The purpose of this first homework assignment is to gain familiarity with MATLAB and some image processing functions.

NOTE: This is the only assignment that will be the same for both the undergraduate (ECE 4881) and graduate (ECE 588) sections.

For this assignment you must use the .mlx format, or MATLAB Live Editor, which allows mixing code and text in the same document, similar to how Jupyter Notebooks, and Mathematica, operate. Use the Text and Code buttons to switch between those modes, adding text and code sections as needed into this document, and answer all questions in this document. You will submit one pdf file showing all work on CANVAS. Once you have completed all sections and have the answers shown that you want, select Save - Export to PDF to generate a PDF file, which will show your code and results, and upload this one file to CANVAS in the Homework 1 assignment. You can use this .mlx file as a starting point and add text and code sections as needed.

Please type your name here:

Aishwarya Dekhane (adekhane@umich.edu)

Note: MATLAB Online uses traditional Unix-like path separators ("/")

It is beneficial to keep projects local to a directory, in this instance

the "RV_W2024" directory is used to hold files for this course,

with a subdirectory of /images holding images. Keeping to this type of

organization is helpful as the number and complexity of projects increase

This command changes the current directory to the RV_W2024 folder (which you must create). The RV_W2024 folder is arbitrary, can name it anything, but it is beneficial to keep all code for a project or a course in one folder. The pwd is a Unix command for Print Working Directory, similar to MS-DOS dir command

```
cd('/MATLAB Drive/RV_W2024')
pwd
```

```
ans =
'/MATLAB Drive/RV_W2024'
```

1. (10 points)

a. Read in the image provided on Canvas titled: ELB_1.JPG, using the imread function. Note, MATLAB is case sensitive. Then use the imshow function to display the image

```
img = imread('ELB_1.JPG');
imshow(img);
```

b. Use the "rot90" function to rotate the image so that the sign is displayed vertically, use imshow

```
%Rotated image in 90 and displayed vertically
rotated_image = rot90(img,-1);
imshow(rotated_image);
```

c. Histograms of the pixel intensity values give useful information about the image. Separate the red, green, and blue components

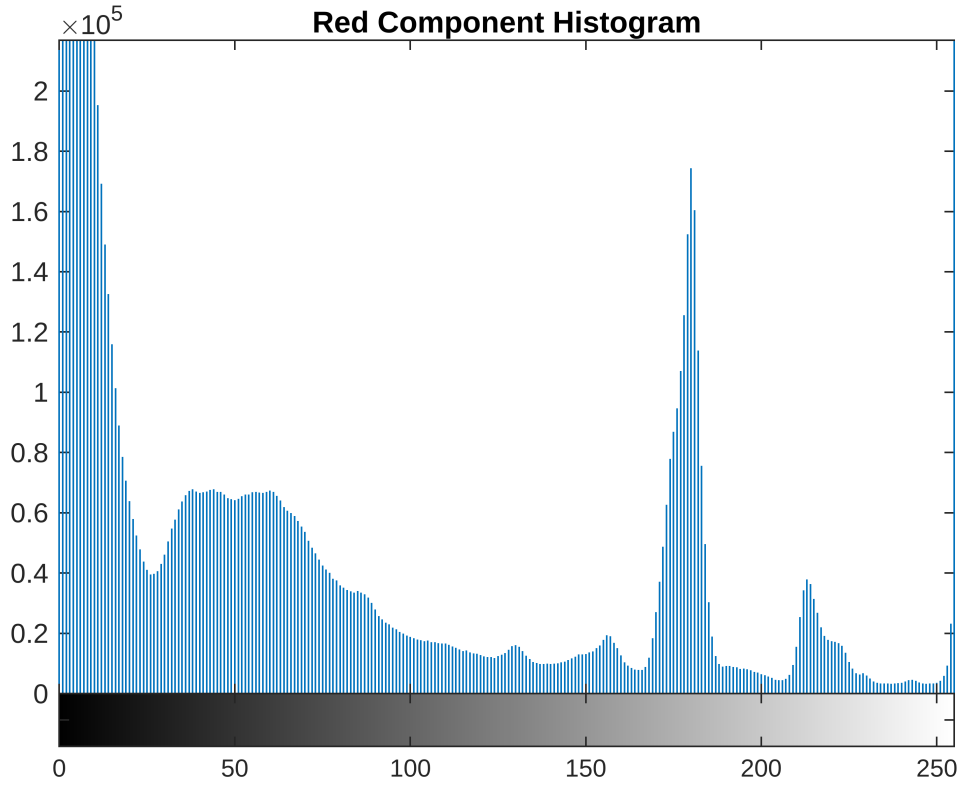
and plot theire histograms. MATLAB stores truecolor images in a separate "plane" for each color, for exampele if using an image named TestImage, then TestImage(:,:,1) hold the red component, (TestImage(:,:,2) holds the green component, and TestImage(:,:,3) holds the blue component. The colons in TestImage(:,:,1) is MATLAB shorthand notation for "all rows, all column, dimension 1", Assign the individual color components to variables and plot the histograms using imhist.

```
% Separated the red, green, and blue components
red_component = img(:,:,1);
green_component = img(:,:,2);
blue_component = img(:,:,3);
```

