EC 247 – Computing for Engineers

Assignment 1

Graphical User Interface Design

Lecturers In-charge – Ms. Lakmini Malasinghe / Ms. Swapna Premasiri

Submitted By:

EN 14538506 – Harischandra P.A.G.

14th of September 2015

1 Abstract

This report was written for an image analyzer which was made using mat lab GUI and mat lab built in function. Mat lab provides image processing tool box which contains comprehensive set of standard algorithms, functions. Using this toolbox user can analysis image, visualize image develop new algorithms based on image processing.

Mat lab GUI means graphical user interface, GUI provides graphically devices and components that make user friendly interface perform vary tasks. When use GUI it’s not necessary to know how function are works. GUI components can be toolbars, menus, pushbutton, check boxes, list boxes and sliders. Displaying data can be done using axes or charts. These each component has their own call back functions which are auto generated when GUI run once. When a component is selected in mat lab GUI its own callback function is generated in an .m file. The execution of each callback is triggered by a particular user action such as a button push, mouse click, selection of a menu item, or the cursor passing over a component. This image analyzer contains basic image analyzing concepts.

Table of Contents

Table of Figures

2 Introduction

This assignment was to build an image analyzer using Mat lab graphics user interface (GUI).a GUI is a user interface which graphically display window with buttons, text boxes, pop menu, sliders. Etc. most functions that we perform using mat lab command window can be easily perform using mat lab UI, rather than typing function in Command window. When creating a UI need to consider about few thing for instance lay out contrails of User interface, function that runs when a UI controls is used (callback), gather input which are entered by user. Mat lab GUI is easy to use since user doesn't need to understand about coding stuff and how it works. GUI created with Mat lab can perform vary operations like any type of computation, read data files, write data files, communicate with other GUIs, display data as texts, picture, 2D plots, 3D plots etc.

This report contest detailed discussion for image processing tasks that can perform using my Image analyzer. Some features are directly related to lab session that we have done already gray-scale transformations, RGB to Black & white, transformations, contour plotting, image resizing, inverting colors of an image, and some are that I added myself referring mat lab image processing tool box such as arithmetic operations, adding noise, filtering image crop, rotating etc. and for GUI push buttons and edit text static text were done in lab session. I have added axes to display image, slider, pop menu, and text box, edit text to gather user input, dialog box to gather user input data etc.

In discussion session I have include all functions that I have used in my image analyzer with a description and coding parts., all functions were taken from mat lab image processing tool box. For Image processing Operations that can perform using image analyzer, I have attached images that before analyzed and after analyzed to compare them at once.

3 Discussion

This Image analyzer can perform few image analyzing operations using mat lab built in faction. User Interface also have user input text, numerical data, loading image from any directory without typing the file path and save image in any directory. Last analyzed image can save using save button in image analyzer.

3.1 Read an Image

This image analyzer has Load button just below the display which used to read an image. This button will be displayed on axes 1, the image which at given path in file directory. When Load button is clicked it will pop up the image dialog box which includes file formats in the list which can read. When the image is selected and clicked open *imgetfile* returns the full path in the return value *filename* and return value of *user_canceled* will be false. If cancel button or close button of the window was clicked it returns true to return value of *user_canceled* and return value of filename is set with an empty string. Then a massage box will be display and mention '*Image Not Loaded'*. Mat lab Function for Open Image dialog box is,

Figure 3.1 Load button in UI

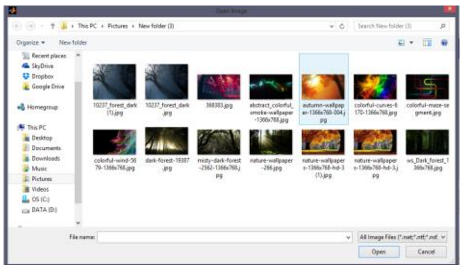
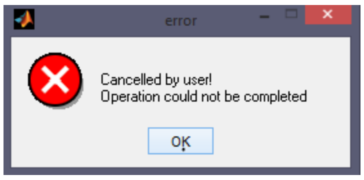


Figure.3.2 file directory opens when click load button



Cancelled by user!
Operation could not be completed

Figure 3.3 error massage box when cancelled

[Filename, user_canceled] = imgetfile

In the code *filename* has been modified as *impath* and using *imread ('impath'*), it read image from graphics file in *impath*.

'Handle' is a very important property in GUI. Because this assigned at run time. User does not get to specify the handle .but used in many time in call back functions.

