FINAL REPORT
ON
"Image Processing in MATLAB"

A report submitted in partial fulfilment of the requirement for the award

ofThe degree of

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted to:

Mr. Sisir Kumar Jena

Assistant Professor

Submitted by:

Srijan Ghosh

Roll No:1000012219

Course-Section:CSE-IOT

Branch:IOT

Year(semester):2nd (IV th)

SCHOOL OF COMPUTING DIT UNIVERSITY, DEHRADUN

(State Private University through State Legislature Act No. 10 of 2013 of Uttarakhand and approved by UGC)

Mussoorie Diversion Road, Dehradun, Uttarakhand - 248009, India.

2021

A PROJECT REPORT
ON
"Image Processing in MATLAB"

BACHELOR OF TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING

Submitted to:
Mr. Sisir Kumar Jena, Assistant Professor

Submitted by :
Srijan Ghosh(CSE-IOT)
Year(semester):2nd (IV th)

TRAINING UNDERTAKEN AT: IRDE, DRDO

Under Guidance of : Mr Kanchan Chandra, Scientist "E"

INSTRUMENTS RESEARCH AND DEVELOPMENT ESTABLISHMENT
DRDO, MINISTRY OF DEFENCE, RAIPUR ROAD, DEHRADUN
MONTH: MAY-JULY, YEAR: 2021

DECLARATION

I here by certify that the work, which is being presented in the Summer Training Report, entitled **"Image Processing in MATLAB"**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Technology** and submitted to the DIT University is an authentic record of my work carried out during the period 28th May to 28th July from Instruments Research and Development Establishment(IRDE), DRDO lab, Dehradun under the guidance of Mr Kanchan Chandra, Scientist "E".

Date: 30/07/2021

Souyanghosh

Signature of the Candidate

K. Chandra.
30/07/21

Signature of Guide

ACKNOWLEDGEMENT

It was a great opportunity and learning experience for me to complete my semester internship at IRDE, DRDO,Dehradun.I am extremely grateful to the organization for the confidence bestowed in me during the internship. It gives me immense pleasure to express my regards and profound gratitude to my Guide Mr. Kanchan Chandra , Scientist 'E', and for his expert guidance during the course of my internship. I am greatly indebted for his consideration and his valuable time towards my projects and suggestions that helped me to shape my project to excellence. I would like to thank Mr Vaibhav Gupta(HR)for facilitating the internship at IRDE,DRDO,Dehradun. Besides, I am also thankful to Dr B.K. Das, Director, IRDE, for giving me opportunity and immense support to carry out this Internship

I would like to express my gratitude towards HOD, School of computing,DIT University and Internal Guide Mr Sisir Kumar Jena for their immense support

Name:Srijan Ghosh

Roll No. :190112004

SAP ID:1000012219

ABSTRACT

Instruments Research and Development Establishment (IRDE), a laboratory of Defense Research & Development Organization (DRDO), based in Dehradun, is devoted to research, design, development and technology transfer in optical and electro-optical instrumentation primarily for the Defense Services.

During the internship period Image processing in Matlab and GUI usage is conducted. Lots of concepts like Image Actualization, Zoom,Gain,Offset,histogram,histogram equalization etc is focused upon. GUI construction, with usage of Sliders,Buttons etc have been made. Finally a GUI is made which can be used for various operation om Image like changing colour content,RGB to grayscale,Zoom, Gain, offset, etc.

CERTIFCATE FROM COMPANY

Attach copy of certificate duly signed that candidate get after training

(In case of training hasn't been completed by the duration, you need to submit the ongoingtraining certificate from the company.)

This is to certify that Mr Srijan Ghosh, B. Tech CSE-IOT student of DIT university, Dehradun have successfully completed his Internship on Image Processing in MATLAB with GUI under the guidance of Mr Kanchan Chandra, Scientist 'E' at IRDE, DRDO in Dehradun during the period 28th May to 28th July,2021

During the internship, he was sincere, dedicated to work and performance was excellent.

I wish him a bright future.

Project Guide:

K. Chandra 30/07/21

(Kanchan Chandra)

Scientist 'E', IRDE, Dehradun

TABLE OF CONTENT

List of Figures

Chapter 1 -- Organization Overview

1.1 Introduction to the organization

The Instruments Research and Development Establishment (IRDE) was initially established as the establishment of Inspectorate of Scientific Stores in the year 1939 at Rawalpindi which is in present day Pakistan. At that time the organization used to examine tale-communication equipment which were being used by the Army in India. This AHSP went through many organizational changes and changes in in location. After such changes it became Technical Development Establishment (Instruments & Electronics) which covered AHSP and R&D functions used in the fields of Instruments and electronics. It was then located at Dehradun. In later years this establishment has shed some of its R&D and AHSP responsibilities. And then came to consistence as its present form I.e IRDE from 18 February 1960.

many optical instruments like binoculars, goggles, fire control instruments for small arms, mortars & field guns and fire control system for up-gunned T-55, Vijayanta & MBT Arjun tank has been developed by IRDE in past few years. IRDE has also developed Laser based instruments like Range Finders, designators, seekers and proximity sensors for various defence applications. IRDE is known for the development of image intensifier and thermal imager based night vision devices. Instruments Research and Development Establishment (IRDE), is devoted to research, design, development and technology transfer in optical and electro-optical instrumentation and image processing primarily for the all the three defence Services.

Figure 1.1 IRDE,DRDO,Dehradun

1.2 Services provided by company

Vision of Instruments R and D Establishment, Dehradun

To achieve excellence in the fields of Optics & Electro-optical Instrumentation with a commitment to provide high quality Instruments for defence Services.

Mission of Instruments R and D Establishment, Dehradun

- To design and develop state-of-the-art Night Vision Devices & Thermal Imagers.
- To design and develop compact Laser Based Instruments.
- To design and develop Integrated Electro-optical Surveillance and Fire Control Systems.
- To carry out research in the area of Photonics.

Chapter 2 –Introduction to Training

2.1 Introduction

My Internship was from IRDE,DRDO. Instruments Research and Development Establishment (IRDE), is devoted to research, design, development and technology transfer in optical and electro-optical instrumentation and image processing primarily for the all the three defence Services.Vision of the Organization is to achieve excellence in the fields of Optics & Electro-optical Instrumentation with a commitment to provide high quality Instruments for defence Services.many optical instruments like binoculars, goggles, fire control instruments for small arms, mortars & field guns and fire control system for up-gunned T-55, Vijayanta & MBT Arjun tank has been developed by IRDE in past few years. IRDE has also developed Laser based instruments like Range Finders, designators, seekers and proximity sensors for various defence applications. IRDE is known for the development of image intensifier and thermal imager based night vision devices.The Online internship started from 28th may,2021 to 28th July, 2021.This was a 8 week online Internship. Introduction to Image Processing was covered in first two weeks.During this time Image, Image formation, Pixel, focal plane array etc other introductory topics were explained. Next I was introduced to the types of images, Images stored as matrix and coding exercise related to it were covered. I was also Introduced to Graphical User Interface of Matlab which helped me to make few GUIs.

I was introduced to different operations on images like RGB to grayscale, Crop,Zoom,Colour content,offset,gain,Histogram equalization etc. Making of GUI was the best part.Graphical user interface provides the user an interactive environment.A graphical user interface (GUI) is an interface through which a user interacts with electronic devices such as computers and smartphones through the use of icons, menus and other visual indicators or representations (graphics). GUIs graphically display information and related user controls, unlike text-based interfaces, where data and commands are strictly in text. GUI representations are manipulated by a pointing device such as a mouse, trackball, stylus, or by a finger on a touch screen.Once the GUI is created the user need not know anything about the coding section. Using GUI we can perform any computations, communicate with any other UI's, create tables etc. MATLAB GUI contains several user interface tools like radio buttons,axes,check box, tables, sliders,list box,panels..etc. GUI interface is an event driven programming. Flow of the application is based

on events. Events may be click,double click,key press etc. Callback functions will be executed once an event occurs. We can edit the properties of each callback functions for making suitable response from the GUI as the user interacts.

I have made different GUIs for conversion of RGB to grayscale, Changing colour content,Zooming Image,Changing Gain, histogram equalization etc

At last All the GUIs were made into one GUI.The experience was Very well. I came across many coding challenges which improved my skills a lot.

2.2 Motivation

Photography is one of my hobbies. I used to click a lot of Images. But as we know we can't have perfect images all the times, thus I used to use Editing software.So I felt that I need to know what is the process behind these software. This was my Initial Motivation. I then tried to Search about this online. Then I came across Image processing.I found that Image processing has a lot to it as compared to Just editting Images.It can be used in Military application, face recognition, space exploration,Medical Imaging, etc. These application were fascinating. So I decided to do Internship on Image Processing. I came to Know that IRDE,DRDO can provide me with Internship on It. I was excited as IRDE,DRDO is one of the esteemed organization. Although due to COVID, Everything was online. But having a DRDO scientist as a Guide was a big Motivation

The Guide gave a lot of Motivation by giving us coding challenges,helping us to understand not only the programming aspect but also the real life application and the theory behind it.MATLAB was a new platform for me.But the Guide Motivated me a lot So I was able to successfully complete this Internship.

2.3 Scope

Image processing aims to transform an image into digital form and performs some process on it, to get an enhanced image or take some utilized information from it. It is a method that develops to convert the image into digital form and perform some operations to obtain specific models or to extract useful information from it. The input of this method is a video section or an image, such as a photograph. The output corresponds to the desired or attention part of the picture. Generally, the Image Processing system treats images as two-dimensional signals when applying predetermined

signal processing methods.There are different purposes of image processing like visualization i.e Observing objects that are difficult to see, image sharpening and restoration i.e Improving noisy images.image retrieval i.e Attractive and high-resolution image search, pattern recognition i.e Defining various objects in an image, image recognition i.e Detecting objects in an image etc. Some of the important applications of image processing in the field of science and technology include computer vision, remote sensing, feature extraction, face detection, forecasting, optical character recognition, finger-print detection, optical sorting, argument reality, microscope imaging, lane departure caution system, Non-photorealistic representation, medical image processing, and morphological imaging.Some Important Application are as follows:

1. Image Sharpening and Resolution

2. Medical field

   - Gamma-ray imaging
   - PET scan
   - X-Ray Imaging
   - Medical CT scan
   - UV imaging

3. Robot Vision

4. Pattern Recognition

5. Video Processing

2.4 Correlation to Course Subject

Since Image Processing is a Coding based work so it is related to the programming concepts of IB201, CS101 and CS221, although they are on Java, C and python respectively, but the application of loops, conditional statements, etc other programming concept were of a lot of help. The use of matrix, understanding its concept etc were also required for which MA102 Engineering Mathematics-II was helpful. These were the subjects of 1st and 2nd year which were a help in this Internship.

In the subjects of the next semesters this Internship will be of help. Computer Graphics the subject that will be taught in next semesters which is co-related to Image processing in MATLAB with GUI. mage processing is a subset of computer vision. A computer vision system uses the image processing algorithms to try and perform emulation of vision at human scale.

The fields of computer vision and computer graphics deal with the acquisition, processing, analysis and rendering of visual information in different representations such as images, videos, volumetric data, and 3D models. This includes disciplines such as image processing, medical imaging, video processing, pattern recognition, visualization, virtual and augmented reality, computer games, and includes aspects of human computer interaction and machine learning.

Chapter 3 –Modules of the Training

1. **Introduction**
   - **Image**
   - **Pixel**
   - **Focal plane array**
   - **Formation of image**
   - **Image processing definition**
2. **Types of image**
   - **Binary Image**
   - **Grayscale Image**
   - **Coloured Image**
   - **Coding Exercise**
3. **RGB to Grayscale conversion**
   - **Average Method**
   - **Weighted Method**
   - **Matlab Function**
   - **Coding Exercise**
4. **Crop,Resize and Zoom operation on an Image**
   - **Crop**
   - **Resize**
   - **Zoom**
   - **Optical Zoom Vs Digital Zoom**
   - **Coding Exercise**
5. **Gain,offset and Histogram equalization**
   - **Histogram**
   - **Gain**
   - **Offset**
   - **Histogram Equalization**
   - **Coding Exercise**

6. Edge detection

- **Definition**
- **Types**
- **Matlab Function**
- **Coding Exercise**

Chapter 4- Technologies/Concepts Learned

4.1 Software & Hardware Requirements

The software used in the Image processing is MATLAB.MATLAB is a proprietary multi-paradigm programming language and numeric computing environment developed by MathWorks.

4.1.1 Getting started with MATLAB:

It is both a programming language as well as a programming environment. It allows the computation of statements in the command window itself.

4.1.2 Command Window:

In this window one must type and immediately execute the statements, as it requires quick prototyping. These statements cannot be saved. Thus, this is can be used for small, easily executable programs.

4.1.3 Editor (Script):

In this window one can execute larger programs with multiple statements, and complex functions These can be saved and are done with the file extension '.m'

4.1.4 Workspace:

In this window the values of the variables that are created in the course of the program (in the editor) are displayed.

This window displays the exact location(path) of the program file being created.

MATLAB Library comes with a set of many inbuilt functions. These functions mostly perform mathematical operations like sine, cosine and tangent. They perform more complex functions too like finding the inverse and determinant of a matrix, cross product and dot product

Although MATLAB is encoded in C, C++ and Java, it is a lot easier to implement than these three languages. For example, unlike the other three, no header files need to be initialised in the beginning of the document and for declaring a variable, the data type need not be provided. It provides an easier alternative for vector operations. They can be performed using one command instead of multiple statements in a for or while loop.

Writing a MATLAB program:

4.1.5 Using Command Window:

Only one statement can be typed and executed at a time. It executes the statement when the enter key is pressed. This is mostly used for simple calculations.

4.1.6 Using Editor:

Multiple lines of code can be written here and only after pressing the run button (or F5) will the code be executed. It is always a good practice to write clc, clear and close all in the beginning of the program.

Any statement followed by % in MATLAB is considered as a comment

4.1.7 Vector Operations:

Operations such as addition, subtraction, multiplication and division can be done using a single command instead of multiple loops We can also extract separate rows and columns by using the colon(:) operator. Consider a matrix A of size 3X3.