Plot the histograms of each color

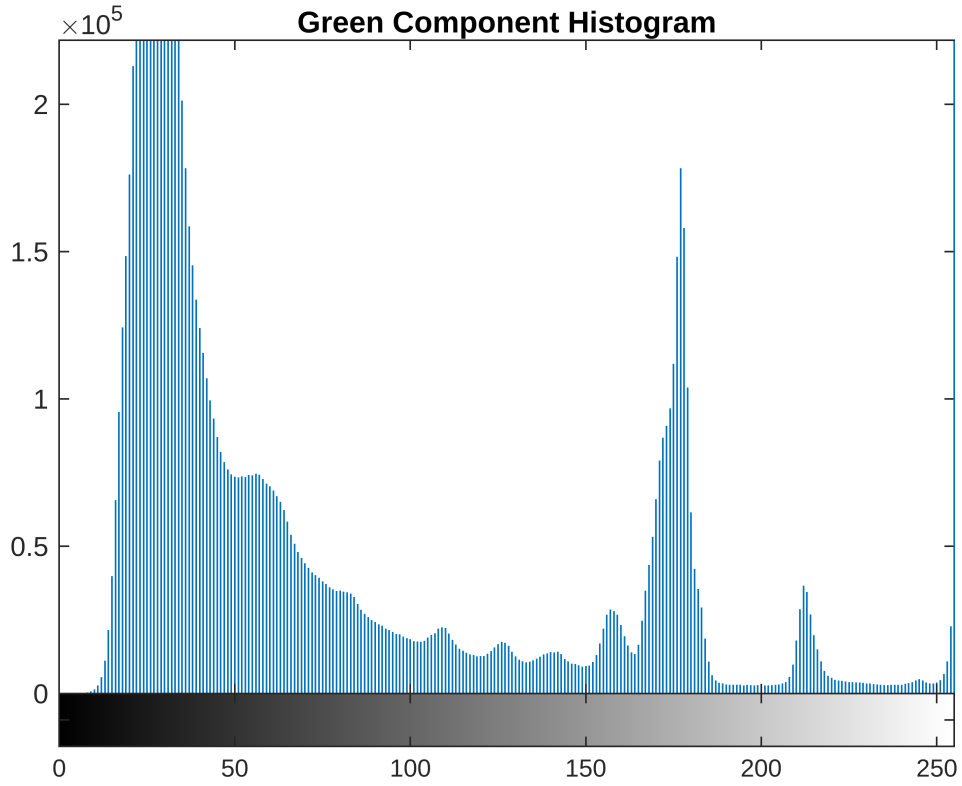
Histogram of red color:

```
% Ploted histogram of the red component
figure;
imhist(red_component);
title('Red Component Histogram');
```



