**BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

**A picture containing text, gear, metalware

Description automatically generatedBUET**

**Department of Electrical and Electronic Engineering**

**Course No:** HUM-272

**Course Title :** Developing English Skills Laboratory **Section:** A1

Reading and Writing Skills Assignment

**Date of Submission:** 17/02/2022

**Name:** Vivek Chowdhury

**Student ID:** 1906031

**Level:** 2 **Term:** 1

**Department:** EEE

**Objective:**

The project’s objective is to process an image using several image processing techniques.

**Features:**

Our image processing toolbox contains:

1. Grayscale Conversion.
2. Binary Conversion.
3. Filtering (Mean, Median, Max, Min).
4. Edge Detection (Sobel, Prewitt).
5. Image Enhancement (Histogram Equalization, Log Transformation, Power Law Transformation).
6. Segmentation. (Otsu, Adaptive).

**Possible Application:**

There are many applications of our project. Applying edge detection makes it easier for computer vision systems(and human physicians) to detect anomalies in medical images. Grayscale representations are often used for extracting descriptors instead of operating on color images directly, as grayscale simplifies the algorithm and reduces computational requirements. The median filter is used to remove noise from an image. Max-Min filter blurs the image by replacing each pixel with the difference of the highest pixel and the lowest pixel within the specified window size. A mean filter is used for smooth images. The segmentation method is very helpful for medical imaging. Image enhancement improves the clarity of images of human viewing. So, our project can be implemented widely in the medical sector for better representation of the human organs.

**Grayscale Conversion:**

A grayscale (or graylevel) image is simply one in which the only colors are shades of gray. The reason for differentiating such images from any other sort of color image is that less information needs to be provided for each pixel. In fact a `gray' color is one in which the red, green and blue components all have equal intensity in RGB space, and so it is only necessary to specify a single intensity value for each pixel, as opposed to the three intensities needed to specify each pixel in a full color image.

Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white. If the levels are evenly spaced then the difference between successive graylevels is significantly better than the graylevel resolving power of the human eye.

Grayscale images are very common, in part because much of today's display and image capture hardware can only support 8-bit images. In addition, grayscale images are entirely sufficient for many tasks and so there is no need to use more complicated and harder-to-process color images.

**Necessary Function:**

**Syntax**

[I = rgb2gray(RGB)](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#d123e1234734)

[newmap = rgb2gray(map)](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#d123e1234760)

**Description**

[I](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-I) = rgb2gray([RGB](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-RGB)) converts the true color image RGB to the grayscale image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. If you have Parallel Computing Toolbox™ installed, rgb2gray can perform this conversion on a GPU.

[newmap](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-newmap) = rgb2gray([map](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-map)) returns a grayscale colormap equivalent to map.

**Binary Conversion:**

A **binary image** is one that consists of pixels that can have one of exactly two colors, usually black and white. Binary images are also called *bi-level* or *two-level*, Pixelart made of two colours is often referred to as *1-Bit* or *1bit*. This means that each pixel is stored as a single bit—i.e., a 0 or 1. The names *black-and-white*, *B&W*, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images. In Photoshop parlance, a binary image is the same as an image in "Bitmap" mode.

Binary images often arise in digital image processing as masks or thresholding and dithering. Some input/output devices, such as laser printers, fax machines, and bilevel computer displays, can only handle bilevel images.

A binary image can be stored in memory as a bitmap, a packed array of bits. A 640×480 image requires 37.5 KiB of storage. Because of the small size of the image files, fax machine and document management solutions usually use this format. Most binary images also compress well with simple run-length compression schemes.

Binary images can be interpreted as subsets of the two-dimensional integer lattice *Z*2; the field of morphological image processing was largely inspired by this view.

**Necessary Function:**

**Syntax**

[BW = im2bw(I,level)](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115587)

[BW = im2bw(X,cmap,level)](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115621)

[BW = im2bw(RGB,level)](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115639)

**Description**

[BW](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115883) = im2bw([I](https://www.mathworks.com/help/images/ref/im2bw.html" \l "d123e115685),[level](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115840)) converts the grayscale image I to binary image BW, by replacing all pixels in the input image with luminance greater than level with the value 1 (white) and replacing all other pixels with the value 0 (black).

This range is relative to the signal levels possible for the image's class. Therefore, a level value of 0.5 corresponds to an intensity value halfway between the minimum and maximum value of the class.

[BW](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115883) = im2bw([X](https://www.mathworks.com/help/images/ref/im2bw.html" \l "d123e115724),[cmap](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115763),[level](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115840)) converts the indexed image X with colormap cmap to a binary image.

[BW](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115883) = im2bw([RGB](https://www.mathworks.com/help/images/ref/im2bw.html" \l "d123e115801),[level](https://www.mathworks.com/help/images/ref/im2bw.html#d123e115840)) converts the truecolor image RGB to a binary image.

**\*newer version of matlab recommends imbinarize( ) function.**

**Filtering:**

**Mean-Filter:**

In mathematics, functions are classified into two groups, linear and non-linear. A function f is said to be linear if f(x + y) = f(x) + f(y). Otherwise, f is non-linear. A linear filter is an extension of the linear function. An excellent example of a linear filter is the mean filter. The coefficients of mean filter F (Table 4.1) are 1’s. To avoid scaling the pixel intensity after filtering, the whole image is then divided by the number of pixels in the filter; in the case of a 3-by-3 subimage we divide it by 9.

