local Binary Pattern

The LBP examines how a pixel and its neighbours are related, encoding this relationship into a binary word. Being resistant to changes in contrast, this enables the detection of patterns and features. It is a method for extracting features. The original LBP operator uses decimal numbers to represent the pixels in an image. These numbers are referred to as LBPs or LBP codes because they encode the local structure surrounding each pixel. By removing the value of the centre pixel, every pixel in a 3 x 3 neighbourhood is compared to its eight neighbours. Negative values are stored as 0 and the other values as 1 in the output. A binary number is created for each given pixel by adding up each of these binary values, beginning with the one of the pixel's top-left neighbour. The specified pixel is then given a label using the corresponding decimal value of the created binary number. The terms "LBPs" or "LBP codes" refer to the resulting binary numbers. [13]

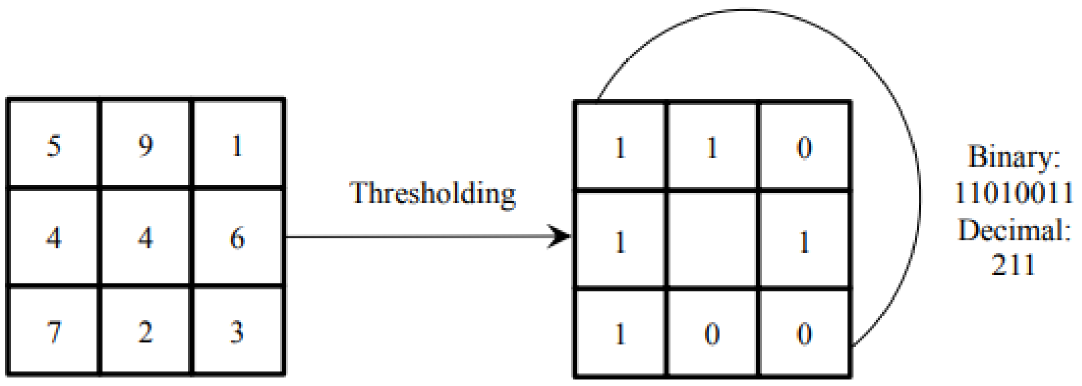


Figure 3.5.1: Basic Local Binary Pattern Operator

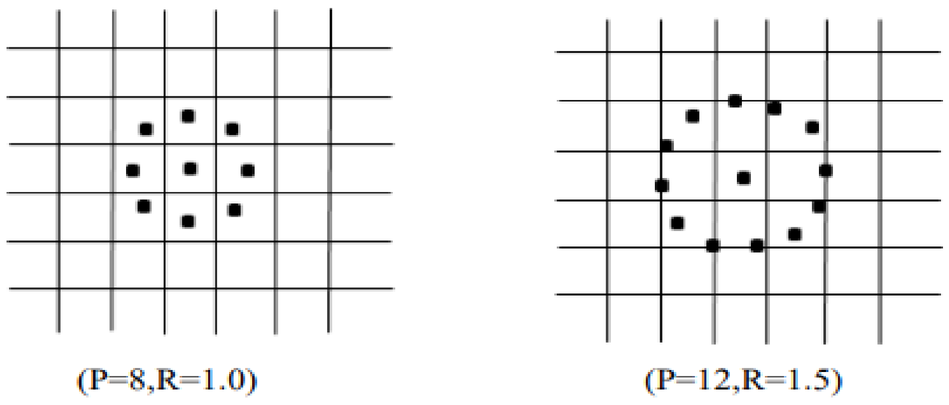


Figure 3.5.2: Example for extended Local Binary Pattern

The limitation of the basic LBP operator is that its small 3×3 neighbourhood cannot capture the dominant features with large scale structures. As a result, to deal with the texture at different scales, the operator was later extended to use neighbourhoods of different sizes. Using circular neighbourhoods and bilinearly interpolating the pixel values allow any radius and number of pixel in the neighborhood. Examples of the extended LBP are shown above (Figure 3.5.2), where (P, R) denotes sampling points on a circle of radius of R. Further extension of LBP is to user uniform patterns. A LBP is called uniform if it contains at most two bitwise transitions from 0 to 1 or vice versa when the binary string is considered circular. E.g.00000000, 001110000 and 11100001 are uniform patterns. A histogram of a labelled image f1(x, y) can be defined as [13]

$$Hi = \sum x, y\, I(f1(x, y) = i), i = 0, \ldots, n - 1$$

(1)

Where n is the number of different labels produced by the LBP operator and

$$I(A) = \begin{cases} 1\; A\; is\; true \\ 0\; A\; is\; false \end{cases}$$

(2)

This histogram contains information about the distribution of the local micro-patterns, such as edges, spots and flat areas, over the whole image. For efficient face representation, feature extracted should retain also spatial information. Hence, face image is divided into m small regions R0, R1, ..., Rm and a spatially enhanced histogram is defined as

$$Hi = \sum x, y\, I(f1(x, y) = i) I(x, y) \in Rj$$

(3)

3.6 Support Vector Machines

The dimensionality of data obtained from the feature extraction method is very high so it is reduced using classification. Features should take different values for object belonging to different class so classification will be done using Support Vector Machine algorithm.

SVM is widely applied in a range of pattern recognition tasks. Modern statistical learning theories serve as the basis of SVM, a cutting-edge machine learning methodology. SVM is capable of almost perfect class separation. SVMs are trained to classify facial expressions based on the suggested attributes. SVMs are the maximal hyperplane classification technique that, in general, promises strong generalisation performance because they are based on findings from statistical learning theory. [13]

Kernel functions are used to quickly transform input data that might not be linearly separable into a high dimensional feature space where linear methods can subsequently be used. SVMs are particularly suited to a dynamic, interactive approach to expression recognition since they display strong classification accuracy even with a limited quantity of training data available. [14]

When the hyper plane and the training data of any class are the largest, an optimal separation is attained. The decision surface is this dividing hyper plane. Many classification tasks, including text categorization, genetic analysis, and face detection, have been effectively completed using SVM. [15]

Given a training set of labelled samples:

$$D = \{(xi, yi) | xi \in Rn, yi \in \{-1,1\}\}_{i=1}^{p} \tag{1}$$

A SVM tries to find a hyperplane to distinguish the samples with the smallest errors.

$$w.x - b = 0 \tag{2}$$

For an input vector xi, the classification is achieved by computing the distance from the input vector to the hyperplane. The original SVM is a binary classifier.

CHAPTER 4

METHODOLOGY

4.1 System Design Steps

Step 1: Image selection

(Reads the given image input)

Step 2: Adding selected images into the database

(Stores the selected images in the database and trains them)

Step 3: Face expression recognition

(The selected image is processed to recognize the expression from the database)

Step 4: Database information

(Illuminates the complete details of the database and compares it with the detected expression)

Step 5: Removal of database

(Terminates the prevailing database from the directory)

Step 6: Program information

(Showcases the information regarding software)

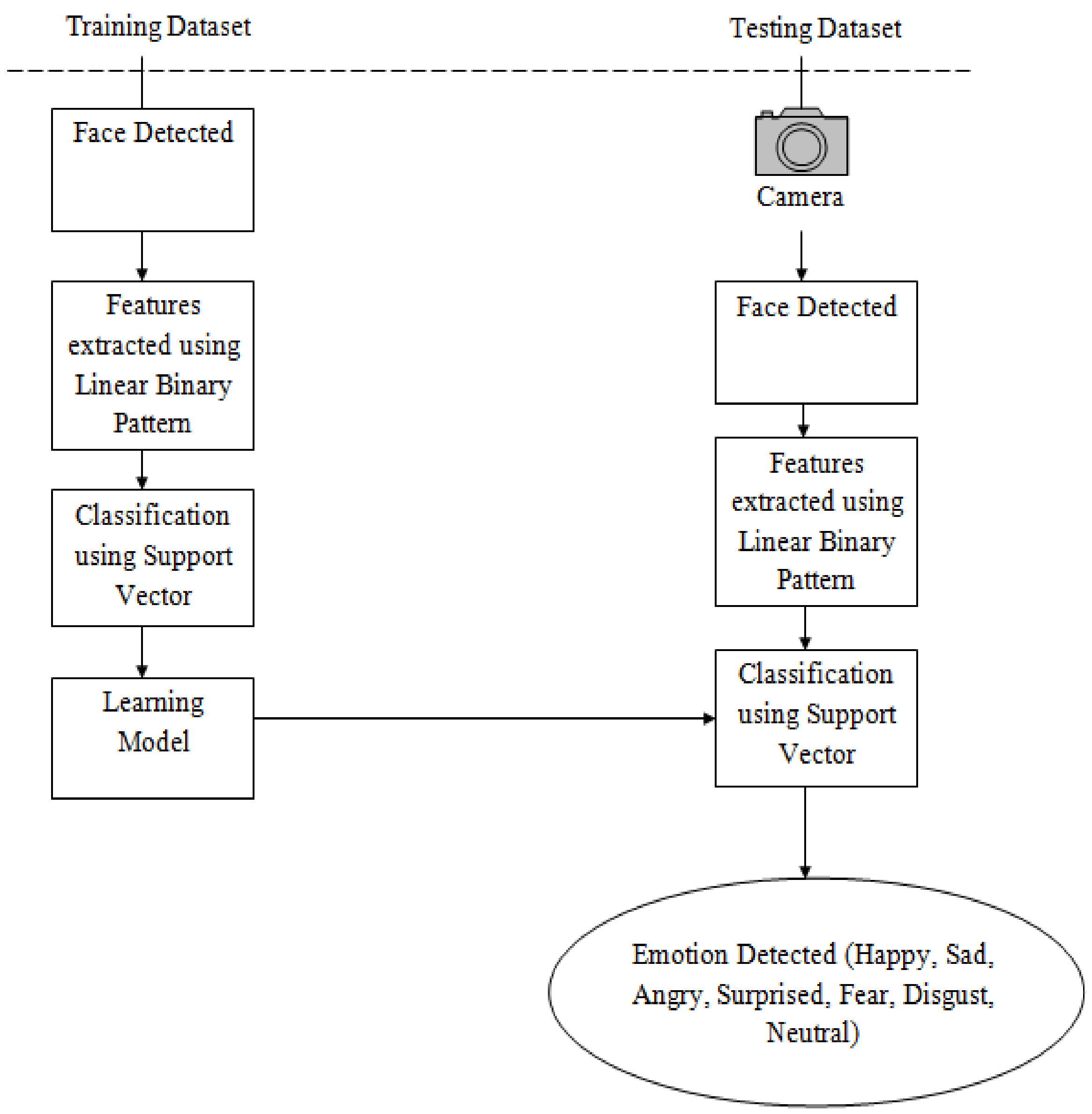
Step 7: Source code of the process

(Emphasizes the source code to be used)

Step 8: Exit

(Exiting from program; quit)

4.2 System Design



Training Dataset
Testing Dataset
Face Detected
Camera
Features extracted using Linear Binary Pattern
Face Detected
Classification using Support Vector
Features extracted using Linear Binary Pattern
Learning Model
Classification using Support Vector
Emotion Detected (Happy, Sad, Angry, Surprised, Fear, Disgust, Neutral)

Figure 4.2.1: System Design

4.3 System Flowchart

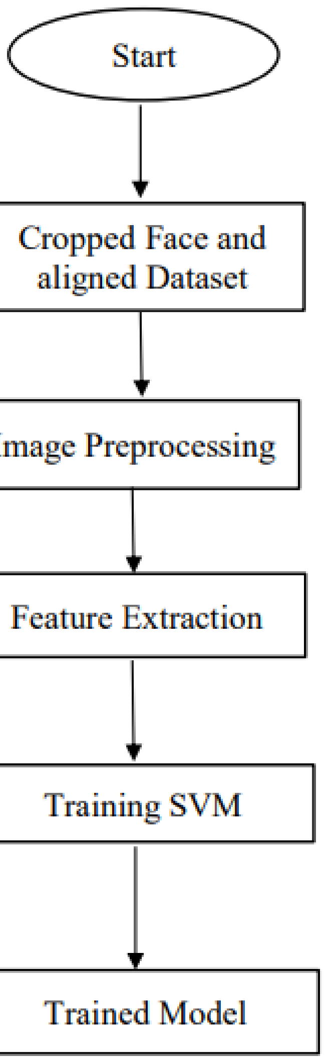


Figure 4.3.1: Flowchart for Training Images

For training the images, we require to detect the face and crop it. The features are then extracted using Linear Binary Pattern (LBP) and classified using Support vector Machine (SVM).

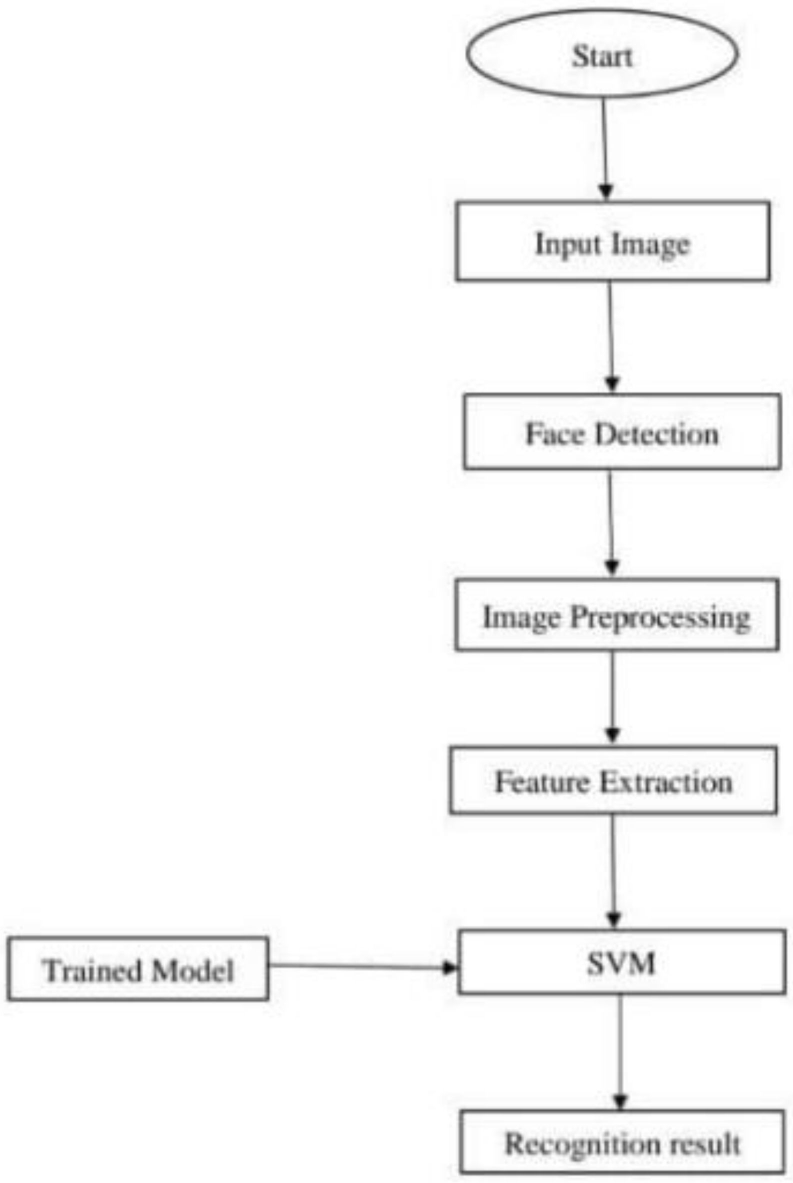


Figure 4.3.2: Flowchart of Testing

4.4 Pre-processing Steps

4.4.1 To Read an Image

Command: imread()

Syntax: B= imread(filename)

B= imread(filename,fmt)

B= imread(__ ,idx)

B= imread(__, Name, Value)

[B, map] = imread(__)

[B, map, transparency] = imread(__)

Description:

B = imread(filename) reads the image from the given file while suggesting the format of the file from its contents. If filename contains several images, imread scans the very first image in the file.