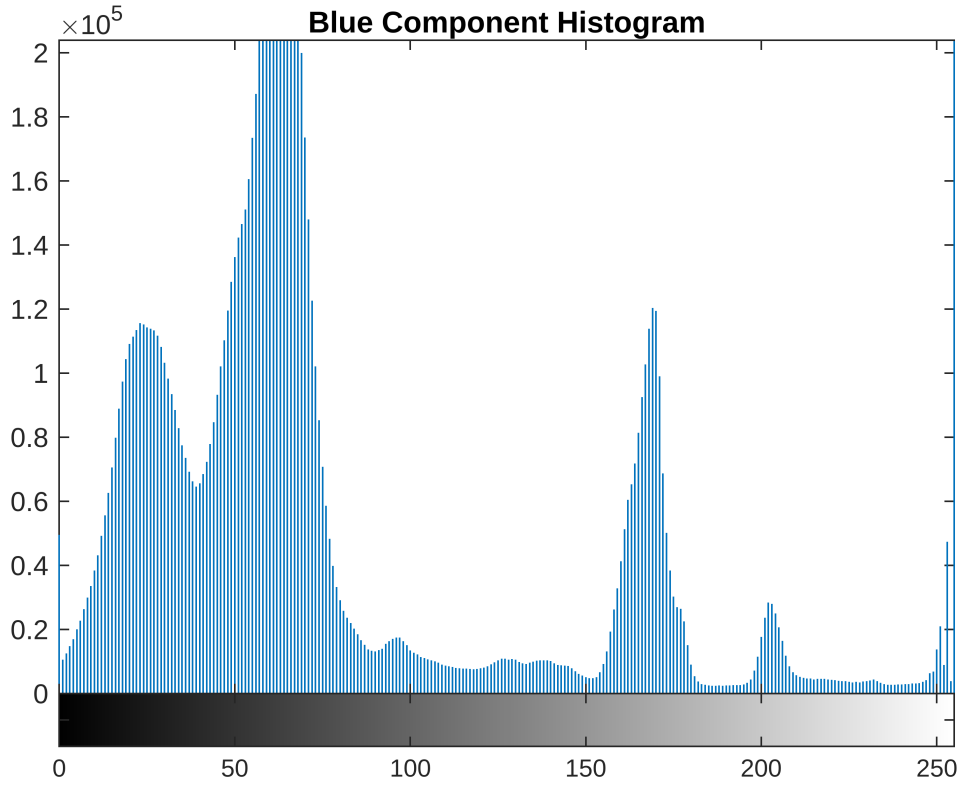
Histogram of green color

```
% Ploted histogram of the green component
figure;
imhist(green_component);
title('Green Component Histogram');
```



Histogram of blue color

```
% Ploted histogram of the blue component
figure;
imhist(blue_component);
title('Blue Component Histogram');
```



2. (10 points)

Use of grayscale images is common in computer vision, as some algorithms are more effective in grayscale format

or at least development of algorithms is simpler when avoiding complications that the three color planes add to algorithms.

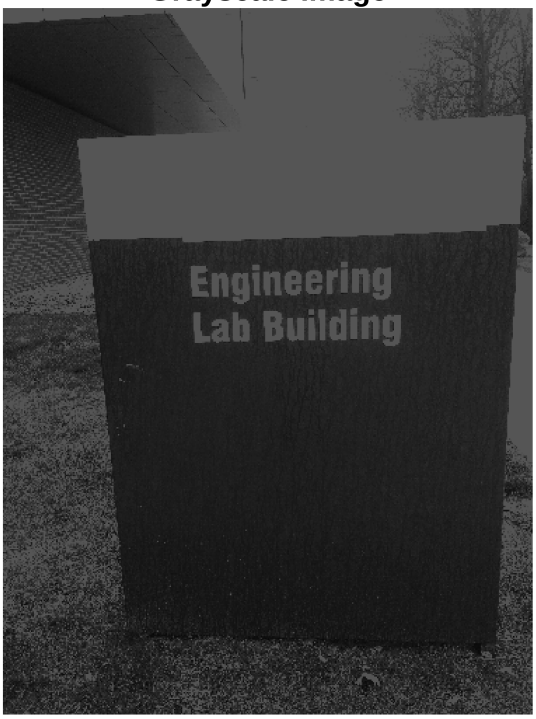
a. Convert the image to grayscale by adding all components together and dividing by 3. This is a simple, basic approach

display the image formed using imshow and comment on the result

```
%Converted the image to grayscale
gray_img = (rotated_image(:,:,1) + rotated_image(:,:,2) +
rotated_image(:,:,3)) / 3;

%Displaying the grayscale image
imshow(gray_img);
title('Grayscale Image');
```

Grayscale Image



This code converts the image to grayscale by averaging the red, green, and blue components for each pixel and then displays the resulting grayscale image. When you convert an image to grayscale using this method, you essentially lose color information and retain only the luminance information. The resulting image may appear less vibrant compared to the original color image. However, it can be beneficial for certain computer vision algorithms that are more effective or simpler to implement in grayscale format, as you mentioned.

