CALIFORNIA STATE UNIVERSITY, NORTHRIDGE

Digital Image Restoration and Enhancement Application in MATLAB

A thesis submitted in partial fulfillment of the requirements
For the degree of Master of Science in
Software Engineering

By

Ayrin Golestanian

December 2020

The graduate project of Ayrin Golestanian is approved:

Robert D McIlhenny, Ph.D.	Date
Jeff Wiegley, Ph.D.	Date
Katya Mkrtchyan, Ph.D., Chair	Date

California State University, Northridge

Dedication

To my family, who walked with me through this long journey and always cared for and supported me.

Table of Contents

List of Figures

List of Tables

ABSTRACT

DIGITAL IMAGE RESTORATION AND ENHANCEMENT APPLICATION IN MATLAB

By

Ayrin Golestanian

Master of Science in Software Engineering

Image processing is a technique to apply some operation on an image and extract or change the data for a specific use. Image processing is a correction of a deformed image and is already being used by many corporations such as medical, agriculture, and production automation; thus, it can be accepted and applied in more fields.

This thesis is for CSUN students who are interested in work on image processing. The application is written in MATLAB, and this thesis goal is to help students to understand the image and the processes that can apply to it to change an image matrix for different purposes. Moreover, it is a general idea to display how image processing functions. The application works on few forms of digital images such as RGB, Grayscale, and Black and white. The application has the function to convert colorful images into grayscale or Black and White and prepare them for further analysis. This application can add noise or deformation to a picture to show the problems caused by taking an image and how to improve the image resolution. It has Morphological operations, and also, it will show edge detection with different techniques.

Introduction

1.2 Background

1.1.1 Image Processing Technology

An image plays an essential role as a mass communication medium in today's society and carries valuable information. Human vision has an intricate mechanism and can collect data from an image to process it; however, it is not enough to extract specific information by looking at it [1]. The technology came in to help humans to manipulate images to reach a particular set of goals. Digital image processing is a technique that operates a set of computer algorithms to perform a fascinating work on an image, such as create an image, process, display, etc., to extract some useful information [2]. Image processing becomes a practical technique in many branches, such as environmental science, biology, vehicle guidance, robots, satellite, medical, etc.

Degradation and deformation happen during formatting a digital image. In these cases, image restoration techniques come in use to reduce the issue. Such methods will help to recover or reconstruct the digital image and improve the quality of it. Examples of digital image enhancement techniques are noise reduction, filtering, deblurring, and fixing the contrast [3]. The following is an example of image processing in the medical field. In the medical field, the resolution of the taken images plays the most critical role. Most of the MRI or CT scan images are in low resolution, contrast, or geometric deformation. Also, ultrasound causes lots of noise while taking images. Moreover, a patient can move in the duration of taking an X-ray, and the outcome will be a blurred image. Today's medical field image processing techniques can remove the degradation and deformation to have a clear vision to diagnoses the disease in the early stage [4].

1.3 MATLAB

MATLAB is an interactive programming platform for technical computing. MATLAB most uses are in developing an algorithm, data analysis, embedded system, Image processing and computer vision, robotics, math, and computation [5]. The MATLAB means matrix laboratory. For matrix computation, EISPACK and LINPACK, the Fortran packages, developed in the 1970s. Furthermore, MATLAB was developed 1984 to help easy access to those packages [6].

MATLAB has a visual system, and it has two- and three-dimensional image processing and data visualization high-level commands [7]. The GUI environment used for this thesis is App Designer in MATLAB 2018b. The GUI will have buttons, input field state elements, single and multi-option components to interact with each other to show the result. Using GUI is going to make the work easier and understandable for the user.

1.4 Motivation

The image processing topic was new for me before I start the project. I search a lot to understand the principle of this topic. My thesis research will introduce the basics of Image processing to students who like to work on digital images. The MathWorks website has tons of information and small codes. However, for someone new in this field, the codes are scattered and must retype, connect, then run to see the result. This application contains the main concepts and displays different effects on an image by choosing a few options.