Calendar

Description automatically generated

Calendar

Description automatically generated with low confidence

**Advantages of the mean filter**

• Removes noise.

• Enhances the overall quality of the image, i.e. mean filter brightens an image. **Disadvantages of the mean filter**

• In the process of smoothing, the edges get blurred.

**Necessary Algorithm:**

To apply the mean filter, we need a mask of a 3x3 matrix with an equal weight (1/9). We did the convolution of the image through the mask using a for loop to loop through the 9 pixels of the filter and take their mean. The mean value is the pixel value of the pixel at the center pixel location.

**Necessary Function:**

**Syntax**

[B = imfilter(A,h)](https://www.mathworks.com/help/images/ref/imfilter.html#d123e140289)

[B = imfilter(A,h,options,...)](https://www.mathworks.com/help/images/ref/imfilter.html#d123e140312)

**Description**

[B](https://www.mathworks.com/help/images/ref/imfilter.html#btsmcj2-1-B) = imfilter([A](https://www.mathworks.com/help/images/ref/imfilter.html" \l "btsmcj2-1-A),[h](https://www.mathworks.com/help/images/ref/imfilter.html#btsmcj2-1-h)) filters the multidimensional array A with the multidimensional filter h and returns the result in B.

[B](https://www.mathworks.com/help/images/ref/imfilter.html#btsmcj2-1-B) = imfilter([A](https://www.mathworks.com/help/images/ref/imfilter.html" \l "btsmcj2-1-A),[h](https://www.mathworks.com/help/images/ref/imfilter.html#btsmcj2-1-h),[options](https://www.mathworks.com/help/images/ref/imfilter.html#btsmcj2-1-options),...) performs multidimensional filtering according to one or more specified options.

**Median Filter:**

Median filter is one of the most popular non-linear filters. A sliding window is chosen and is placed on the image at the pixel position (i, j). All pixel values under the filter are collected. The median of these values is computed and is assigned to (i, j) in the filtered image. For example, consider a 3-by-3 sub-image with values 5, 7, 6, 10, 13, 15, 14, 19, 23. To compute the median, the values are arranged in ascending order, so the new list is: 5, 6, 7, 10, 13, 14, 15, 19, and 23. Median is a value that divides the list into two equal halves; in this case it is 13. So the pixel (i, j) will be assigned 13 in the filtered image. The median filter is commonly used in removing salt-and-pepper noise and impulse noise. Salt-and-pepper noise is characterized by black and white spots randomly distributed in an image.

**Necessary Function:**

**Syntax**

[J = medfilt2(I)](https://www.mathworks.com/help/images/ref/medfilt2.html#d123e212871)

[J = medfilt2(I,[m n])](https://www.mathworks.com/help/images/ref/medfilt2.html#d123e212889)

[J = medfilt2(**\_\_\_**,padopt)](https://www.mathworks.com/help/images/ref/medfilt2.html#d123e212906)

**Description**

[J](https://www.mathworks.com/help/images/ref/medfilt2.html#bupa05x-1-B) = medfilt2([I](https://www.mathworks.com/help/images/ref/medfilt2.html#bupa05x-1-A)) performs median filtering of the image I in two dimensions. Each output pixel contains the median value in a 3-by-3 neighborhood around the corresponding pixel in the input image.

[J](https://www.mathworks.com/help/images/ref/medfilt2.html#bupa05x-1-B) = medfilt2([I](https://www.mathworks.com/help/images/ref/medfilt2.html#bupa05x-1-A),[[m n]](https://www.mathworks.com/help/images/ref/medfilt2.html#bupa05x-1-mn)) performs median filtering, where each output pixel contains the median value in the m-by-n neighborhood around the corresponding pixel in the input image.

[J](https://www.mathworks.com/help/images/ref/medfilt2.html#bupa05x-1-B) = medfilt2(**\_\_\_**,[padopt](https://www.mathworks.com/help/images/ref/medfilt2.html" \l "bupa05x-1-padopt)) controls how medfilt2 pads the image boundaries.

**Syntax**

[C = cat(dim,A,B)](https://www.mathworks.com/help/matlab/ref/double.cat.html#f70-456595)

[C = cat(dim,A1,A2,…,An)](https://www.mathworks.com/help/matlab/ref/double.cat.html#f70-456598)

**Description**

C = cat([dim](https://www.mathworks.com/help/matlab/ref/double.cat.html" \l "d123e149443),[A](https://www.mathworks.com/help/matlab/ref/double.cat.html#mw_8e40eb41-9d14-41da-9357-a1cbac7b4211),[B](https://www.mathworks.com/help/matlab/ref/double.cat.html#mw_cd6decd9-9f5d-491f-b579-a64d3bfef5e7)) concatenates B to the end of A along dimension dim when A and B have compatible sizes (the lengths of the dimensions match except for the operating dimension dim).

C = cat([dim](https://www.mathworks.com/help/matlab/ref/double.cat.html#d123e149443),[A1,A2,…,An](https://www.mathworks.com/help/matlab/ref/double.cat.html#mw_1751ec86-3e52-4609-8282-397a6ac6bb25)) concatenates A1, A2, … , An along dimension dim.

You can use the square bracket operator [] to concatenate. For example, [A,B] or [A B] concatenates arrays A and B horizontally, and [A; B] concatenates them vertically.