b. Convert the original, upright image from uint8 to double, using the im2double function, and use imshow to display

```
% Converted to double
double_image = im2double(rotated_image);
% Displayed the double image
imshow(double_image);
title('Double Image');
```

Double Image

Obtain the individual color planes for red, green, and blue for the image converted to double format and plot

```
% Converted to double
double_image = im2double(rotated_image);

% Separated the red, green, and blue components
red_component = double_image(:,:,1);
green_component = double_image(:,:,2);
blue_component = double_image(:,:,3);

%Plotting Red component histogram
figure;
imshow(red_component);
title('Red Component');
```

Red Component

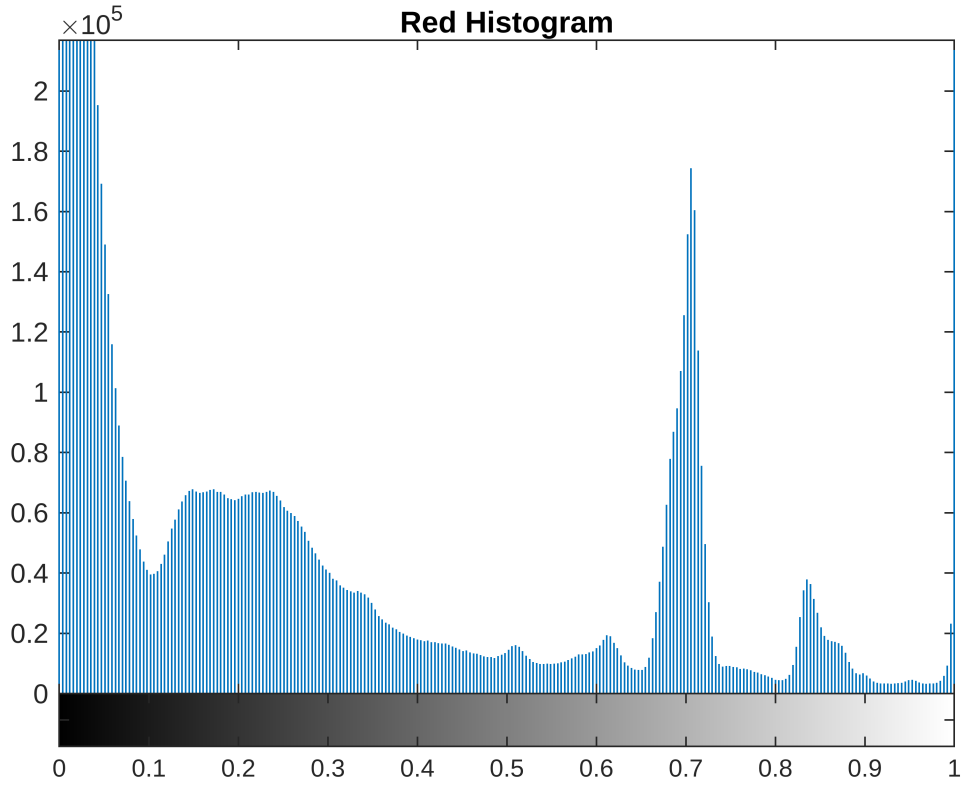

```
figure;
imshow(green_component);
title('Green Component');
```