4.2 Implementation Details and screenshots

4.2.1 Introduction:

Before Starting Image processing one must know about Image. Image is a visual representation or a picture produced in an electronic display. An Image is made up of pixels.Pixels are the smallest unit of an image.A pixel is the basic logical unit in digital graphics. Pixels are combined to form a complete image.Image are represented in matrix.

4.2.2 How is Image formed?

The light reflected from a image falls on the focal plane array of a Camera. Focal plane array is a image sensor consisting of an array of light sensing pixels at the focal plane of the lens.An image sensor is a semiconductor device that can convert optical images into digital signals. It is widely used in digital cameras and other electronic optical devices. The image sensor uses the photoelectric conversion function of the photoelectric device to convert the light image on the photosensitive surface into an electrical signal in a proportional relationship with the light image.

The photosensitive device is the core component of the industrial camera. There are two kinds of image sensors which are CMOS (Complementary Metal Oxide Semiconductor) and CCD (Charged Coupled Device).The CMOS sensor uses the most commonly used CMOS process of general semiconductor circuits. It has the characteristics of high integration, low power consumption, fast speed, low cost, etc. In recent years, it has developed rapidly in a wide dynamic range and low illumination. CCD and CMOS have their advantages in different application scenarios.

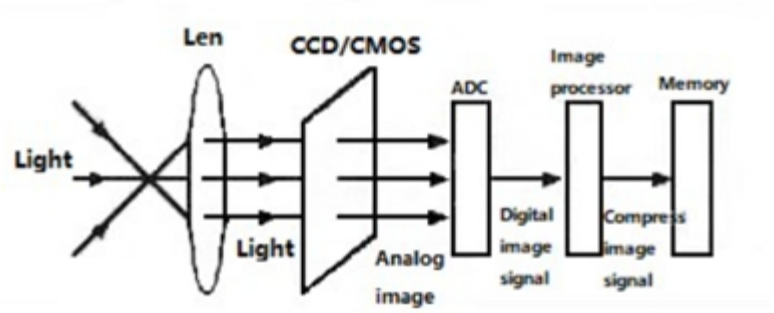


Figure 4.1 Working process of image sensor

An image sensor is a semiconductor device that can convert optical images into digital signals. The tiny photosensitive material implanted on the sensor is called a pixel. The more pixels there are on a sensor, the higher the resolution of the picture it provides. It functions as a film, but it converts image pixels into digital signals.

The manufacturing technology of CMOS is no different from that of general computer chips. It is mainly a semiconductor made of silicon and germanium so that it has a negatively charged N pole and a positively charged P pole semiconductor coexisting on the CMOS. The currents generated by the two complementary effects can be recorded and converted into images by the processing chip. Later, it was discovered that CMOS can also be used as an image sensor in digital photography after processing. CMOS is a complementary metal-oxide-semiconductor, which is mainly a semiconductor made of two elements of silicon and germanium and realizes basic functions through negatively and positively charged transistors on CMOS. The current generated by these two complementary effects can be recorded and interpreted as an image by the processing chip. A CMOS image sensor is a typical solid-state imaging sensor and has a common historical origin with CCD. CMOS image sensors are usually composed of image sensor cell array, row driver, column driver, timing control logic, AD converter, data bus

output interface, control interface, etc. These parts are usually integrated on the same silicon chip. The working process can be divided into reset, photoelectric conversion, integration, and readout.

A CMOS image sensor is a typical solid-state imaging sensor and has a common historical origin with CCD. CMOS image sensors are usually composed of image sensor cell array, row driver, column driver, timing control logic, AD converter, data bus output interface, control interface, etc. These parts are usually integrated on the same silicon chip. The working process can be divided

into reset, photoelectric conversion, integration, and readout.

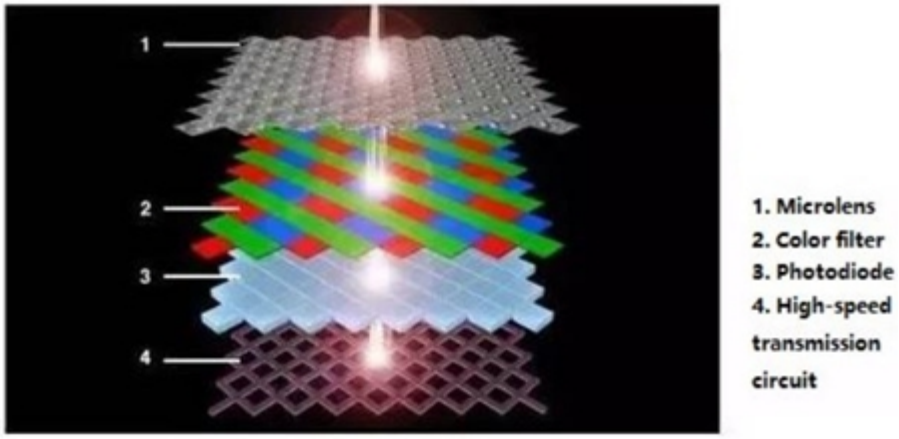


Figure 4.2 Composition of CMOS sensor

4.2.3 Image Processing:

Image processing consists of the manipulation of images using digital computers. Its use has been increasing exponentially in the last decades. Its applications range from medicine to entertainment, passing by geological processing and remote sensing. Multimedia systems, one of the pillars of the modern information society, rely heavily on digital image processing.

The purpose of image processing is to extract some information from an image or prepare it for a certain task, such as viewing, printing, or transmitting. Image quality may be poor, so the image may need to be preprocessed.

Preprocessing can be in done in spatial or frequency domain using a variety of techniques. Frequency-domain processing can be implemented through a Fourier transform. For example, if the camera was moved while

taking the image, a restoration technique may be needed to eliminate motion blur or defocus. After noise has been reduced or the image has been

enhanced in some way, one can look closely at an object of interest in the image. Thresholding is a simple way to isolate (segment) objects in an image if the background has very low intensity. On the other hand, one may be

interested in finding points, lines, or other features in the image, such as intensity, color, shape, or texture of an object. The Hough transform will help find lines in an image that intersect at different points. In some noise-removal cases, the noise is of aspecific shape, e.g., in the case where there are never more than four consecutive image pixels in the area. Erosion and dilation

offer a powerful spatial-domain technique to remove noise from an image. Often, it may be desirable to zoom in to some detailed parts of an image to observe something interesting. The more sophisticated multi resolution-analysis power of the wavelet transform is needed in these cases. With the advent of the Internet and the widespread use of pictures and movies, the need for image compression is greater than ever. Volumes of data must be stored, retrieved, and transmitted quickly, and clever compression methods save time and space.

4.2.4 Types of Images

a) Binary Images: It is the simplest type of image. It takes only two values i.e, Black and White or 0 and 1. The binary image consists of a 1-bit image and it takes only 1 binary digit to represent a pixel. Binary images are mostly used for general shape or outline.Binary images are generated using threshold operation. When a pixel is above the threshold value, then it is turned white('1') and which are below the threshold value then they are turned black('0')

b) Grayscale Images: Grayscale images are monochrome images, Means they have only one color. Grayscale images do not contain any information about color. Each pixel determines available different gray levels.A normal grayscale image contains 8 bits/pixel data, which has 256 different gray levels. In medical images and astronomy, 12 or 16 bits/pixel images are used.

c) Coloured Images: Colour images are three band monochrome images in which, each band contains a different color and the actual information is stored in the digital image. The color images contain gray level information in each spectral band.The images are represented as red, green and blue (RGB images). And each color image has 24 bits/pixel means 8 bits for each of the three color band(RGB).

Figure 4.3 Binary Image

Figure 4.4 Grayscale Image

Figure 4.5 Coloured Image

4.2.5 Coding Exercise 1 : Create a matrix with 480 rows and 640 column to display a cross in center.

Code:

```
I=zeros(480,640);
I(239:241,295:345)=1;
I(215:265,319:321)=1;
figure
imshow(I)
```

OUTPUT-

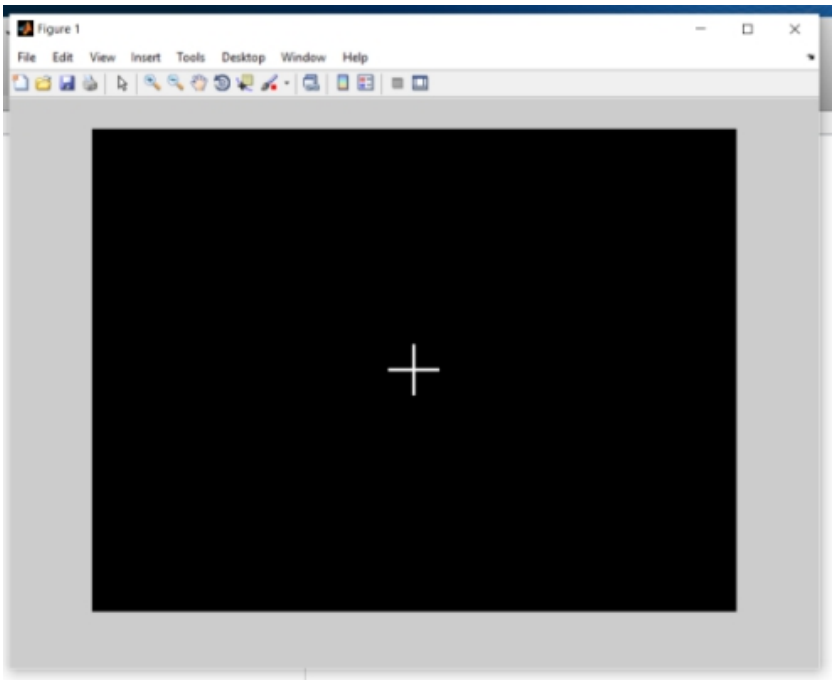


Figure 4.6 Output of coding Exercise 1

4.2.6 Coding Exercise 2 : Write a GUI based program which generates different colors with changing RGB values using sliders.

Code:

```
function varargout = excercise2(varargin)
% EXCERCISE2 MATLAB code for excercise2.fig
%      EXCERCISE2, by itself, creates a new EXCERCISE2 or raises the existing
```

```
%      singleton*.
%
%      H = EXCERCISE2 returns the handle to a new EXCERCISE2 or the handle to
%      the existing singleton*.
%
%      EXCERCISE2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in EXCERCISE2.M with the given input arguments.
%
%      EXCERCISE2('Property','Value',...) creates a new EXCERCISE2 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before excercise2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to excercise2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help excercise2
% Last Modified by GUIDE v2.5 25-Jul-2021 11:45:22
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @excercise2_OpeningFcn, ...
                   'gui_OutputFcn',  @excercise2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before excercise2 is made visible.
function excercise2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to excercise2 (see VARARGIN)
% Choose default command line output for excercise2
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes excercise2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = excercise2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on slider movement.
function RedSlide_Callback(hObject, eventdata, handles)
global Red;
Red = get(handles.RedSlide,'value');
```

```
set(handles.edit7,'String',Red);
global Blue;
red_i = ones(480,640);
red_i=red_i*Red;
global color_i;
color_i = ones(480,640,3);
global Green;
blue_i = ones(480,640);
blue_i=blue_i*Blue;
green_i = ones(480,640);
green_i=green_i*Green;
color_i(:,:,1)= red_i;
color_i(:,:,2) =green_i;
color_i(:,:,3) =blue_i;
imshow(color_i);
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function RedSlide_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RedSlide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function BlueSlide_Callback(hObject, eventdata, handles)
% hObject    handle to BlueSlide (see GCBO)
global Blue
global color_i;
```

```
Blue = get(handles.BlueSlide,'value');
set(handles.edit9,'String',Blue);
global Red;
red_i = ones(480,640);
red_i=red_i*Red;
color_i = ones(480,640,3);
global Green;
blue_i = ones(480,640);
blue_i=blue_i*Blue;
green_i = ones(480,640);
green_i=green_i*Green;
color_i(:,:,1)= red_i;
color_i(:,:,2) =green_i;
color_i(:,:,3) =blue_i;
imshow(color_i);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function BlueSlide_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BlueSlide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function GreenSlide_Callback(hObject, eventdata, handles)
% hObject    handle to GreenSlide (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
global Green;
global color_i;
Green = get(handles.GreenSlide,'value');
set(handles.edit8,'String',Green);
global Red;
red_i = ones(480,640);
red_i=red_i*Red;
color_i = ones(480,640,3);
global Blue;
blue_i = ones(480,640);
blue_i=blue_i*Blue;
green_i = ones(480,640);
green_i=green_i*Green;
color_i(:,:,1)= red_i;
color_i(:,:,2) =green_i;
color_i(:,:,3) =blue_i;
imshow(color_i);
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function GreenSlide_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GreenSlide (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function edit1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes during object creation, after setting all properties.
function axes2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes2
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
```

```
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit5_Callback(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit5 as text
%        str2double(get(hObject,'String')) returns contents of edit5 as a double
% --- Executes during object creation, after setting all properties.
function edit5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
% --- Executes during object creation, after setting all properties.
```

```
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes during object creation, after setting all properties.
function axes1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to axes1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: place code in OpeningFcn to populate axes1
function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double
% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double
% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

end

OUTPUT-

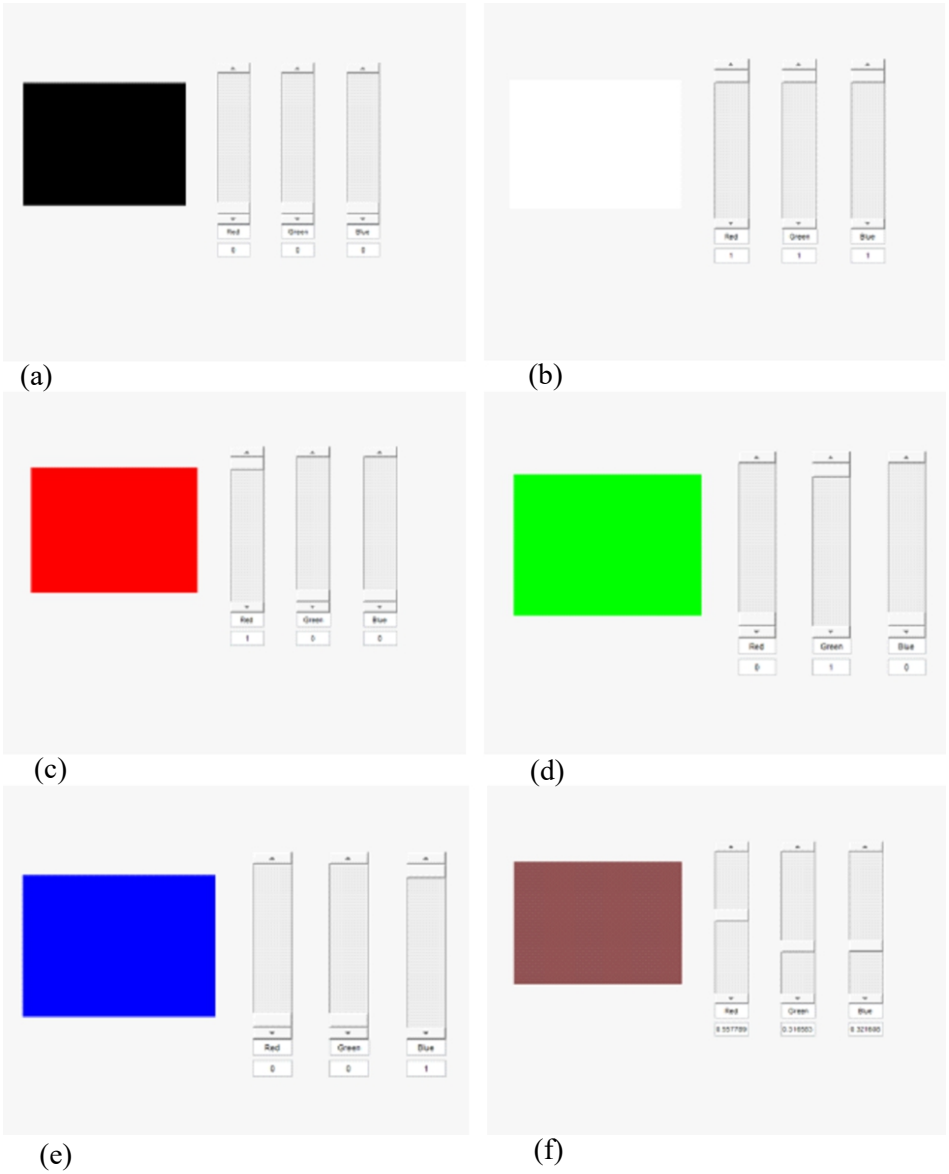


Figure 4.7 Output of coding Exercise 2- (a)When all the colour components are zero then black colour appear. (b)When all the colour components are one then white colour appear. (c),(d),(e) when one is high their corresponding color appear and (f) a random colour

4.2.7 Coding Exercise 3 : Display image in MATLAB

Code:

```
clear all
I=imread('C:\Users\DELL\Desktop\lab-image.jpg');
imshow(I);
```

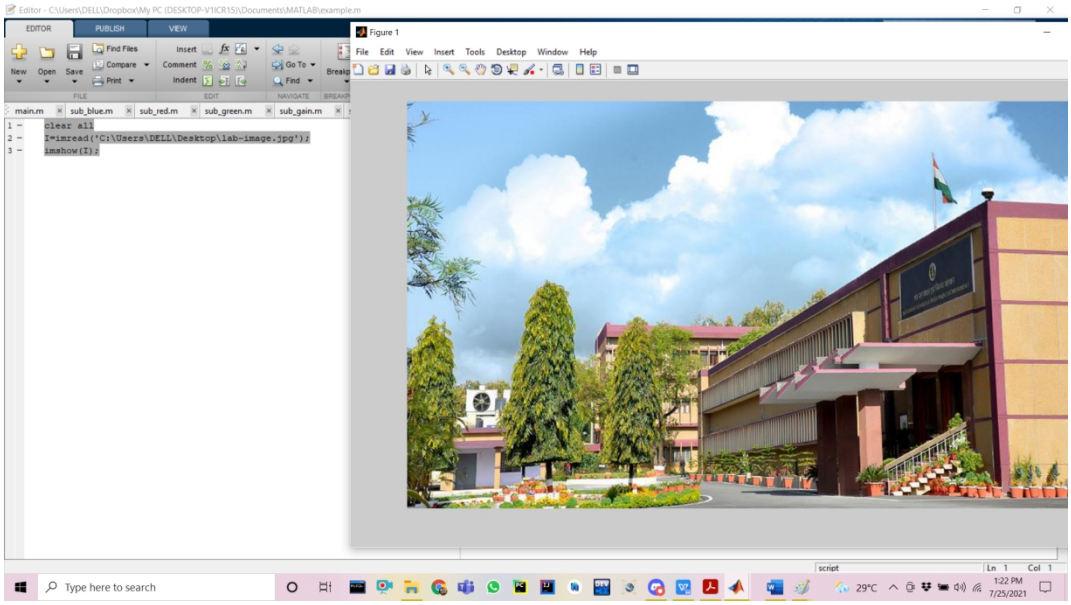


Figure 4.8 Output of coding Exercise 3

4.2.8 Coding Exercise 4 : Write a GUI based program with menu to select any image from system and then change its RGB contents using Sliders.

Code:

```
function varargout = excercise3(varargin)
% EXCERCISE3 MATLAB code for excercise3.fig
%      EXCERCISE3, by itself, creates a new EXCERCISE3 or raises the existing
%      singleton*.
%
%      H = EXCERCISE3 returns the handle to a new EXCERCISE3 or the handle to
%      the existing singleton*.
%
%      EXCERCISE3('CALLBACK',hObject,eventData,handles,...) calls the local
```

```
%      function named CALLBACK in EXCERCISE3.M with the given input arguments.
%
%      EXCERCISE3('Property','Value',...) creates a new EXCERCISE3 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before excercise3_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to excercise3_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help excercise3
% Last Modified by GUIDE v2.5 23-Jun-2021 10:40:04
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @excercise3_OpeningFcn, ...
                   'gui_OutputFcn',  @excercise3_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
```

```
% --- Executes just before excercise3 is made visible.
function excercise3_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to excercise3 (see VARARGIN)
% Choose default command line output for excercise3
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes excercise3 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = excercise3_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in togglebutton1.
function togglebutton1_Callback(hObject, eventdata, handles)
% hObject    handle to togglebutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of togglebutton1
% --- Executes on slider movement.
function slider_red_Callback(hObject, eventdata, handles)
% hObject    handle to slider_red (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
value_red = get(handles.slider_red,'value');
set(handles.red_number,'string',value_red);
I=handles.ii;
I =(im2double(I));
I_red =value_red.*I(:,:,1);
I_green =I(:,:,2);
I_blue = I(:,:,3);
Im(:,:,1)= I_red;
Im(:,:,2)= I_green;
Im(:,:,3)= I_blue;
imshow(Im);
handles.ii= Im;
guidata(hObject, handles);
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider_red_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_red (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function slider_green_Callback(hObject, eventdata, handles)
% hObject    handle to slider_green (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
value_green = get(handles.slider_green,'value');
```

```
set(handles.green_number,'string',value_green);
I=handles.ii;
I =(im2double(I));
I_red =I(:,:,1);
I_green =value_green.*I(:,:,2);
I_blue = I(:,:,3);
Im(:,:,1)= I_red;
Im(:,:,2)= I_green;
Im(:,:,3)= I_blue;
imshow(Im);
handles.ii= Im;
guidata(hObject, handles);
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider_green_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_green (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function slider_blue_Callback(hObject, eventdata, handles)
% hObject    handle to slider_blue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
value_blue = get(handles.slider_blue,'value');
set(handles.blue_number,'string',value_blue);
I=handles.ii;
```

```
I =(im2double(I));
I_red =I(:,:,1);
I_green =I(:,:,2);
I_blue = value_blue.*I(:,:,3);
Im(:,:,1)= I_red;
Im(:,:,2)= I_green;
Im(:,:,3)= I_blue;
imshow(Im);
handles.ii= Im;
guidata(hObject, handles);
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider_blue_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider_blue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
[f d]=uigetfile({'*.jpg;*.tif;*.png;*.gif'},'Select an Image');
file_name=strcat(d,f);
I=imread(file_name);
handles.ii=I;
imshow(I);
```

```
guidata(hObject, handles);
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function red_number_Callback(hObject, eventdata, handles)
% hObject    handle to red_number (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of red_number as text
%        str2double(get(hObject,'String')) returns contents of red_number as a double
% --- Executes during object creation, after setting all properties.
function red_number_CreateFcn(hObject, eventdata, handles)
% hObject    handle to red_number (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor','white');
end
function blue_number_Callback(hObject, eventdata, handles)
% hObject    handle to blue_number (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of blue_number as text
%        str2double(get(hObject,'String')) returns contents of blue_number as a double
% --- Executes during object creation, after setting all properties.
function blue_number_CreateFcn(hObject, eventdata, handles)
% hObject    handle to blue_number (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% --- Executes during object creation, after setting all properties.
function green_number_CreateFcn(hObject, eventdata, handles)
% hObject    handle to green_number (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
function green_number_Callback(hObject, eventdata, handles)
% hObject    handle to green_number (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of green_number as text
%        str2double(get(hObject,'String')) returns contents of green_number as a double
```

OUTPUT-

Figure 4.9 Output of coding Exercise 4

4.2.9 RGB to grayscale conversion

There are two methods to convert it. Both has their own merits and demerits. The methods are:

- Average method
- Weighted method or luminosity method

Average method

Average method is the most simple one. You just have to take the average of three colors. Since its an RGB image, so it means that you have add r with g with b and then divide it by 3 to get your desired grayscale image.

Its done in this way.

Grayscale = (R + G + B / 3)

For example:

Figure 4.10 RGB to Grayscale by Average method

If you have an color image like the image shown above and you want to convert it into grayscale using average method. The following result would appear.

Explanation

There is one thing to be sure, that something happens to the original works. It means that our average method works. But the results were not as expected. We wanted to convert the image into a grayscale, but this turned out to be a rather black image.

Problem

This problem arise due to the fact, that we take average of the three colors. Since the three different colors have three different wavelength and have their own contribution in the formation of image, so we have to take average according to their contribution, not done it averagely using average method. Right now what we are doing is this,

33% of Red, 33% of Green, 33% of Blue

We are taking 33% of each, that means, each of the portion has same contribution in the image. But in reality thats not the case. The solution to this has been given by luminosity method.

Weighted method or luminosity method

You have seen the problem that occur in the average method. Weighted method has a solution to that problem. Since red color has more wavelength of all the three colors, and green is the color that has not only less wavelength then red color but also green is the color that gives more soothing effect to the eyes.

It means that we have to decrease the contribution of red color, and increase the contribution of the green color, and put blue color contribution in between these two.

So the new equation that form is:

New grayscale image = ((0.3 * R) + (0.59 * G) + (0.11 * B)).

According to this equation, Red has contribute 30%, Green has contributed 59% which is greater in all three colors and Blue has contributed 11%.

Applying this equation to the image, we get this

Figure 4.11 RGB to Grayscale by Weighted method

Explanation

As you can see here, that the image has now been properly converted to grayscale using weighted method. As compare to the result of average method, this image is more brighter.

Function

Matlab has an inbuilt function for conversion of RGB to Grayscale

Syntax

I = rgb2gray(RGB)

I = rgb2gray(RGB) converts

the truecolor image RGB to the grayscale intensity image I. rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

Examples

Convert an RGB image to a grayscale image.I = imread('board.tif');

J = rgb2gray(I);

figure, imshow(I), figure, imshow(J);

4.2.10 Coding Exercise 5 : Write a GUI based program with menu to select any image from system to convert RGB to greyscale using Mathematical formula in which the percentage of RGB values are changed with sliders to find a perfect ratio of RGB in the equation.

Code:

```
function varargout = excercise_4_22_jun(varargin)
% EXCERCISE_4_22_JUN MATLAB code for excercise_4_22_jun.fig
%      EXCERCISE_4_22_JUN, by itself, creates a new EXCERCISE_4_22_JUN or raises the existing
%      singleton*.
%
%      H = EXCERCISE_4_22_JUN returns the handle to a new EXCERCISE_4_22_JUN or the handle to
%      the existing singleton*.
%
%      EXCERCISE_4_22_JUN('CALLBACK',hObject,eventData,handles,...) calls the local
%          function named CALLBACK in EXCERCISE_4_22_JUN.M with the given input arguments.
%
%      EXCERCISE_4_22_JUN('Property','Value',...) creates a new EXCERCISE_4_22_JUN or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before excercise_4_22_jun_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to excercise_4_22_jun_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help excercise_4_22_jun
% Last Modified by GUIDE v2.5 25-Jun-2021 10:55:12
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @excercise_4_22_jun_OpeningFcn, ...
                   'gui_OutputFcn',  @excercise_4_22_jun_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before excercise_4_22_jun is made visible.
function excercise_4_22_jun_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to excercise_4_22_jun (see VARARGIN)
% Choose default command line output for excercise_4_22_jun
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes excercise_4_22_jun wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
```

```
function varargout = excercise_4_22_jun_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
global I
[f d]=uigetfile;
file_name=strcat(d,f);
I=imread(file_name);
imshow(I);
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on slider movement.
function RED_SLIDE_Callback(hObject, eventdata, handles)
% hObject    handle to RED_SLIDE (see GCBO)
global value_red;
global value_green;
global value_blue;
global I;
global I_bw_calculation;
value_red = get(handles.RED_SLIDE,'value');
value_green = get(handles.GREEN_SLIDE,'value');
value_blue = get(handles.BLUE_SLIDE,'value');
RED_PERCENT = (value_red./(value_red + value_blue + value_green));
GREEN_PERCENT = (value_green./(value_red + value_blue + value_green));
BLUE_PERCENT = (value_blue./(value_red + value_blue + value_green));
```

```
set(handles.RED,'String',RED_PERCENT);
set(handles.GREEN,'String',GREEN_PERCENT);
set(handles.BLUE,'String',BLUE_PERCENT);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function RED_SLIDE_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RED_SLIDE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function GREEN_SLIDE_Callback(hObject, eventdata, handles)
% hObject    handle to GREEN_SLIDE (see GCBO)
global value_red;
global value_green;
global value_blue;
global I;
value_red = get(handles.RED_SLIDE,'value');
value_green = get(handles.GREEN_SLIDE,'value');
value_blue = get(handles.BLUE_SLIDE,'value');
RED_PERCENT = (value_red./(value_red + value_blue + value_green));
GREEN_PERCENT = (value_green./(value_red + value_blue + value_green));
BLUE_PERCENT = (value_blue./(value_red + value_blue + value_green));
set(handles.RED,'String',RED_PERCENT);
set(handles.GREEN,'String',GREEN_PERCENT);
```