**Max-Min Filter:**

The Max filter enhances the bright points in an image. In this filter the maximum value in the sub-image replaces the value at (i, j). The min filter is used to enhance the darkest points in an image. In this filter, the minimum value of the sub-image replaces the value at (i, j).

**Necessary Algorithm:**

Firstly, we will do a zero padding to the image. Then a 3×3 mask is made. We did the convolution of the image through the mask using a for loop to loop through the 9 pixels of the filter and take their max or min value. The max or min value is the pixel value of the pixel at the center pixel location.

**Necessary function:**

Here, we used some basic functions like size( ), min( ), max( ).

**Edge Detection:**

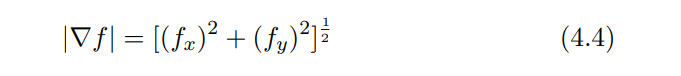
**First Derivative Filters** :

An image is not a continuous function and hence derivatives are calculated using discrete approximations. Let us look at the definition of gradient of a continuous function and then extend it to discrete cases. If f(x, y) is a continuous function, then the gradient of f as a vector is given by:

Background pattern

Description automatically generated with low confidence

where fx = ∂f/∂x is known as the partial derivative of f with respect to x, it represents change of f along the horizontal direction and fy = ∂f/∂y is known as the partial derivative of f with respect to y, it represents change of f along the vertical direction. The magnitude of the gradient is a scalar quantity and is given by:



For computational purposes, we consider the simplified version of the gradient is given by Equation 4.5 and angle is given by Equation 4.6.

Text

Description automatically generated with low confidence

One of the most popular first derivative filters is the Sobel filter. The Sobel filter or mask is used to find horizontal and vertical edges as given in Table 4.3.

Graphical user interface, text

Description automatically generated

To understand how filtering is done, let us consider a sub-image of size 3-by-3 given in Table 4.4 and multiply the sub-image with horizontal and vertical Sobel masks. The corresponding output is given in Table 4.5.

Text

Description automatically generated

Since fx is the partial derivative of f in x direction which is change of f along horizontal direction, the partial can be obtained by taking the difference between the third row and the first row in the horizontal mask, so fx = (f7 + 2f8 + f9) − (−f1 − 2f2 − f3). Likewise, fy is the partial of f in y direction which is change of f in vertical direction, the partial can be obtained by taking the difference between the third column and the first column in the vertical mask, so fy = (f3 + 2f6 + f9) − (−f1 + 2f4 − f7). Simplying fx and fy the discrete gradient at f5 is given by the Equation 4.7. |f5| = |f7+2f8+f9+f1+2f2+f3|+|f3+2f6+f9+f1−2f4+f7| (4.7)

Graphical user interface, text, application

Description automatically generated

The important features of the Sobel filter are:

• The sum of the coefficients in the mask image is 0. This means that the pixels with constant grayscale are not affected by the derivative filter.

• The side effect of derivative filters is creation of additional noise. Hence, coefficients of +2 and −2 are used in the mask image to produce smoothing.

Another popular first derivative filter is Prewitt. The masks for the Prewitt filter are given in Table 4.6.

Graphical user interface, text

Description automatically generated with medium confidence

As in the case of Sobel, the sum of the coefficients in Prewitt is also 0. Hence this filter does not affect pixels with constant grayscale. However, the filter does not reduce noise as can be seen in the values of the coefficients. Slightly modified Sobel and Prewitt filters can be used to detect one or more types of edges. Sobel and Prewitt filters to detect diagonal edges are given in Tables 4.7 and 4.8.

Text

Description automatically generated with low confidence

Text

Description automatically generated with medium confidence

**Necessary Algorithm:**

Firstly, we will convolute the image with the sobel/prewitt x operator & will find Gx. We will do the same with the sobel/prewitt y operator & find Gy. Now, we will use the gradient approximation formula sqrt(Gx^2+Gy^2) & compare it with a threshold value T. If it is greater than T, it is an edge otherwise it will not be an edge. We have to do this for each pixel.

**Necessary Functions:**

**Syntax**

[I = im2gray(RGB)](https://www.mathworks.com/help/matlab/ref/im2gray.html#d123e687911)

**Description**

[I](https://www.mathworks.com/help/matlab/ref/im2gray.html#mw_38e7de21-4196-4eeb-b3c3-6aae03b9cb2f) = im2gray([RGB](https://www.mathworks.com/help/matlab/ref/im2gray.html#mw_740e9357-1a33-45fb-a068-e8f7ff38fd3d)) converts the specified truecolor image RGB to a grayscale intensity image I. The im2gray function accepts grayscale images as inputs and returns them unmodified.

The im2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

**Syntax**

[BW = edge(I)](https://www.mathworks.com/help/images/ref/edge.html#d123e91739)

[BW = edge(I,method)](https://www.mathworks.com/help/images/ref/edge.html#d123e91760)

[BW = edge(I,method,threshold)](https://www.mathworks.com/help/images/ref/edge.html#d123e91782)

[BW = edge(I,method,threshold,direction)](https://www.mathworks.com/help/images/ref/edge.html#d123e91799)

[BW = edge(**\_\_\_**,'nothinning')](https://www.mathworks.com/help/images/ref/edge.html#d123e91824)

[BW = edge(I,method,threshold,sigma)](https://www.mathworks.com/help/images/ref/edge.html#d123e91844)

[BW = edge(I,method,threshold,h)](https://www.mathworks.com/help/images/ref/edge.html#d123e91869)

[[BW,threshOut] = edge(**\_\_\_**)](https://www.mathworks.com/help/images/ref/edge.html#d123e91896)

[[BW,threshOut,Gv,Gh] = edge(**\_\_\_**)](https://www.mathworks.com/help/images/ref/edge.html#d123e91910)

**Description**

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge([I](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-I)) returns a binary image BW containing 1s where the function finds edges in the grayscale or binary image I and 0s elsewhere. By default, edge uses the Sobel edge detection method.