Green Component

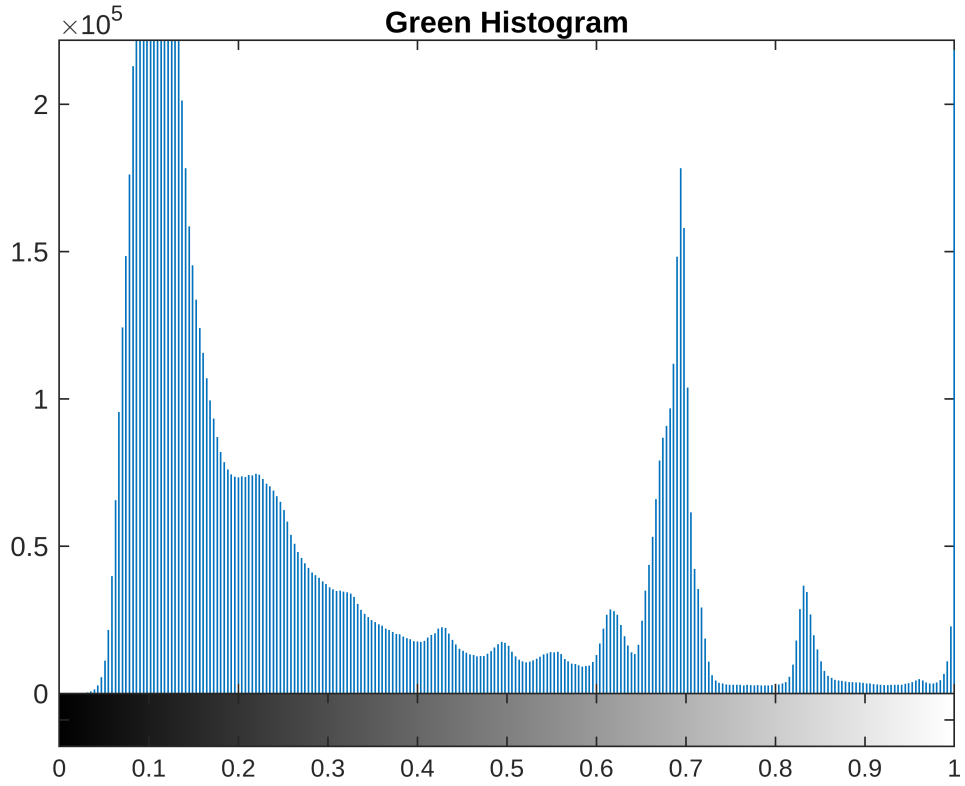

```
figure;
imshow(blue_component);
title('Blue Component');
```

Blue Component

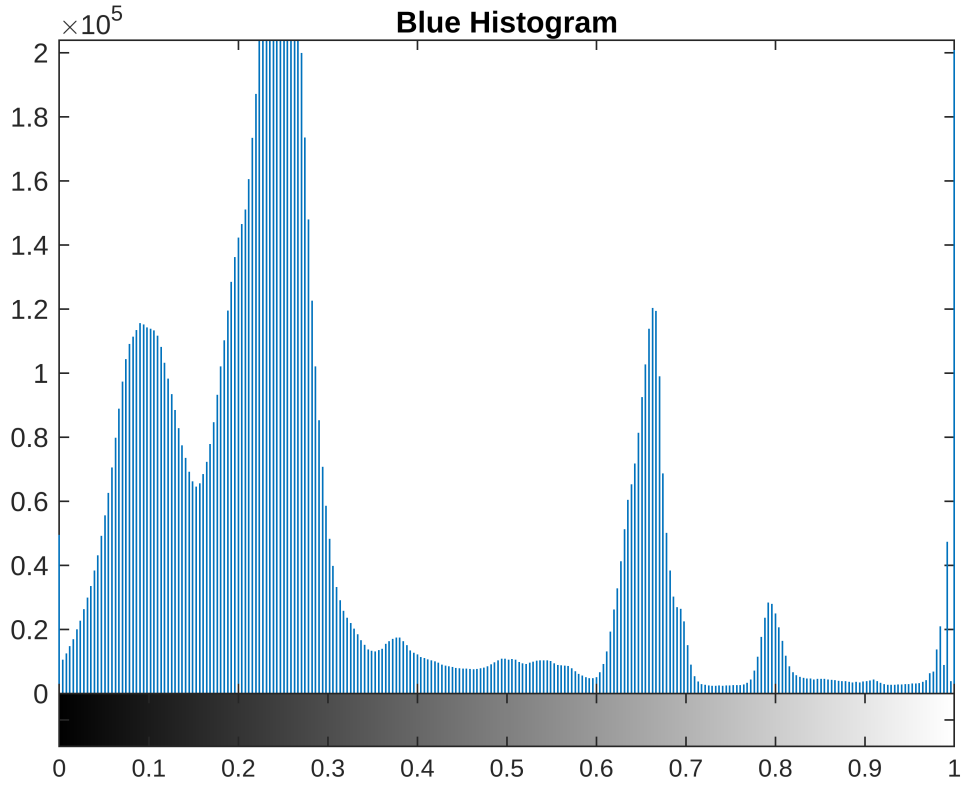

```
figure;
imhist(red_component);
title('Red Histogram');
```




```
figure;
imhist(green_component);
title('Green Histogram');
```




```
figure;
imhist(blue_component);
title('Blue Histogram');
```