3.2 Save an Image

In this call back function two Mat lab inbuilt functions were used to save the file. Firstly using *uipitfile* the dialog box displays for saving files then using *imwrite* the image file writes to graphics file in given format. In all call back functions a variable called *imfinal* is assigned to last updated image. This the variable that used in save function. This *imfinal* variable always updated when an operation is executed.
Imfinal variable is taken as a global variable so that it can be used for all functions as last updated image.

```
[FileName, PathName] = uiputfile ('*.png', 'Save As');
```

Above function also returns the path to PathName if the operation is cancelled it retunes false for both arguments.

```
x = fullfile (PathName, FileName);
```

X is a variable that assigned for gather all data which were in *uiputfile'*s return values.

```
Imwrite (imfinal, x, 'png');
```

Additional parameter also can specify for out file such as JPEG, TIFF, PNG, PPM, BMP files by putting more arguments. If the leave only the files name it returns in ans.

Figure3.4 save button in IU

A wait bar was used in saving operation so it run until the operation is done.

```
waitbar( x ,'Processing...');
```

'X' is the fractional length of wait bar.*'X'* must be between 0 and 1.

3.3 Reset to Original Image

When Reset button is clicked the original image will be replaced at reset variable in reset button's callback function. And the *imfinal* variable in save button's callback function also replaced with this original image in this reset callback function. In this function two global variables are used to reset operation code as follows

```
global im imfinal
reset = im;
axes (handles.axes2);
imshow (reset);
imfinal = reset;
```

3.4 Display Image

Mat lab UI has a function called *axes (…);* to create axes graphics objects. All function in this image analyzer has this axes function to display original and analyzed images separately. After executing axes command *imshow (…)* to display the analyzed or original image in relevant axes.

axes (handles.axes2);

handles.axes2 means that the current axes should be set to axes 2. Any graphic figure now applied to axes 2 otherwise the graphic figure will be overridden the default function. Then *imshow(X);* function is executed and display the image X in axes 2, since the axes 2 has been pointed before. When an axes is dropped on GUI workspace it has x axes and t axes with labeled scale. In this case our task is to display image so those axes scales can be removed change setting in property inspector-tick and Y-tick should be cleared.

imshow (X);

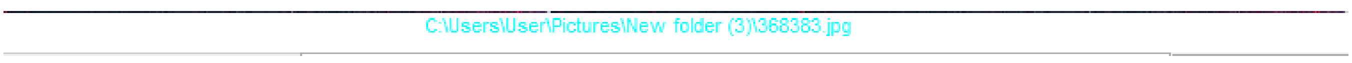
Above function displays the image X in a handles graphics figure.

3.5 Display Image Path

An edit text was used to display the read image's path in file directory so that user can whether correct image has been loaded in to the image analyzer database.

set (handles.text3,'String',impath);

set (…) functions was used to set the path in text box. *Handles.text3* means the string in *impath* is set to edit text which has tag called '*text3'.*



C:\Users\User\Pictures\New folder (3)\368383.jpg

Figure 3.5 file path displays in edit text

Faction *set (…)* allows users to obtain object for a single property at run time. Usually used in edit text.

3.6 Gray Scale Image

This function was used to covert RGB image or color map to grayscale image.

Grayimage = rgb2gray (imgrgb);

rgb2gray (*imrgb*) function coverts the RGB image also known as true color image in variable '*imrgb'* to a grayscale image by eliminating the hue and saturation. True color image is type of 3D numeric array. Converted gray scale image is returned as a numeric array.

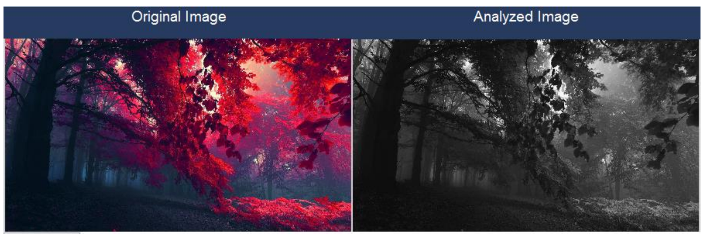


Figure 3.6 original vs grayscale image

3.7 Command line

A command line was added to operate user enter operations. But it can only run basics operations by typing used component name (string).