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge([I](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-I),[method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005)) detects edges in image I using the edge-detection algorithm specified by method.

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge([I](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-I),[method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005),[threshold](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-threshold)) returns all edges that are stronger than threshold.

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge([I](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-I),[method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005),[threshold](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-threshold),[direction](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-direction)) specifies the orientation of edges to detect. The Sobel and Prewitt methods can detect edges in the vertical direction, horizontal direction, or both. The Roberts method can detect edges at angles of 45° from horizontal, 135° from horizontal, or both. This syntax is valid only when method is 'Sobel', 'Prewitt', or 'Roberts'.

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge(**\_\_\_**,'nothinning') skips the edge-thinning stage, which can improve performance. This syntax is valid only when [method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005) is 'Sobel', 'Prewitt', or 'Roberts'.

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge([I](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-I),[method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005),[threshold](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-threshold),[sigma](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-sigma)) specifies sigma, the standard deviation of the filter. This syntax is valid only when method is 'log' or 'Canny'.

[BW](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-BW) = edge([I](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-I),[method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005),[threshold](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-threshold),[h](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-h)) detects edges using the 'zerocross' method with a filter, h, that you specify. This syntax is valid only when method is 'zerocross'.

[[BW](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-BW),[threshOut](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-threshOut)] = edge(**\_\_\_**) also returns the threshold value.

[[BW](https://www.mathworks.com/help/images/ref/edge.html" \l "buo5g3w-1-BW),[threshOut](https://www.mathworks.com/help/images/ref/edge.html#buo5g3w-1-threshOut),[Gv](https://www.mathworks.com/help/images/ref/edge.html#d123e92432),[Gh](https://www.mathworks.com/help/images/ref/edge.html#d123e92463)] = edge(**\_\_\_**) also returns the directional gradient magnitudes. For the Sobel and Prewitt methods, Gv and Gh correspond to the vertical and horizontal gradients. For the Roberts methods, Gv and Gh correspond to the gradient at angles of 45° and 135° from horizontal, respectively. This syntax is valid only when [method](https://www.mathworks.com/help/images/ref/edge.html#d123e92005) is 'Sobel', 'Prewitt', or 'Roberts'.

**Image Enhancement:**

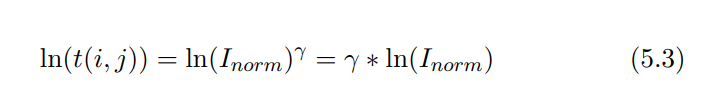
**Power Law Transformation:**

Power law transformation, also known as gamma-correction, is used to enhance the quality of the image. The power transformation at (i, j) is given by

**A picture containing text

Description automatically generated**

here k and γ are positive constants and I is the intensity value of the pixel in the input image at (i, j). In most cases k = 1. If γ = 1 (Figure 5.4), then the mapping is linear and the output image is the same as the input image. When γ < 1, a narrow range of dark or low intensity pixel values in the input image get mapped to a wide range of intensities in the output image, while a wide range of bright or high intensity pixel values in the input image get mapped to a narrow range of high intensities in the output image. The effect from values of γ > 1 is opposite that of values γ < 1. Considering that the intensity range is between [0, 1], Figure 5.4 illustrates the effect of different values of γ for k = 1. The human brain uses gamma-correction to process an image, hence gamma-correction is a built-in feature in devices that display, acquire, or publish images. Computer monitors and television screens have built-in gamma-correction so that the best image contrast is displayed in all the images. In an 8-bit image, the intensity values range from [0, 255]. If the transformation is applied according to Equation 5.2, and for γ > 1 the output pixel intensities will be out of bounds. To avoid this scenario, the pixel intensities are normalized, I(i, j) /max(I) = Inorm. For k = 1, replacing I(i, j) with Inorm and then applying natural log, ln on both sides of Equation 5.2 will result in



now basing both sides by e will give us

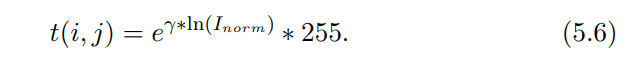


since e^ln(x) = x, the left side in the above equation will simplify to

**A picture containing text

Description automatically generated**

to have the output in the range [0, 255] we multiply the right side of the above equation by 255 which results in



Diagram

Description automatically generated

**Necessary Functions:**

Here, we used some basic functions like size( ),max( ),uint8( ) (8-bit unsigned integer arrays).