Convert the double image to grayscale using equal portions of red, green, and blue components and display

```
% Converted to grayscale using equal portions of red, green, and blue
components
gray_image = (double_image(:,:,1) + double_image(:,:,2) +
double_image(:,:,3)) / 3;

% Displayed the grayscale image
imshow(gray_image);
title('Grayscale Image using Equal Portions of RGB');
```

Grayscale Image using Equal Portions of RGB

Can convert images to grayscale using a different mixture of colors, as human perception is stronger in the green color wavelength. Convert the image, in double format, to grayscale using this ratio: (1 red, 2 green, 1 blue). Compare the result to the previous image result using equal mixtures.

```
% Defined the weights for each color channel
red_weight = 1;
green_weight = 2;
blue_weight = 1;

% Converted to grayscale using the defined weights
gray_image_custom = (double_image(:,:,1) * red_weight + double_image(:,:,2)
* green_weight + double_image(:,:,3) * blue_weight);

% Displayed the grayscale image using custom weights
figure;
imshow(gray_image_custom);
title('Grayscale Image with Custom Color Weights');
```

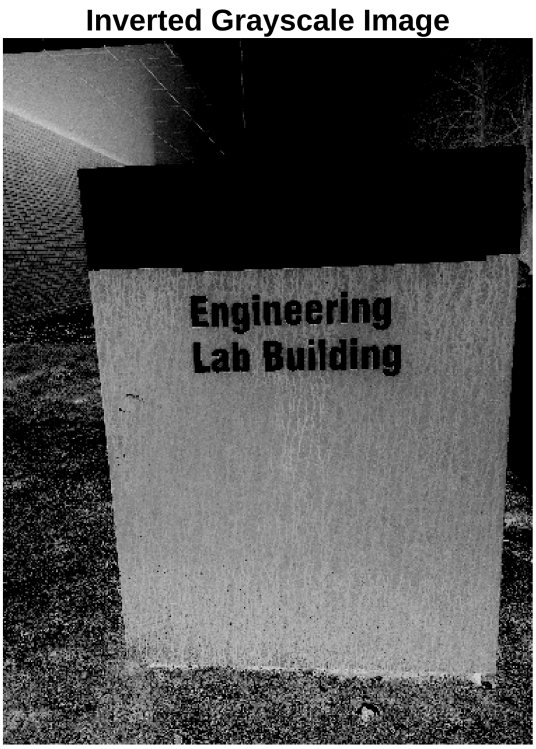
Grayscale Image with Custom Color Weights

3. (5 points)

a. Invert the ELB_1 grayscale image you created using the 1, 2, 1 mixture of colors, in uint8 format and show with imshow

```
% Convert the grayscale image to uint8 format and invert
int_image = 255 - im2uint8(gray_image_custom);

% Display the inverted image
imshow(int_image);
title('Inverted Grayscale Image');
```



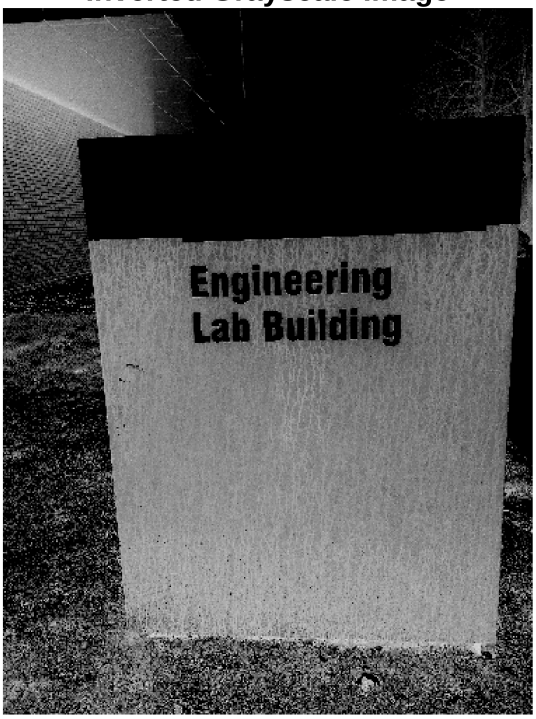
b. Invert the ELB_1 grayscale image you created using the 1,2, 1 mixture of colors in double format and show with imshow

```
double_image = im2double(gray_image_custom);

% Inverted the image using 1, 2, 1 mixture of colors
inverted_image = 1 - double_image;

% Display the inverted image
imshow(inverted_image);
title('Inverted Grayscale Image');
```