B = imread(filename,fmt) The additional information provided by it describes the format of the file with the default file extension supplied by fmt. If the filename supplied by filename cannot be located, imread searches for a file called filename.fmt.

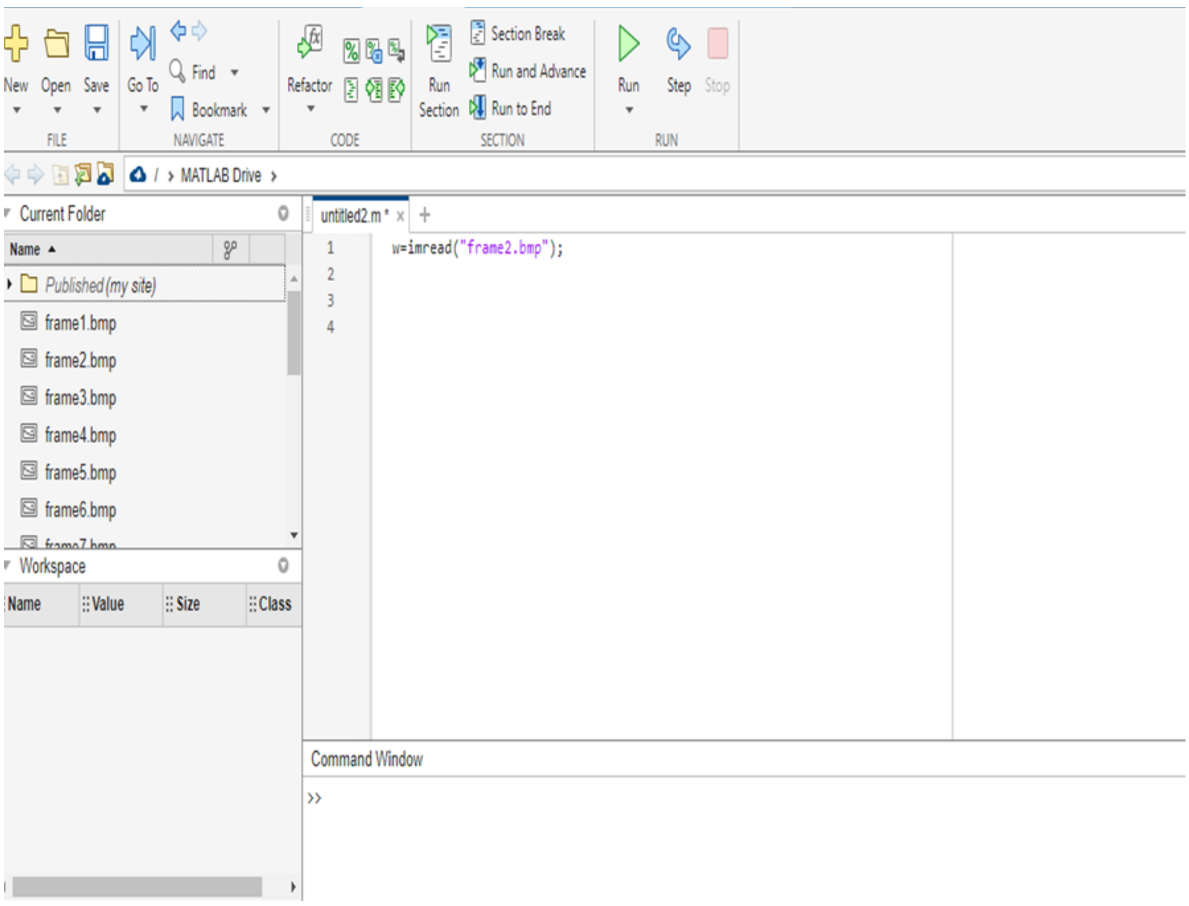
B = imread(__,idx) opens a multi-image file and reads the supplied image or pictures. Just the following file types support this syntax: GIF, PGM, PBM, PPM, CUR, ICO, TIF, SVS, and HDF4. Both the filename input and the fmt specification are required.

B = imread(__, Name, Value) allows the specification of format-specific parameters using one or even more name-value pair arguments in addition to any input arguments from the preceding syntaxes.

[B, map] = imread(__) reads the appropriate colormap for the indexed picture in filename into map and writes it to B. The colormap values are automatically scaled into the [0,1] range in the image file.

[B, map, transparency] = imread(__) returns the transparency of the image in addition. Only PNG, CUR, and ICO files are compatible with this syntax. Transparency in PNG files is represented via the alpha channel, if one is available. The AND (opacity) mask is what it is for CUR and ICO files.

Command Window:



```
w=imread("frame2.bmp");
```

```
>>
```

Figure 4.4.1.1 imread() Command

4.4.2 To Write an Image

Command: imwrite( )

Syntax: imwrite(B, filename)

imwrite(B, map, filename)

imwrite(__, fmt)

imwrite(__, Name, Value)

Description:

imwrite(B, filename) determines the file format based on the extension and writes the image data in A to the file identified by filename. The new file is created by imwrite in the current folder.

imwrite(B, map, filename) creates a file with the name filename, which contains the indexed image in A and its associated colormap map.

imwrite(__,fmt) writes the picture in the format indicated by fmt, regardless of the file extension in filename. Any of the syntaxes allow you to specify fmt after the input arguments.

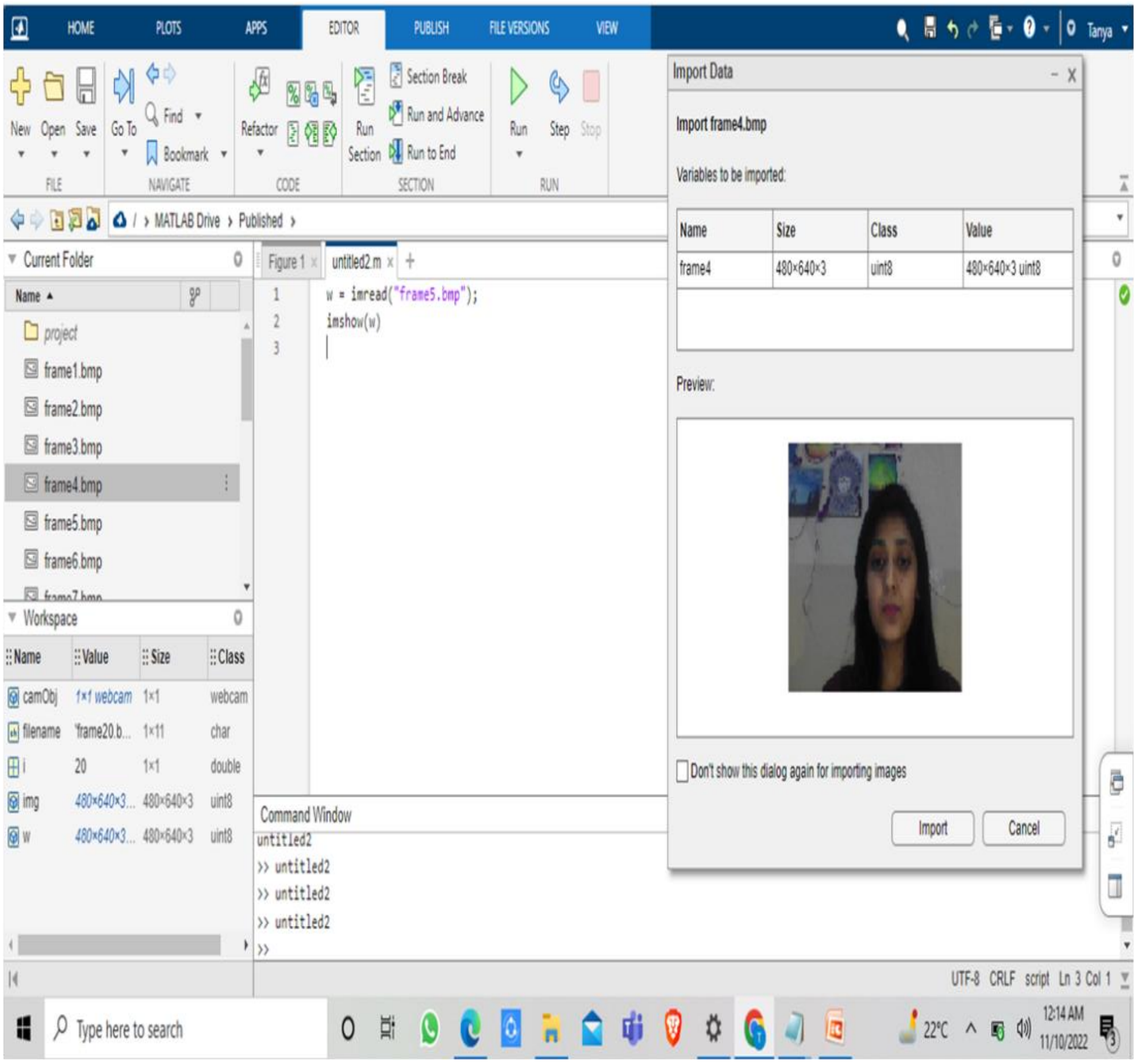
imwrite(__, Name, Value) provides one or more name-value arguments to provide additional features for output GIF, HDF, JPEG, PBM, PGM, PNG, PPM, and TIFF files.

4.4.3 To Display an Image

Command: imshow

imshow is a function that can be used to display picture data. The image is displayed in a figure window using the imshow function after being read into the workspace. (imread command does not store any image whereas imshow command diaplays the image).

Command Window:



```
w = imread("frame5.bmp");
imshow(w)
```

Command Window

```
untitled2
>> untitled2
>> untitled2
>> untitled2
>>
```

Workspace:

Name	Value	Size	Class
camObj	1×1 webcam	1×1	webcam
filename	'frame20.b...	1×11	char
i	20	1×1	double
img	480×640×3...	480×640×3	uint8
w	480×640×3...	480×640×3	uint8

Import frame4.bmp

Variables to be imported:

Name	Size	Class	Value
frame4	480×640×3	uint8	480×640×3 uint8

Don't show this dialog again for importing images

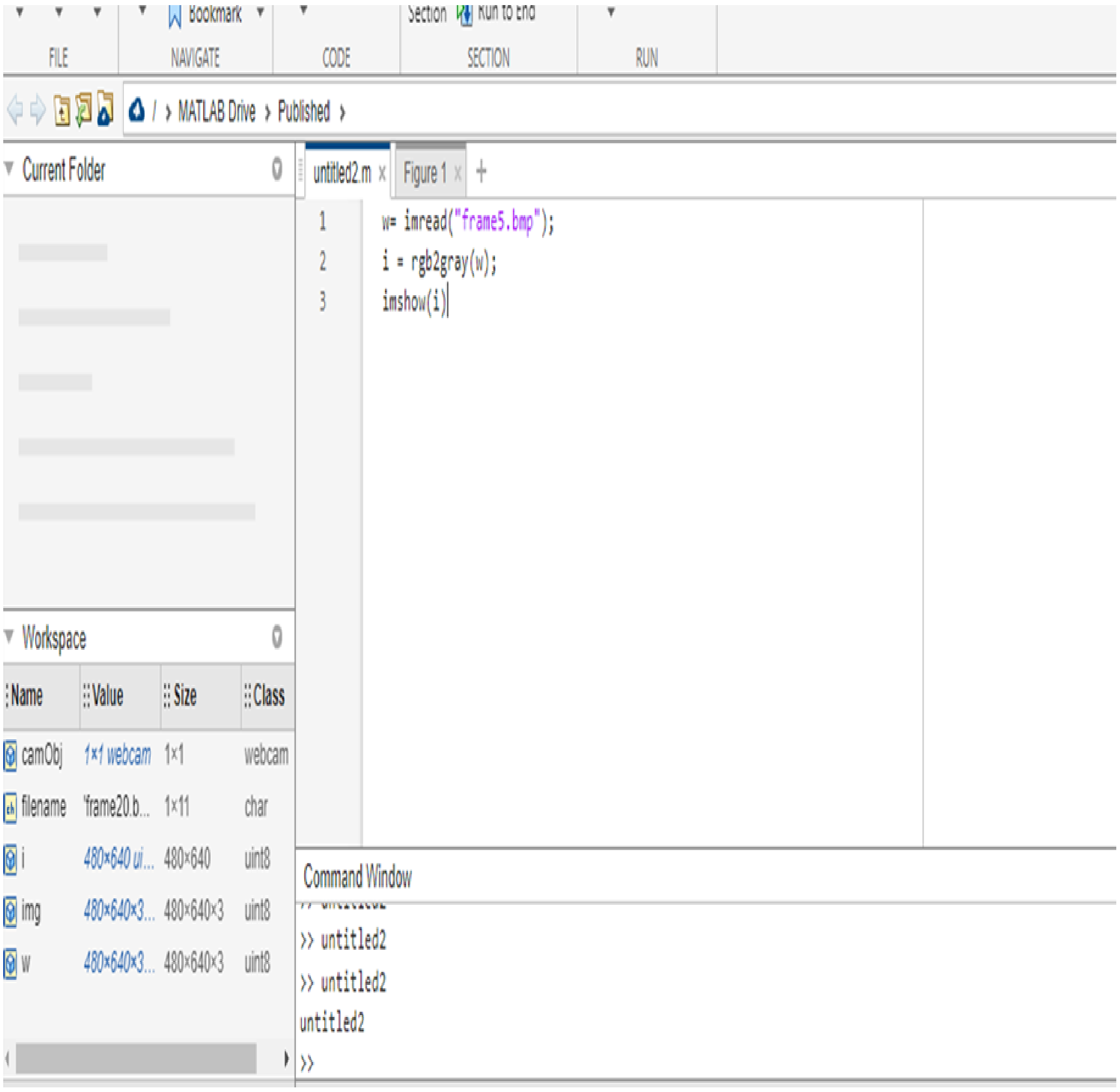
Figure 4.4.3.1 imshow() Command

4.4.4 To Convert RGB Image to Gray Image

Command: rgb2gray(rgbimage)

Rgb2gray( ) creates an image that is grayscale from the truecolor image RGB. By removing the hue and saturation data while keeping the luminance, the rgb2gray function converts RGB images to grayscale.