cmd_sting = get (handles.edit3,'String');
Using function *get (…)* user input text is returned as a string. *'handles.edit3'* means it handling edit3 edit text.
Using a switch case statements the operation is run.

```
switch cmd_sting
   case  'load'
      pushbutton6_Callback (hObject, eventdata, handles);
   case  'save'
      save_Callback (hObject, eventdata, handles);
   case 'reset'
      reset_Callback (hObject, eventdata, handles);
   case 'gray scale'
      grayscale_Callback (hObject, eventdata, handles);
…
```

Case called the call back function directly when matching relevant string is entered from *edit.text3*

Figure 3.7 Command line

3.8 Black & White image

This function was used to covert true color image to black & white image. This concept also known as covert RGB to a binary image.

imblack = im2bw (im , answer_t);

'im' is the input image and '*answer_t'* is the threshold value. Input image can be a grayscale image, if not this function automatically converts the image to a grayscale then it converts to a binary image by thresholding. Here the threshold level can set by user. Binary image has zeros and ones. When function is executed pixels in input image with luminance greater that threshold level will be replaced with value one (1), and pixel luminance less than of threshold level will be replaces with zeros (0). Threshold range is in between 0 to 1.the input image can be a class of uint8, uint16, single, int16 or double. Input image must be non-sparse. Output of this function is class of logical.

UI has modified to ask user enter value for threshold level. This was done by using dialog box to gather user input. When entered a value it is returned as a string .To convert into class of double *str2double (…)* function has been used.

```
prompt= {'Threshold'};
name = 'Enter Threshold';
defaultans = {'0.3'};
answer = inputdlg(prompt,name,1,defaultans);
```

```
answer_t = str2double(answer);
imblack = im2bw(im , answer_t);
axes(handles.axes2);
imshow(imblack);
imfinal = imblack;

answer = inputdlg(prompt,name,1,defaultans);
```

Above *inputdlg (,)* function prompt a dialog box named '*Enter Threshold'*. Default threshold level was set to 0.3.

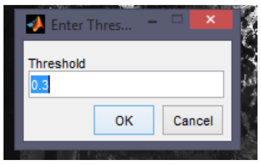


Figure 3.9 Threshold Value enter Dialog box

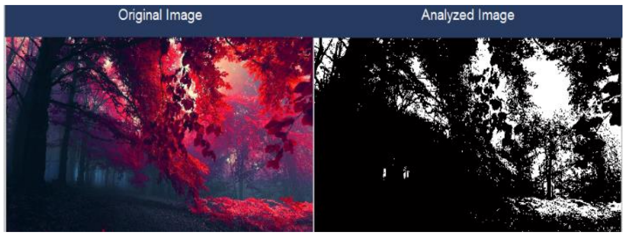


Figure 3.8 original vs BW image threshold 0.3

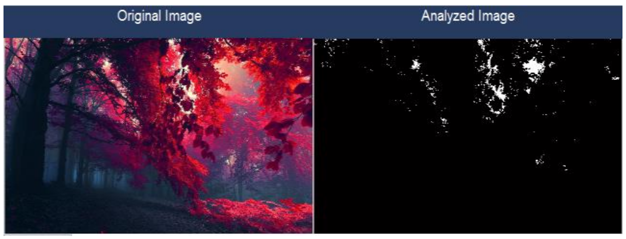


Figure 3.10 original vs BW image threshold 0.7

3.9 Gray-slice Image

This function was used to convert grayscale image to indexed image using multilevel thresholding. This function returns an indexed image. Gray slice uses threshold values to change image intensity. And the class of output image is depends on threshold value. If value is less than 256 then output image is of class unit8.

3.10 G to Indexed image

This function was used to convert a grayscale image or a binary image to an indexed image.

```
grayimage2 = rgb2gray (im2);
prompt = {'Map'};
name = 'Enter Colormap';
defaultans = {'64'};
answer = inputdlg(prompt,name,1,defaultans);
answer_dub = str2double (answer);
[X, map] = gray2ind (grayimage2, answer_dub);
axes (handles.axes2);
imshow (X, map);
imfinal = X;


[X, map] = gray2ind (grayimage2, answer_dub);
```

‘answer_dub’ is a user input value which specifies the size of the color map. *‘answer_dub’* has been converted to class of double using *‘str2double’*.thhis values must be an integer between 1 and 65536. If n is omitted, it has set been defaults to integer 64.