Inverted Grayscale Image



4. (5 points)

a. What is the numeric result of this calculation if all values are represented in unit8 format:

(255 + 255 +255)/3

Ans: (255 + 255 +255)/3 = 765/3 = 255 is the calculation in uint8 format

b. What is the numeric result of this calculation if all values are represented in unit8 format:

(253/3 + 253/3 + 253/3)

Ans: (253/3 + 253/3 + 253/3) =253/3 * 3= 253

c. What is the numeric value of this calculation if all values are represented in double format:

(255 +255 +255)/3

Ans: (255 +255 +255)/3=765/3 = 255 is the numeric result of this calculation is in double format

5. (5 points)

In the human eye, cones do not function in the dark, as they are not sensitive enough. What is the response of rods in bright light, knowing that rods are very sensitive to light. Do you believe rods provide much meaningful signal in bright light? Why or why not?

Ans: In bright light, like when the sun is shining or in a well-lit room, the response of rods is not as strong as it is in dim light. Rods are specialized cells in the retina that are super sensitive to light and are mainly responsible for vision in low light conditions, like when it's dark. However, when it's bright, the cones, another type of cells in the retina, take over. Cones are what give us color vision and work best in bright light.

In my opinion, in bright light, rods don't provide much meaningful signal. That's because they're built to work well in the dark, so they're not as active or sensitive when it's bright. It's like trying to use a flashlight during the day – it's not very effective because there's already so much light around. So, even though rods are super important for seeing in the dark, they kinda take a back seat when it's bright out, letting the cones do their thing instead.

6. (10 points)

a. Consider a manufacturing system where components are assembled. Components can be one of : red, green, blue, and white. Describe an approach to reliably determining the color of components using an imaging system, keep in mind that cost will be a factor, so minimizing the number of components while keeping reliable color determination is a facor. (Note, consider image formation, a lengthy answer is not required)

Ans: In my perspective, In a manufacturing system where components are assembled and color determination is important, we can use a simple yet effective approach to reliably determine the color of components while keeping costs in mind.

One approach could be to use a camera or imaging system to capture images of the components as they move along the assembly line. These images can then be processed using software to analyze the color of each component.

To minimize costs and complexity, we can use a limited set of colors for the components, such as red, green, blue, and white. By restricting the number of colors, we simplify the color determination process and reduce the complexity of the imaging system.

In the imaging system, we can use basic color detection algorithms to identify the predominant color of each component in the image. These algorithms can analyze the distribution of colors within the component and determine which color category it belongs to (red, green, blue, or white).

By using a limited set of colors and simple color detection algorithms, we can reliably determine the color of components while keeping costs low. This approach allows for efficient assembly line operations without sacrificing accuracy in color determination.

b. Consider that a working system exists, but due to changes in the assembly area the amout of light present can change, either the illumination increases or decreases. What impact may this have on a previously functioning system and how may the impact of illumination changes be limited on the decision of which color a component is?

Ans: Changes in illumination levels can affect color determination systems in manufacturing environments by causing shifts in color appearance and potentially leading to misclassification of colors. To mitigate these effects:

1. Maintain consistent lighting conditions.
2. Regularly calibrate the imaging system.
3. Implement white balance correction techniques.
4. Use adaptive algorithms that adjust parameters based on illumination levels.

7. (5 points)

a. What are the wavelengths of visible light?

Ans: Visible light consists of electromagnetic waves with wavelengths ranging from approximately 380 nanometers (nm) to 750 nanometers (nm). This range of wavelengths corresponds to the colors of the rainbow, with shorter wavelengths associated with violet light and longer wavelengths associated with red light. Here's a breakdown of the approximate wavelengths for each color:

- Violet: 380 nm to 450 nm
- Blue: 450 nm to 495 nm
- Green: 495 nm to 570 nm
- Yellow: 570 nm to 590 nm
- Orange: 590 nm to 620 nm
- Red: 620 nm to 750 nm

These wavelengths represent the spectrum of visible light that the human eye can perceive.

b. What color has the longest wavelength? What color has the shortest wavelength?

Ans: The color with the longest wavelength in the visible spectrum is red. Red light has a wavelength ranging from approximately 620 nanometers (nm) to 750 nanometers (nm). On the other hand, the color with the shortest wavelength in the visible spectrum is violet. Violet light has a wavelength ranging from approximately 380 nanometers (nm) to 450 nanometers (nm).