Command Window:



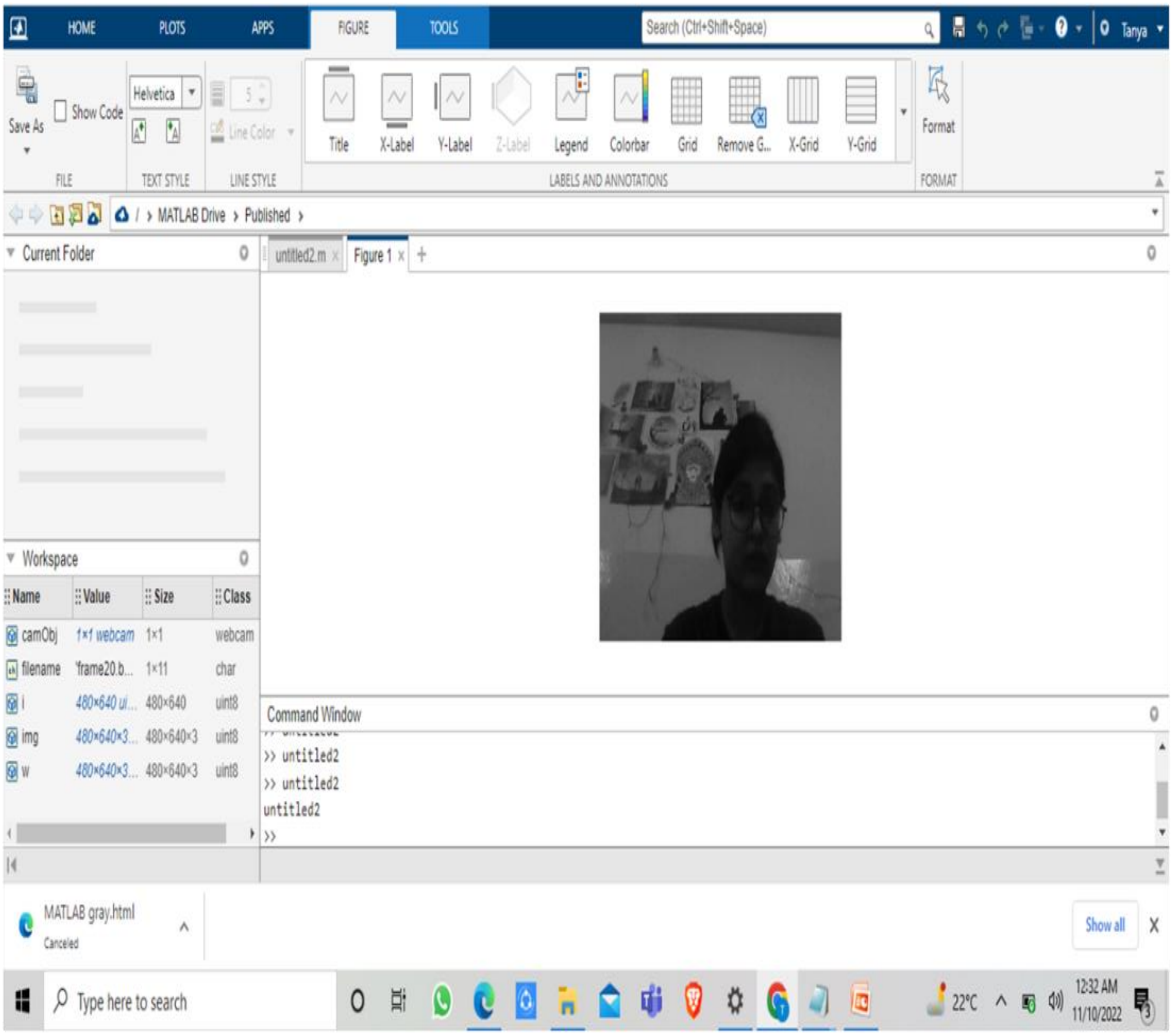
```
w= imread("frame5.bmp");
i = rgb2gray(w);
imshow(i)
```

Name	Value	Size	Class
camObj	1×1 webcam	1×1	webcam
filename	'frame20.b...	1×11	char
i	480×640 ui...	480×640	uint8
img	480×640×3...	480×640×3	uint8
w	480×640×3...	480×640×3	uint8

```
>> untitled2
>> untitled2
untitled2
>>
```

Figure 4.4.4.1 rgb2gray() Command

Results:



Name	Value	Size	Class
camObj	1×1 webcam	1×1	webcam
filename	'frame20.b...	1×11	char
i	480×640 ui...	480×640	uint8
img	480×640×3...	480×640×3	uint8
w	480×640×3...	480×640×3	uint8

```
>> untitled2
>> untitled2
untitled2
>>
```

Figure 4.4.4.2 Results of rgb2gray() Command

4.4.5 To Read Pixels of an Image

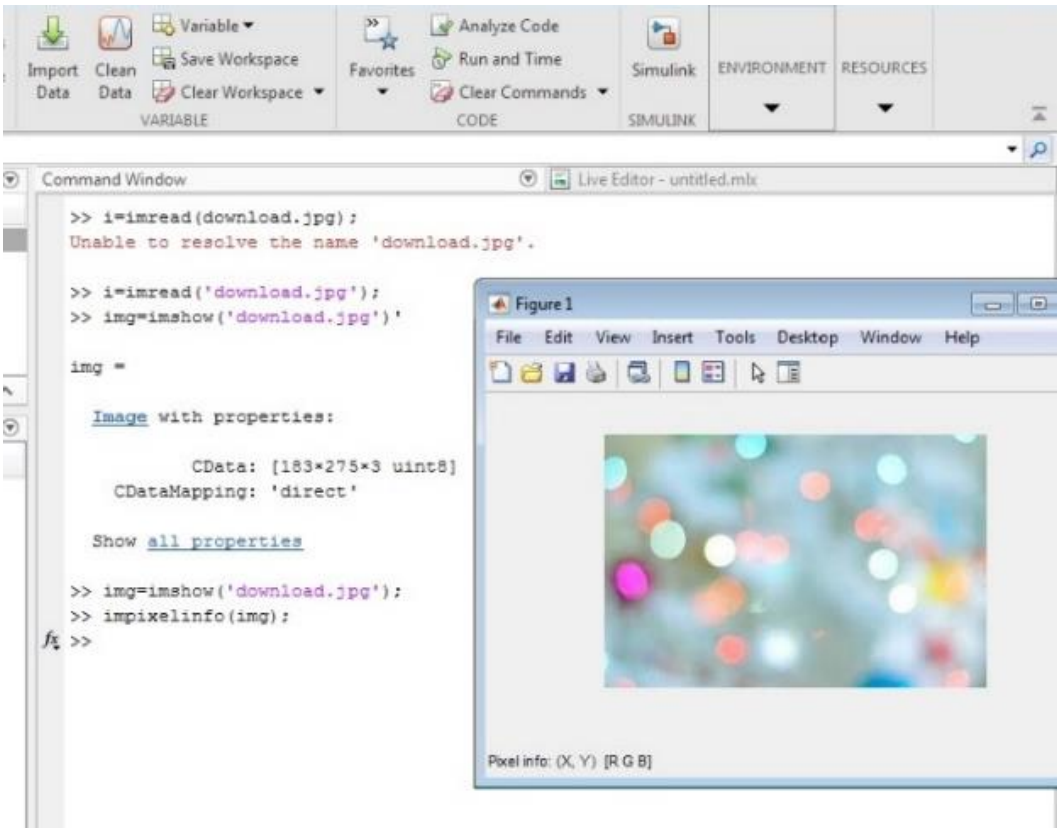
Command: impixelinfo( )

Impixelinfo( ) is a pixel information tool. It displays the information of the pixels of the image wherever the cursor is placed over. The information is displayed such as :

Pixel info (X, Y) (R, G, B); where X and Y denotes the x and y coordinates and R, G, B denotes the intensity red, green, blue colour because each pixel is made up of these three colour components only.

Command Window:

Image 1:

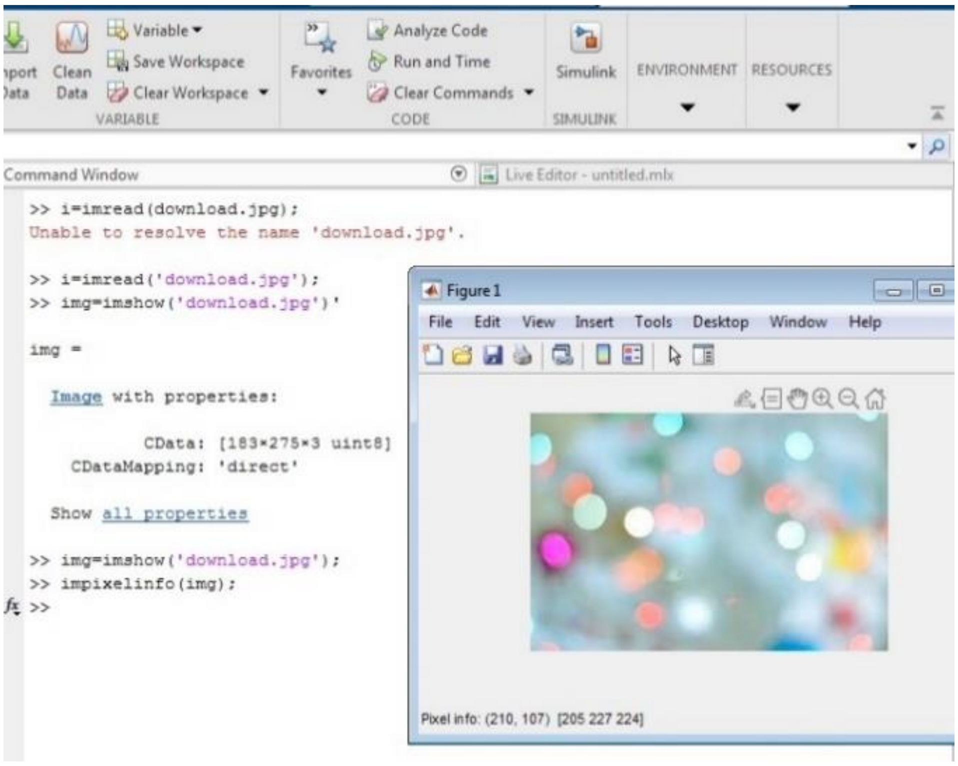


```
>> i=imread(download.jpg);
Unable to resolve the name 'download.jpg'.

>> i=imread('download.jpg');
>> img=imshow('download.jpg')'

img =

  Image with properties:

           CData: [183×275×3 uint8]
     CDataMapping: 'direct'

  Show all properties

>> img=imshow('download.jpg');
>> impixelinfo(img);
fx >>
```

```
Pixel info: (X, Y)  [R G B]
```

Figure 4.4.5.1 impixelinfo() Command

Image 2: (with cursor movements…)

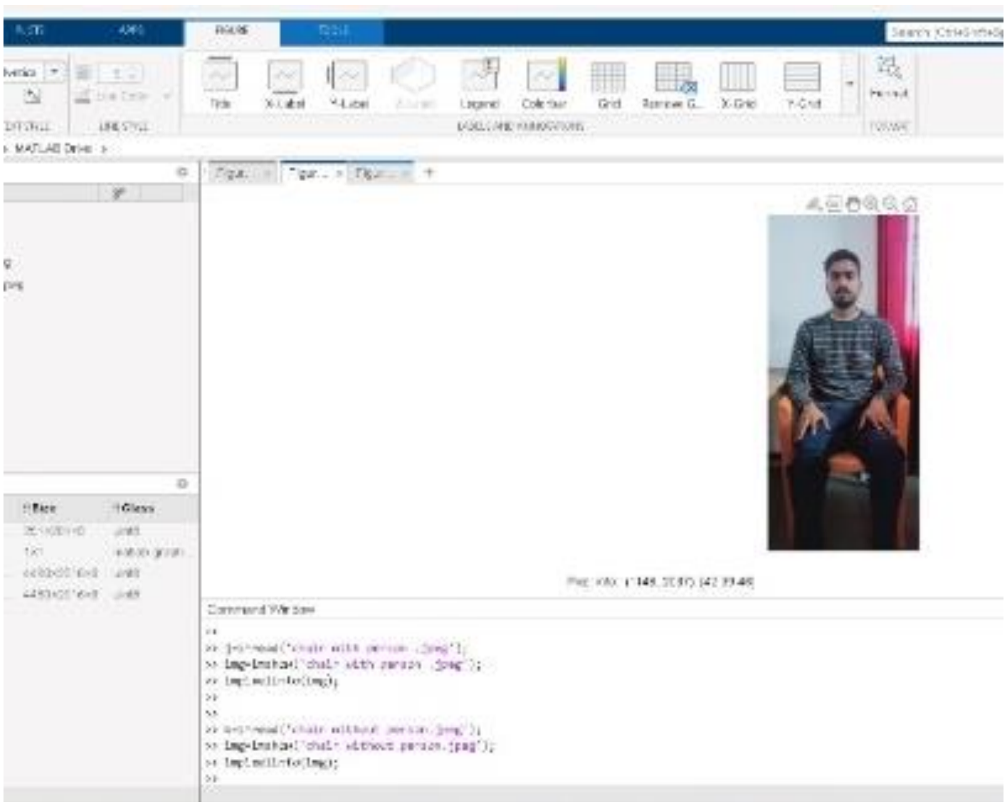


```
>> i=imread(download.jpg);
Unable to resolve the name 'download.jpg'.

>> i=imread('download.jpg');
>> img=imshow('download.jpg')'

img =

  Image with properties:

           CData: [183×275×3 uint8]
     CDataMapping: 'direct'

  Show all properties

>> img=imshow('download.jpg');
>> impixelinfo(img);
fx >>
```

```
Pixel info: (210, 107)  [205 227 224]
```

Figure 4.4.5.2 impixelinfo() Command with Cursor Movement

To observe the differences between pixels in an image, we initially performed the experiment with a person seated on a chair, and for the second, we used the chair with no one seated on it. The results are shown below:

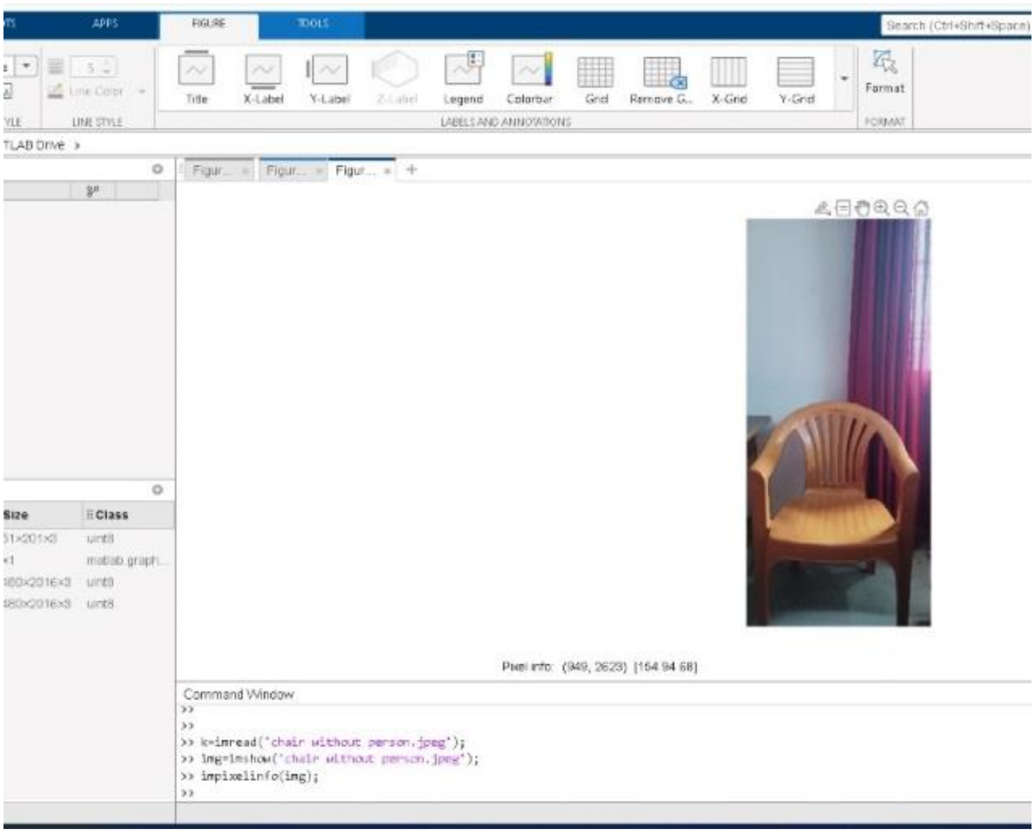


Pixel info: (1148, 2030) [42 33 48]