1.5 Similar Work

There are many strong and powerful image processing tools out there written for companies working in these fields. For example, Varex Imaging works with x-Ray in the medical field [8]. However, this project is for small use, and the purpose is to introduce this technique to new users.

In this project's research, I found many designs about image processing on the MathWorks website. They were single applications that focus only on one aspect of this topic, such as designing a 3-dimensional graph after analyzing the image's colors. Also, APPS in MATLAB provides a section for Image Processing and Computer vision. Such as Image region Analyzer, Image viewer, segmentation, etc. Moreover, I found online applications with fixed data. It means the user could not upload an image and must work with the developer's default image. In some cases, no passing parameters, and it was impossible to see the result with different inputs. I tried to find the general use of methods in this field and sum them up in this application.

1.6 My Approach

This application's first step is to Converting an RGB image into grayscale and RGB into black and white. This method changes the image's data type so further the user can apply other masks to these pictures. There are histograms to show the distribution of color in each pixel in the image.

Filters are other features of this application. They are smoothing the image and change frequencies in it. The filters are Average, Median, Wiener, and Gaussian. Also, the other process in this application is edge detection. The edge detection method uses to find the boundaries of an object inside of an image. The techniques that operate in this thesis are Sobel, Canny, Prewitt, and Roberts. The application has Morphological processing, with Erosion, Dilation opening, and closing procedures.

Image

2.1 Digital Image

Digital images create when we use a camera, scan an image using a scanner, or create pictures using graphic design tools. Thus, a digital image is another version of a real image generated with numbers for a computer to handle it [9].

2.1.1 Image Layout

A digital image has two dimensions filled with values called pixels or picture elements [9]. Usually, they are ordered in an array, and the size of an image depends on their pixel array size. The image's width is defined by the number of columns of the array, and the height is the number of rows. The pixel array is a $M \times N$ matrix where M is the column and N the row [3].

$$i = \begin{bmatrix} i(1,1) & i(1,2) & \cdots & i(1,M) \\ \vdots & \ddots & & \vdots \\ i(N,1) & i(N,2) & \cdots & i(N,M) \end{bmatrix} \qquad (1.1)$$

To refer to a specific pixel in the image matrix, we need a coordinator, and it is defined by x and y. In digital image processing in MATLAB, the X coordinate starts from left to right, and the Y coordinate starts from the top left and goes down. It means the Y coordinate is flipped, and the (0,0) coordinate is located on the top left corner (Figure 2.1) [10].

Figure 2. 1 The 2-Dimensional coordinate space of the digital image

Usually, in the cartesian system (0,0) located on the bottom left side, it is flipped in the digital image because it follows the scanning pattern of an electron beam designed for television. The beam scans from the top left corner of the screen to display an image and moves to the right. When scanning reaches at the end of the line, it does not move from right to left. It comes one row down and again starts from the left to right [11].

2.2 Pixel

Pixel is a word that is invented for "picture element" [9]. In some cases, it appears as (x, y) and sometimes column and row (N, M). A pixel represents the smallest property of an image and contains numbers that describes the quality of that image, such as resolution and brightness [12]. Figure (2.2) is an example of creating a 4×4 array and display it as an image. Every element of the array represent a color number, and The image() function specifies that number to 1 pixel in the image. The result is a 4×4 grid of pixels, and the color bar next to the image represents the current image colormap (Figure 2.3) [13].

```
I = [0 5 7 9; 12 14 22 27;...
     30 38 41 47; 52 59 62 72];
 image(I);
 colorbar;
```

```
I =

     0     5     7     9
    12    14    22    27
    30    38    41    47
    52    59    62    72
```

