

Problem

Write your own MATLAB Routines for the following:

1. Histogram, Normalized Histogram, Histogram equalization
2. Contrast Stretching
3. Bit-plane slicing
4. Power Law Transformation
5. Intensity Level Slicing (both cases)

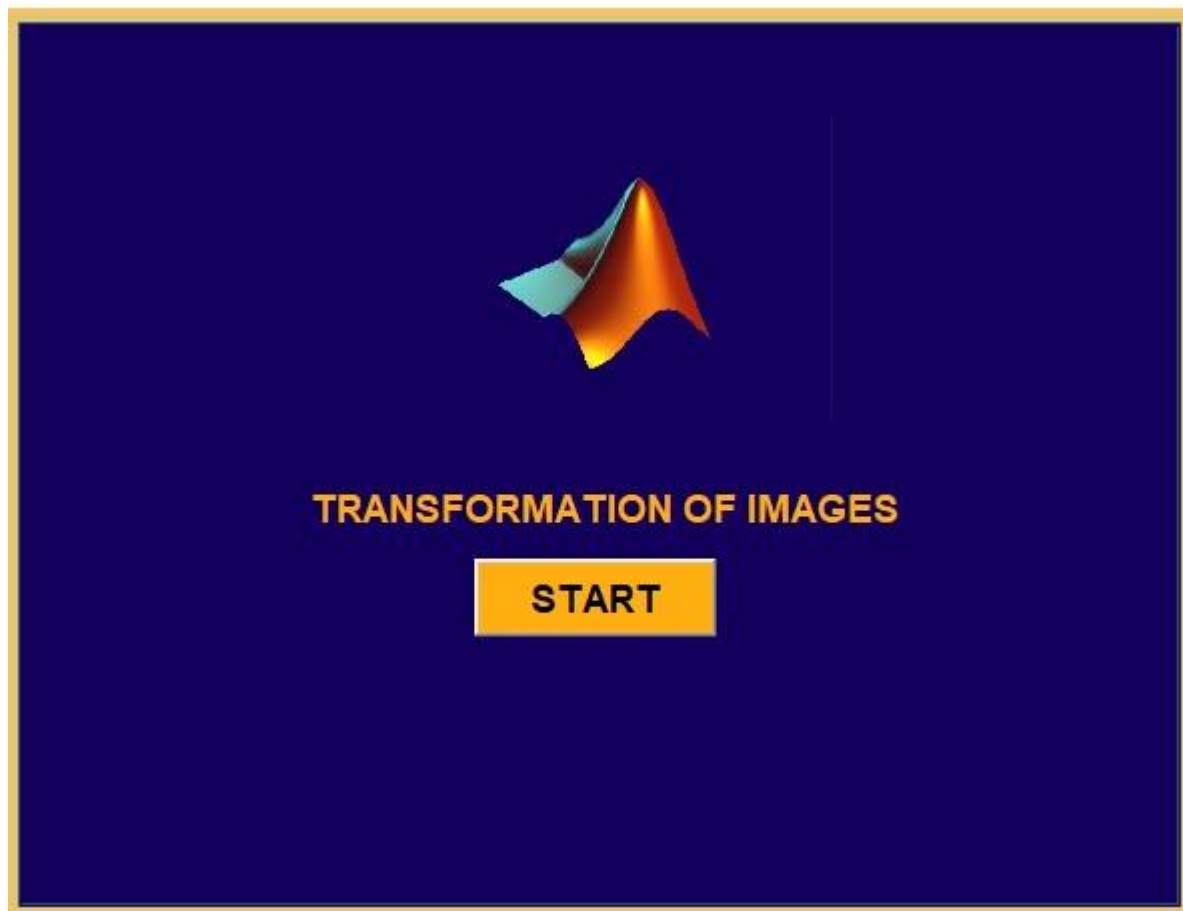
All Inputs should be in the form of Images. Design a graphical user interface which prompts the user to select an image file, initially, from any directory from the computer system. Display the options in points 1-5 in the form of radio buttons. For every transformation (1-5) appropriate range of operations should be provided in the form of text inputs or slide bars, radio buttons etc. All outputs should be in the form of images, and additionally, in the form of graphs (e.g. in case of histograms, or contrast stretching) demonstrating the initial transformation function or the effect of Transformation after application.

Explanation

In this digital image processing application, We can upload an image from any location in our system and apply different transformation and make histogram of image. This application has different function like Histogram, Normalized Histogram, Histogram equalization, Contrast Stretching, Bit plane slicing, Power Law Transformation, Intensity Level Slicing (both cases). We can upload an image from any location in our system and apply these transformation on that selected image.