Command Window

```
>>
>> j=imread('chair with person .jpeg');
>> img=imshow('chair with person .jpeg');
>> impixelinfo(img);
>>
>>
>> k=imread('chair without person.jpeg');
>> img=imshow('chair without person.jpeg');
>> impixelinfo(img);
>>
```

Figure 4.4.5.3: Pixels with a Person Seated on Chair



Pixel info: (949, 2623) [154 94 68]

Command Window

```
>>
>>
>> k=imread('chair without person.jpeg');
>> img=imshow('chair without person.jpeg');
>> impixelinfo(img);
>>
```

Figure 4.4.5.4: Pixels with an Empty Chair

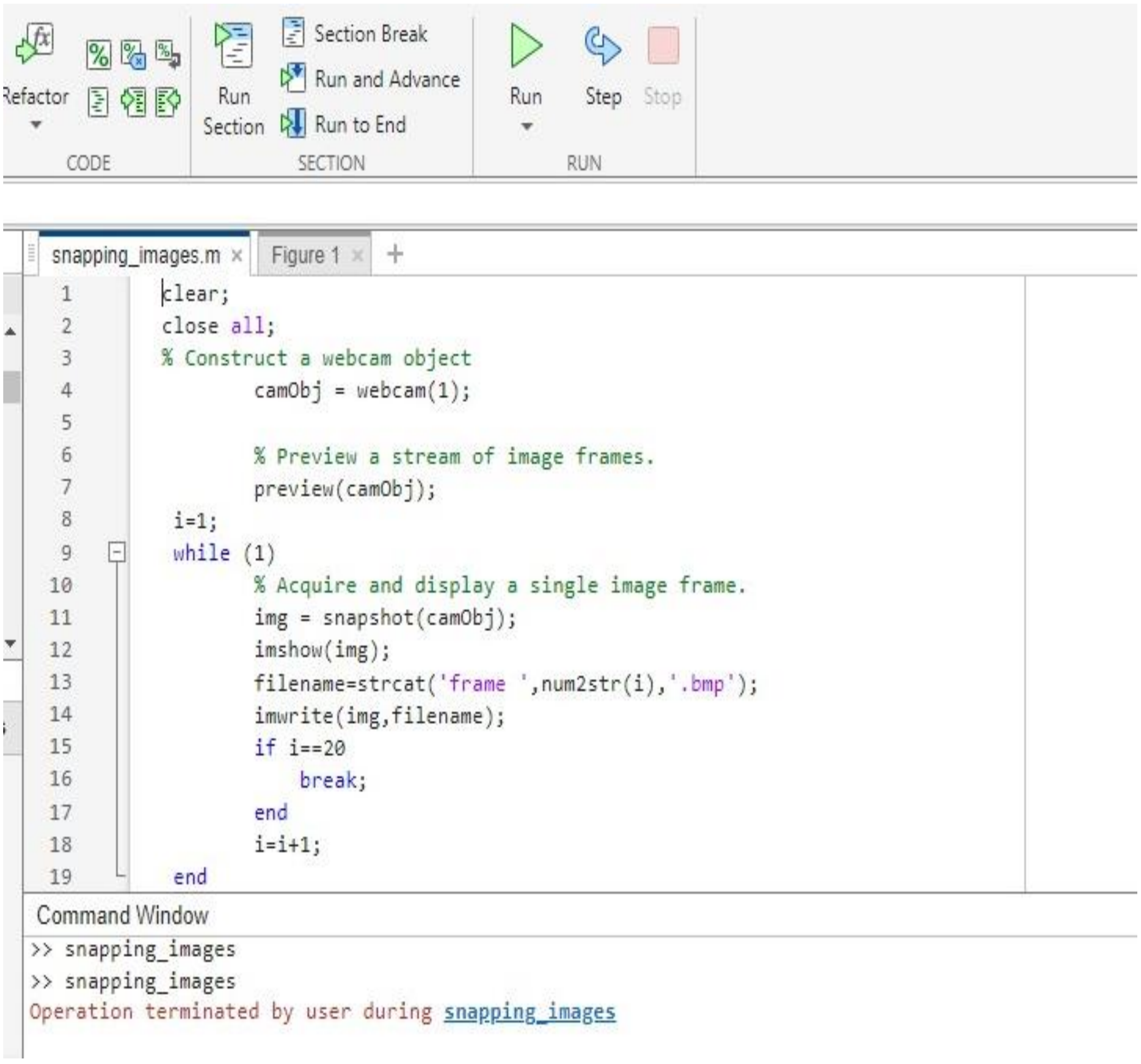
CHAPTER 5

EXPERIMENTS

5.1 Snapping Images from Webcam

To detect micro-expressions, continuous image capture is required. It is done by creating a webcam object, followed by a while loop. The snaps are stored using the.bmp extension, as it is ideal for storing original, uncompressed images with high pixel quality. In the following example, we have run the while loop for i = 20; that means the camera will continuously take 20 snaps before exiting the loop.

Command Window:



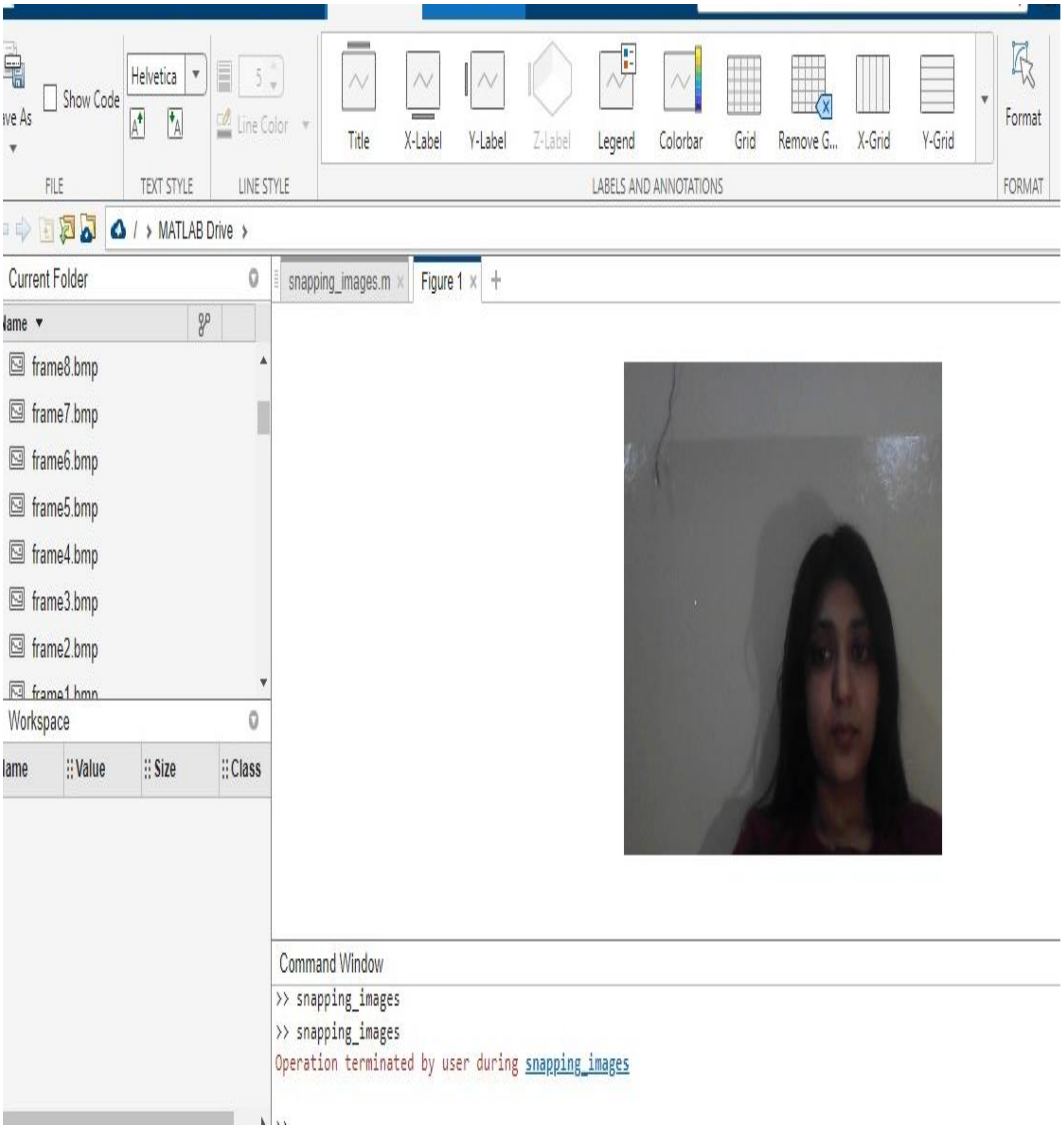
```
clear;
close all;
% Construct a webcam object
        camObj = webcam(1);

        % Preview a stream of image frames.
        preview(camObj);
 i=1;
 while (1)
        % Acquire and display a single image frame.
        img = snapshot(camObj);
        imshow(img);
        filename=strcat('frame ',num2str(i),'.bmp');
        imwrite(img,filename);
        if i==20
            break;
        end
        i=i+1;
 end
```

```
>> snapping_images
>> snapping_images
Operation terminated by user during snapping_images
```

Figure 5.1.1: Code for Snapping Images from Webcam

Result:

As seen, the camera has taken 20 snaps, which are stored in MATLAB as "frame1.bmp," "frame2.bmp," and so on.

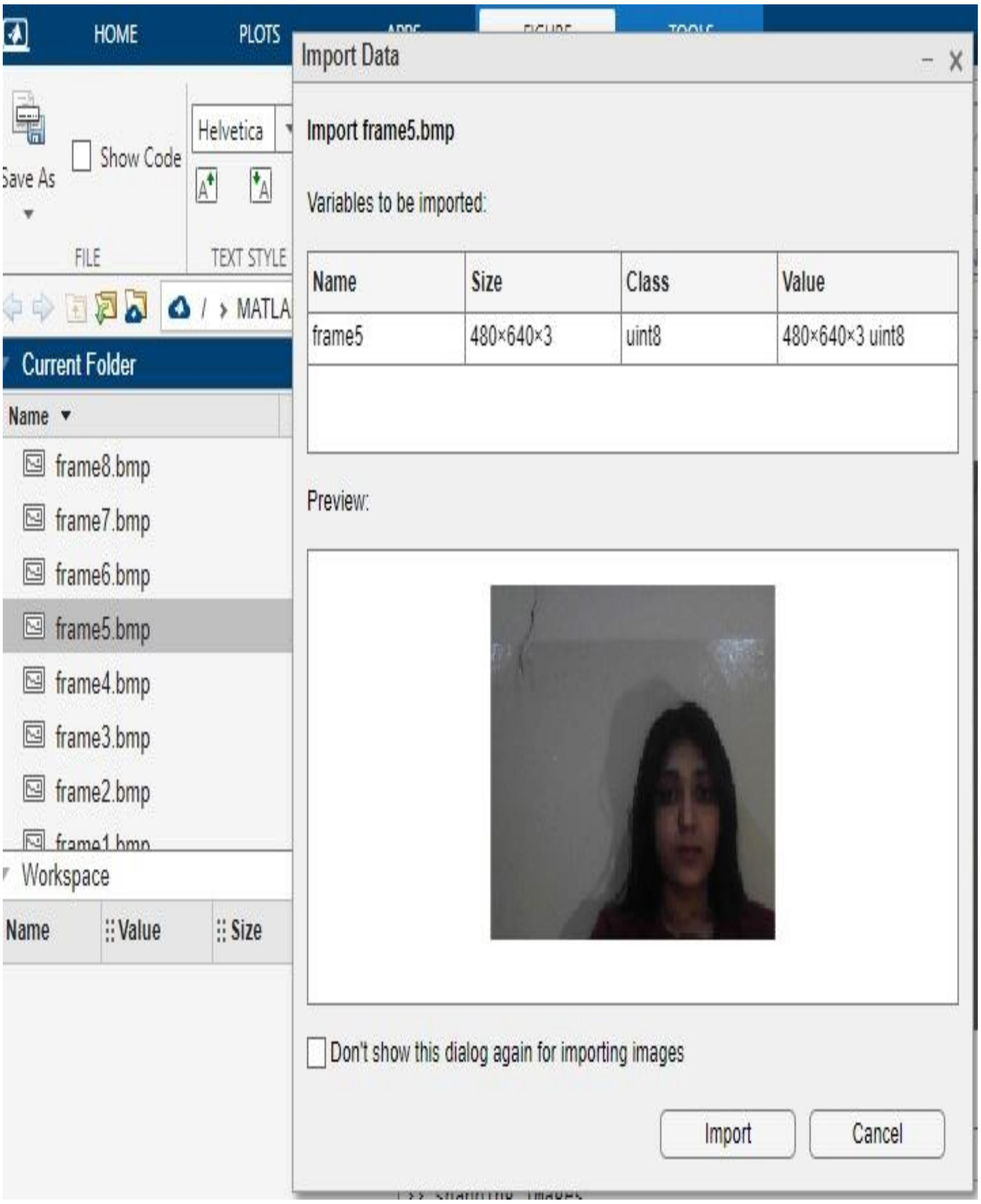


frame8.bmp
frame7.bmp
frame6.bmp
frame5.bmp
frame4.bmp
frame3.bmp
frame2.bmp
frame1.bmp

```
>> snapping_images
>> snapping_images
Operation terminated by user during snapping_images
```

Figure 5.1.2: Result for Snapping Images from Webcam

The size of each snap taken is 480*64*3 pixels with class unit 8, which means it is an 8-bit image. It can be easily imported from MATLAB. The preview window for the same is shown below:



Import frame5.bmp

Variables to be imported:

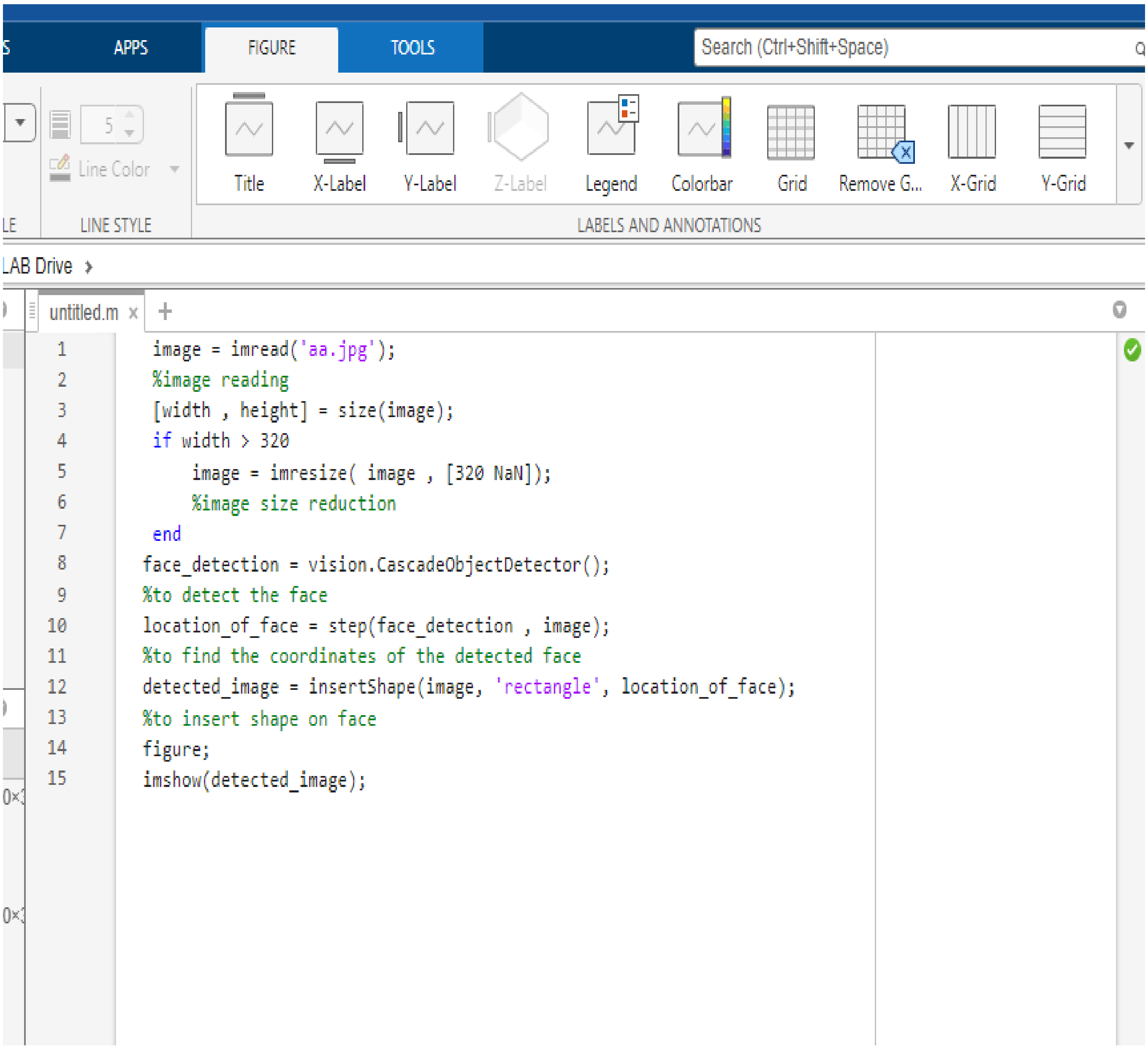
Name	Size	Class	Value
frame5	480×640×3	uint8	480×640×3 uint8

Figure 5.1.3: Result if we Import the Images Snapped through Webcam

5.2 Detection of Face through Image

It is very simple to detect faces from images using MATLAB. It just requires following toolboxes: the image processing toolbox and the Vision system toolbox. The image is initially read from the source folder. As each image is unique in its dimensions and size, we use the "imresize" object to shrink each image to 320 pixels in order to standardise them all. The "vision.CascadeObjectDetector" object is used to detect faces. By finding the detected face's coordinates, a rectangle box is placed on the detected face.