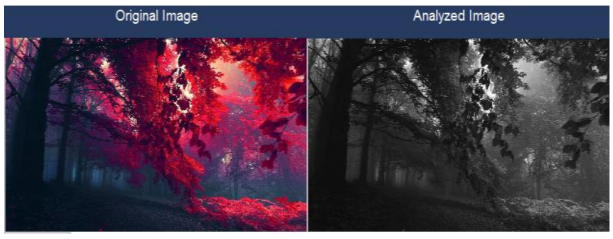


Figure 3.11 original image vs gray-slice image

3.11 Invert Colors of an Image

This function was used compliment an image. Image that was used to compliment can be a binary, grayscale or a RGB image. After complement is done image still remain its class and size.

im3 = imcomplement (im2);

When a binary image is complemented, all ones become zeros and all zeros become ones. In other words Black and white pixels are reversed. When a RGB image is complemented, every each pixel’s intensity value will be subtracted from its maximum value which is supported by its class.in output image dark areas become lighter and light areas become darker.

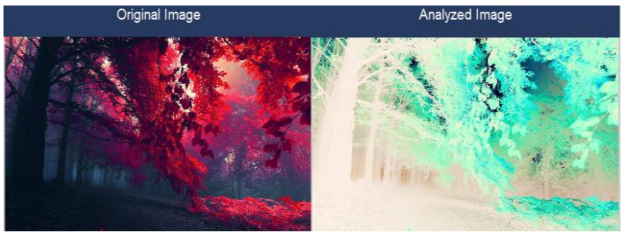


Figure 3.12original image vs pixel color inverted image

3.12 Crop an Image

Image cropping was done with below function. When button is clicked a menu with few operation displays.

crp = imcrop (im2);

Input image *im2* can be a grayscale, RGB image or a logical array. But must be a real one and non-sparse. *Imcrop (…)* accepts any class that *imshow* function is accepted.

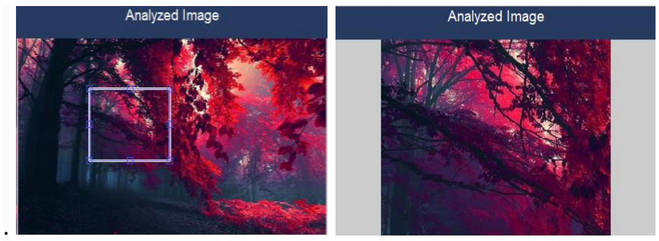


Figure 3.13 Image cropping

3.13 Image Brightness

Brightness of an image was set to change using a slider bar and *imadd (...);* function this also can be done using *imadjust (…)* If there any change in slider it should be updated in code.so that *handles.brt = brt;* was used to update handles structure

```
brt = im2;
axes(handles.axes2);
imshow(brt);
handles.brt = brt;
% Update handles structure
guidata(hObject, handles);
```

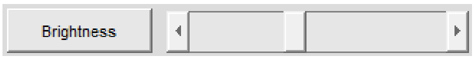


Figure 3.14 Brightness slider

Using slider its position was taken slider. Minimum position of the slider was set 0.0 and maximum of it was set to 255.0. Slider position was taken to return value *'Value'* in function *get (hObject,'Value')*.this value is double since the scale of slider was give as in class of double (0.0 - 255.0).then this return value directly put into the *imadd (…);* function as *st_val.*

```
guidata (hObject, handles);
```

Above Function was used to store UI data. This function stores the variable '*handles'* where the object specified by '*hObject'*.the data can be any Mat lab variable. Code in slider's callback as follows.

```
brt = handles.brt;
%returns position of slider
st_val = get (hObject,'Value')
%position from slider
add = imadd(brt,st_val);
axes (handles.axes2);
imshow (add);
imfinal = add;
```

Faction *get (…)* allows users to obtain object for a single property at run time.

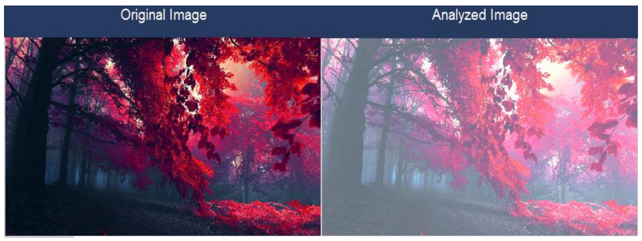


Figure 3.15 brightness when slider at above position

3.14 Filter Menu

In image analyzing filtering is an important operation. This image analyzer includes five filtering methods. They are *'motion', 'Circular averaging filter (pillbox)',' laplacian' ,'log', 'sobel'.* In some filter we can change their parameters .Input dialog boxes were used to gather those user input data. And a pop menu was used make easier to operate these filtering tabs.