Figure 2. 2 Code to create an image of matrix data

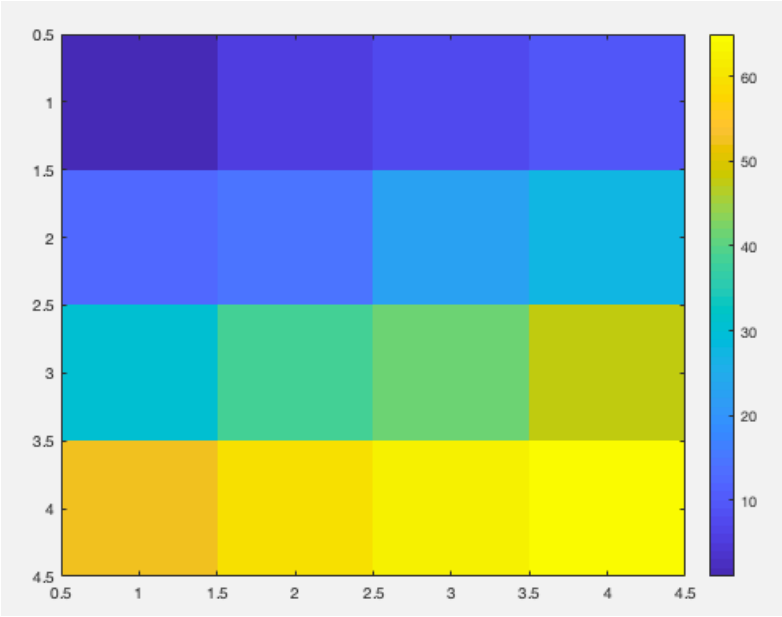

Figure 2. 3 Display image of matrix data

In figure (2.4), the magnified section demonstrates the pixels of the image. Choosing the pixel region tool from MATLAB Image Viewer helps users move the mouse over the picture, extract a portion, and see the shade of colors and their brightness. Each pixel carries one color and the pixel information [13].

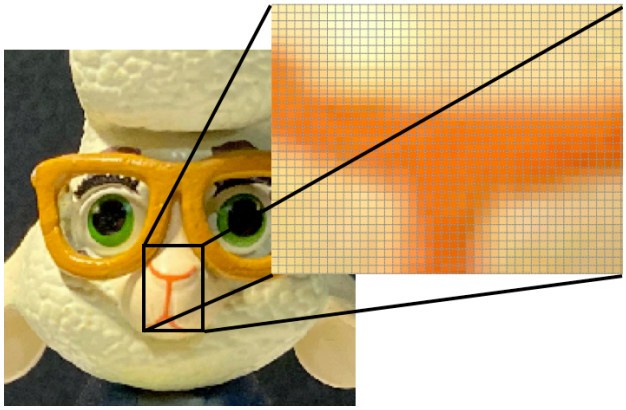

Figure 2. 4 Magnified portion to display the pixel

2.3 Size

The number of pixels in an image represents the Size of a digital image, and it should not confuse with real word image size. If they say the image's size is 1284×1024, it means 1284 pixels are from the left side to the right and 1024 pixels from top to bottom. And by multiplying these numbers, we can find 1,310,720 pixels on that image [10].

2.3.1 Resolution

The number of two-dimension pixels grid with the size of the data stored in a digital image is called resolution [10]. Resolution of an image measures with a pixel per inch(ppi). Ppi refers to pixel array, and it is different for Dpi, which stands for dot per inch and uses for resolution of a physical printed image [11]. To illustrate, if an image has a size of 34501024, it has a total of 3,532,800 pixels and a resolution of 3.5 megapixels [14]

2.4 Reading and Displaying Image in MATLAB

After knowing the digital image, pixel, and resolution, I can now show a few commands to display a digital image in MATLAB.

To read a file from the current folder you only need to use the imread() function.

```
I = imread('INFO.png');
```

The function imread() is case sensitive and must type the file name in the exact name that you saved before. We pass the value of the function into 'I' argument for further use.