```
set(handles.BLUE,'String',BLUE_PERCENT);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function GREEN_SLIDE_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GREEN_SLIDE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function BLUE_SLIDE_Callback(hObject, eventdata, handles)
% hObject    handle to BLUE_SLIDE (see GCBO)
global value_red;
global value_green;
global value_blue;
global I;
value_red = get(handles.RED_SLIDE,'value');
value_green = get(handles.GREEN_SLIDE,'value');
value_blue = get(handles.BLUE_SLIDE,'value');
global RED_PERCENT;
global GREEN_PERCENT;
global BLUE_PERCENT;
RED_PERCENT = (value_red./(value_red + value_blue + value_green));
GREEN_PERCENT = (value_green./(value_red + value_blue + value_green));
BLUE_PERCENT = (value_blue./(value_red + value_blue + value_green));
set(handles.RED,'String',RED_PERCENT);
```

```
set(handles.GREEN,'String',GREEN_PERCENT);
set(handles.BLUE,'String',BLUE_PERCENT);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function BLUE_SLIDE_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BLUE_SLIDE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function RED_Callback(hObject, eventdata, handles)
% hObject    handle to RED (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of RED as text
%        str2double(get(hObject,'String')) returns contents of RED as a double
% --- Executes during object creation, after setting all properties.
function RED_CreateFcn(hObject, eventdata, handles)
% hObject    handle to RED (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function GREEN_Callback(hObject, eventdata, handles)
% hObject    handle to GREEN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of GREEN as text
%        str2double(get(hObject,'String')) returns contents of GREEN as a double
% --- Executes during object creation, after setting all properties.
function GREEN_CreateFcn(hObject, eventdata, handles)
% hObject    handle to GREEN (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function BLUE_Callback(hObject, eventdata, handles)
% hObject    handle to BLUE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of BLUE as text
%        str2double(get(hObject,'String')) returns contents of BLUE as a double
% --- Executes during object creation, after setting all properties.
function BLUE_CreateFcn(hObject, eventdata, handles)
% hObject    handle to BLUE (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
```

```
end
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
global I;
global RED_PERCENT;
global GREEN_PERCENT;
global BLUE_PERCENT;
I_bw_calculation=(((I(:,:,1)).*RED_PERCENT)+((I(:,:,2)).*GREEN_PERCENT)+((I(:,:,3)).*BL
UE_PERCENT));
imshow(I_bw_calculation);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

OUTPUT-

Figure 4.12 Output of coding Exercise 5

Observation:

Best ratio: (approx value)

Red:0.301037

Green:0.590472

Blue:0.108491

4.2.11 Crop,Resize and Zoom operation on an Image

a) **Crop:** To "crop" an image is to remove or adjust the outside edges of an image (typically a photo) to improve framing or composition, draw a viewer's eye to the image subject, or change the size or aspect ratio. In other words, image cropping is the act of improving a photo or image by removing the unnecessary parts.

b) **Resize:** Resize is to change the size of a image.Image interpolation occurs when you resize or distort your image from one pixel grid to another. Image resizing is necessary when you need to increase or decrease the total number of pixels, whereas remapping can occur when you are correcting for lens distortion or rotating an image.

c) **Zoom:** Zooming simply means enlarging a picture in a sense that the details in the image became more visible and clear. Zooming an image has many wide applications ranging from zooming through a camera lens, to zoom an image on internet e.t.c.

d) Optical Zoom vs digital Zoom

These two types of zoom are supported by the cameras.

- **Optical Zoom:** The optical zoom is achieved using the movement of the lens of your camera. An optical zoom is actually a true zoom. The result of the optical zoom is far better then that of digital zoom. In optical zoom, an image is magnified by the lens in such a way that the objects in the image appear to be closer to the camera. In optical zoom the lens is physically extend to zoom or magnify an object.

- **Digital Zoom:** Digital zoom is basically image processing within a camera. During a digital zoom, the center of the image is magnified and the edges of the picture got crop out. Due to magnified center, it looks like that the object is closer to you.During a digital zoom, the pixels got expand , due to which the quality of the image is compromised.The same effect of digital zoom can be seen after the image is taken through your computer by using an image processing toolbox / software, such as Photoshop.The following picture is the result of digital zoom done through one of the following methods given below in the zooming methods.

4.2.12 Coding exercise 6 : write a code to crop an Image

Code:

```
clear all
I=imread('C:\Users\DELL\Desktop\lab-image.jpg');
I_gray=rgb2gray(I);
imshow(I_gray,[]);
[R C]=size(I_gray);
I_c=I((R./4):(3.*R./4), (C./4):(3.*C./4));
figure
imshow(I_c, [])
```

OUTPUT-

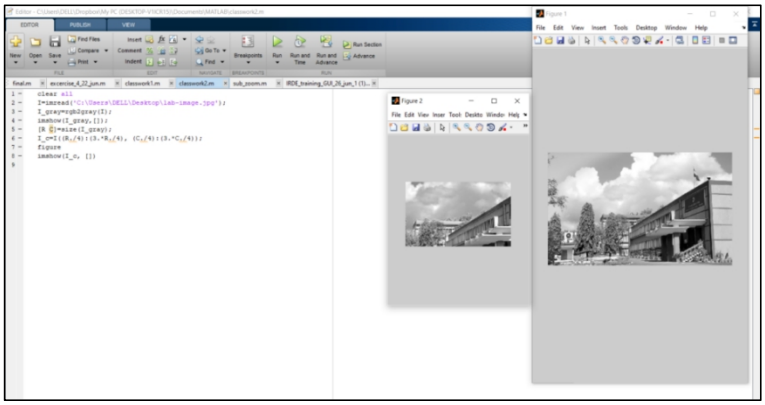


Figure 4.13 Output of coding Exercise 6

4.2.13 Coding exercise 7 : Write a program to zoom an image.

Code:

```
clear all
I=imread('C:\Users\DELL\Desktop\lab-image.jpg');
I_gray=rgb2gray(I);
imshow(I_gray,[]);
[R C]=size(I_gray);
I_c=I((R./4):(3.*R./4), (C./4):(3.*C./4));
I_z=imresize(I_c,2);
figure
imshow(I_z,[])
```

OUTPUT-

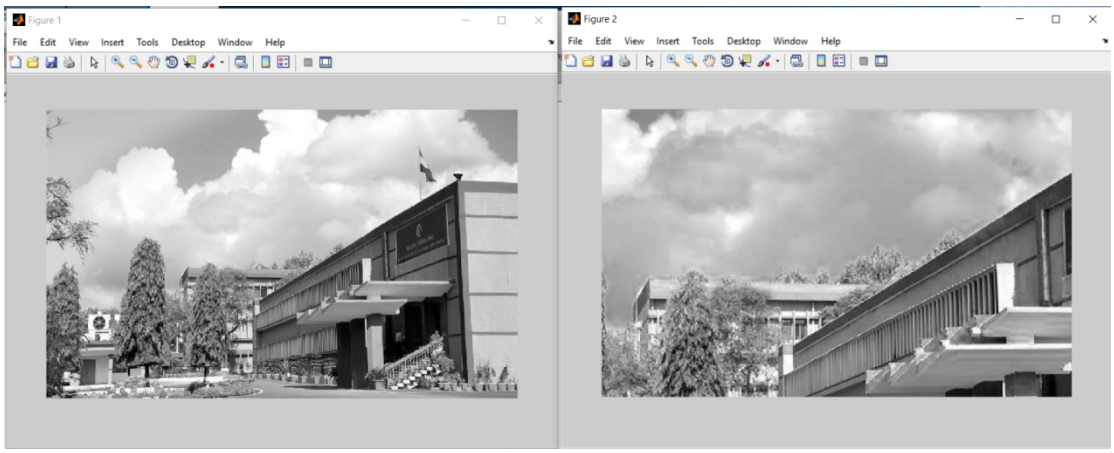


Figure 4.14 Output of coding Exercise 7

4.2.14 Coding exercise 8 : Write a GUI based program with menu to select any image from system and zoom it.

Code:

```
function varargout = june27(varargin)
% JUNE27 MATLAB code for june27.fig
```

```
%      JUNE27, by itself, creates a new JUNE27 or raises the existing
%      singleton*.
%
%      H = JUNE27 returns the handle to a new JUNE27 or the handle to
%      the existing singleton*.
%
%      JUNE27('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in JUNE27.M with the given input arguments.
%
%      JUNE27('Property','Value',...) creates a new JUNE27 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before june27_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to june27_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help june27
% Last Modified by GUIDE v2.5 27-Jul-2021 15:18:31
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @june27_OpeningFcn, ...
                   'gui_OutputFcn',  @june27_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
```

```
end
if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before june27 is made visible.
function june27_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to june27 (see VARARGIN)
% Choose default command line output for june27
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes june27 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = june27_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
I_change=handles.R;
```

```
m=(get(handles.slider1,'Value'))+1;
set(handles.edit1,'String',m);
[R C]=size(I_change);
R1=round(R.*(1-1./m)./2)+1
R2=round(R.*(1+1./m)./2)
C1=round(C.*(1-1./m)./2)+1
C2=round(C.*(1+1./m)./2)
I_c=handles.R(R1:R2,C1:C2);
I_z1=imresize(I_c,m);
I_change=handles.G;
[R C]=size(I_change);
R1=round(R.*(1-1./m)./2)+1
R2=round(R.*(1+1./m)./2)
C1=round(C.*(1-1./m)./2)+1
C2=round(C.*(1+1./m)./2)
I_c=handles.G(R1:R2,C1:C2);
I_z2=imresize(I_c,m);
I_change=handles.B;
[R C]=size(I_change);
R1=round(R.*(1-1./m)./2)+1
R2=round(R.*(1+1./m)./2)
C1=round(C.*(1-1./m)./2)+1
C2=round(C.*(1+1./m)./2)
I_c=handles.B(R1:R2,C1:C2);
I_z3=imresize(I_c,m);
n(:,:,1)=I_z1;
n(:,:,2)=I_z2;
n(:,:,3)=I_z3;
imshow(n,[])
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
```

```
[f d]=uigetfile({'*.jpg;*.tif;*.png;*.gif'},'Select an Image');
file_name=strcat(d,f);
I=imread(file_name);
handles.R = I(:,:,1);
handles.G = I(:,:,2);
handles.B = I(:,:,3);
imshow(I);
guidata(hObject, handles);
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
End
```

OUTPUT-

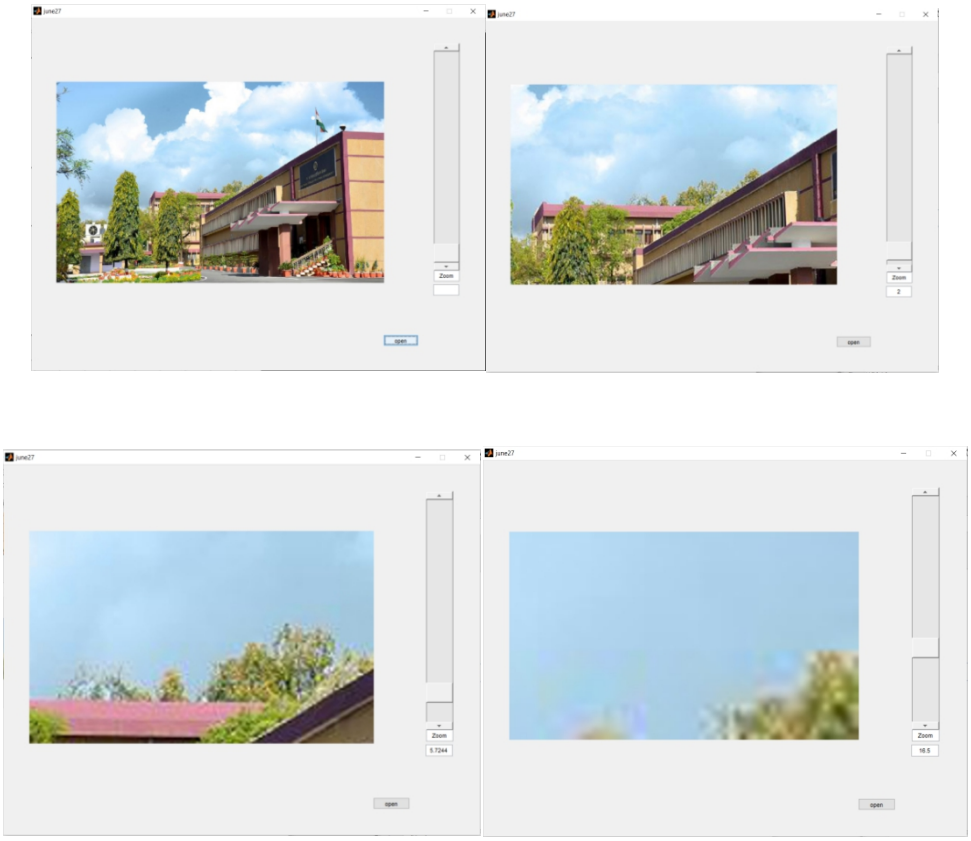


Figure 4.15 Output of coding Exercise 8

4.2.15 Coding exercise 9 : Write a GUI based program with menu to select any image from system and to perform changing of color content, conversion to greyscale and zoom.