**Log Transformation:**

Log transformation is used to enhance pixel intensities that are otherwise missed due to a wide range of intensity values or lost at the expense of high intensity values. If the intensities in the image range from [0, L − 1] then the log transformation at (i, j) is given by



where k = (L – 1)/ log(1 + |Imax|) and Imax is maximum magnitude value and I(i, j) is the intensity value of the pixel in the input image at (i, j). If both I(i, j) and Imax are equal to L − 1 then t(i, j) = L − 1. When I(i, j) = 0, since log(1) = 0 will give t(i, j) = 0. While the end points of the range get mapped to themselves, other input values will be transformed by the above equation. The log can be of any base; however, common log (log base 10) or natural log (log base e) are widely used. The inverse of the above log transformation when the base is e is given by

t^-1 (x) = e^(x/k) − 1 which does the opposite of the log transformation. Similar to the power law transformation with γ < 1, the log transformation also maps a small range of dark or low intensity pixel values in the input image to a wide range of intensities in the output image, while a wide range of bright or high intensity pixel values in the input image get mapped to narrow range of high intensities in the output image. Considering the intensity range is between [0, 1], Figure 5.6 illustrates the log and inverse log transformations.

Diagram

Description automatically generated

**Necessary Functions:**

Here, we also used some basic functions & uint8( ).

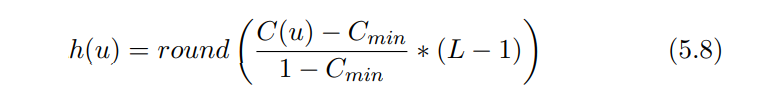
**Histogram Equalization:**

The histogram of an image is a discrete function, its input is the gray level value and the output is the number of pixels with that gray level value and can be given as h(xn) = yn. In a grayscale image, the intensities of the image take values between [0, L−1]. Low gray level values in the image (the left side of the histogram) correspond to dark regions and high gray level values in the image (the right side of the histogram) correspond to bright regions. In a low contrast image, the histogram is narrow, whereas in a high contrast image, the histogram is spread out. In histogram equalization, the goal is to improve the contrast of an image by rescaling the histogram so that the histogram of the new image is spread out and the pixel intensities range over all possible gray level values. The rescaling of the histogram will be performed by using a transformation. To ensure that for every gray level value in the input image there is a corresponding output, a one-to-one transformation is required; that is every input has a unique output. This means the transformation should be an increasing function. This will ensure that the transformation is invertible. Before histogram equalization transformation is defined, the following should be computed:

Text, letter

Description automatically generated

The histogram equalization transformation can be defined as follows:



where Cmin is the minimum cumulative distribution value in the image. For a grayscale image with range between [0, 255], if C(u) = Cmin then h(u) = 0. If C(u) = 1 then h(u) = 255. The integer value for the output image is obtained by rounding Equation 5.8. Let us consider an example to illustrate the probability, CDF and histogram equalization. Figure 5.8 is an image of size 5 by 5. Let us assume that the gray levels of the image range from [0, 255].

Table

Description automatically generated

The probabilities, CDF as C for each gray level value for Figure 5.8 along with the output of histogram equalization transformation are given in Figure 5.9.

Table

Description automatically generated

**Necessary Functions:**

Create the histogram. For the example image, showing grains of rice, imhist creates a histogram with 64 bins. The imhist function displays the histogram, by default. The histogram shows a peak at around 100, corresponding to the dark gray background in the image.

figure;

imhist(I);

Chart, histogram

Description automatically generated

histeq

Enhance contrast using histogram equalization

**Syntax**

[J = histeq(I)](https://www.mathworks.com/help/images/ref/histeq.html#d123e110363)

[J = histeq(I,n)](https://www.mathworks.com/help/images/ref/histeq.html#d123e110383)

[J = histeq(I,hgram)](https://www.mathworks.com/help/images/ref/histeq.html#d123e110409)

[newmap = histeq(X,map)](https://www.mathworks.com/help/images/ref/histeq.html#d123e110432)

[newmap = histeq(X,map,hgram)](https://www.mathworks.com/help/images/ref/histeq.html#d123e110449)

[[**\_\_\_**,T] = histeq(**\_\_\_**)](https://www.mathworks.com/help/images/ref/histeq.html#d123e110483)

**Description**

[J](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-J) = histeq([I](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-I)) transforms the grayscale image I so that the histogram of the output grayscale image J has 64 bins and is approximately flat.

[J](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-J) = histeq([I](https://www.mathworks.com/help/images/ref/histeq.html" \l "bvhelue-1-I),[n](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-n)) transforms the grayscale image I so that the histogram of the output grayscale image J with n bins is approximately flat. The histogram of J is flatter when n is much smaller than the number of discrete levels in I.

[J](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-J) = histeq([I](https://www.mathworks.com/help/images/ref/histeq.html" \l "bvhelue-1-I),[hgram](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-hgram)) transforms the grayscale image I so that the histogram of the output grayscale image J with length(hgram) bins approximately matches the target histogram hgram.

[newmap](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-newmap) = histeq([X](https://www.mathworks.com/help/images/ref/histeq.html" \l "bvhelue-1-X),[map](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-map)) transforms the values in the colormap so that the histogram of the gray component of the indexed image X is approximately flat. The transformed colormap is newmap.

[newmap](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-newmap) = histeq([X](https://www.mathworks.com/help/images/ref/histeq.html" \l "bvhelue-1-X),[map](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-map),[hgram](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-hgram)) transforms the colormap associated with the indexed image X so that the histogram of the gray component of the indexed image (X,newmap) approximately matches the target histogram hgram. The histeq function returns the transformed colormap in newmap. length(hgram) must be the same as size(map,1).