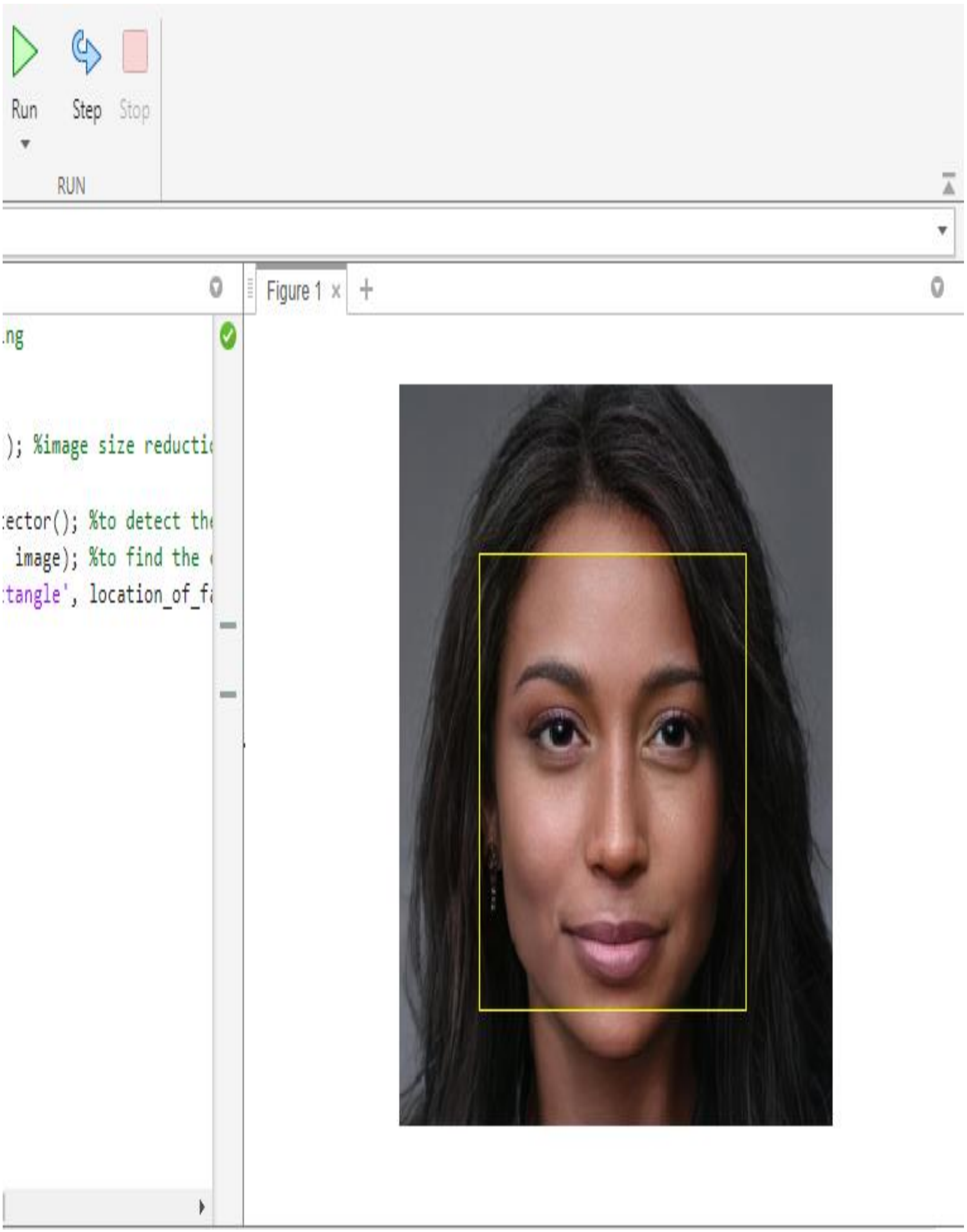
Command Window:



```
 image = imread('aa.jpg');
 %image reading
 [width , height] = size(image);
 if width > 320
     image = imresize( image , [320 NaN]);
     %image size reduction
 end
face_detection = vision.CascadeObjectDetector();
%to detect the face
location_of_face = step(face_detection , image);
%to find the coordinates of the detected face
detected_image = insertShape(image, 'rectangle', location_of_face);
%to insert shape on face
figure;
imshow(detected_image);
```

Figure 5.2.1: Code for Detection of Face through Image

Result:



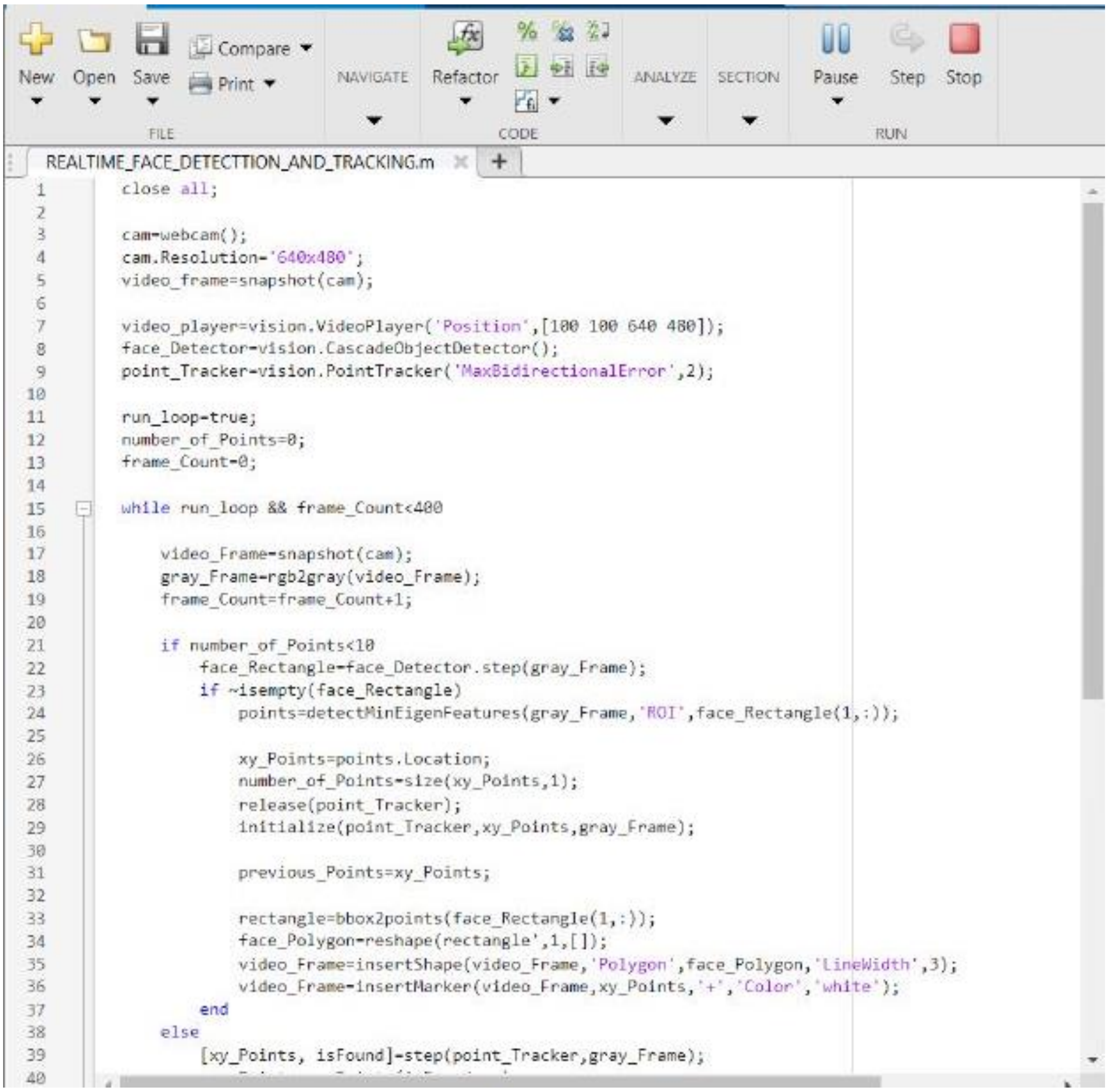
```
.ng

); %image size reducti

ector(); %to detect th
 image); %to find the
tangle', location_of_f
```

Figure 5.2.2: Detection of Face through Image [8]

5.3 Real-Time Face Detection

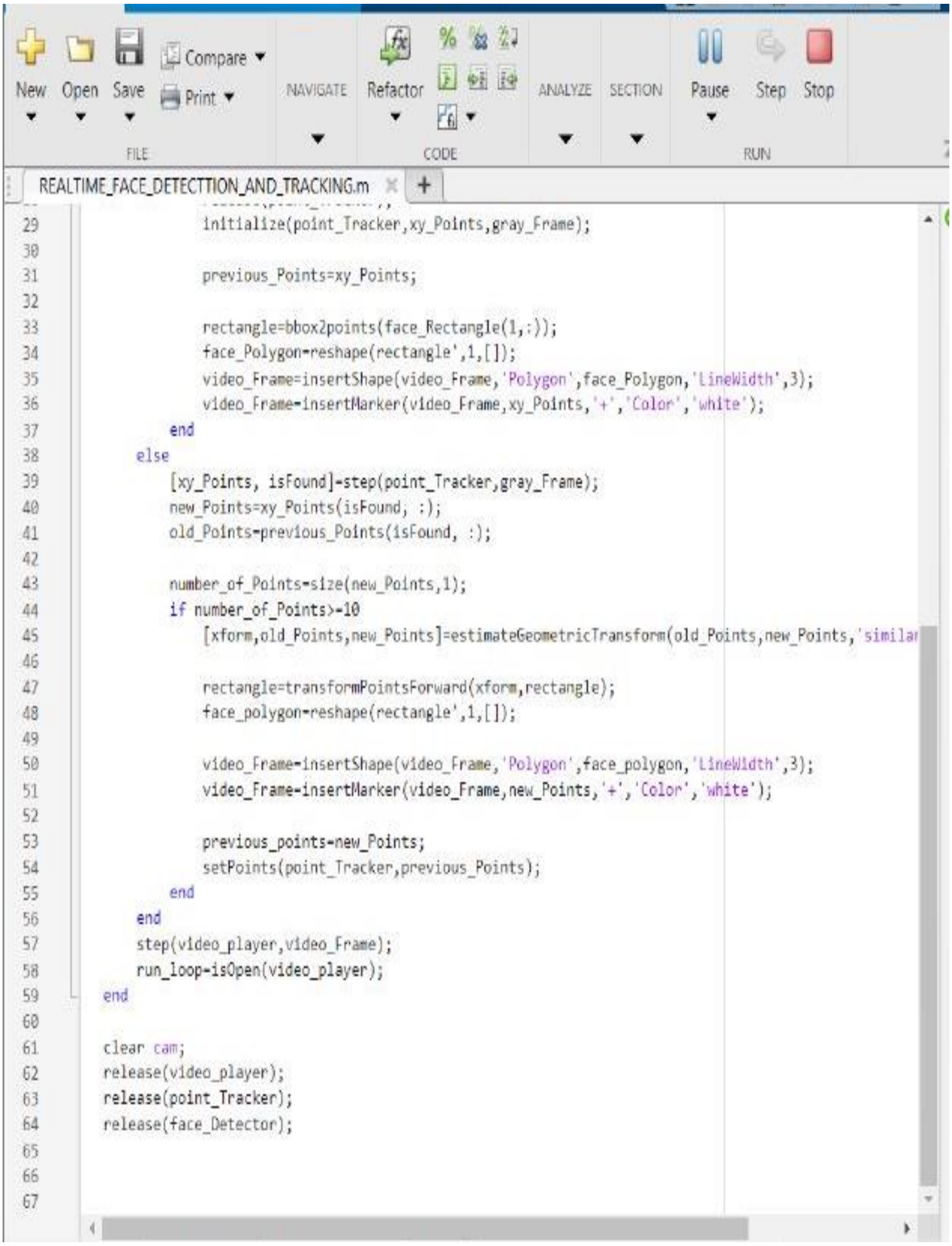
This example demonstrates the use of feature points to automatically recognise and track a face. The method used in this illustration maintains focus on the face even when the subject tilts their head or turns their body away from or towards the camera. Many computer vision applications, such as activity recognition, vehicle safety, and surveillance, depend on accurate object detection and tracking. We have divided the problem of tracking the face continuously in three parts to create a basic face tracking system. First, look for the face. Identifying facial features to track comes next, followed by tracking the face. To detect the face, “vision.CascadeObjectDetector” object is used. It uses Viola-jones detection algorithm. For tracking the face continuously Kanade-Lucas-Tomasi (KLT) algorithm is used

Command Window:



```
close all;

cam=webcam();
cam.Resolution='640x480';
video_frame=snapshot(cam);

video_player=vision.VideoPlayer('Position',[100 100 640 480]);
face_Detector=vision.CascadeObjectDetector();
point_Tracker=vision.PointTracker('MaxBidirectionalError',2);

run_loop=true;
number_of_Points=0;
frame_Count=0;

while run_loop && frame_Count<400

    video_Frame=snapshot(cam);
    gray_Frame=rgb2gray(video_Frame);
    frame_Count=frame_Count+1;

    if number_of_Points<10
        face_Rectangle=face_Detector.step(gray_Frame);
        if ~isempty(face_Rectangle)
            points=detectMinEigenFeatures(gray_Frame,'ROI',face_Rectangle(1,:));

            xy_Points=points.Location;
            number_of_Points=size(xy_Points,1);
            release(point_Tracker);
            initialize(point_Tracker,xy_Points,gray_Frame);

            previous_Points=xy_Points;

            rectangle=bbox2points(face_Rectangle(1,:));
            face_Polygon=reshape(rectangle',1,[]);
            video_Frame=insertShape(video_Frame,'Polygon',face_Polygon,'LineWidth',3);
            video_Frame=insertMarker(video_Frame,xy_Points,'+','Color','white');
        end
    else
        [xy_Points, isFound]=step(point_Tracker,gray_Frame);
```