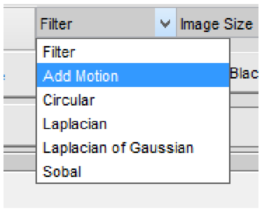


Figure 3.16 Filter menu

contents = get (hObject, 'Value');

Above function is an in built Mat lab function also shown in callback function as *contents = cellstr(get (hObject, 'String')).* This function returns filter content as cell array.hObjects is used to handle to filter pop menu. And dialog will be appear when a filter operation is clicked, if there are any parameter that user can change.

```
prompt = {'Motion'};
name = 'Linear Motion';
defaultans = {'40'};

answer_m = str2double (inputdlg (prompt, name, 1, defaultans));
prompt = {'Angle'};
name = 'Clockwise Angle';
defaultans = {'50'};
answer_t = str2double (inputdlg (prompt,name,1,defaultans));
h = fspecial ('motion', answer_m, answer_t);
filt = imfilter (im2, h);
axes (handles.axes2);
imshow (filt);
```

Above function is for analyze the linear motion of a camera from dialog box two parameters were taken and they were covert to class of double using *str2double (...)* function.

```
h = fspecial ('disk', answer_r); %Circular averaging filter (pillbox)
filt = imfilter (im2, h, 'replicate');
```

Above function is Circular averaging filter it returns square matrix of size 2*radius+1.the default radius was set to 5.

```
h = fspecia l('laplacian', answer_a);
```

```
filt = imfilter(im2,h,'replicate');
```

Above function was used to analyze the two dimensional laplacian operator. The parameter '*answer_a'* controls the shape of laplacian and values of it range of 0.0 to1.0.

```
h = fspecial ('log', answer_s); %Laplacian of Gaussian filter
filt = imfilter (im2,h,'replicate');
```

Above function returns a rotationally symmetric Laplacian of Gaussian filter '*answer_s'* is the standard deviation known as sigma.

```
h = fspecial ('sobel'); %Sobel horizontal edge-emphasizing filter
filt = imfilter (im2,h,'replicate');
```

Above function returns return output image includes with emphasizes horizontal edges. This done by using the smoothing effect by approximating a vertical gradient.

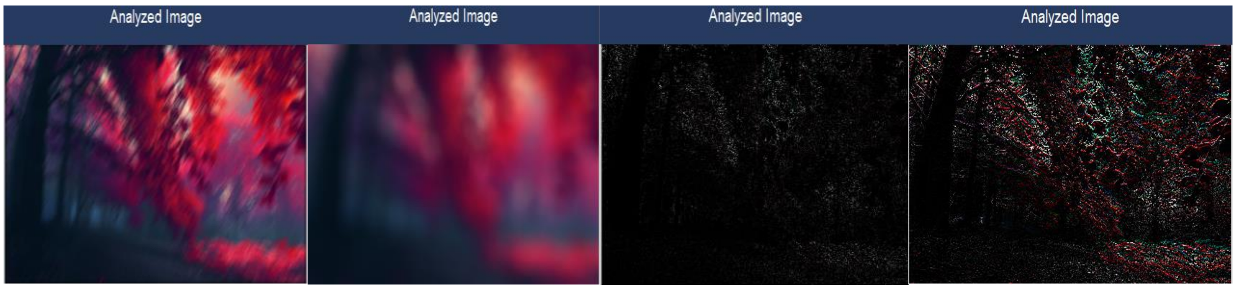


Figure 3.17 filter operation motion 40 at angle 50, circular averaging disc radius 50, laplacian at default sigma and sobal filter

3.15 Image Resize

Image resign was done using *imresize (...)* function. A popup menu was used to have few predefined size .and other sizes can set by clicking '*other'* tab. and also this image size can be directly enter using edit text.

Figure 3.18 user enter image size through edit text

```
contents = get(hObject,'Value')
switch contents

  case 2
    imre = imresize(im2,[20 20]);
    axes(handles.axes2);
    imshow(imre);
```

When clicked a label in popup menu return value is taken by variable '*contents'* and '*hObjects*' is to handle the popup menu.