GUI SECTION

START SCREEN



When user run application then above screen will appear, to perform operations of image processing, click on start button.

OPERATIONS SCREEN

When we press the start button then following screen is appeared.

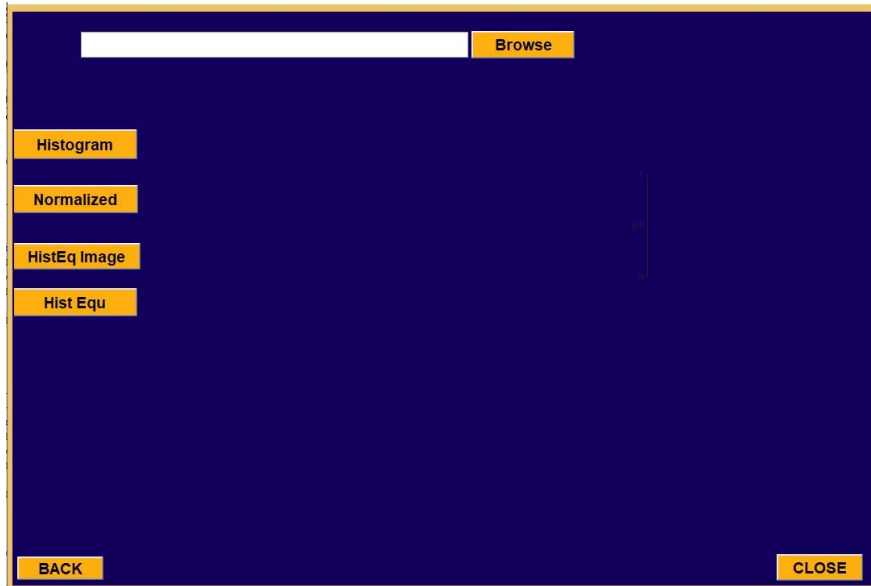


The above one picture is a dashboard screen that contain list of operations. In this screen user select a button to perform a particular operation. if user want to go back at start screen, then click on back button and if user want to close the screen, then he/she need to press the close button.

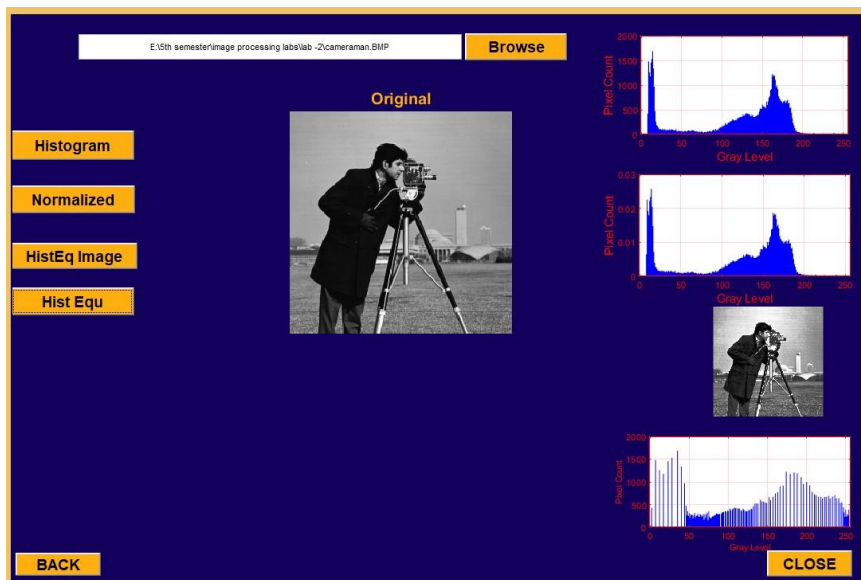
HISTOGRAMS SCREEN

In this screen firstly we need to browse a image from computer directory and then apply the operations on this image by pressing button of particular operation.