Figure 5.3.1: Code for Real-Time Face Detection (part 1)

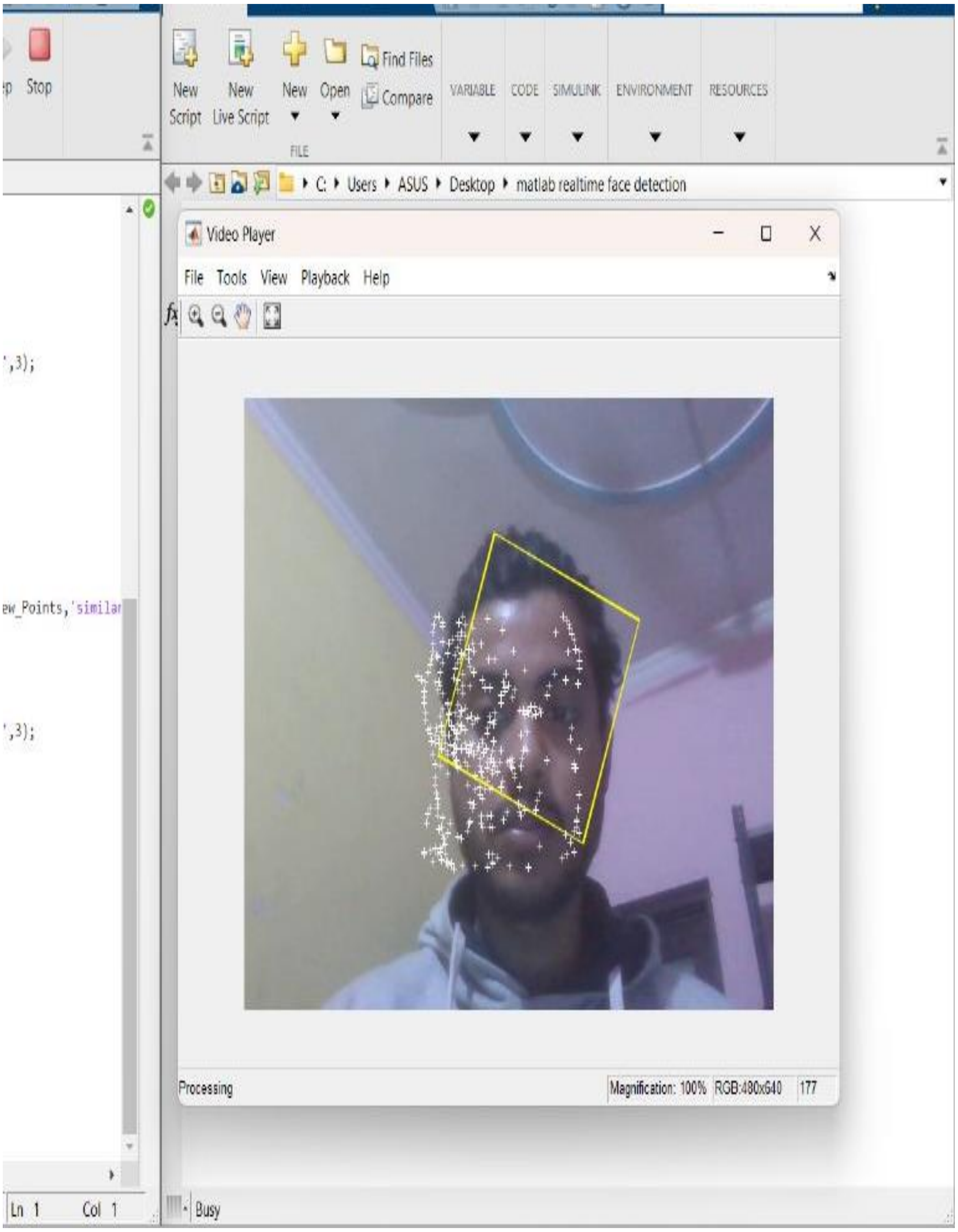


REALTIME_FACE_DETECTTION_AND_TRACKING.m

```
            initialize(point_Tracker,xy_Points,gray_Frame);

            previous_Points=xy_Points;

            rectangle=bbox2points(face_Rectangle(1,:));
            face_Polygon=reshape(rectangle',1,[]);
            video_Frame=insertShape(video_Frame,'Polygon',face_Polygon,'LineWidth',3);
            video_Frame=insertMarker(video_Frame,xy_Points,'+','Color','white');
        end
    else
        [xy_Points, isFound]=step(point_Tracker,gray_Frame);
        new_Points=xy_Points(isFound, :);
        old_Points=previous_Points(isFound, :);

        number_of_Points=size(new_Points,1);
        if number_of_Points>=10
            [xform,old_Points,new_Points]=estimateGeometricTransform(old_Points,new_Points,'similar

            rectangle=transformPointsForward(xform,rectangle);
            face_polygon=reshape(rectangle',1,[]);

            video_Frame=insertShape(video_Frame,'Polygon',face_polygon,'LineWidth',3);
            video_Frame=insertMarker(video_Frame,new_Points,'+','Color','white');

            previous_points=new_Points;
            setPoints(point_Tracker,previous_Points);
        end
    end
    step(video_player,video_Frame);
    run_loop=isOpen(video_player);
end

clear cam;
release(video_player);
release(point_Tracker);
release(face_Detector);
```

Figure 5.3.2: Code for Real-Time Face Detection (part 2)

Result:



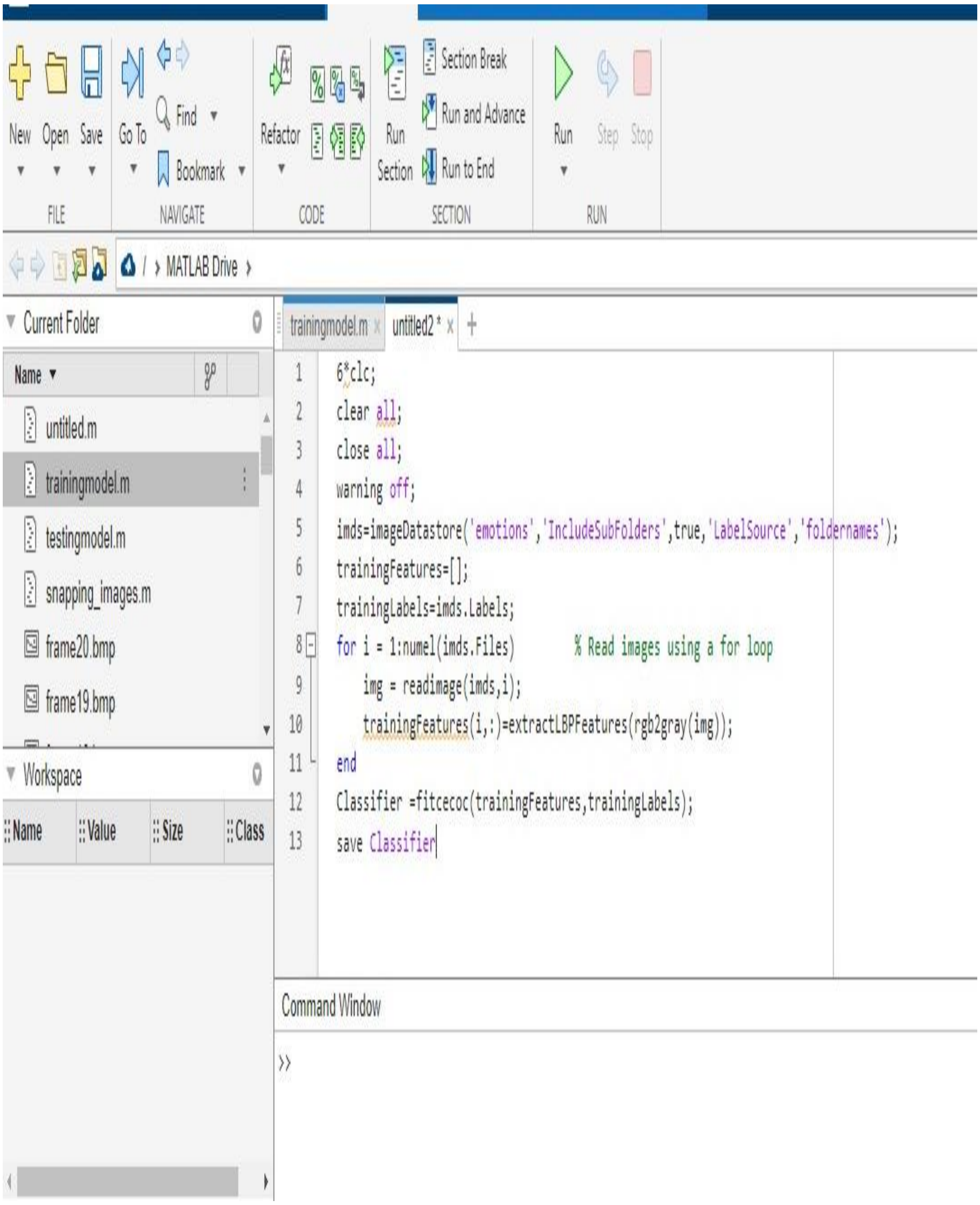
```
',3);
ew_Points,'similar
',3);
```

Figure 5.3.3: Result of Real-Time Face Detection

5.4 Training Model

Now that we have our dataset, we need to train our model to match our expectations.

Command Window:



```
6*clc;
clear all;
close all;
warning off;
imds=imageDatastore('emotions','IncludeSubFolders',true,'LabelSource','foldernames');
trainingFeatures=[];
trainingLabels=imds.Labels;
for i = 1:numel(imds.Files)        % Read images using a for loop
    img = readimage(imds,i);
    trainingFeatures(i,:)=extractLBPFeatures(rgb2gray(img));
end
Classifier =fitcecoc(trainingFeatures,trainingLabels);
save Classifier
```

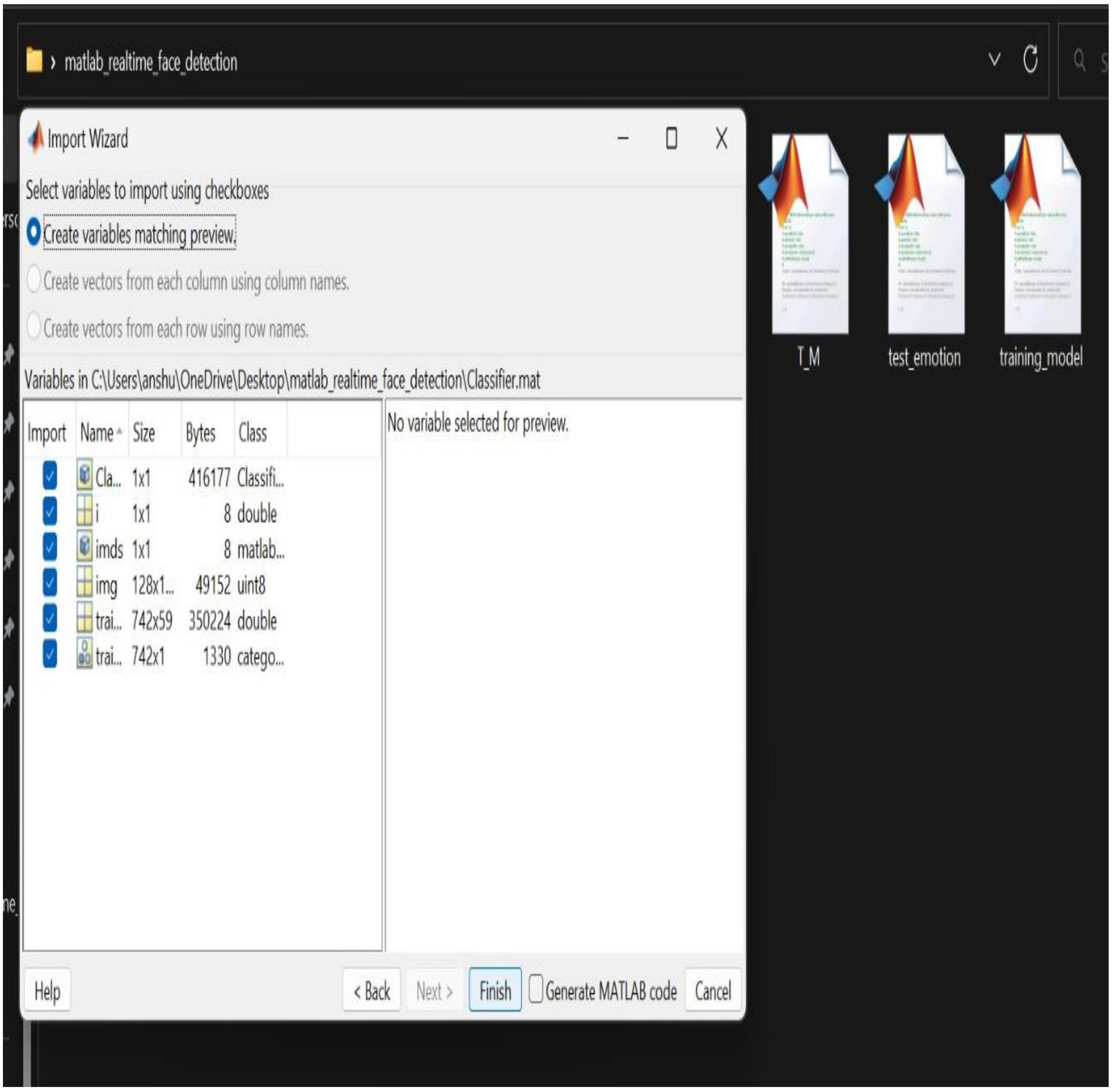
```
>>
```

Figure 5.4.1: Code for Training Model

5.5 Classifier File used for Database Training

When we execute our code for training model, MATLAB automatically creates a standard file known as Classifier file. After analysing the documents, classifier selects the appropriate label from a list of labels that we create and applies it to the content.

Command Window:



Select variables to import using checkboxes

[✓ Create variables matching preview.]

Create vectors from each column using column names.

Create vectors from each row using row names.

Variables in C:\Users\anshu\OneDrive\Desktop\matlab_realtime_face_detection\Classifier.mat

Import	Name	Size	Bytes	Class
✓	Cla...	1x1	416177	Classifi...
✓	i	1x1	8	double
✓	imds	1x1	8	matlab...
✓	img	128x1...	49152	uint8
✓	trai...	742x59	350224	double
✓	trai...	742x1	1330	catego...

No variable selected for preview.