Code:

```
function varargout = june27ex2(varargin)
% JUNE27EX2 MATLAB code for june27ex2.fig
%      JUNE27EX2, by itself, creates a new JUNE27EX2 or raises the existing
%      singleton*.
%
%      H = JUNE27EX2 returns the handle to a new JUNE27EX2 or the handle to
%      the existing singleton*.
%
%      JUNE27EX2('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in JUNE27EX2.M with the given input arguments.
%
%      JUNE27EX2('Property','Value',...) creates a new JUNE27EX2 or raises the
```

```
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before june27ex2_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to june27ex2_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help june27ex2
% Last Modified by GUIDE v2.5 02-Jul-2021 18:09:49
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @june27ex2_OpeningFcn, ...
                   'gui_OutputFcn',  @june27ex2_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before june27ex2 is made visible.
function june27ex2_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
```

```
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to june27ex2 (see VARARGIN)
% Choose default command line output for june27ex2
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes june27ex2 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = june27ex2_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in menu.
function menu_Callback(hObject, eventdata, handles)
[f d]=uigetfile({'*.jpg;*.tif;*.png;*.gif'},'Select an Image');
file_name=strcat(d,f);
global I
I=imread(file_name);
imshow(I);
global red_value;
global green_value;
global blue_value;
red_value=1;
green_value=1;
blue_value=1;
```

```
% hObject    handle to menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in gray.
function gray_Callback(hObject, eventdata, handles)
global I
imgg=I;
gray= rgb2gray(imgg);
imshow(gray);
I=gray;
% hObject    handle to gray (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
global I
imgg=I;
m=(get(handles.slider1,'Value'))+1;
set(handles.edit1,'String',m);
if(ndims(I)>2)
   I_change=imgg(:,:,1);
   [R C]=size(I_change);
   R1=round(R.*(1-1./m)./2)+1
   R2=round(R.*(1+1./m)./2)
   C1=round(C.*(1-1./m)./2)+1
   C2=round(C.*(1+1./m)./2)
   I_c=I_change(R1:R2,C1:C2);
   I_z1=imresize(I_c,m);
   I_change=imgg(:,:,2);;
   [R C]=size(I_change);
   R1=round(R.*(1-1./m)./2)+1
```

```
    R2=round(R.*(1+1./m)./2)
    C1=round(C.*(1-1./m)./2)+1
    C2=round(C.*(1+1./m)./2)
    I_c=I_change(R1:R2,C1:C2);
    I_z2=imresize(I_c,m);
    I_change=imgg(:,:,3);;
    [R C]=size(I_change);
    R1=round(R.*(1-1./m)./2)+1
    R2=round(R.*(1+1./m)./2)
    C1=round(C.*(1-1./m)./2)+1
    C2=round(C.*(1+1./m)./2)
    I_c=I_change(R1:R2,C1:C2);
    I_z3=imresize(I_c,m);
    n(:,:,1)=I_z1;
    n(:,:,2)=I_z2;
    n(:,:,3)=I_z3;
    imshow(n,[])
    I=n;
else
     I_change=imgg;
    [R C]=size(I_change);
    R1=round(R.*(1-1./m)./2)+1
    R2=round(R.*(1+1./m)./2)
    C1=round(C.*(1-1./m)./2)+1
    C2=round(C.*(1+1./m)./2)
    I_c=I_change(R1:R2,C1:C2);
    I_z=imresize(I_c,m);
    n=I_z;
    imshow(n,[])
    I=n;
end
```

```
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes on slider movement.
```

```
function slider2_Callback(hObject, eventdata, handles)
global I;
global red_value;
global green_value;
global blue_value;
red_value=1+get(handles.slider2,'value');
set(handles.edit4,'String',red_value);
red=I(:,:,1);
green=I(:,:,2);
blue=I(:,:,3);
z(:,:,1)=red_value.*red;
z(:,:,2)=green_value.*green;
z(:,:,3)=blue_value.*blue;
imshow(z);
I=z;
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
```

```
global I;
global red_value;
global green_value;
global blue_value;
blue_value=1+get(handles.slider3,'value');
set(handles.edit2,'String',blue_value);
red=I(:,:,1);
green=I(:,:,2);
blue=I(:,:,3);
z(:,:,1)=red_value.*red;
z(:,:,2)=green_value.*green;
z(:,:,3)=blue_value.*blue;
imshow(z);
I=z;
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function slider4_Callback(hObject, eventdata, handles)
global I
```

```
global red_value;
global green_value;
global blue_value;
green_value=1+get(handles.slider4,'value');
set(handles.edit3,'String',green_value);
red=I(:,:,1);
green=I(:,:,2);
blue=I(:,:,3);
z(:,:,1)=red_value.*red;
z(:,:,2)=green_value.*green;
z(:,:,3)=blue_value.*blue;
imshow(z);
I=z;
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit3_Callback(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit3 as text
%        str2double(get(hObject,'String')) returns contents of edit3 as a double
% --- Executes during object creation, after setting all properties.
function edit3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
End
```

OUTPUT-

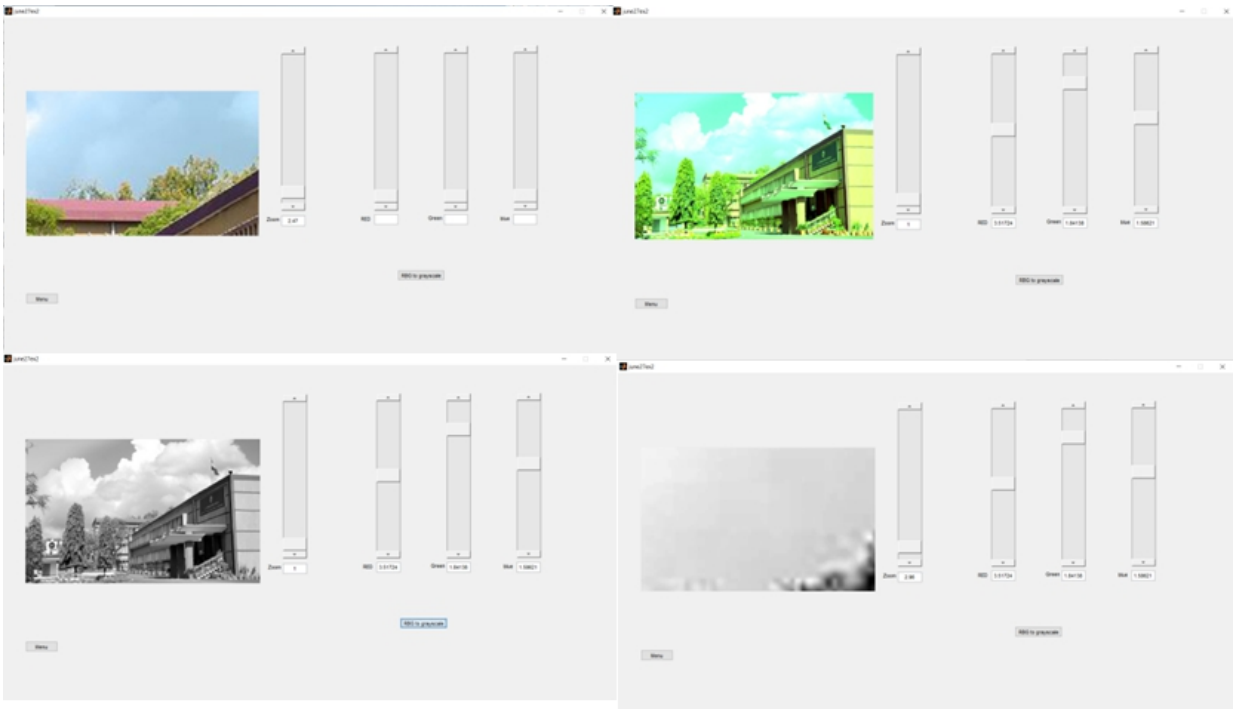


Figure 4.16 Output of coding Exercise 9

4.2.16 Gain,Offset and Histogram equalization

a) **Histogram:**A histogram obtains the PDF of image pixels (intensity, color, etc.). A histogram is usually obtained by plotting the number of pixels with intensity levels at different range intervals.Histogram of an image provides a global description of the appearance of an image.Information obtained from histogram is very large in quality.Histogram of an image represents the relative frequency of occurrence of various gray levels in an image. A compressed histogram often indicates an image with a poor visual contrast. A well-distributed histogram often has a higher contrast and better visibility of detail.

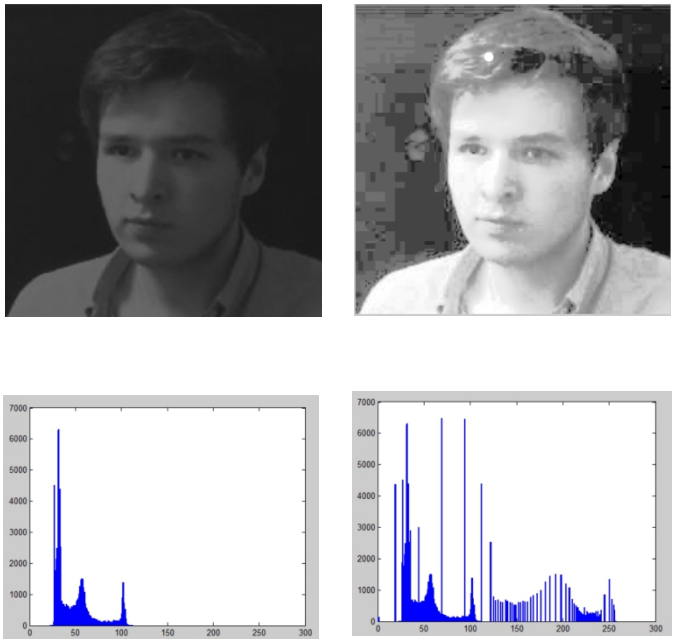


Figure 4.17 Photos and their Histogram

b) **Gain:** The contrast of an image depends on the Gain factor. Gain is in product with image matrix.

c) **Offset:** The brightness of a image depends on the offset factor. Offset is added to image matrix.

d) The formulae to change gain and offset of a Image is:

New image= offset +(old image)*gain

e) Histogram equalization:

Histogram equalization is the process of applying a point operation on a known image histogram such that the output image has a flat histogram(equally many pixels at every gray level). Histogram equalization is useful when putting two images in the same footing for comparison.A point operation f (G) is applied on input image A(x, y) to transform A(x, y) into B(x, y). It turns out that the point operation needed for such histogram equalization is simply a cumulative distribution function (CDF). Let us assume that given the histogram of A(x, y) by HA(G), we want to find the histogram of B(x, y) expressed as HB(G). HB(G) is the equalized histogram.

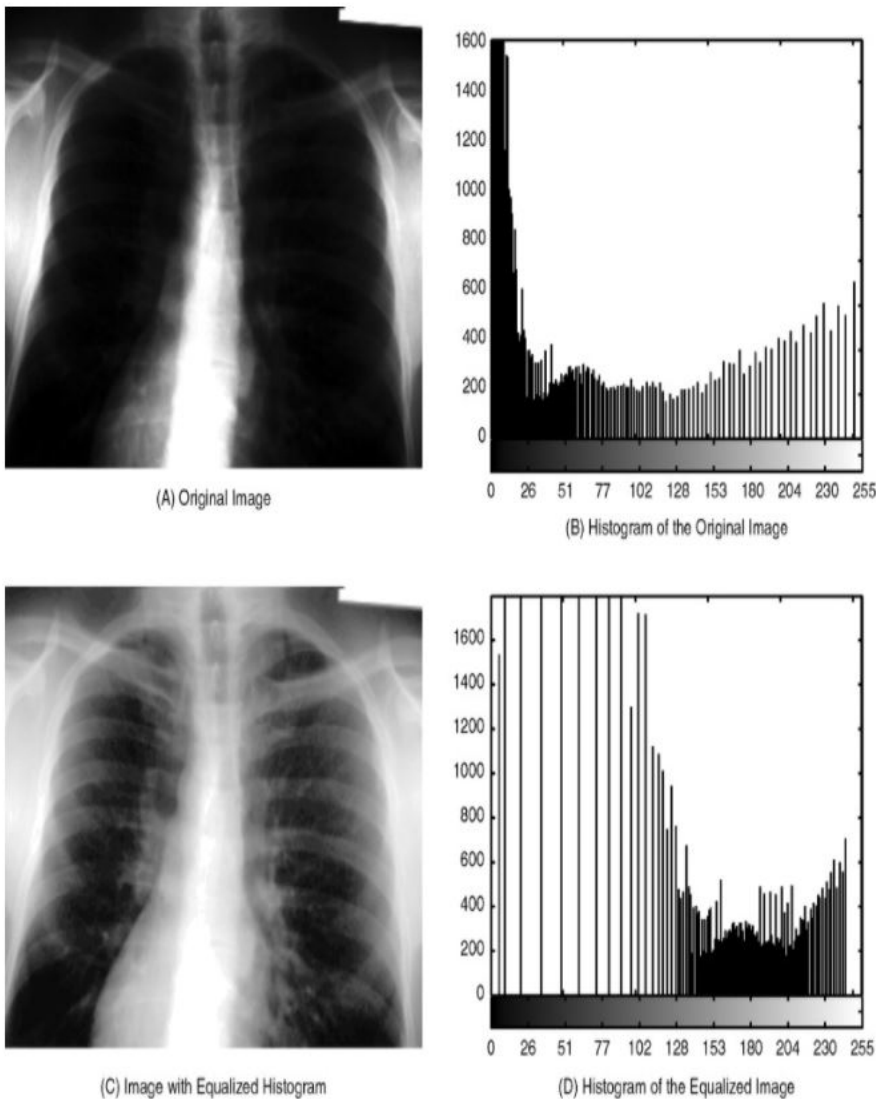


Figure 4.18 Photos and their Histogram

4.2.17 Coding exercise 10 : Display histogram of an image.

Code:

```
clear all
I=imread('C:\Users\DELL\Desktop\download.jpg');
I=rgb2gray(I);
I=I+20;
imshow(I)
p=size(I);
ele=numel(I);
t=zeros(1,257);
% Histogram plotting
for k=1:p(1)
    for l=1:p(2)
   a=I(k,l);
   t(a+1)=t(a+1)+1;
    end
end
L=t;
figure
bar(L,'b')
```

OUTPUT

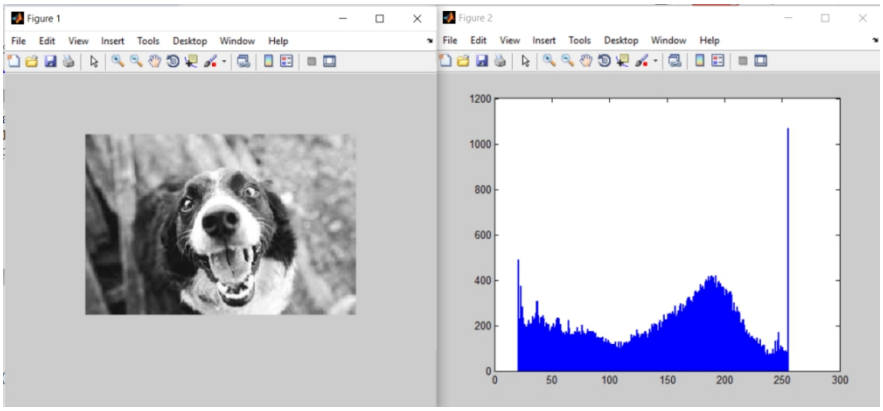


Figure 4.19 Output of coding Exercise 10

4.2.18 Coding exercise 11 : Write a GUI based program with menu to select any image from system, then change the offset and gain of an image with sliders and display its histogram to observe change in Image and histogram with change in offset and gain.