To display an image on the screen, can use imshow().

```
imshow(I);
```

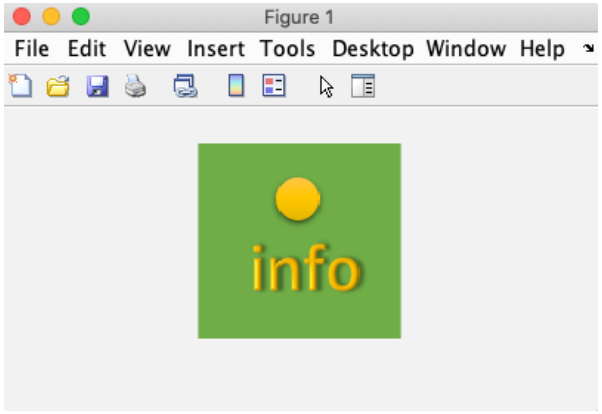

Figure 2. 5 Displayed image in the MATLAB

2.5 Image format

The data stored in the image describes the format of that image. Each structure has its specific use and has advantages and disadvantages. For example, TIFF is suitable for printing purposes, and web designers more often use PNG. The images used in this project are PNG and JPEG.

2.5.1 Lossy compression

Loss-prone or lossy are terms used for an image that lost part of its data. More often, to store images, we compress them; thus, it requires less space and can transfer them faster. When we compress an image, some data disappears from the original file. When the image is lossy compressed, it is impossible to go back. And if this process happens continuously, the image will lose more data [11].

2.5.2 Image format types

The Joint Photographic Experts Group or JPEG sometimes comes in '.jpg' or '.jpeg' format. JPEG is a lossy format because it is a reduced size image. Its best use in digital cameras, printing, web design, PowerPoint, etc. [15] [11]. The second format is PNG in this project. PNG stands for Graphics Interchange Format files, and it is a lossy type too. This format supports sixteen million colors and designed to replace GIF [15] [11].

2.6 Opening Image in the Project

In this application, I design a panel that contains a window and a load button to open an image file. The MATLAB 2018b has no option for displaying an image. That is why we have to use the 'Axe' option to display a plot and graph. Have to clear the values in x and y-axis and generate code to display the image in that Aux.

The application runs, and the user in this section can press the load button to load an image. To give functionality to the button, I had to create a callback function. In the function section having some codes will open a file from a browser. To open a modal dialog box, I use uigetfile(). This command opens a dialog box in the current folder where you run your program file. It allows the user to select or enter a file name if it is valid. To avoid displaying all kinds of files, I used

filters to choose only .png .jpeg and .jpg files in the dialog box. When the user clicks to open, the file name and path will save in [filename, pathname]. The strcat() concatenate and keeps the file name and its full path and pass it into an argument. Then, imread used that information to open the file. And ismshow() displays the image on the designed location on the panel.

```
function LoadImage_btnPushed(app, event)
    global rgbImage fullpath;
[filename, pathname] = uigetfile({'*.png; *.jpg; *.jpeg'},...
                        'Browse File');
fullpath = strcat (pathname,filename);
rgbImage = imread(fullpath);

imshow(rgbImage,'Parent',app.DisplayOriginalImage);
end
```

Figure 2. 6 The code to open a file

After pressing the 'Load' button, the file browser window opens, and the user can choose the specific image format file from any location from the computer, and it displays in the panel after loading an image (Figure 2.7).

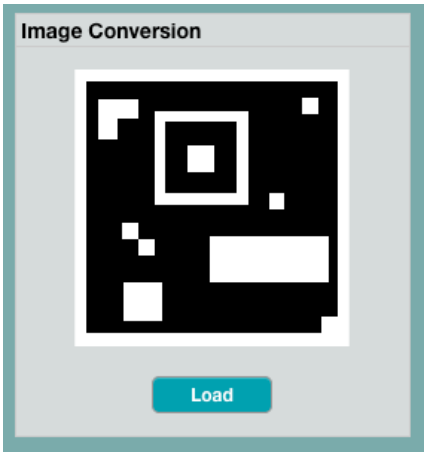

Figure 2. 7 Panel after load an image

Color

3.1 Color components

The digital image carries a significant amount of information about the color and can detect the image's quality by analyzing it. The principal identifiers of color are hue, saturation, and lightness. These three properties together create a high or low-quality image.