[**\_\_\_**,[T](https://www.mathworks.com/help/images/ref/histeq.html#bvhelue-1-T)] = histeq(**\_\_\_**) also returns the transformation T that maps the gray component of the input grayscale image or colormap to the gray component of the output grayscale image or colormap.

rgb2hsv

Convert RGB colors to HSV

**Syntax**

[HSV = rgb2hsv(RGB)](https://www.mathworks.com/help/matlab/ref/rgb2hsv.html#d123e1235135)

[hsvmap = rgb2hsv(rgbmap)](https://www.mathworks.com/help/matlab/ref/rgb2hsv.html#d123e1235150)

**Description**

[HSV](https://www.mathworks.com/help/matlab/ref/rgb2hsv.html#d123e1235308) = rgb2hsv([RGB](https://www.mathworks.com/help/matlab/ref/rgb2hsv.html#d123e1235236)) converts the red, green, and blue values of an RGB image to hue, saturation, and value (HSV) values of an HSV image.

[hsvmap](https://www.mathworks.com/help/matlab/ref/rgb2hsv.html#mw_4d5ac071-bd48-4a3c-bcd4-ef9a91b500d9) = rgb2hsv([rgbmap](https://www.mathworks.com/help/matlab/ref/rgb2hsv.html" \l "mw_8c4f27fd-0016-40db-a730-3f7cd09027d8)) converts an RGB colormap to an HSV colormap.

hsv2rgb

Convert HSV colors to RGB

**Syntax**

[RGB = hsv2rgb(HSV)](https://www.mathworks.com/help/matlab/ref/hsv2rgb.html#d123e617834)

[rgbmap = hsv2rgb(hsvmap)](https://www.mathworks.com/help/matlab/ref/hsv2rgb.html#d123e617849)

**Description**

[RGB](https://www.mathworks.com/help/matlab/ref/hsv2rgb.html#d123e618051) = hsv2rgb([HSV](https://www.mathworks.com/help/matlab/ref/hsv2rgb.html#d123e617939)) converts the hue, saturation, and value (HSV) values of an HSV image to red, green, and blue values of an RGB image.

[rgbmap](https://www.mathworks.com/help/matlab/ref/hsv2rgb.html#mw_12f35c07-754f-495e-98b3-4c86fad546da) = hsv2rgb([hsvmap](https://www.mathworks.com/help/matlab/ref/hsv2rgb.html" \l "mw_a640f759-4824-4291-8c2e-1302edbdc201)) converts an HSV colormap to an RGB colormap.

**Segmentation:**

**Otsu’s Method:**

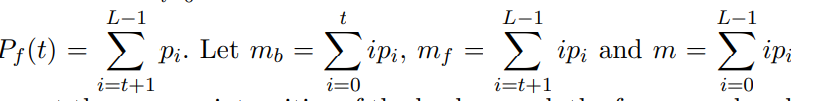
Otsu’s method works best if the histogram of the image is bi-modal but can be applied to other histograms as well. A bi-modal histogram is a type of histogram (similar to Figure 7.1) containing two distinct peaks separated by a valley. One peak is the background and the other foreground. Otsu’s algorithm searches for a threshold value that maximizes the variance between the two groups foreground and background, so that the threshold value can better segment the foreground from the background.

Let L be the number of intensities in the image. For an 8-bit image, L = 2^8 = 256. For a threshold value, t the probabilities, Pi of each intensity is calculated. Then the probability of the background pixels is given by

A close up of a logo

Description automatically generated with low confidence

and the probability of foreground pixels is given by



represent the average intensities of the background, the foreground and the whole image respectively. If vb, vf and v represent the variance of the background, foreground and the whole image respectively. Then the variance within the groups is given by Equation 7.1 and the variance in between the groups is given by Equation 7.2.

Text, letter

Description automatically generated

For different threshold values this process of finding variance within the groups and variance between the groups is repeated. The threshold value that maximizes the variance between the groups or minimizes the variance within the group is considered the Otsu’s threshold. All pixel values with intensities less than the threshold value are assigned a value of zero and all pixel values with intensities greater than the threshold value are assigned a value of one. In the case of a color image, since there are three channels, Red, Green and Blue channel, a different threshold value for each channel is calculated.

**Necessary functions:**

rgb2gray

Convert RGB image or colormap to grayscale

**Syntax**

[I = rgb2gray(RGB)](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#d123e1234734)

[newmap = rgb2gray(map)](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#d123e1234760)

**Description**

[I](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-I) = rgb2gray([RGB](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-RGB)) converts the truecolor image RGB to the grayscale image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance. If you have Parallel Computing Toolbox™ installed, rgb2gray can perform this conversion on a GPU.

[newmap](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-newmap) = rgb2gray([map](https://www.mathworks.com/help/matlab/ref/rgb2gray.html#buiz8mj-1-map)) returns a grayscale colormap equivalent to map.

graythresh

Global image threshold using Otsu's method

**Syntax**

[T = graythresh(I)](https://www.mathworks.com/help/images/ref/graythresh.html#d123e109821)

[[T,EM] = graythresh(I)](https://www.mathworks.com/help/images/ref/graythresh.html#d123e109848)

**Description**

[T](https://www.mathworks.com/help/images/ref/graythresh.html#bvhttkq-level) = graythresh([I](https://www.mathworks.com/help/images/ref/graythresh.html#bvhttkq-I)) computes a global threshold T from grayscale image I, using Otsu's method [[1]](https://www.mathworks.com/help/images/ref/graythresh.html#bviewj6). Otsu's method chooses a threshold that minimizes the intraclass variance of the thresholded black and white pixels. The global threshold T can be used with [imbinarize](https://www.mathworks.com/help/images/ref/imbinarize.html) to convert a grayscale image to a binary image.