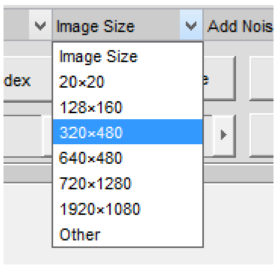


Figure 3.19 Image resize menu

```
case 8
   prompt={'x-axis'};
   name = 'Horizontal';
   defaultans = {'0'};
   answer_xa = str2double(inputdlg(prompt,name,1,defaultans));
   prompt={'y-axis'};
   name = 'Vertical';
   defaultans = {'0'};
   answer_ya = str2double(inputdlg(prompt,name,1,defaultans));
   imre = imresize(im2,[answer_xa answer_ya]);
   axes(handles.axes2);
   imshow(imre)
```

Above code part is for other label in popup menu .this code for gather user input from dialog box. Since the dialog box returns the value as a sting it should be coveted to double using *str2double (…)* function.

Below code is for gather user input data through edit text. Edit text returns a string so that its should be converted to double before passing to the *imresize (…)* function.

```
y_val = str2double (get(hObject,'String'));
x_val= str2double (get(hObject,'String'));
imre1 = imresize (im2,[x_val y_val]);
Axes (handles.axes2);
imshow(imre1);
imfinal = imre1 ;
```

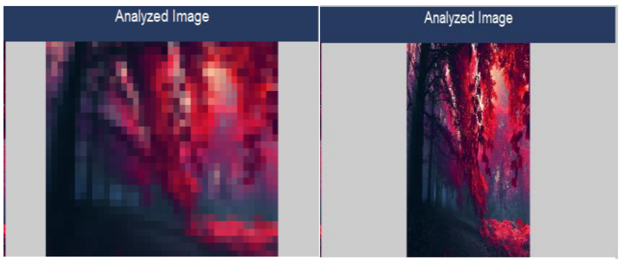


Figure 3.20 resized image size in 30x40 and 640x480

3.16 Add Noise

Noise can added using add noise popup menu .this image analyzer can perform four noise adding operations. When clicked a label in popup menu return value is taken by variable ‘*contents’* and ‘*hObjects*’ is to handle the popup menu.

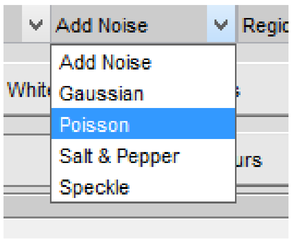


Figure 3.21 add noise menu

contents = get (hObject,'Value')

J = imnoise(im2,'gaussian');
This the Gaussian white noise with constant mean and variance

J = imnoise(im2,'poisson');
Above *'Poisson'* noise added by adding artificial noise to the data.

J = imnoise(im2,'salt & pepper');
Above *'salt & pepper' noise* is simply operates as on and of pixels.

J = imnoise(im2,'speckle');
Above function was used to added multiplicative noise to the image

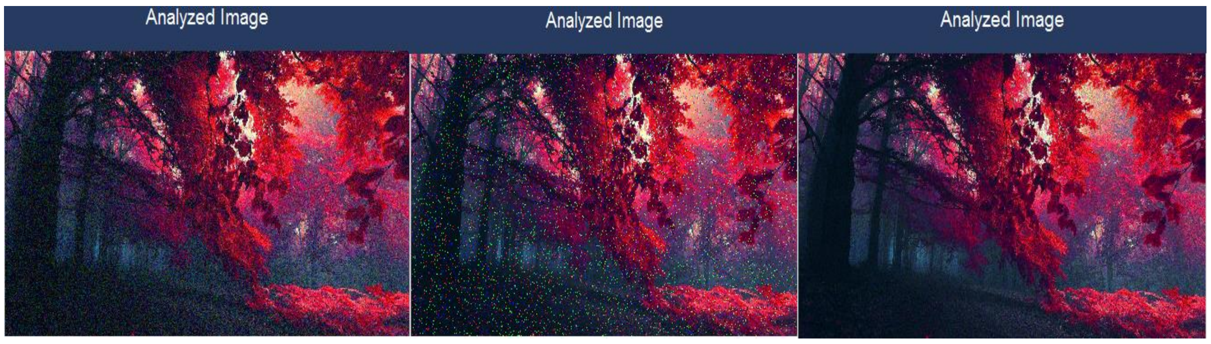


Figure 3.22Nois Gaussian, salt & paper, speckle

3.17 Arithmetic

Arithmetic operations also can be done with images following arithmetic operations are AVAILABLE IN THIS MAGE analyzer simply adding, subtracting, multiplying, dividing can be done with user input data. User input data gathers from input dialog box constant and scalar values can set using dialog box. For arithmetic operation text box was used to label all operations.