3.1.1 Hue

Hue means color or, in other words, a pure tint that contains three primary colors (red, blue, and green) in it. Also, it is a color identifier [16]. The color wheel is a calculated degree of the color. It starts from 0 and ends on 360. 0 and 360 are red, 120 on the color wheel is green, and 240 degree is blue [17].

3.1.2 Saturation

The saturation stands for the pureness and intensity of a hue. Pale colors are unsaturated, but vivid colors are saturated [16]. The vivid colors are bright, and they are shining because they are rich in color, and when it loses its vividness, it is fade and unsaturated. A black and white image is de-saturated because it lost the hue, and there is no intensity [18]. The Saturation calculates by a percentage between 0 to 100. The 0 percent is the black color, and 100 is the full color [17].

3.1.3 Lightness

The Lightness of a color, which sometimes identifies as value or tone, shows how dark or bright a color is. Color with low brightness and low value is 0 or black. The opposite of black, if brightness and tone are high, the color gets close to white or 1. The Lightness measures by percent, so 0 is black, and white(lightness) is 100 percent [17].

3.1.4 Contrast

A high-quality image has the right amount of brightness and contrast. Brightness works on the lightness and darkness of an image, and contrast is the distinction between brightness of the objects in space [19]. Imagine a white bear is walking on a snowy, white background. Visually it is hard to separate the bear from the environment, and this a low contrast if we locate a person in a black jacket in the same space that gives us a good contrast [19]. The range of intensity on the histogram is wide for high contrast images compare to low contrast ones [12].

3.2 Images data type

3.2.1 RGB

In the color model, the two primary colors are red (R) and blue (B), and one of the secondaries is green (G). An RGB image uses these three-color scales for its primaries. RGB is like three-layer colors on top of each other and produces a colorful image [20].

Figure 3. 1 Three layer of an RGB image

They call RGB also a true-color image, and it is a three-dimensional array. In MATLAB RGB array show as M×N×3; thus, each pixel stores the value of R, G, and B in itself to show the actual color of an image in that location. The array number 3 represents a channel number, and the channel order is $R_0G_0B_0$, R_1, G_1, B_1, ..., $R_nG_nB_n$ [11]. RGB in MATLAB can have a datatype of

unit8, unit16 or double, and this represents the range of values in an image. RGB range of value for double is 0 and 1, for unit8 is [0-255], and unit16 [0-65535], and these numbers are the possible numbers of colors in the image [13].

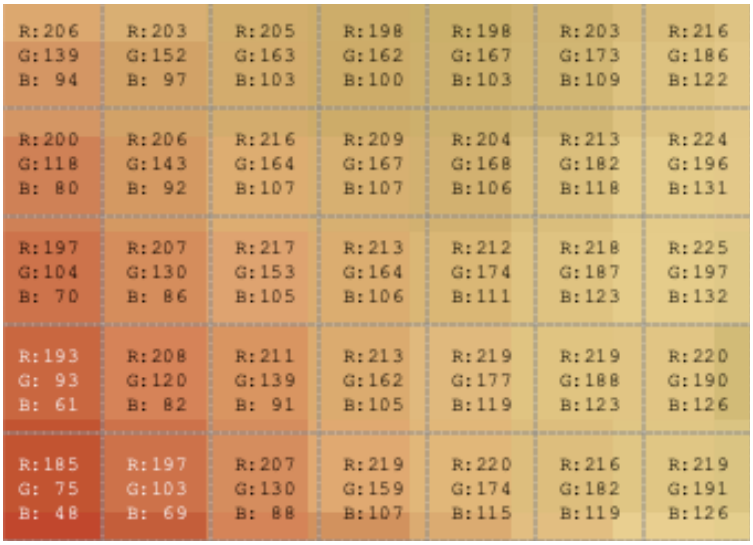

Figure 3. 2 Pixels from an RGB image

In figure 3.2, we can see pixels with different values stored in R, G, and B; a combination of these three-color panels can create the intensity of the image. This image is ‘unit8’, and each color component can have a number in the range of 0 to 255 [21].