SCREEN BEFORE BROWSING ANY IMAGE:



SCREEN AFTER APPLYING OPERATIONS ON IMAGE:

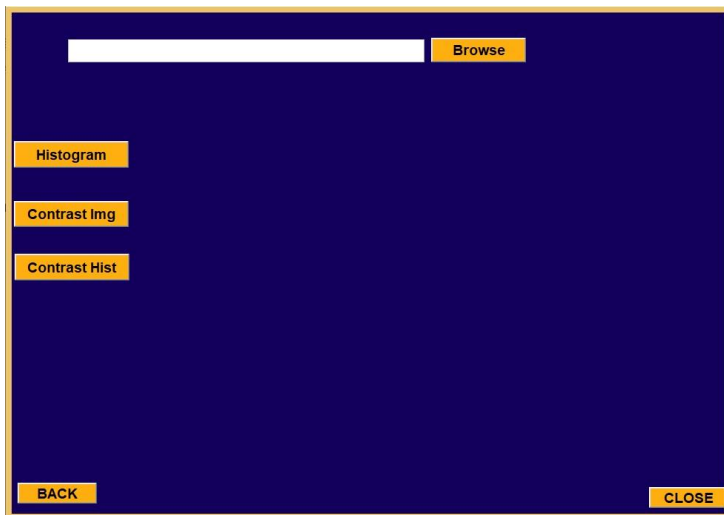


CONTRAST STRETCHING

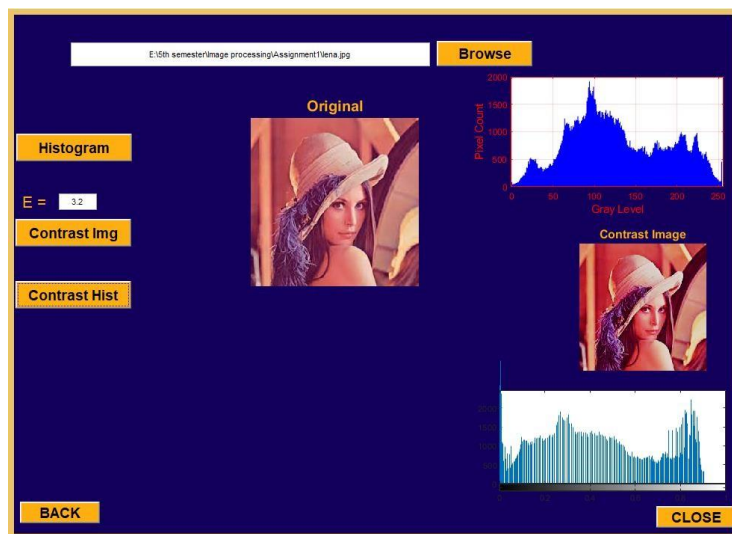
Contrast stretching is a simple image enhancement technique that attempts to improve image contrast by 'stretching' the range of intensity values contained in the image to span a desired range of values.

- Browse Image, by clicking on browse button.
- Apply Operation by pressing button of particular operation.

SCREEN BEFORE BROWSING ANY IMAGE:



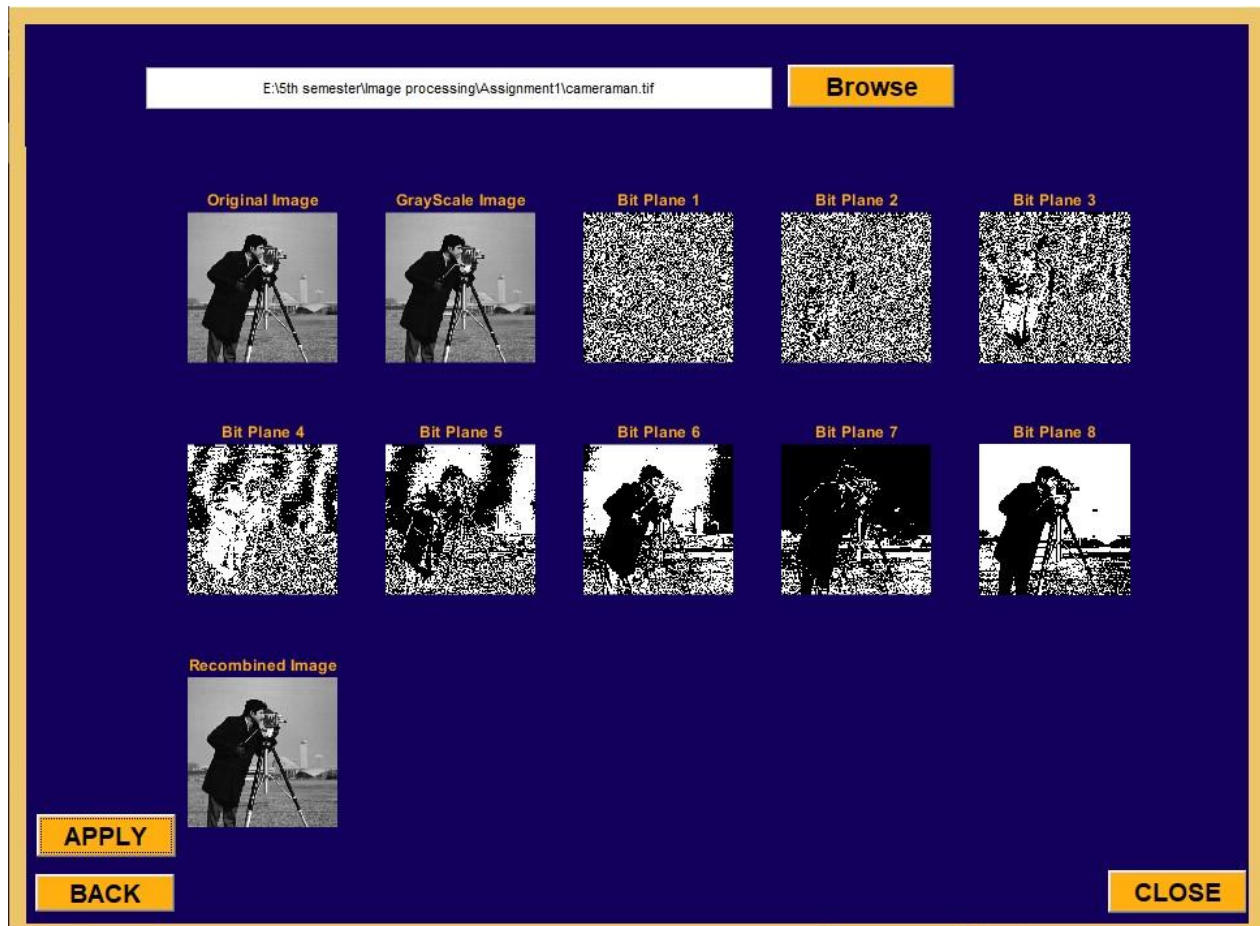
SCREEN AFTER APPLYING OPERATIONS ON IMAGE:



BIT PLANE SLICING

Bit Plane Slicing (BPS) is an image-expression method in which each pixel is represented by one or more bits of byte.

In this screen firstly browse image by pressing ' Browse ' button and then apply operation by pressing ' APPLY ' button.



Power Law Transformation

In this screen firstly we browse a image from computer directory, and then enter the value of gamma in text field and then apply operation by pressing ' APPLY ' button Formula for contrast Power Law is given below:

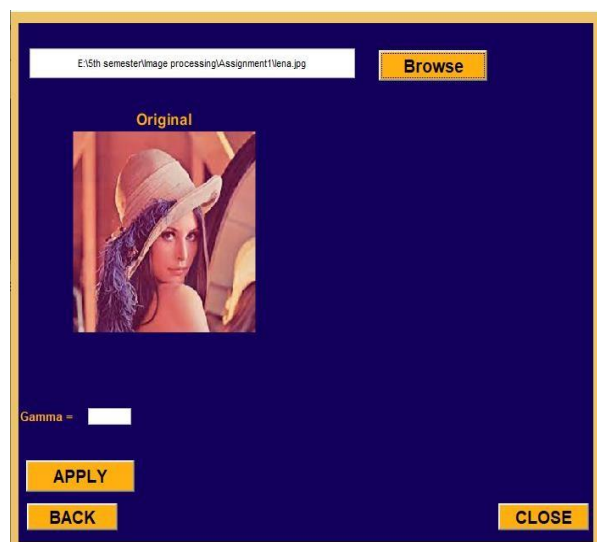
$$s = c * r^Y$$

I used this formula to perform Power Law Transformation, using y value which gamma value we can curve the grayscale either darken the intensity when value of gamma is greater than one or brighter the intensity when value of gamma is less than one. Now we will apply it to the image.