Code:

```
function varargout = july4(varargin)
% JULY4 MATLAB code for july4.fig
%      JULY4, by itself, creates a new JULY4 or raises the existing
%      singleton*.
%
%      H = JULY4 returns the handle to a new JULY4 or the handle to
%      the existing singleton*.
%
%      JULY4('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in JULY4.M with the given input arguments.
%
%      JULY4('Property','Value',...) creates a new JULY4 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before july4_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to july4_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help july4
% Last Modified by GUIDE v2.5 09-Jul-2021 11:21:28
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
```

```
                'gui_OpeningFcn', @july4_OpeningFcn, ...
                'gui_OutputFcn',  @july4_OutputFcn, ...
                'gui_LayoutFcn',  [] , ...
                'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before july4 is made visible.
function july4_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to july4 (see VARARGIN)
% Choose default command line output for july4
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes july4 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = july4_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
global offset_value;
global value_gain;
global I_gray;
value_gain=get(handles.slider2,'value')+1;
offset_value=get(handles.slider1,'value');
global k
set(handles.edit2,'String',offset_value);
set(handles.edit1,'String',value_gain);
k=offset_value+value_gain.*I_gray;
axes(handles.axes1);
imshow(k);
axes(handles.axes2);
imhist(k);
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
```

```
end
% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
global offset_value;
global value_gain;
global I_gray;
value_gain=get(handles.slider2,'value')+1;
offset_value=get(handles.slider1,'value');
global k
set(handles.edit2,'String',offset_value);
set(handles.edit1,'String',value_gain);
k=offset_value+value_gain.*I_gray;
axes(handles.axes1);
imshow(k);
axes(handles.axes2);
imhist(k);
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
function edit1_Callback(hObject, eventdata, handles)
```

```
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double
% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double
% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
global I_gray;
global re;
global k;
global l;
k=0;
l=0;
[f d]=uigetfile({'*.jpg;*.tif;*.png;*.gif'},'Select an Image');
filename=strcat(d,f);
im=imread(filename);
re=rgb2gray(im);
I_gray =rgb2gray(im) ;
axes(handles.axes1);
imshow(I_gray);
axes(handles.axes2);
imhist(I_gray);
global offset_value;
global value_gain;
offset_value=0;
value_gain=1;
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO
global I_gray;
global re
I_gray= re;
axes(handles.axes1);
```

```
imshow(I_gray);
axes(handles.axes2);
imhist(I_gray);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

OUTPUT:

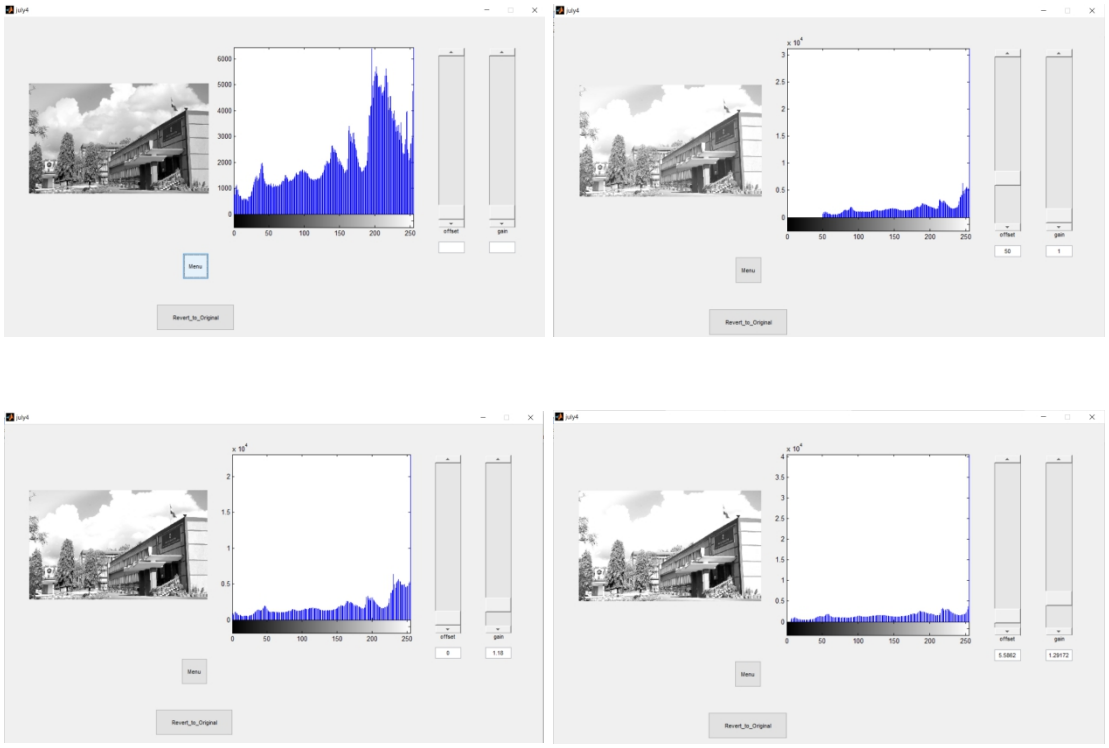


Figure 4.20 Output of coding Exercise 11

4.2.19 Coding exercise 12: Write a GUI based program with menu to select any image from system, to Perform histogram equalization and display its histogram.

Code:

```
function varargout = july11(varargin)
% JULY11 MATLAB code for july11.fig
%      JULY11, by itself, creates a new JULY11 or raises the existing
%      singleton*.
```

```
%
%      H = JULY11 returns the handle to a new JULY11 or the handle to
%      the existing singleton*.
%
%      JULY11('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in JULY11.M with the given input arguments.
%
%      JULY11('Property','Value',...) creates a new JULY11 or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before july11_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to july11_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help july11
% Last Modified by GUIDE v2.5 15-Jul-2021 11:25:20
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @july11_OpeningFcn, ...
                   'gui_OutputFcn',  @july11_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
```

```
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before july11 is made visible.
function july11_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to july11 (see VARARGIN)
% Choose default command line output for july11
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes july11 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = july11_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
global p;
global t;
```

```
[f d]=uigetfile({'*.jpg;*.tif;*.png;*.gif'},'Select an Image');
filename=strcat(d,f);
handles.im=imread(filename);
handles.I_gray=rgb2gray(handles.im);
axes(handles.axes1);
imshow(handles.I_gray,[]);
p=size(handles.I_gray);
ele=numel(handles.I_gray);
t=zeros(1,257);
% Histogram plotting
for k=1:p(1)
    for l=1:p(2)
   a=handles.I_gray(k,l);
   t(a+1)=t(a+1)+1;
    end
end
L=t;
axes(handles.axes2);
bar(L,'b')
guidata(hObject,handles);
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
global p;
global t;
for k=1:256;
    s(k)=floor(256.*sum(t(1:k))./(sum(t(1:256))));
end
%figure
%plot(s,'b')
```

```
% Output gray value
for i=1:(p(1)-1)
    for j=1:(p(2)-1)
        O_I=handles.I_gray(i,j);
        M_I(i,j)=s(O_I+1);
    end
end
axes(handles.axes4);
imshow(M_I,[])
for k=1:(p(1)-1)
    for l=1:(p(2)-1)
   a=M_I(k,l);
  t(a+1)=t(a+1)+1;
    end
end
axes(handles.axes3);
bar(t,'b')
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
global I_gray;
global re;
I_gray = re;
axes(handles.axes1);
imshow(I_gray);
axes(handles.axes2);
imhist(I_gray);
```

% eventdata reserved - to be defined in a future version of MATLAB

% handles   structure with handles and user data (see GUIDATA)

OUTPUT-

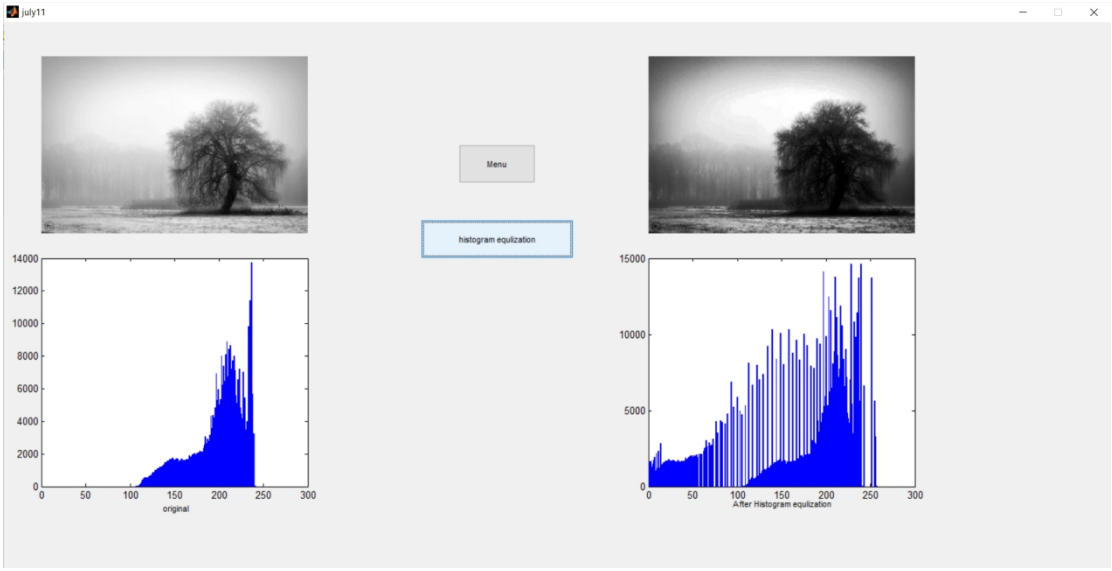


Figure 4.21 Output of coding Exercise 12

4.2.20 a Edge Detection:

Edges are significant local changes of intensity in a digital image. An edge can be defined as a set of connected pixels that forms a boundary between two disjoint regions. There are three types of edges:

- Horizontal edges
- Vertical edges
- Diagonal edges

Edge Detection is a method of segmenting an image into regions of discontinuity. It is a widely used technique in digital image processing like

- pattern recognition

- image morphology
- feature extraction

Edge detection allows users to observe the features of an image for a significant change in the gray level. This texture indicating the end of one region in the image and the beginning of another. It reduces the amount of data in an image and preserves the structural properties of an image.

Edge Detection Operators are of two types:

- **Gradient** – based operator which computes first-order derivations in a digital image like, Sobel operator, Prewitt operator, Robert operator
- **Gaussian** – based operator which computes second-order derivations in a digital image like, Canny edge detector, Laplacian of Gaussian

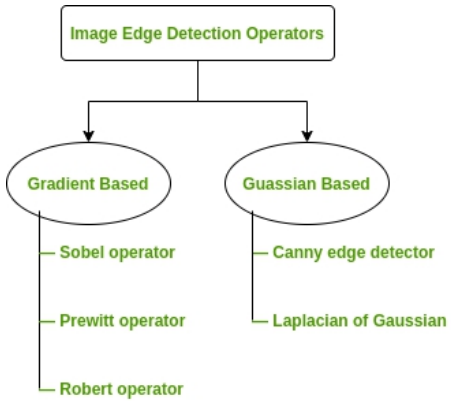


Figure 4.22: Types of Edge Detection Operators

1. Sobel Operator: It is a discrete differentiation operator. It computes the gradient approximation of image intensity function for image edge detection. At the pixels of an image, the Sobel operator produces either the normal to a vector or the corresponding gradient vector. It uses two 3 x 3 kernels or masks which are convolved with the input image to calculate the vertical and horizontal derivative approximations respectively –

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 4.23: Sobel Matrix

2. Prewitt Operator: This operator is almost similar to the sobel operator. It also detects vertical and horizontal edges of an image. It is one of the best ways to detect the orientation and

magnitude of an image. It uses the kernels or masks –

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Figure 4.24: Prewitt Matrix

3. Robert Operator: This gradient-based operator computes the sum of squares of the differences between diagonally adjacent pixels in an image through discrete differentiation. Then the gradient approximation is made. It uses the following 2 x 2 kernels or masks –

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

Figure 4.25: Robert Matrix

4. Laplacian of Gaussian (LoG): It is a gaussian-based operator which uses the Laplacian to take the second derivative of an image. This really works well when the transition of the grey level seems to be abrupt. It works on the zero-crossing method i.e when the second-order derivative crosses zero, then that particular location corresponds to a maximum level. It is called an edge location. Here the Gaussian operator reduces the noise and the Laplacian operator detects the sharp edges.

5. Canny Operator: It is a gaussian-based operator in detecting edges. This operator is not susceptible to noise. It extracts image features without affecting or altering the feature. Canny edge detector have advanced algorithm derived from the previous work of Laplacian of Gaussian operator. It is widely used an optimal edge detection technique. It detects edges based on three criteria:

6. Low error rate
7. Edge points must be accurately localized
8. There should be just one single edge response

Matlab Function for Edge Detection:

BW = edge(I,'algorithm')

4.2.21 Coding Exercise 13: WAP to Display edges of a image.

Code:

```
clear all
I=imread('C:\Users\DELL\Desktop\lab-image.jpg');
I=rgb2gray(I);%(:,21:660,:));
I=double(I);
imshow(I,[]);
%preweet
E_h=[-1 0 1;-1 0 1;-1 0 1];
E_v=[-1 -1 -1;0 0 0;1 1 1];
%sobel
% E_h=[-1 0 1;-2 0 2;-1 0 1];
% E_v=[-1 -2 -1;0 0 0;1 2 1];
th=50;
p=size(I);
I_edge=zeros(p(1),p(2));
for k=2:1:(p(1)-1)
    for l=2:1:(p(2)-1)
  a=I((k-1):(k+1),(l-1):(l+1));
  edge_h=0;
  edge_v=0;
  for i=1:1:3;
     for j=1:1:3;
      aa=E_h(i,j);
      bb=E_v(i,j);
      bb=a(i,j);
      edge_h=edge_h+(E_h(i,j).*a(i,j));
      edge_v=edge_v+(E_v(i,j).*a(i,j));
     end
  end
    if abs(edge_h)>th
       I_edge(k,l)=1;
    end
```

```
    if abs(edge_v)>th
       I_edge(k,l)=1;
    end
    end
end
figure
imshow(I_edge,[]);
```

OUTPUT-

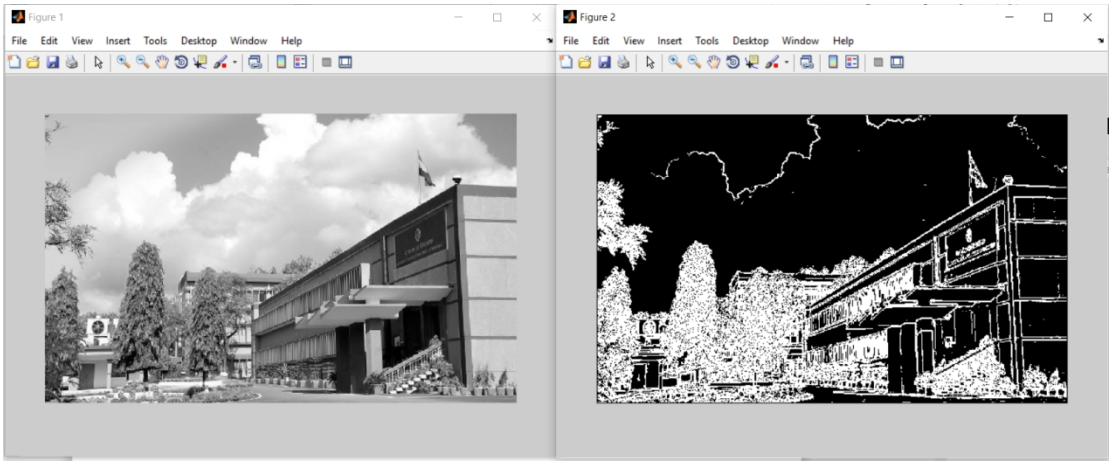


Figure 4.26 Output of coding Exercise 13

Chapter 5 – Learning Outcome

Outcome of this internship is making of GUI which can do various operation on images.

Parts of final GUI:

Slider: for changing colour content, offset, gain and zoom

Buttons: RGB to grayscale,Colour content,Save,Histogram equalization,offset,gain,Zoom,Return to original, edge detection and keep change

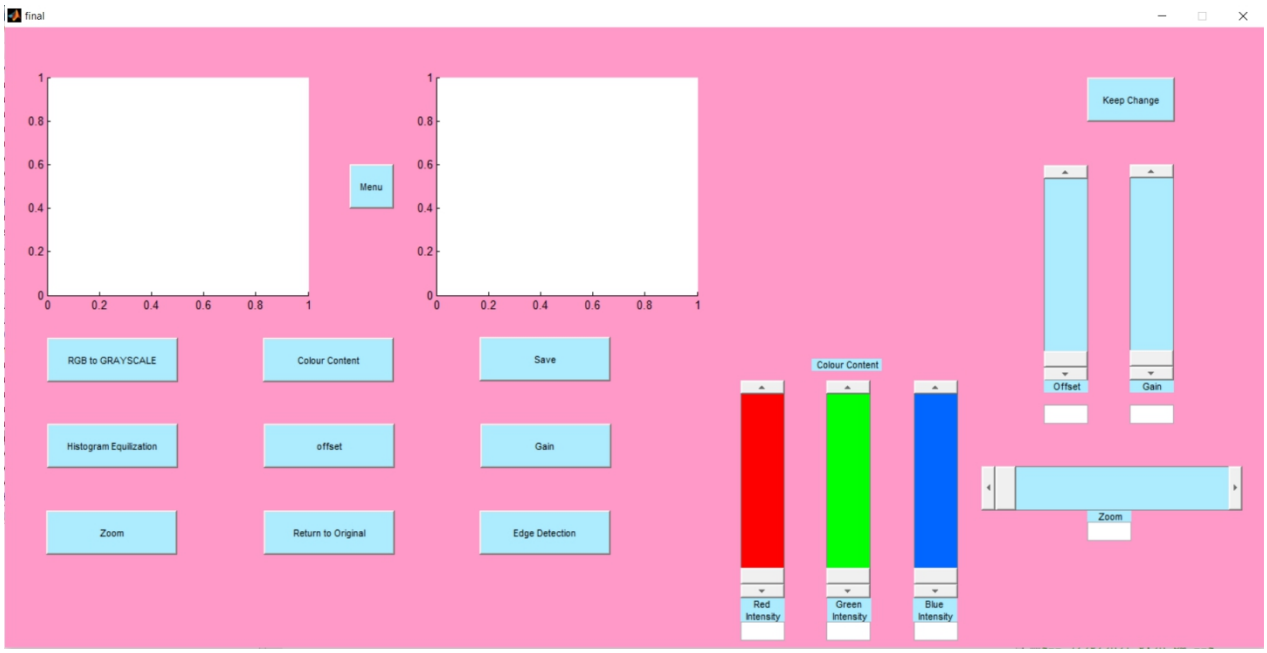
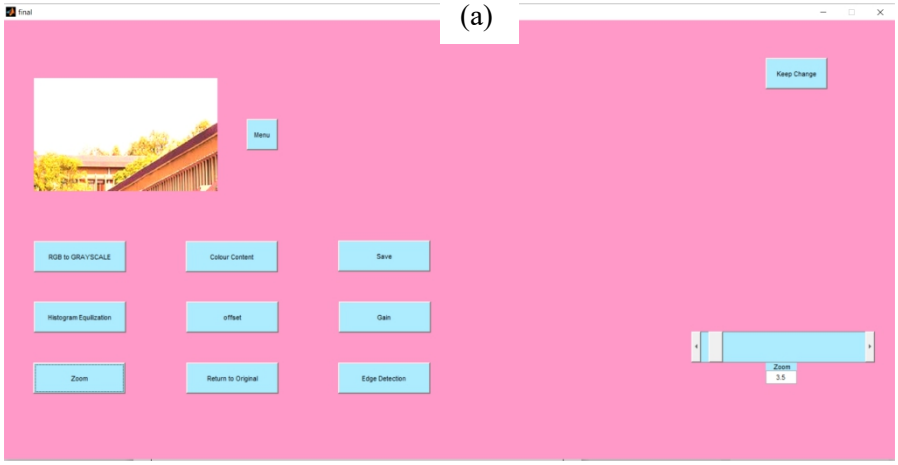


Figure 5.1 Layout of the GUI

Code: In Annexures

Output-

(a)



(b)

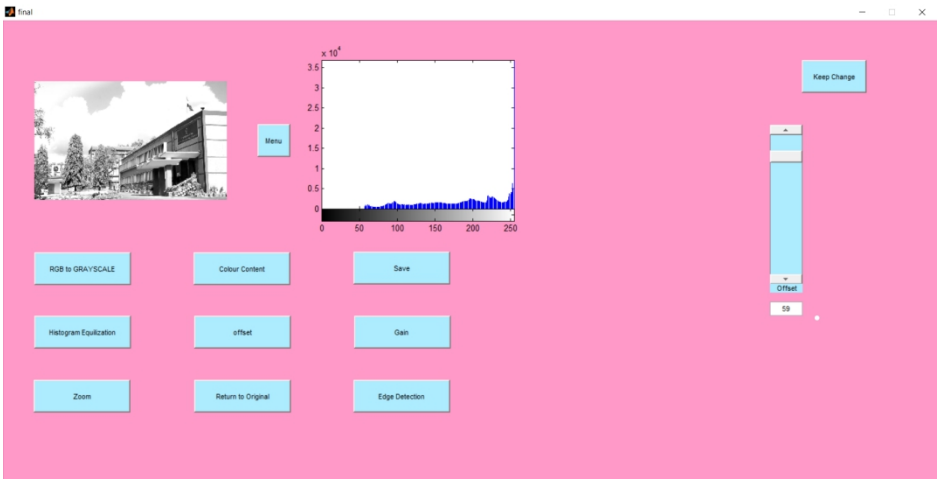
(c)


final
Keep Change
Menu
RGB to GRAYSCALE
Colour Content
Save
Histogram Equilization
offset
Gain
Zoom
Return to Original
Edge Detection

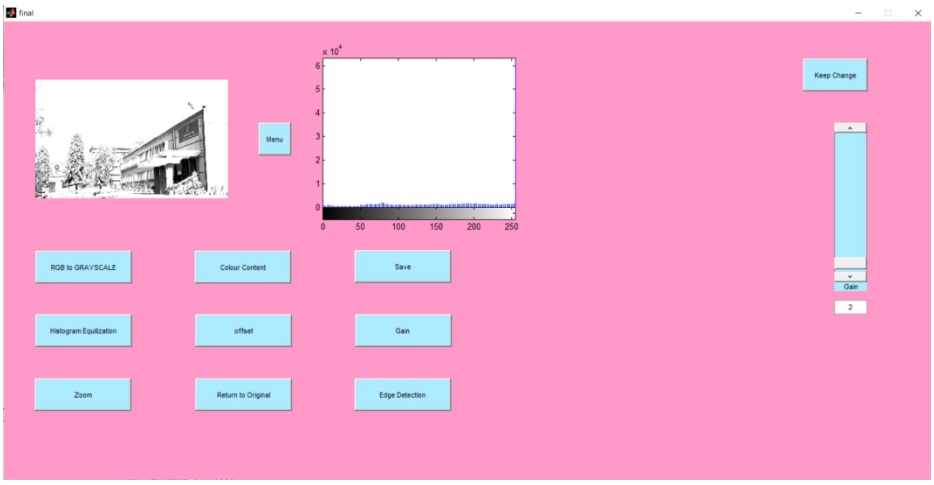
(d)


final
Keep Change
Menu
RGB to GRAYSCALE
Colour Content
Save
Histogram Equilization
offset
Gain
Zoom
Return to Original
Edge Detection

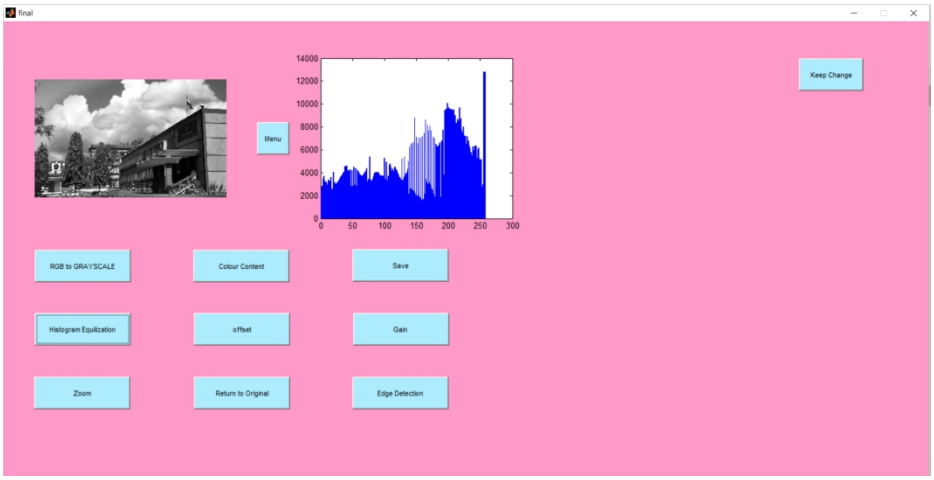
(e)


final
x 10^4
3.5
3
2.5
2
1.5
1
0.5
0
0
50
100
150
200
250
Keep Change
Menu
Offset
59
RGB to GRAYSCALE
Colour Content
Save
Histogram Equilization
offset
Gain
Zoom
Return to Original
Edge Detection

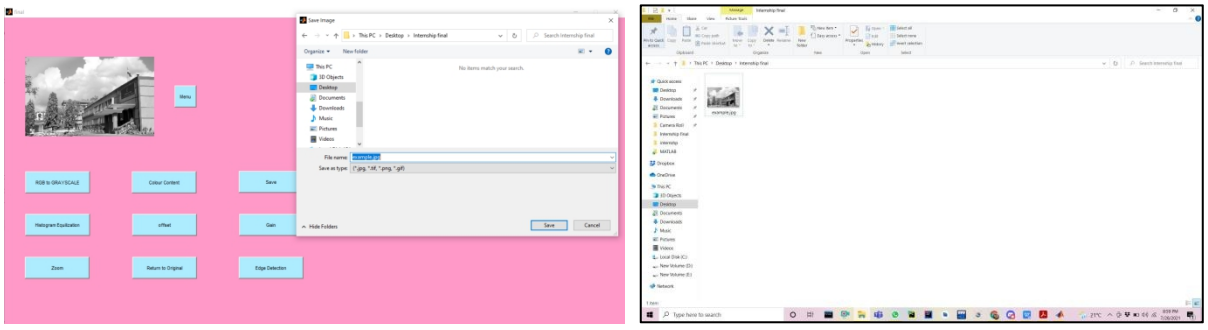
(f)



(g)



(h)



(i)

Figure 5.2 Output of the GUI

(a) Colour content (b) zoom (c) edge (d) return to original (e) RGB to grayscale (f) offset (g) gain (h) histogram equalization (i)save

Chapter 6 – Conclusion & Future scope

Conclusion

The aim of this internship was to understand Image Processing using MATLAB.Many image processing operations have been implemented on different images in order to enhance the quality of the image, For example colour content, zoom, edge detection,grayscale,offset,gain control, histogram equalization etc.Most difficulty which faced was when implementing Histogram equalization in the final GUI, this was because it generated image of single matrix and other functions used three matrices. Thus after operating histogram equalization, values of the single matrix had to be assigned to all the three matrices, in order to make it executable in other functions.This work has been a challenge to coding skills, leading to improvement of them to a larger extent especially in the field of Image processing.

Future scope

Image processing technology extracts information from images and integrates it for a wide range of applications. Here, we've outlined the most prominent fields where image processing could bring significant benefits.

In production automation

Image processing applications can make it possible for machines to act as more self-sufficient and ensure the quality of products. Assuming processing systems work faster than humans, inline quality controls like 100% controls can be very quickly implemented. Damaged parts can be replaced or corrected, which would lead to more efficacies of production facilities. With the help of advanced image processing technologies, even an entire production facility can be managed.

In agricultural landscape

Key concerns in agriculture include quality of yield and water stress. Irrigation monitoring and providing information can be made possible by tracking satellite imaging of the fields. Processing of infrared images can act as an additional means to monitor and analyze irrigation. This analysis can then be utilized in pre-harvesting operations for deciding whether to harvest or not. Growth of weeds can also be detected by using a combination of machine learning

and image processing algorithms and techniques. Quality of yields can be ensured by the reliable and accurate method of image processing through sorting and grading of fresh products.

Biomedical and other healthcare applications

3D imaging is a process where a 2D image is converted into a 3D image by creating the optical illusion of depth. The next step is rendering where colors and textures are included in the 3D model to make it look realistic. With such 3D imaging and rendering, doctors can see extremely high quality 3D images of organs that they couldn't have seen otherwise. This, in turn, can help them carry out delicate surgeries and make accurate diagnoses.

Disaster management

Drone aircraft monitoring environmental and traffic conditions can use image processing to capture high resolution real-time videos and photographs. In case of natural or other disasters like flood, earthquake, fire etc, knowing which disaster-struck areas the authorities need to focus upon can help save lives by reaching quickly to those trapped and bring them out safely. Even monitoring the progress and ensuring co-ordination during such rescue operations can be made easier with real-time image processing techniques.

Real time image processing

Real Time Image Processing,as the name suggests,frames gets processed on the flow without any delay & loss of frames,as soon as it comes from image-sensor(camera).Basic criteria to perform real time image processing is:

Processing Time Per Frame < Capturing Time Per Frame.

Featuring Extraction & Detection,Pixel Manipulation or whatever needs to be done on every frame as soon as its arrives from camera sensor.

Bibliography

- www.quora.com
- www.skyfilabs.com
- www.geeksforgeeks.org
- www.javatpoint.com
- www.tutorialspoint.com
- www.mathworks.com
- MATLAB Help
- Google Images

Anexures

A. Code

Code of Final GUI:

```
function varargout = final(varargin)
% FINAL MATLAB code for final.fig
%      FINAL, by itself, creates a new FINAL or raises the existing
%      singleton*.
%
%      H = FINAL returns the handle to a new FINAL or the handle to
%      the existing singleton*.
%
%      FINAL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in FINAL.M with the given input arguments.
%
%      FINAL('Property','Value',...) creates a new FINAL or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before final_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to final_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES
% Edit the above text to modify the response to help final
% Last Modified by GUIDE v2.5 26-Jul-2021 11:43:05
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @final_OpeningFcn, ...
                   'gui_OutputFcn',  @final_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
% --- Executes just before final is made visible.
```

```
function final_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to final (see VARARGIN)
% Choose default command line output for final
handles.output = hObject;
% Update handles structure
guidata(hObject, handles);
% UIWAIT makes final wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = final_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
global k;
k=0;
```

```
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global im;
global m;
global re;
global p;
global t;
[f d]=uigetfile({'*.jpg;*.tif;*.png;*.gif'},'Select an Image');
filename=strcat(d,f);
im=imread(filename);
m=im;
re=im;
axes(handles.axes1);
imshow(im,[]);
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
```

```
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global im;
global m;
m=im;
m=rgb2gray(m);
axes(handles.axes1);
imshow(m,[]);
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global im;
```

```
global m;
if(ndims(im)>2)
   imm=rgb2gray(im);
else
   imm=im;
end
I_edge=edge(imm,'prewitt',0.11);
m=I_edge;
axes(handles.axes1);
imshow(m);
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global re;
global m;
m=re;
axes(handles.axes1);
imshow(m,[]);
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
```

```
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','on');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','on');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','on');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off')
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','on') %hide the current axes
```

```
set(get(handles.axes3,'children'),'visible','on') %hide the current axes contents
global p;
global t;
global m;
global im;
if(ndims(im)>2)
   I_gray=rgb2gray(im);
else
   I_gray=im;
end
p=size(I_gray);
ele=numel(I_gray);
t=zeros(1,257);
% Histogram plotting
for k=1:p(1)
    for l=1:p(2)
   a=I_gray(k,l);
   t(a+1)=t(a+1)+1;
    end
end
for k=1:256;
   s(k)=floor(256.*sum(t(1:k))./(sum(t(1:256))));
end
% Output gray value
for i=1:(p(1)-1)
    for j=1:(p(2)-1)

        O_I=I_gray(i,j);
        M_I(i,j)=s(O_I+1);

    end
end

axes(handles.axes1);
imshow(M_I,[]);

% Histogram plotting
for k=1:(p(1)-1)
    for l=1:(p(2)-1)
   a=M_I(k,l);
  t(a+1)=t(a+1)+1;
    end
end
for k=1:(p(1)-1)
    for l=1:(p(2)-1)
        m(k,l,1)=M_I(k,l);
```

```
    end
end
for k=1:(p(1)-1)
    for l=1:(p(2)-1)
        m(k,l,2)=M_I(k,l);
    end
end
for k=1:(p(1)-1)
    for l=1:(p(2)-1)
        m(k,l,3)=M_I(k,l);
    end
end
L=t;

axes(handles.axes3);
bar(L,'b')

% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','on');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','on');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','on');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','on') %hide the current axes
set(get(handles.axes3,'children'),'visible','on')
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','on');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','on');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','on');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','on') %hide the current axes
set(get(handles.axes3,'children'),'visible','on') %hide the current axes contents
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','on');
set(handles.slider6, 'visible','on');
set(handles.slider7, 'visible','on');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','on');
set(handles.text10, 'visible','on');
set(handles.text12, 'visible','on');
set(handles.text11, 'visible','on');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
```

```
set(handles.edit6, 'visible','on');
set(handles.edit8, 'visible','on');
set(handles.edit7, 'visible','on');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global im
global mm
mm=im;
global red_value;
global green_value;
global blue_value;
red_value=1;
green_value=1;
blue_value=1;

% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global im;
[f d]=uiputfile({'*.jpg;*.tif;*.png;*.gif'},'Save Image');
```

```
filename=strcat(d,f);
imwrite(im,filename);
guidata(hObject,handles);

% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% --- Executes on slider movement.
function slider1_Callback(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider2_Callback(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
```

```
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider3_Callback(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
set(handles.slider5, 'visible','off');
set(handles.slider6, 'visible','off');
set(handles.slider7, 'visible','off');
set(handles.slider8, 'visible','off');
set(handles.slider9, 'visible','off');
set(handles.slider11, 'visible','off');
set(handles.text13, 'visible','off');
set(handles.text10, 'visible','off');
set(handles.text12, 'visible','off');
set(handles.text11, 'visible','off');
set(handles.text14, 'visible','off');
set(handles.text15, 'visible','off');
set(handles.text17, 'visible','off');
set(handles.edit6, 'visible','off');
set(handles.edit8, 'visible','off');
set(handles.edit7, 'visible','off');
set(handles.edit9, 'visible','off');
```

```
set(handles.edit10, 'visible','off');
set(handles.edit12, 'visible','off');
set(handles.axes3,'visible','off') %hide the current axes
set(get(handles.axes3,'children'),'visible','off') %hide the current axes contents
global im;
global m;
global p;
global t;
im=m;
handles.I_gray=rgb2gray(im);
p=size(handles.I_gray);
ele=numel(handles.I_gray);
t=zeros(1,257);
% Histogram plotting
for k=1:p(1)
    for l=1:p(2)
   a=handles.I_gray(k,l);
   t(a+1)=t(a+1)+1;
    end
end
guidata(hObject,handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
function edit1_Callback(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit1 as text
%        str2double(get(hObject,'String')) returns contents of edit1 as a double


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
```

```
function edit2_Callback(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit2 as text
%        str2double(get(hObject,'String')) returns contents of edit2 as a double


% --- Executes during object creation, after setting all properties.
function edit2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
function edit4_Callback(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'String') returns contents of edit4 as text
%        str2double(get(hObject,'String')) returns contents of edit4 as a double
% --- Executes during object creation, after setting all properties.
function edit4_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
% --- Executes on slider movement.
function slider5_Callback(hObject, eventdata, handles)
global mm;
global m;
global red_value;
global green_value;
global blue_value;
red_value=1+get(handles.slider5,'value');
set(handles.edit6,'String',red_value);
```

```
red=mm(:,:,1);
green=mm(:,:,2);
blue=mm(:,:,3);
z(:,:,1)=red_value.*red;
z(:,:,2)=green_value.*green;
z(:,:,3)=blue_value.*blue;
m=z;
axes(handles.axes1);
imshow(m);

% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider5_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function slider6_Callback(hObject, eventdata, handles)
global mm;
global m;
global red_value;
global green_value;
global blue_value;
green_value=1+get(handles.slider6,'value');
set(handles.edit8,'String',green_value);
red=mm(:,:,1);
green=mm(:,:,2);
blue=mm(:,:,3);
z(:,:,1)=red_value.*red;
z(:,:,2)=green_value.*green;
z(:,:,3)=blue_value.*blue;
m=z;
axes(handles.axes1);
imshow(m);
% hObject    handle to slider6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
% --- Executes on slider movement.
function slider7_Callback(hObject, eventdata, handles)
global mm;
global m;
global red_value;
global green_value;
global blue_value;
blue_value=1+get(handles.slider7,'value');
set(handles.edit7,'String',blue_value);
red=mm(:,:,1);
green=mm(:,:,2);
blue=mm(:,:,3);
z(:,:,1)=red_value.*red;
z(:,:,2)=green_value.*green;
z(:,:,3)=blue_value.*blue;
m=z;
axes(handles.axes1);
imshow(m);
% hObject    handle to slider7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider
% --- Executes during object creation, after setting all properties.
function slider7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

```
function edit6_Callback(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit6 as text
%        str2double(get(hObject,'String')) returns contents of edit6 as a double
% --- Executes during object creation, after setting all properties.
function edit6_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit7_Callback(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit7 as text
%        str2double(get(hObject,'String')) returns contents of edit7 as a double

% --- Executes during object creation, after setting all properties.
function edit7_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit8_Callback(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double
```

```
% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function slider8_Callback(hObject, eventdata, handles)
value_offset=get(handles.slider8,'Value');
set(handles.edit9,'String',value_offset);
global mm;
global im;
global m
if(ndims(im)>2)
   mm=rgb2gray(im);
else
   mm=im;
end
m= mm + value_offset;
axes(handles.axes1);
imshow(m,[]);
axes(handles.axes3);
imhist(m);

% hObject    handle to slider8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider8_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
```

```
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end


% --- Executes on slider movement.
function slider9_Callback(hObject, eventdata, handles)
value_gain=1 + get(handles.slider9,'Value');
set(handles.edit10,'String',value_gain);
global mm;
global im;
global m
if(ndims(im)>2)
    mm=rgb2gray(im);
else
    mm=im;
end
m= mm.*value_gain;
axes(handles.axes1);
imshow(m,[]);
axes(handles.axes3);
imhist(m);

% hObject    handle to slider9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end

function edit9_Callback(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
% Hints: get(hObject,'String') returns contents of edit9 as text
%        str2double(get(hObject,'String')) returns contents of edit9 as a double


% --- Executes during object creation, after setting all properties.
function edit9_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

function edit10_Callback(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit10 as text
%        str2double(get(hObject,'String')) returns contents of edit10 as a double

% --- Executes during object creation, after setting all properties.
function edit10_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end

% --- Executes on slider movement.
function slider11_Callback(hObject, eventdata, handles)
global im;
global m;
me=(get(handles.slider11,'value')).^1+1;
set(handles.edit12,'String',me);
if(ndims(im)>2)
   Red=im(:,:,1);
   Green=im(:,:,2);
   Blue=im(:,:,3);
```

```
    [R C]=size(Red);
    R1=round(R.*(1-1./me)./2)+1
    R2=round(R.*(1+1./me)./2)
    C1=round(C.*(1-1./me)./2)+1
    C2=round(C.*(1+1./me)./2)
    I_C=Red(R1:R2,C1:C2);
    I_Z1=imresize(I_C,me);

    [R C]=size(Green);
    R1=round(R.*(1-1./me)./2)+1
    R2=round(R.*(1+1./me)./2)
    C1=round(C.*(1-1./me)./2)+1
    C2=round(C.*(1+1./me)./2)
    I_C=Green(R1:R2,C1:C2);
    I_Z2=imresize(I_C,me);

    [R C]=size(Blue);
    R1=round(R.*(1-1./me)./2)+1
    R2=round(R.*(1+1./me)./2)
    C1=round(C.*(1-1./me)./2)+1
    C2=round(C.*(1+1./me)./2)
    I_C=Blue(R1:R2,C1:C2);
    I_Z3=imresize(I_C,me);

    z(:,:,1)=I_Z1;
    z(:,:,2)=I_Z2;
    z(:,:,3)=I_Z3;
    axes(handles.axes1);
    imshow(z,[])
    m=z;
  else
    img=im;
   [R C]=size(img);
   R1=round(R.*(1-1./me)./2)+1
   R2=round(R.*(1+1./me)./2)
   C1=round(C.*(1-1./me)./2)+1
   C2=round(C.*(1+1./me)./2)
   I_c=img(R1:R2,C1:C2);
   I_z=imresize(I_c,me);
   axes(handles.axes1);
   imshow(I_z,[]);
   m=I_z;
  end
```

```
% hObject    handle to slider11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider


% --- Executes during object creation, after setting all properties.
function slider11_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end



function edit12_Callback(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit12 as text
%        str2double(get(hObject,'String')) returns contents of edit12 as a double


% --- Executes during object creation, after setting all properties.
function edit12_CreateFcn(hObject, eventdata, handles)
% hObject    handle to edit12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

B: Supporting Documents

- Field guide to Image Processing by Khan M. Iftekharuddin and Abdul A. Awwal