3.2.2 Grayscale

Grayscale or intensity image is a two-dimensional array. This format assigns a numerical number to each pixel, and that number represents the intensity of that image. This type of image is a single layer [21].

Figure 3. 3 Grayscale image

145	173	190	192	181	193	191	191	175
133	160	186	195	185	194	193	187	181
129	143	180	187	189	194	198	186	170
119	129	155	182	193	200	189	181	162
108	124	138	162	203	199	195	182	168
94	98	120	143	165	196	188	166	169
74	100	104	131	152	179	182	180	177
76	86	101	114	144	161	175	185	188

Figure 3. 4 Values stored in a grayscale image pixel

Grayscale images can take directly from a camera or convert a true-color image into grayscale. Also, it is possible to create a grayscale image using R, G, and B channels to combine these colors in a way to produce different shades of gray. The outcome will be a grayscale image that has hue, saturation, and lightness [21]. A grayscale image has some advantages compared to RGB. The true-color image has 24 bits (8 bit for each color), but when we convert it to grayscale has only required 8 bits to store pixels. Because of the grayscale image structure, it is beneficial in most image processing techniques [10].

3.2.3 Black and white

Black and white images are also called binary. They are two-dimensional arrays with value of 0 and 1 in each pixel [10]. The region of interest in a binary image is marked with 1, and it uses 0 for background color [21].

Figure 3. 5 Binary image

1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	1	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	1	1	1	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	0	0	0	0	0	0

Figure 3. 6 Values stored in a binary image pixel

3.3 Image conversion

In MATLAB with a simple function, we can convert a true-color image into grayscale, grayscale, or true-color into a binary image. This process reduces amount of information from a file. For example, a converted grayscale has fewer data compared to an RGB image. However, the essential and significant information is maintained, such as edges and regions, etc. In this project, I converted RGB into grayscale and binary because this type of formatting plays a critical role in this application.

3.3.1 RGB to grayscale

The function MATLAB uses to convert the RGB image into grayscale is rgb2gray. This command clears the hue and saturation of the colormap image.

I = rgb2gray(RGB_Image);

To rgb2gray function first reads the true-color image. Then extracts three colors of the R, G, and B and saves each color separately in three two-dimensional files. It means now we have three two-dimensional files, and each one of them contains information about R, G, and B. After that, the function creates a new two-dimensional matrix, then weighted sum of R, G, and B's components and transfers the new values into an (i,j) location inside the file [22].

The rgb2gray algorithm: NewMatrix (i , j) = (0.2989×R) + (0.5870×G) + (0.1140×B) [21].

Figure 3. 7 RGB into grayscale

3.3.2 Grayscale to black and white

To convert grayscale into a binary image, thresholding is the most common method. If we do not fix a number, MATLAB will use its default by finding the best threshold with analyzing the image. Otherwise, we can generate a number that works as a borderline. If the gray value is above the number, it will turn into 1, it means white, and below the number into 0, black [23].

$$g(x,y) = \begin{cases} 1 & if\ f(x,y) \geq Thresh \\ 0 & below\ the\ Thresh \end{cases} \qquad (3.1)$$

To convert gray scale to black and white MATLAB uses im2bw() function.

I = im2bw(Gray_Image);

MATLAB reads a file; if it is an RGB image, it will convert it into a grayscale then will start the process. The function im2bw() creates an image array with the same row and columns as the grayscale image and makes all values of that array equal to 0. Now the comparison starts; this function reads (x_0,y_0) from the grayscale image, and if it is greater than or equal to the threshold, the value of new arrays (i_0,j_0) turns into 1; otherwise 0. In this method, im2bw fills the entire new

array's row and columns with 0 and 1s. The figure 3.8 displays the conversion of a grayscale image into black and white one.