SCREEN BEFORE BROWSING ANY IMAGE

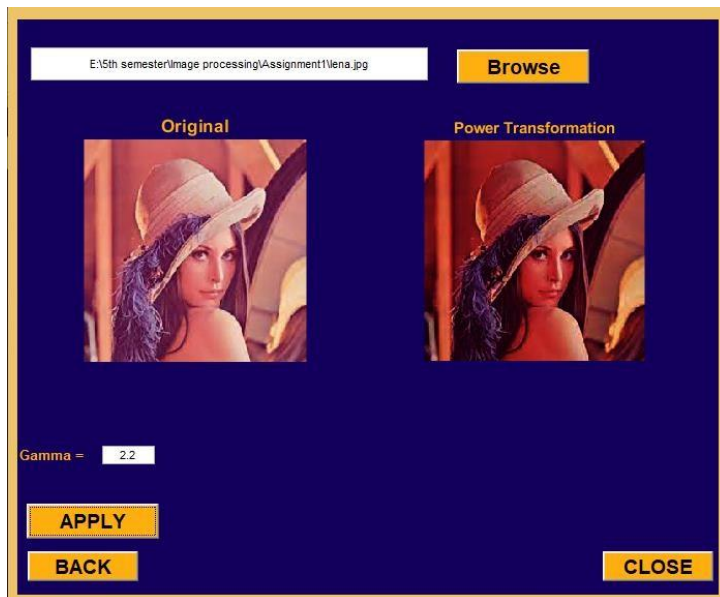


SCREEN AFTER BROWSING IMAGE

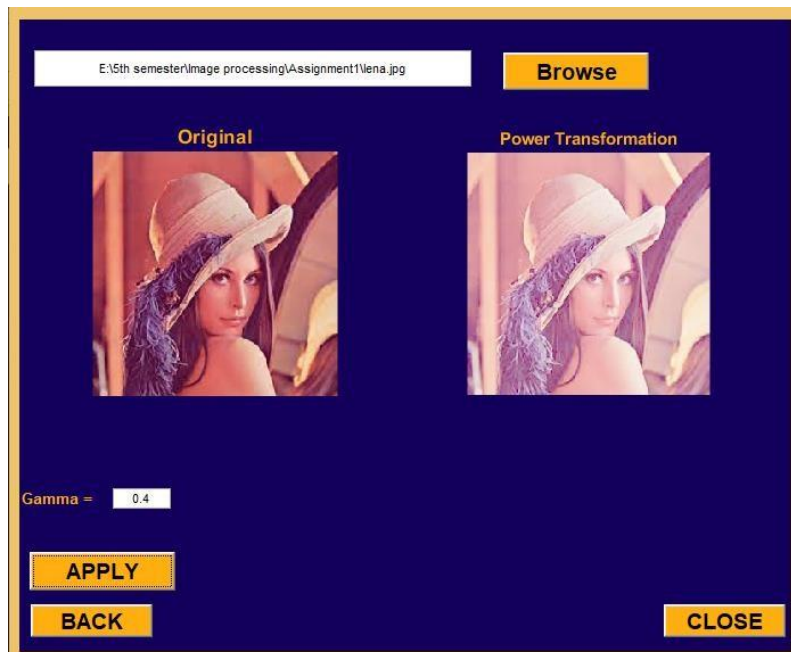


SCREEN AFTER APPLYING OPERATION ON IMAGE:

- When value of gamma = 2.2



- When value of gamma = 0.4



Intensity Level Slicing (both cases)

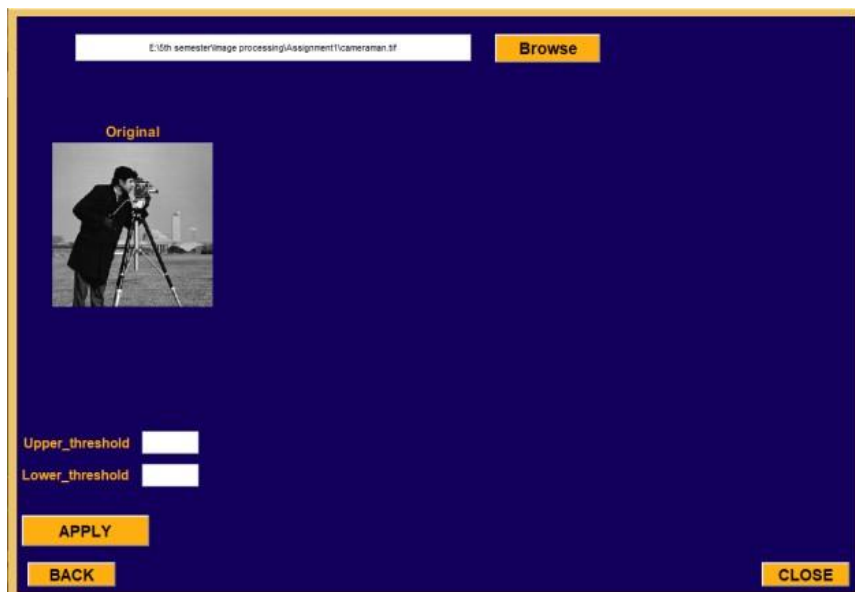
- Firstly, browse image by pressing browse button.
- Enter the upper threshold (upper value of interest) in upper text label.
- Enter the lower threshold (lower value of interest) in lower text label.
- And press 'APPLY' button for perform Operation.