[[T](https://www.mathworks.com/help/images/ref/graythresh.html#bvhttkq-level),[EM](https://www.mathworks.com/help/images/ref/graythresh.html#bvhttkq-EM)] = graythresh([I](https://www.mathworks.com/help/images/ref/graythresh.html#bvhttkq-I)) also returns the effectiveness metric, EM.

imbinarize

Binarize 2-D grayscale image or 3-D volume by thresholding

**Syntax**

[BW = imbinarize(I)](https://www.mathworks.com/help/images/ref/imbinarize.html#d123e129119)

[BW = imbinarize(I,method)](https://www.mathworks.com/help/images/ref/imbinarize.html#d123e129150)

[BW = imbinarize(I,T)](https://www.mathworks.com/help/images/ref/imbinarize.html#d123e129175)

[BW = imbinarize(I,'adaptive',Name,Value)](https://www.mathworks.com/help/images/ref/imbinarize.html#d123e129193)

**Description**

[BW](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-BW) = imbinarize([I](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-I)) creates a binary image from 2-D or 3-D grayscale image I by replacing all values above a globally determined threshold with 1s and setting all other values to 0s. By default, imbinarize uses Otsu's method, which chooses the threshold value to minimize the intraclass variance of the thresholded black and white pixels [[1]](https://www.mathworks.com/help/images/ref/imbinarize.html#mw_e4a539f4-6c30-4ea2-bd07-1aad59fc9cea). imbinarize uses a 256-bin image histogram to compute Otsu's threshold. To use a different histogram, see [otsuthresh](https://www.mathworks.com/help/images/ref/otsuthresh.html).

[BW](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-BW) = imbinarize([I](https://www.mathworks.com/help/images/ref/imbinarize.html" \l "bu1w0rc-1-I),[method](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-method)) creates a binary image from image I using the thresholding method specified by method: 'global' or 'adaptive'.

[BW](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-BW) = imbinarize([I](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-I),[T](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-T)) creates a binary image from image I using the threshold value T. T can be a global image threshold, specified as a scalar luminance value, or a locally adaptive threshold, specified as a matrix of luminance values.

[BW](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-BW) = imbinarize([I](https://www.mathworks.com/help/images/ref/imbinarize.html#bu1w0rc-1-I),'adaptive',[Name,Value](https://www.mathworks.com/help/images/ref/imbinarize.html" \l "namevaluepairarguments)) creates a binary image from image I using name-value pairs to control aspects of adaptive thresholding.

**Adaptive Thresholding:**

A global threshold might not provide accurate segmentation. Adaptive thresholding helps solve this problem. In the adaptive thresholding, the image is first divided into small sub-images. Threshold value for each sub-image is computed and is used to segment the sub-image. The threshold value for the sub-image can be computed using mean or median or Gaussian methods. In the case of mean method, the mean of sub-image is used as a threshold, while for median method, the median of the sub-image is used as a threshold. A custom formula can also be used to compute the threshold, for example we can use an average of maximum and minimum pixel values in the sub-image. By appropriate programming, any of the histogram based segmentation methods can be converted into an adaptive thresholding method.

**Necessary Functions:**

We will use imbinarize( ) here. Details are given in the previous section.

**User Interface:**

Graphical user interface

Description automatically generated

**Input & Output Methods:**

User will upload an image via the ‘upload’ option. After uploading, he can choose any technique from the options & he can see the processed image beside the original image. He can also use sequence operation ( applying multiple operations to an image) by selecting it. He may also use the undo & reset option if needed.

Finally, he can also download the processed image using the ‘download’ option.