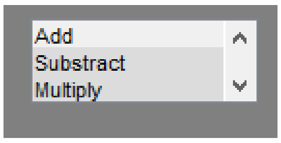


Figure 3.23text box for arithmetic operation list

[impath1, user_canceled1] = imgetfile;
Text box is also operates as popup menu in the switch case operation was done.

contents = get (hObject,'Value');
contents is the case in the switch case code

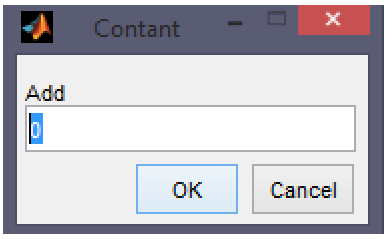


Figure 3.24 User input data using dialog box

contant = str2double(inputdlg(prompt,name,1,defaultans));
From dialog box, parameter was taken and it was covert to class of double using *str2double (...)* function.

K = imadd (I,contant); function was used to add constant to the loaded image. Contrast of the image will be increased

K = imsubtract (I,contant); function was used to subtract constant from image. Contrast of the image will be decreased

K = immultiply (I,Scaler); function was used to multiply image by a scalar . Contrast of the image will be increased

K = imdivide (I,Scaler); function was used to divide image by a scalar. Contrast of the image will be decreased

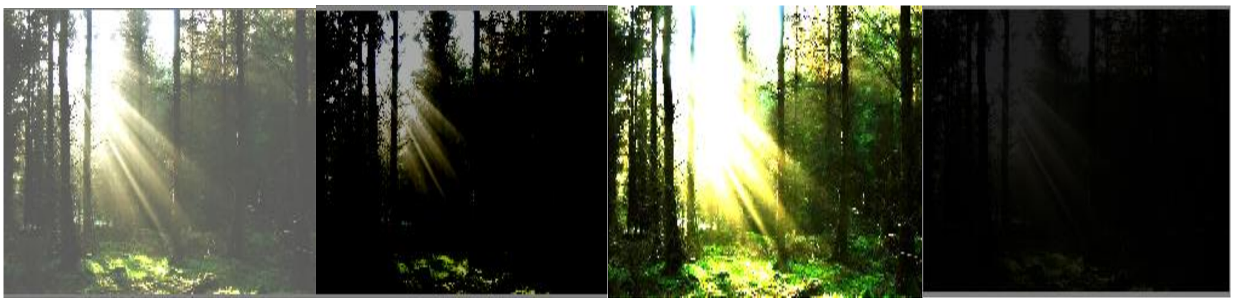


Figure 3.25 image 1 constant 80 was added, image2 constant 40 was subtracted, image3 multiplied by scale 5, image4 divided by scale 8

3.18 Image Rotating

In this operation image rotates around its center point. If user gave a positive value for angle image rotates counter clockwise and if negative angle was given its rotate in anti-clock wise direction.

```
 answer_an = str2double (inputdlg (prompt,name,1,defaultans));
```

From dialog box, a string was taken and it was covert to class of double using *str2double (...)* function.

```
im_r_v = imrotate (im2,answer_an);
```

Above function was used to user enter angle function.

```
im5 = imrotate (im,90);
im4 = imrotate (im,-90);
```

Above two function are for two push buttons one rotate right other for rotate left.

A waiting bar was used t since this operation take some time.

```
w = waitbar(0, 'Processing...');

p_steps = 50;

for p_step = 1:p_steps
im4 = imrotate (im,-90);
%computations take place here
Waitbar (p_step / p_steps);
end
close (w);
```

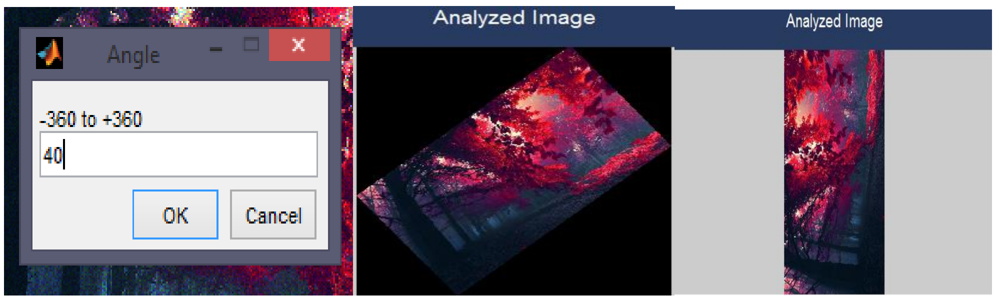


Figure 3.26 image 1 entering angle 40 degree, image 2 40degree rotated image, image 3 using Rotate-L button