SCREEN BEFORE BROWSING ANY IMAGE



The screenshot shows a dark blue application window with a yellow border. At the top, there is a white text input field and a yellow 'Browse' button. Below this, on the left side, are two labels: 'Upper_threshold' and 'Lower_threshold', each followed by a white text input field. At the bottom left, there is a yellow 'APPLY' button. At the bottom center, there is a yellow 'BACK' button. At the bottom right, there is a yellow 'CLOSE' button.

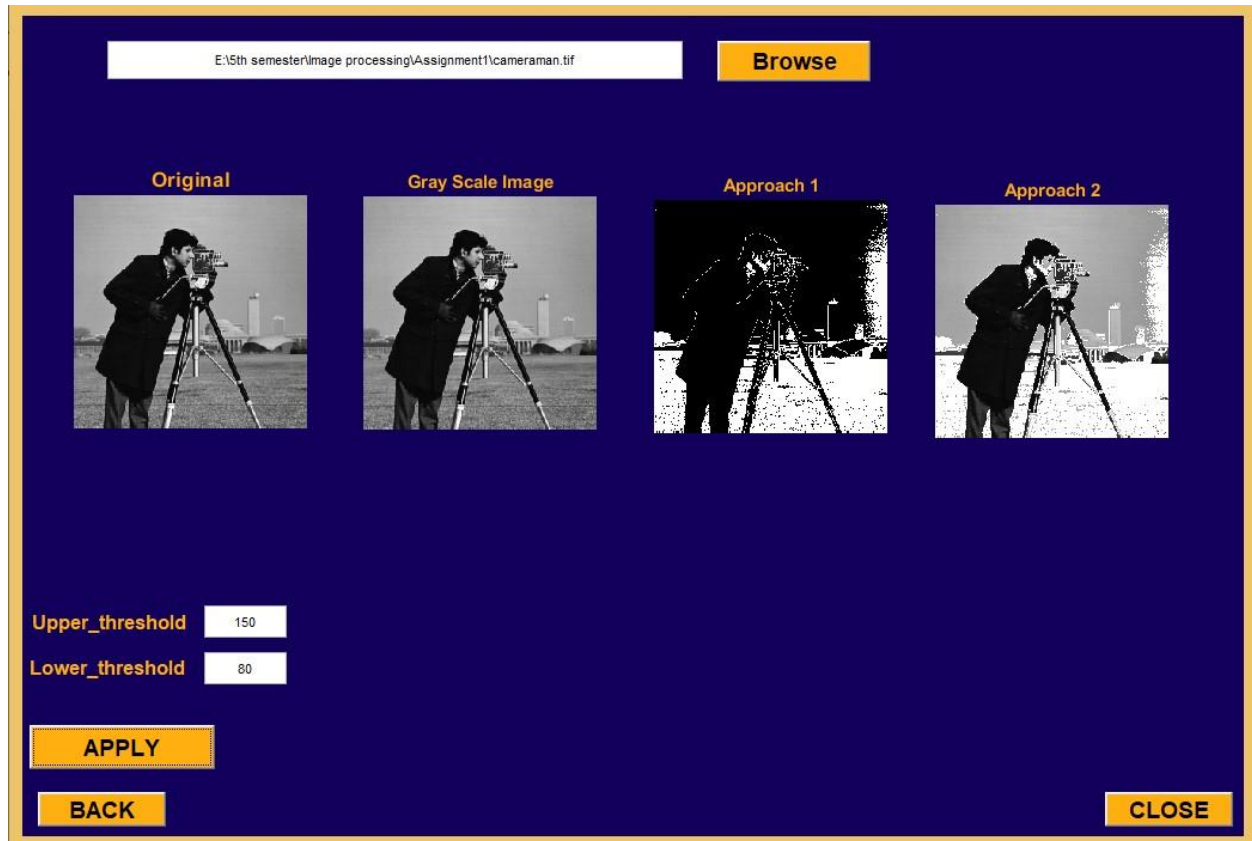
SCREEN AFTER BROWSING IMAGE



The screenshot shows the same application window as before, but now with an image displayed. The image is a black and white photograph of a person taking a picture with a camera on a tripod. Above the image, the word 'Original' is written in yellow. The rest of the interface, including the 'Browse' button, threshold input fields, 'APPLY' button, 'BACK' button, and 'CLOSE' button, remains the same as in the previous screenshot.