**An Example:**

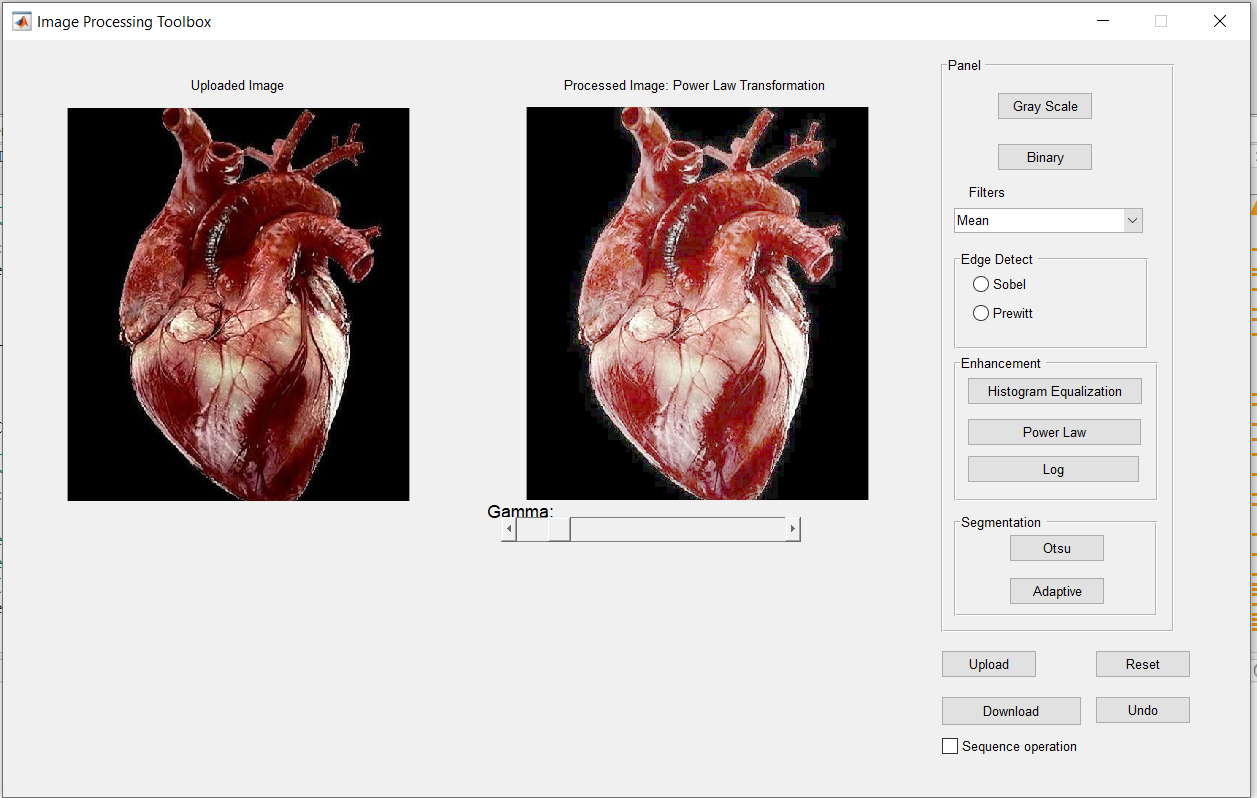


Fig. Image Enhancement of Human Heart (Power Law Method)

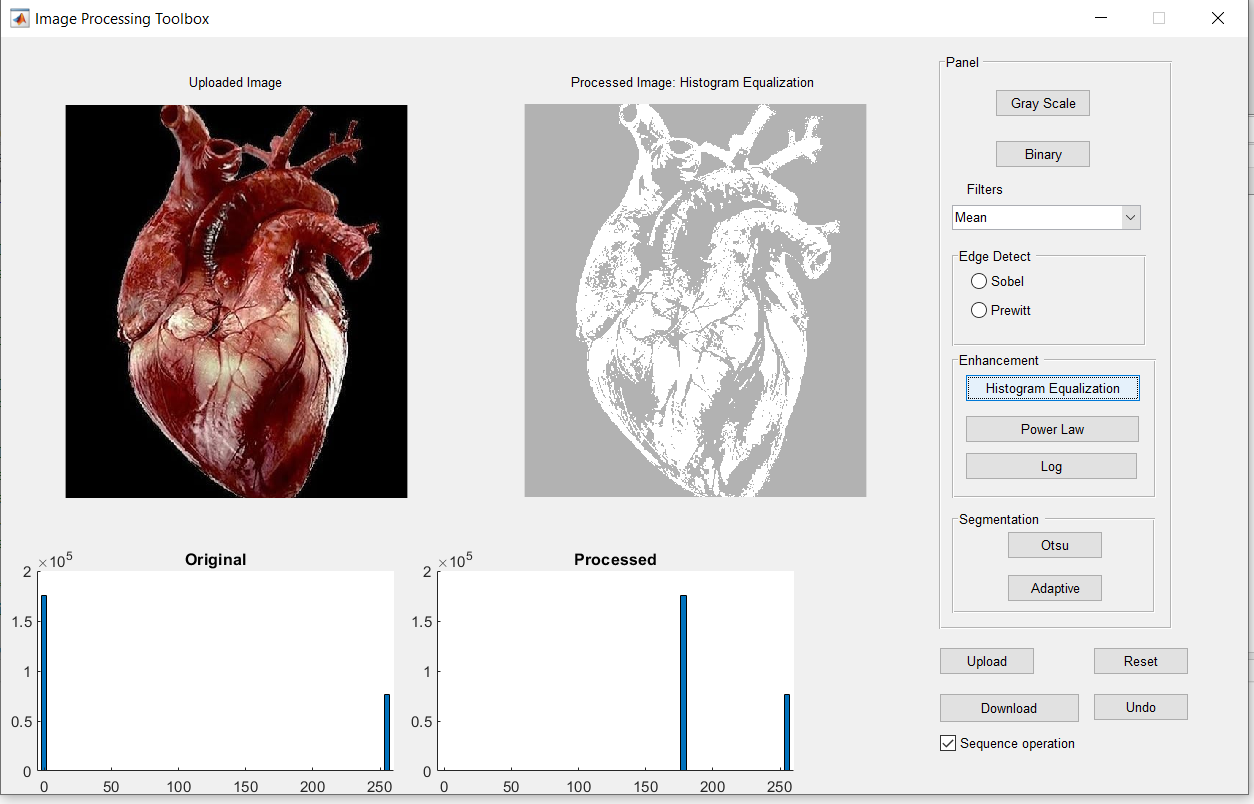


Fig. Sequence operation ( Adaptive & Histogram Equalization)