4 Conclusions

- Mat lab provides image processing tool box which contains comprehensive set of standard algorithms, functions.
- GUI provides graphically devices and components that make user friendly interface perform vary tasks. When use GUI it's not necessary to know how function are works
- For this Image larger files are not accepted. GUI get not responding when large size of image was loaded. This due to memory overflow
- Mat lab GUI's components are objects. All of these object has unique handle. Every each object has own properties that can change in property inspector.

5 References

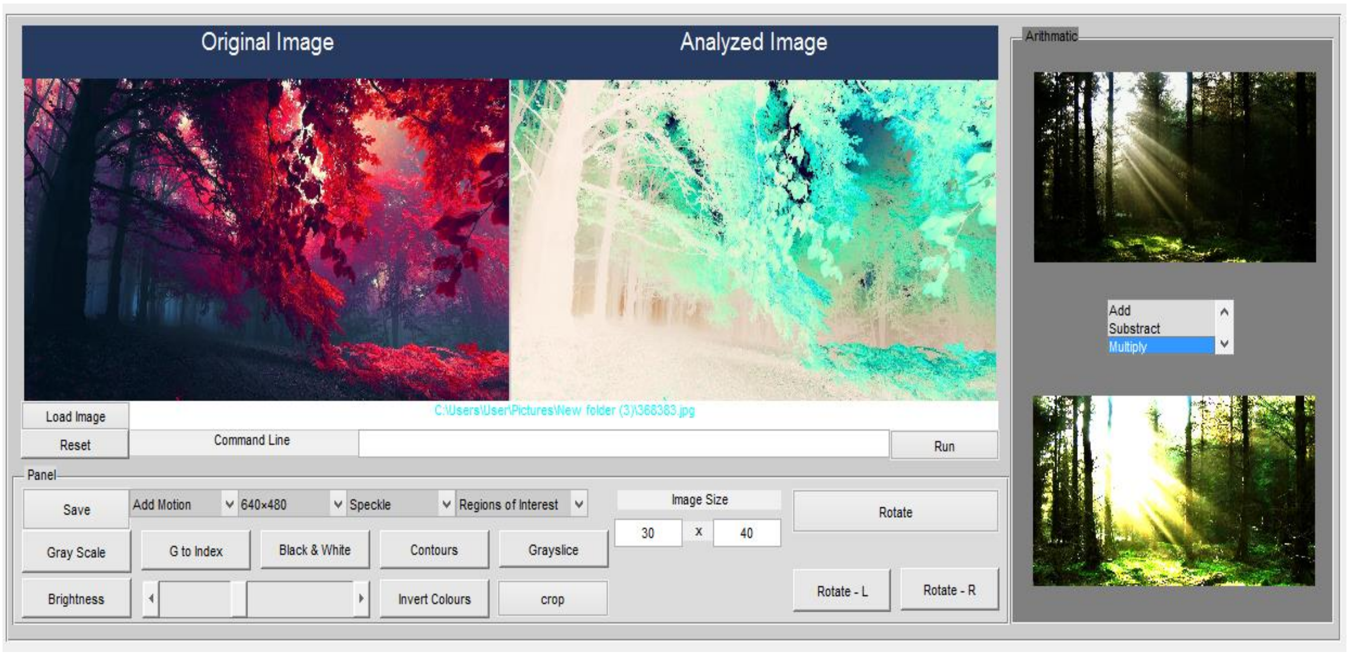
[4] "Matlab Mathworks," MATLAB, [Online]. Available: http://in.mathworks.com.html. [Accessed 02 09 2015].

[5] "Image Processing Toolbox Functions," Matlab Mathworks, [Online]. Available: http://in.mathworks.com/help/images/functionlist.html. [Accessed 01 09 2015].

[6] "Tagged Questions," Stackoverflow, [Online]. Available: http://stackoverflow.com/questions/tagged/matlab. [Accessed 06 09 2015].

6 Appendix

6.1 User Interface



C:\Users\User\Pictures\New folder (3)\368383.jpg

Image Size 30 x 40

Figure 6.1 User interface of image analyzer