SCREEN AFTER APPLYING OPERATION ON IMAGE:

When upper threshold = 150 and lower threshold = 80



CODE FOR BROWSE IMAGE AND THEN SHOW

IMAGE

```
global Img;

%Browse path of image
[Filename,Pathname]=uigetfile('*.','File Selector');
name=strcat(Pathname,Filename);

%Read Image
a = imread(name);

set(handles.edit1,'string',name);
axes(handles.axes1);

%reize the Image by 256 X 256
Img=imresize(a, [256,256]);

%show Image
imshow(Img);
title("Original",'FontSize', 15,'color',[1 .69 0.06]);
```

Code To Plot A Histogram of Image

```
%set Variable
global Img
O_img = Img

%code to determine the histogram of Image
[rows, columns] = size(O_img);
counts = zeros(1, 256);

for col = 1 : columns
    for row = 1 : rows

        % Get the gray level.
        grayLevel = O_img(row, col);
        % Add 1 because graylevel zero goes into index 1 and so on.

        counts(grayLevel+ 1) = counts(grayLevel+1) + 1;
    end
end

% Plot the histogram.
grayLevels = 0 : 255;

.. .. ^

bar(grayLevels, counts, 'BarWidth', 1, 'FaceColor', 'b');
xlabel('Gray Level', 'FontSize', 11);
ylabel('Pixel Count', 'FontSize', 11);

grid on;

%change The color of gridLine
ch = gca % Get handle to current axes.

ch.XColor = 'r'; % Red
ch.YColor = 'r'; % Red
```

CODE OF HISTOGRAM NORMALIZATION OF IMAGE

```
function Normalized_Button_Callback(hObject, eventdata, handles)

%set Variable
global Img
O_img = Img

%code to determine the histogram of Image
[rows, columns] = size(O_img);
counts = zeros(1, 256);

for col = 1 : columns
    for row = 1 : rows

        % Get the gray level.
        grayLevel = O_img(row, col);
        % Add 1 because graylevel zero goes into index 1 and so on.

        counts(grayLevel+ 1) = counts(grayLevel+1) + 1;
    end
end

% Plot the histogram.
grayLevels = 0 : 255;
counts=counts/(rows*columns);
axes(handles.axes3);
bar(grayLevels, counts, 'BarWidth', 1, 'FaceColor', 'b');
xlabel('Gray Level', 'FontSize', 11);
ylabel('Pixel Count', 'FontSize', 11);
```

CODE FOR HISTOGRAM EQUALIZATION

```
%set global Variabl
global Img;

O_img = Img

% Count the totoal number of Pixel present in image
numofpixels=size(O_img,1)*size(O_img,2);

%type casting
HE=uint8(zeros(size(O_img,1),size(O_img,2)));

%for frequency calculation
freq=zeros(256,1);

%The probability of each occurrence is calculated by probf.
probf=zeros(256,1);

probc=zeros(256,1);

cum=zeros(256,1);
output=zeros(256,1);

%freq counts the occurrence of each pixel value.
```

```

for i=1:size(O_img,1)
    for j=1:size(O_img,2)
        value=O_img(i,j);
        freq(value+1)=freq(value+1)+1;
        probf(value+1)=freq(value+1)/numofpixels;
    end
end

sum=0;
no_bins=255;

%The cumulative distribution probability is calculated.
for i=1:size(probf)
    sum=sum+freq(i);
    cum(i)=sum;
    probc(i)=cum(i)/numofpixels;
    output(i)=round(probc(i)*no_bins);
end

for i=1:size(O_img,1)
    for j=1:size(O_img,2)
        HE(i,j)=output(O_img(i,j)+1);
    end
end

```

```

axes(handles.axes4);
% plot Image After Histogram Equalization
imshow(HE);

```

CODE FOR CONTRAST-STRECHING

```
function Contrast_Img_Callback(hObject, eventdata, handles)

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

%Img -> image that we browse
global Img
global g

% find the mean of image.
m=mean(Img(:));

% Take value of E from User
E=str2double(get(handles.e,'String'))

%formula to calculate contrast streching
g=1./(1 + (m./(double(Img) + eps)).^E)

%Display Image of Contrast Streching
axes(handles.axes3);
imshow(g);
title("Contrast Image",'FontSize', 12,'color',[1 .69 0.06])
```