Figure 5.5.1: Classifier File used for Database Training

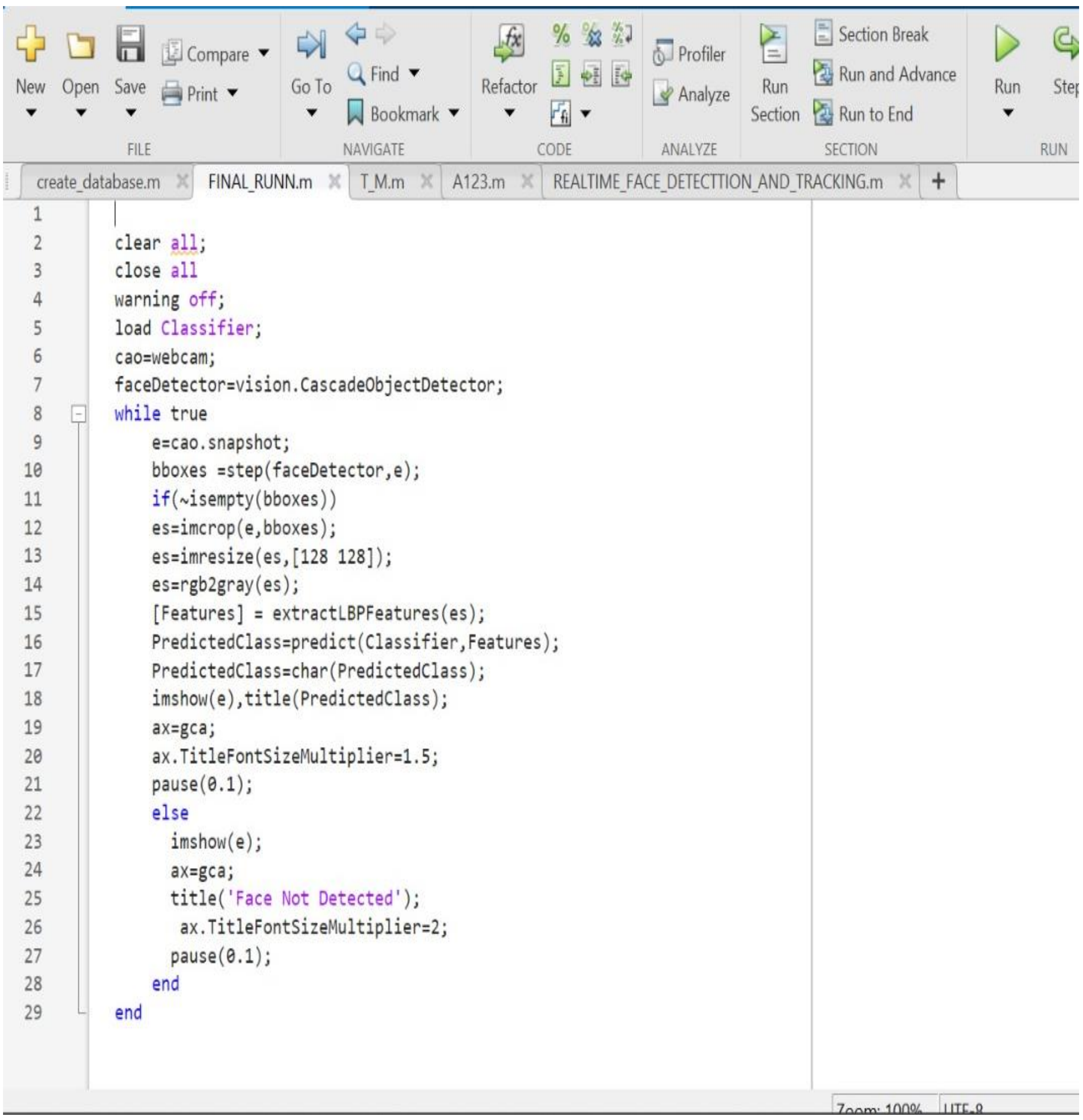
CHAPTER 6

RESULT AND DISCUSSION

6.1 Final Execution

Now since our dataset is built and also trained it is time to test the model. After running the code, it detects the face and compares it with the images stored in our dataset and gives us the results.

Command Window:

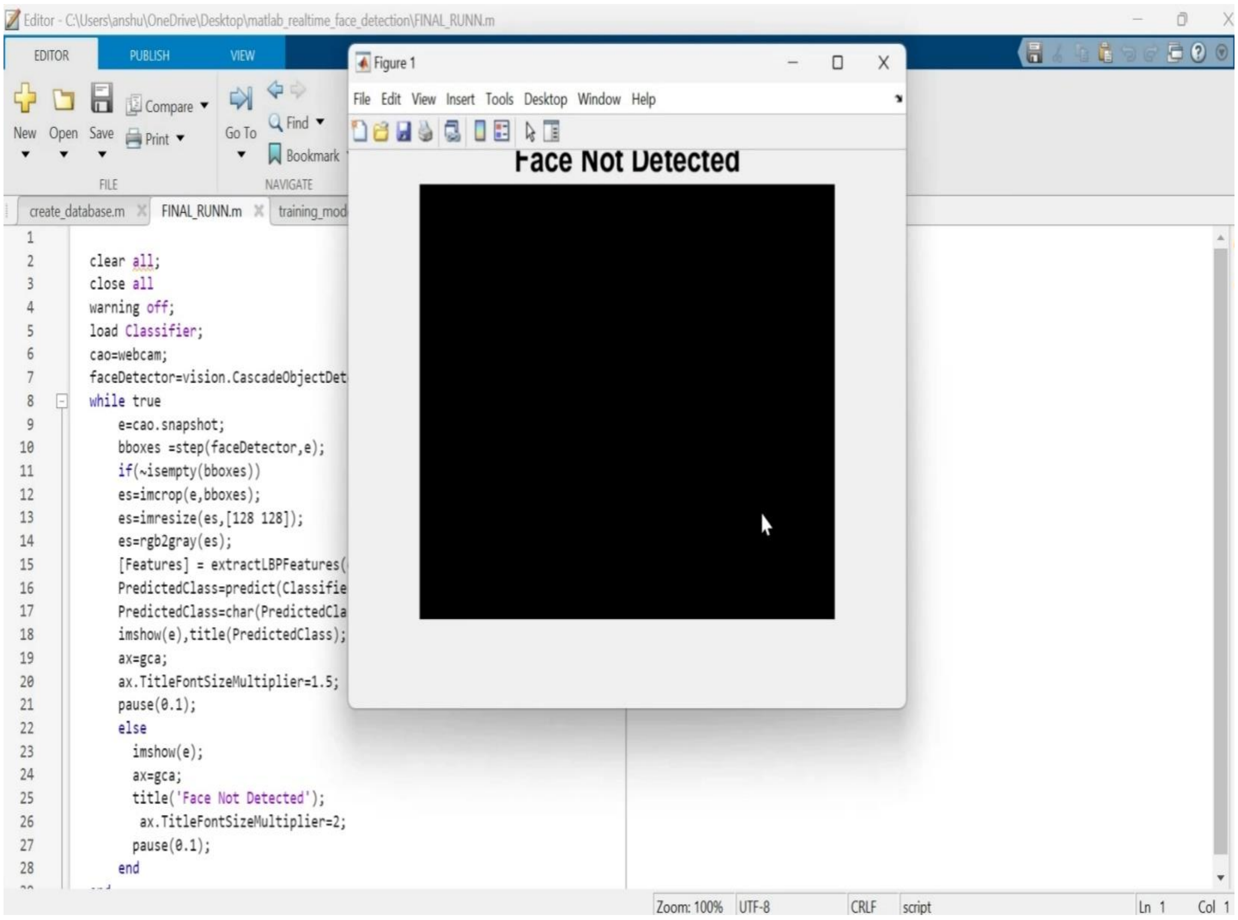


```

clear all;
close all
warning off;
load Classifier;
cao=webcam;
faceDetector=vision.CascadeObjectDetector;
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(~isempty(bboxes))
    es=imcrop(e,bboxes);
    es=imresize(es,[128 128]);
    es=rgb2gray(es);
    [Features] = extractLBPFeatures(es);
    PredictedClass=predict(Classifier,Features);
    PredictedClass=char(PredictedClass);
    imshow(e),title(PredictedClass);
    ax=gca;
    ax.TitleFontSizeMultiplier=1.5;
    pause(0.1);
    else
      imshow(e);
      ax=gca;
      title('Face Not Detected');
       ax.TitleFontSizeMultiplier=2;
      pause(0.1);
    end
end
```

Figure 6.1.1: Code for Final Execution

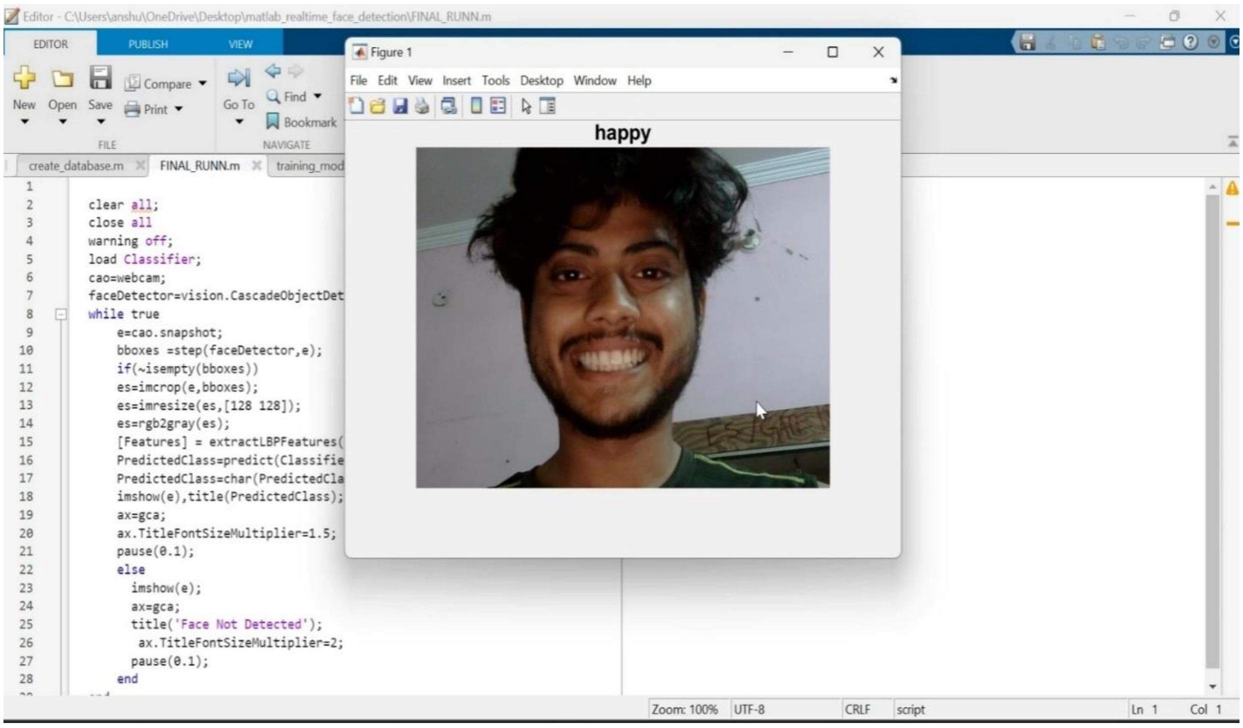
Result:

The emotions detected by our "Real-Time Facial Emotion Recognition" model are shown as below:



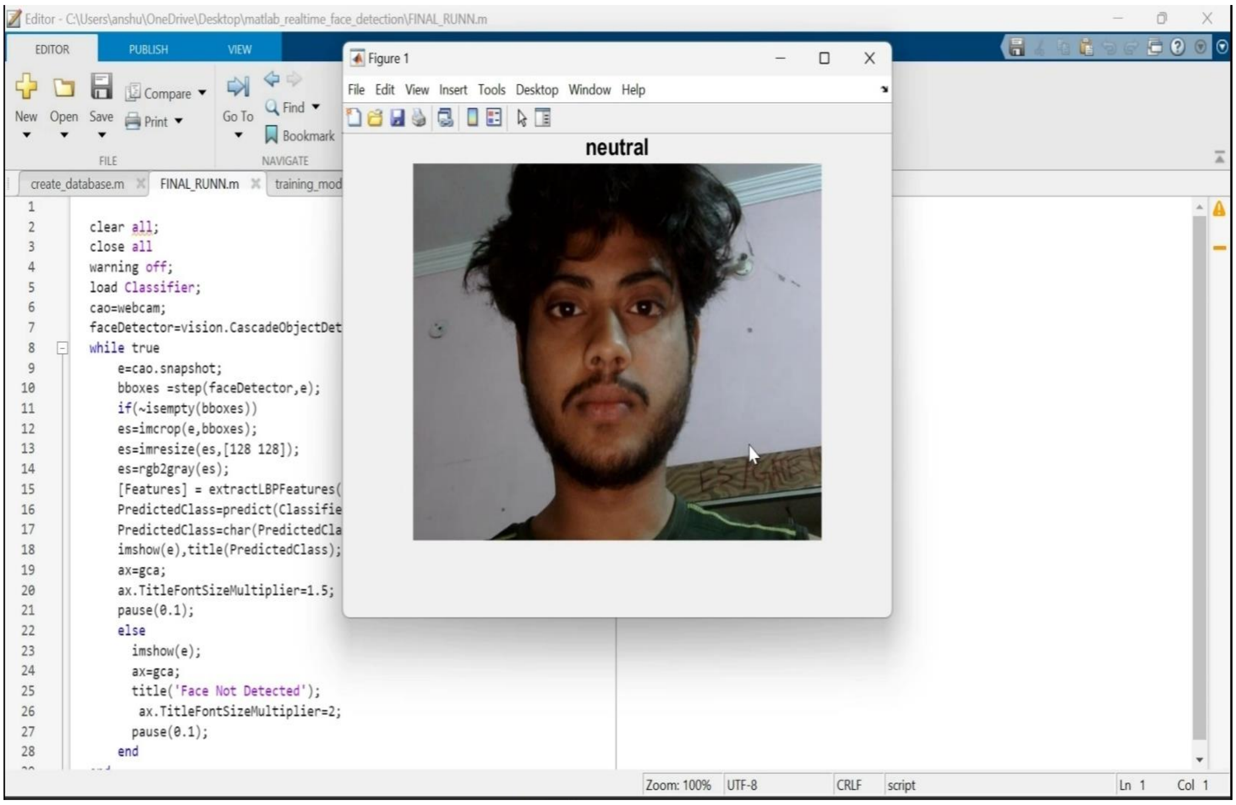
```
clear all;
close all
warning off;
load Classifier;
cao=webcam;
faceDetector=vision.CascadeObjectDet
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(~isempty(bboxes))
    es=imcrop(e,bboxes);
    es=imresize(es,[128 128]);
    es=rgb2gray(es);
    [Features] = extractLBPFeatures(
    PredictedClass=predict(Classifie
    PredictedClass=char(PredictedCla
    imshow(e),title(PredictedClass);
    ax=gca;
    ax.TitleFontSizeMultiplier=1.5;
    pause(0.1);
    else
      imshow(e);
      ax=gca;
      title('Face Not Detected');
       ax.TitleFontSizeMultiplier=2;
      pause(0.1);
    end
```

Figure 6.1.2: Result when Face not Detected



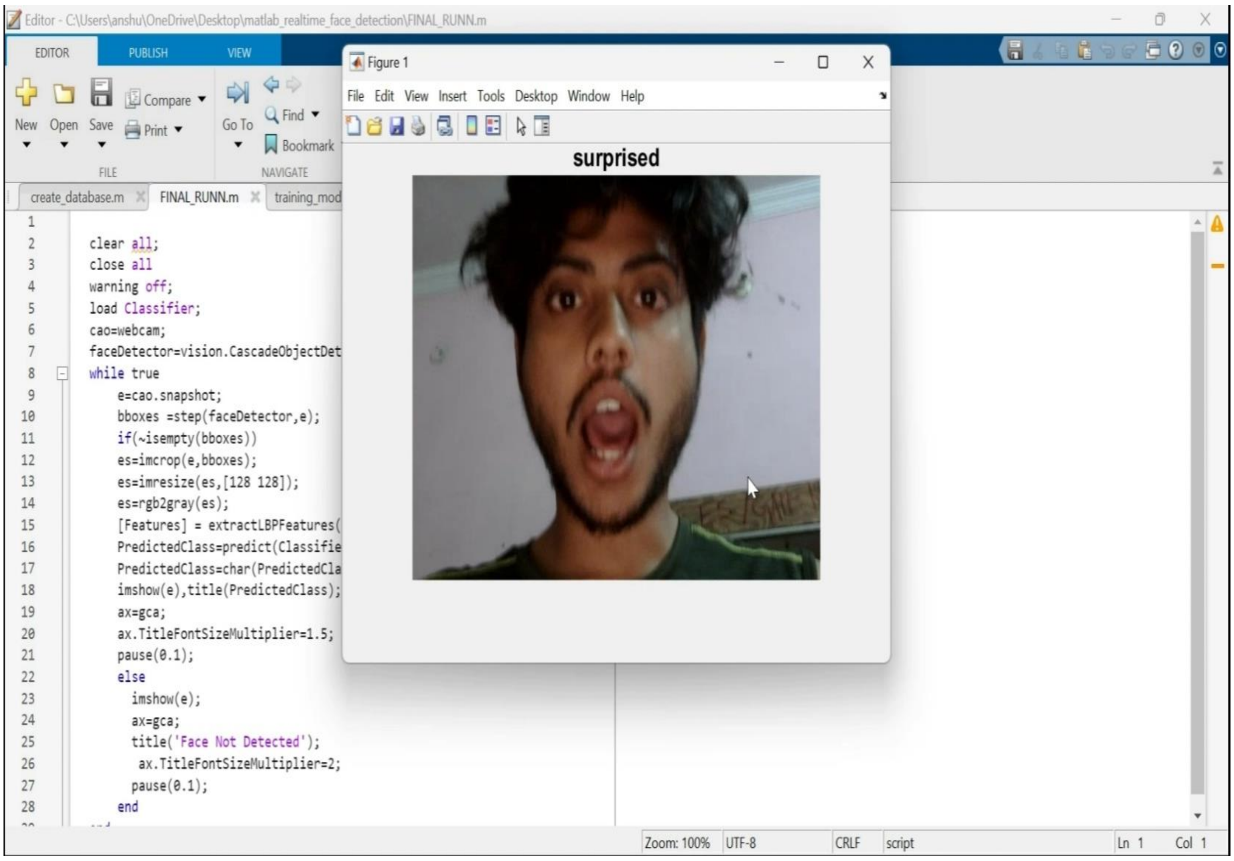
```
clear all;
close all
warning off;
load Classifier;
cao=webcam;
faceDetector=vision.CascadeObjectDet
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(~isempty(bboxes))
    es=imcrop(e,bboxes);
    es=imresize(es,[128 128]);
    es=rgb2gray(es);
    [Features] = extractLBPFeatures(
    PredictedClass=predict(Classifie
    PredictedClass=char(PredictedCla
    imshow(e),title(PredictedClass);
    ax=gca;
    ax.TitleFontSizeMultiplier=1.5;
    pause(0.1);
    else
      imshow(e);
      ax=gca;
      title('Face Not Detected');
       ax.TitleFontSizeMultiplier=2;
      pause(0.1);
    end
```

Figure 6.1.3: Result when Happy Face Detected



```
clear all;
close all
warning off;
load Classifier;
cao=webcam;
faceDetector=vision.CascadeObjectDet
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(~isempty(bboxes))
    es=imcrop(e,bboxes);
    es=imresize(es,[128 128]);
    es=rgb2gray(es);
    [Features] = extractLBPFeatures(
    PredictedClass=predict(Classifie
    PredictedClass=char(PredictedCla
    imshow(e),title(PredictedClass);
    ax=gca;
    ax.TitleFontSizeMultiplier=1.5;
    pause(0.1);
    else
      imshow(e);
      ax=gca;
      title('Face Not Detected');
       ax.TitleFontSizeMultiplier=2;
      pause(0.1);
    end
```

Figure 6.1.4: Result when Neutral Face Detected



```
clear all;
close all
warning off;
load Classifier;
cao=webcam;
faceDetector=vision.CascadeObjectDet
while true
    e=cao.snapshot;
    bboxes =step(faceDetector,e);
    if(~isempty(bboxes))
    es=imcrop(e,bboxes);
    es=imresize(es,[128 128]);
    es=rgb2gray(es);
    [Features] = extractLBPFeatures(
    PredictedClass=predict(Classifie
    PredictedClass=char(PredictedCla
    imshow(e),title(PredictedClass);
    ax=gca;
    ax.TitleFontSizeMultiplier=1.5;
    pause(0.1);
    else
      imshow(e);
      ax=gca;
      title('Face Not Detected');
       ax.TitleFontSizeMultiplier=2;
      pause(0.1);
    end
```

Figure 6.1.5: Result when Surprised Face Detected

CHAPTER 7

CONCLUSION AND FUTURE SCOPE

7.1 Conclusion

In this project, a method for classifying the facial expressions is proposed. Several applications, including robotics vision, video surveillance, digital cameras, security, and human-computer interaction, benefit from face detection and extraction of facial expressions. The goal of this project was to create a system for recognising facial expressions by applying computer visions and improving advanced feature extraction and classification.

Seven different facial expressions of photos of various people from various datasets have been evaluated for this research. Happy, sad, neutral, fear, disgust, rage, and surprise are among the emotions that were examined. In this study, facial expression pre-processing of collected facial images is followed by feature extraction using Local Binary Patterns and classification of facial expressions based on training datasets of facial images using Support Vector Machines. One occurrence of emotion can be recognised in within 5 seconds. Any emotion-aware mobile applications can use the system due to its excellent performance and short time requirement. We aim to expand on our work in a subsequent study and take additional emotional input modalities into account.

7.2 Future Scope

Over the past ten years, face expression recognition technologies have significantly advanced. Recognition of spontaneous expressions has undeniably taken centre stage in place of recognition of posed expressions. Under face registration mistakes, quick processing times, and high correct recognition rates (CRR), promising results can be produced, and our system has made considerable performance improvements. The system is entirely autonomous and is equipped to handle an image feed. It can identify unprompted facial expressions. Digital cameras that employ our method can only take pictures of people who are grinning. In security systems that can recognise a person, he can present himself in any way. When someone enters a room in a house, the lights and television can be adjusted to their preferences. A deaf patient's ailment or level of pain can be understood by doctors using this approach.

Our technology can be utilised in shopping centres, mini-marts, and other places to observe customer comments in order to improve operations, as well as to detect and track a user's mental condition. Better face recognition rates are guaranteed by the wide experimental evaluation of the face expressional system. After looking at methods for dealing with expression variance, the problem of face classification and the best integration of colour and depth information may be further explored in the future. Future research can be directed at gene alleles that match the geometric components of facial expressions. To meet the needs of various security models, such as criminal detection, governmental confidential security breaches, etc., the genetic property evolution framework for the facial expressional system can be investigated.

REFERENCES

[1] Nagi, Jawad, Syed Khaleel Ahmed, and Farrukh Nagi. “A MATLAB based face recognition system using image processing and neural networks.” In *4th International Colloquium on Signal Processing and its Applications*, vol. 2, pp. 83-8. 2008.

[2] Gupta, Neha, and Navneet Kaur. “Design and implementation of emotion recognition system by using MATLAB.” *International Journal of Engineering Research and Applications (IJERA)* 3, no. 4 (2013): 2002-2006.

[3] Bhadangkar, Dasharath K & Pujari, Jagadeesh & Yakkundimath, Rajesh. (2022). “Identification and Recognition of Facial Expressions Using Image Processing” a survey. 10.13140/RG.2.2.32068.09602.

[4] Chavan, P. M., Manan C. Jadhav, Jinal B. Mashruwala, Aditi K. Nehete, and Pooja A. Panjari. “Real time emotion recognition through facial expressions for desktop devices.” *International Journal of Emerging Science and Engineering (IJESE)* 1, no. 7 (2013): 104-108.

[5] Liu, Zhentao, Min Wu, Weihua Cao, Luefeng Chen, Jianping Xu, Ri Zhang, Mengtian Zhou, and Junwei Mao. “A facial expression emotion recognition-based human-robot interaction system.” *IEEE/CAA Journal of Automatica Sinica* 4, no. 4 (2017): 668-676.

[6] Murugappan M., Mutawa A., “Facial geometric feature extraction based emotional expression classification using machine learning algorithms”, *PLOS ONE*, vol.16, no.2, pp. e0247131, 2021.

[7] A. Kartali, M. Roglić, M. Barjaktarović, M. Đurić-Jovičić and M. M. Janković, “Real-time Algorithms for Facial Emotion Recognition: A Comparison of Different Approaches,” *2018 14th Symposium on Neural Networks and Applications (NEUREL)*, Belgrade, Serbia, 2018, pp. 1-4, doi: 10.1109/NEUREL.2018.8587011.

[8] Hauntington, A. H. (2020, August 12). *Humanface*.https://unsplash.com/s/photos/human-face.https://unsplash.com/photos/jzY0KRJopEI

[9] Pahuja, H., Ranjan, P., Ujlayan, A. and Goyal, A., 2022. Convolution Neural Network Based Visual Speech Recognition System for Syllable Identification. Recent Advances in Computer Science and Communications (Formerly: Recent Patents on Computer Science), 15(1), pp.139-150.

[10] Pahuja, H., Ranjan, P. and Ujlayan, A., 2017. GPS enabled efficient sound source localisation using microphone array. International Journal of Signal and Imaging Systems Engineering, 10(3), pp.120-135.

[11] Refat, Abu & Singh, Bikash & Rahman, M. (2022). SentiNet: A Nonverbal Facial Sentiment Analysis Using Convolutional Neural Network. International Journal of Pattern Recognition and Artificial Intelligence. 10.1142/S0218001422560079.

[12] Sambare, M. (2020) Fer-2013, Kaggle. Available at: https://www.kaggle.com/datasets/msambare/fer2013 /code (Accessed: February 24, 2023).

[13] Parajuli, S., Sunuwar. S., Matang, N., & Shrestha, S. (2017, August 23). A facial expression recognition system a project report. Academia.edu. Retrieved February 24, 2023, from

https://www.academia.edu/29632929/
A_Facial_Expression_Recognition_System_A_Project_Report

[14] Michel, P., & El Kaliouby, R. (2005). Facial expression recognition using support vector machines. In The 10th International Conference on Human-Computer Interaction, Crete, Greece.

[15] Michel, P., & El Kaliouby, R. (2003, November). Real time facial expression recognition in video using support vector machines. In Proceedings of the 5th international conference on Multimodal interfaces (pp. 258-264). ACM.

[16] *MATLAB LOGIN: Matlab & Simulink*. MATLAB Login | MATLAB & Simulink. (n.d.). Retrieved February

27, 2023, from https://matlab.mathworks.com/

ANNEXURE

1. To Read an Image

```
w= imread(“frame2.bmp);
```

2. To Write an Image

```
imwrite(“frame2.bmp);
```

3. To Display an Image

```
w=imread(“frame5.bmp”)
imshow(w);
```

4. To Convert it in Grayscale Image

```
w=imread(“frame5.bmp”)

i=rgb2gray(w)

imshow(i);
```

5. To Read Pixels of an Image

```
i=imread(‘dowmload.jpg’)

img=imshow(‘download.jpg’)

impixelinfo(img);
```

6. Code for Snapping Images from Webcam

```
clear;
close all;
    camObj = webcam(1)
    preview(camObj);
i=1;
while (1)
    img = snapshot(camObj);
    imshow(img);
    filename=strcat('frame ',num2str(i),'.bmp');
    imwrite(img,filename);
if i==20
break;
    end
    i=i+1;
end
```

7. Code for Detection of Face through Image

```
    image = imread('aa.jpg');
    [width, height] = size(image);
    if width > 320
    image = imresize( image , [320 NaN]);
    end
    face_detection = vision.CascadeObjectDetector();
    location_of_face = step(face_detection , image);
    detected_image = insertShape(image, 'rectangle', location_of_face);
    figure;
imshow(detected_image);
```

8. Code for Real-Time Face Detection

```
close all;
cam=webcam();
cam.Resolution='640x480';
video_frame=snapshot(cam);
video_player=vision.VideoPlayer('Position',[100 100 640 480]);
face_Detector=vision.CascadeObjectDetector();
point_Tracker=vision.PointTracker('MaxBidirectionalError',2);
run_loop=true;
number_of_Points=0;
frame_Count=0;
while run_loop && frame_Count<400
 video_Frame=snapshot(cam);
 gray_Frame=rgb2gray(video_Frame);
 frame_Count=frame_Count+1;
if number_of_Points<10
     face_Rectangle=face_Detector.step(gray_Frame);
     if ~isempty(face_Rectangle)
points=detectMinEigenFeatures(gray_Frame,'ROI',face_Rectangle(1,:));
xy_Points=points.Location;
number_of_Points=size(xy_Points,1);
release(point_Tracker);
initialize(point_Tracker,xy_Points,gray_Frame);
previous_Points=xy_Points;
rectangle=bbox2points(face_Rectangle(1,:));
face_Polygon=reshape(rectangle',1,[]);
```

```
    video_Frame=insertShape(video_Frame,'Polygon',face_Polygon,'LineWidth',3);

    video_Frame=insertMarker(video_Frame,xy_Points,'+','Color','white');

    end

    else

        [xy_Points, isFound]=step(point_Tracker,gray_Frame);

        new_Points=xy_Points(isFound, :);

        old_Points=previous_Points(isFound, :);

        number_of_Points=size(new_Points,1);

        if number_of_Points>=10

    [xform,old_Points,new_Points]=estimateGeometricTransform(old_Points,new_Points,'simila
    rity','MaxDistance',4);

    rectangle=transformPointsForward(xform,rectangle);

    face_polygon=reshape(rectangle',1,[]);

    video_Frame=insertShape(video_Frame,'Polygon',face_polygon,'LineWidth',3);

    video_Frame=insertMarker(video_Frame,new_Points,'+','Color','white');

    previous_points=new_Points;

    setPoints(point_Tracker,previous_Points);

        end

end

step(video_player,video_Frame);

run_loop=isOpen(video_player);

    end

    clear cam;

    release(video_player);

    release(point_Tracker);

    release(face_Detector);
```

9. Code for Training Model

```
clear all;

close all;

warning off;

imds=imageDatastore('emotions','IncludeSubFolders',true,'LabelSource','foldernames');

trainingFeatures=[];

trainingLabels=imds.Labels;

for i = 1:numel(imds.Files)         % Read images using a for loop

    img = readimage(imds,i);

trainingFeatures(i,:)=extractLBPFeatures(rgb2gray(img));

end

Classifier =fitcecoc(trainingFeatures,trainingLabels);

save Classifier
```

10. Code for Testing the Model

```
   clear all;

   close all

   warning off;

   load Classifier;

   cao=webcam;

   faceDetector=vision.CascadeObjectDetector;

   while true

e=cao.snapshot;

bboxes =step(faceDetector,e);

if(~isempty(bboxes))

es=imcrop(e,bboxes);

es=imresize(es,[128 128]);

es=rgb2gray(es);

[Features] = extractLBPFeatures(es);

      PredictedClass=predict(Classifier,Features);

      PredictedClass=char(PredictedClass);

      imshow(e),title(PredictedClass);

      ax=gca;

      ax.TitleFontSizeMultiplier=1.5;

      pause(0.1);

      else

        imshow(e);

        ax=gca;
```

```
            title('Face Not Detected');

            ax.TitleFontSizeMultiplier=2;

            pause(0.1);

end

    end
```