CODE FOR BIT PLANE SLICING

```
% --- Executes on button press in Apply_button.
function Apply_button_Callback(hObject, eventdata, handles)
% hObject    handle to Apply_button (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Image that we browse
global Img

%convert Image into gray Scale
grayScale=im2gray(Img);

dImg = double(grayScale);

%get bits of image

b1 = mod(dImg, 2);
b2 = mod(floor(dImg/2), 2);
b3 = mod(floor(dImg/4), 2);
b4 = mod(floor(dImg/8), 2);
b5 = mod(floor(dImg/16), 2);
b6 = mod(floor(dImg/32), 2);
b7 = mod(floor(dImg/64), 2);
b8 = mod(floor(dImg/128), 2);
```

```

%Combine all bits
cb = (2 * (2 * (2 * (2 * (2 * (2 * (2 * b8 + b7)| + b6) + b5) + b4) + b3) + b2) + b1);

%plot original image
subplot(3, 5, 1);
imshow(Img);
title('Original Image','color',[1 .69 0.06]);

%plot gray scale image
subplot(3, 5, 2);
imshow(grayScale);
title('GrayScale Image','color',[1 .69 0.06])

%plot bitwise Image

subplot(3, 5, 3);
imshow(b1);
title('Bit Plane 1','color',[1 .69 0.06]);

subplot(3, 5, 4);
imshow(b2);
title('Bit Plane 2','color',[1 .69 0.06]);

subplot(3, 5, 5);
imshow(b3);
title('Bit Plane 3','color',[1 .69 0.06]);

subplot(3, 5, 6);
imshow(b4);
title('Bit Plane 4','color',[1 .69 0.06]);

subplot(3, 5, 7);
imshow(b5);
title('Bit Plane 5','color',[1 .69 0.06]);

subplot(3, 5, 8);
imshow(b6);
title('Bit Plane 6','color',[1 .69 0.06]);

subplot(3, 5, 9);
imshow(b7);
title('Bit Plane 7','color',[1 .69 0.06]);

subplot(3, 5, 10);
imshow(b8);
title('Bit Plane 8','color',[1 .69 0.06]);

% plotting recombined image |
subplot(3, 5, 11);
imshow(uint8(cb));
title('Recombined Image','color',[1 .69 0.06]);

```

CODE FOR POWER LAW TRANSFORMATION

```
%Img -> image that we browse
global Img;

%value of constant
c=1;

%Take value of gamma from useer
gamma=str2double(get(handles.edit3,'String'));

%Convert image into double
i=im2double(Img)

%Formula of power Law Transformation
s=c*(i.^gamma)

%Display image on axes2
axes(handles.axes2)

%show Image
imshow(s);
title("Power Transformation",'FontSize', 11, 'color',[1 .69 0.06])
```

CODE FOR INTENSITY LEVEL SLICING

```
%Img -> image that we browse
global Img;

%Convert Image Into Gray Scale
g=im2gray(Img);

%retrieve number of rows and columns
[M,N]=size(g);

%Take value of Upper threshold and lower Threshold from user
v1=str2double(get(handles.L_Value,'String'));
v2=str2double(get(handles.U_Value,'String'));

%Approach 1
%Set value of all pixels within range of interest to white and all other to
%black

a=g;
for i=1:M
    for j=1:N
        if g(i,j)>=v1 && g(i,j)<=v2
            a(i,j)=255;
        else
            a(i,j)=0;
        end
    end
end

%Approach 2
%Set value of all pixels within range of interest to white and all other
%remain unchange

b=g;
for i=1:M
    for j=1:N
        if g(i,j)>=v1 && g(i,j)<=v2
            b(i,j)=255;
        else
            b(i,j)=g(i,j);
        end
    end
end
```

```
%show gray scale image on axes 2  
axes(handles.axes2);
```

```
%show Image  
imshow(g);  
title("Gray Scale Image",'FontSize', 11, 'color',[1 .69 0.06])
```

```
%show image on axes 3  
axes(handles.axes3);  
%show Image  
imshow(a);  
title("Approach 1",'FontSize', 11, 'color',[1 .69 0.06])
```

```
%show image on axes 4  
axes(handles.axes4);  
%show Image  
imshow(b);  
-title("Approach 2",'FontSize', 11, 'color',[1 .69 0.06])
```