Figure 3. 8 Grayscale into Black and white

3.4 Histogram

A histogram is a graph and can find in any documents that work with statistics. It has an x and y-axis and bars that show the distribution of data. MATLAB also uses histogram in the image processing, and it displays the intensity values of pixels inside an image. For example, for an 8bit grayscale image, the range of intensities is from [0-255]. On the x-axis, we can see the 0-255, which represents the intensity of the pixels and the y-axes' frequency [12].

The histogram also works for RGB images in MATLAB. It can show them as a two-dimension histogram, or separate each R, G, B channels and desplay in a different histogram. Also, a three-dimension histogram with z-axis.

Figure 3.9 displays the histogram of a black and white image created by imhist(Image) MATLAB command. The color bar at the bottom of the histogram shows the intensity level. And the numbers on the y-axis frequency of pixels with that number of intensities.

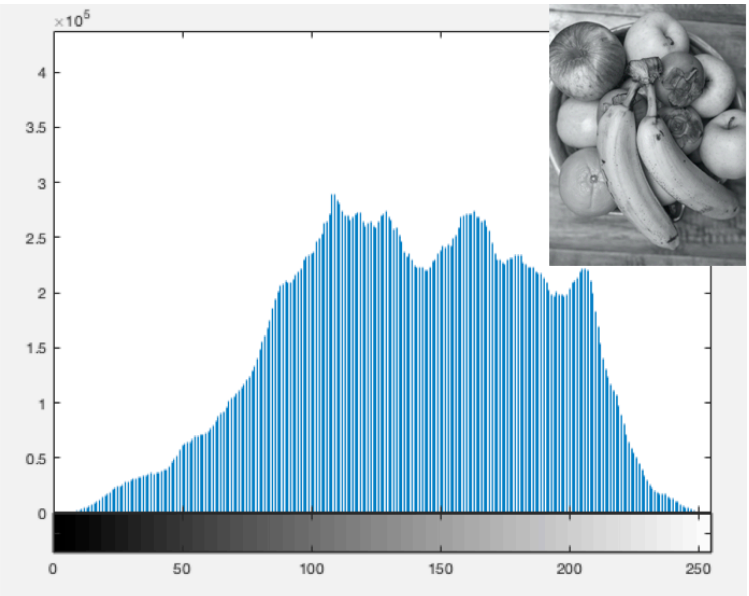

Figure 3. 9 Grayscale image and its histogram

3.5 Convert RGB to grayscale in the Project

In the project, after loading the image from ‘Convert’ panel, user can choose ‘Convert to Grayscale’ button. The callback called the rgb2gray function and convert the image. Then it displays it in the designed window.

3.6 Convert Grayscale to B&W in the Project

Another option of the ‘Convert to' radio button section is ‘Black and White’. This radio button actives another panel named ‘Black and White.' In this section, if the user hits the ‘convert’ button, the image will convert into a grayscale. If ‘Black and White’ is the user's first choice, then the program's image will first convert into grayscale then into black and white. If this were the second choice of the uses after choosing the ‘Grayscale’ radio button, the previous grayscale converted would be used.

3.7 Image Enhancement in the Project

A low-quality image can be dark due to low lighting, or environmental phenomena affect the quality, such as fog or rain. These types of images are called Haze, and they lack both contrast and light. MATLAB has a technique to fix these problems. The imreducehaze() function improves the visibility of an image. The first step is to invert the original image to see the low-light areas in it. These areas are a bit hazy, so to reduce them imreducehaze() function is helpful. To see the result of improvement areas, have to invert the image for the second time. Figure 3.10 from the project reveals the differences between the low-light and improved ones next to each other. As a result, the histogram of the improved image has a wide range of intensity.

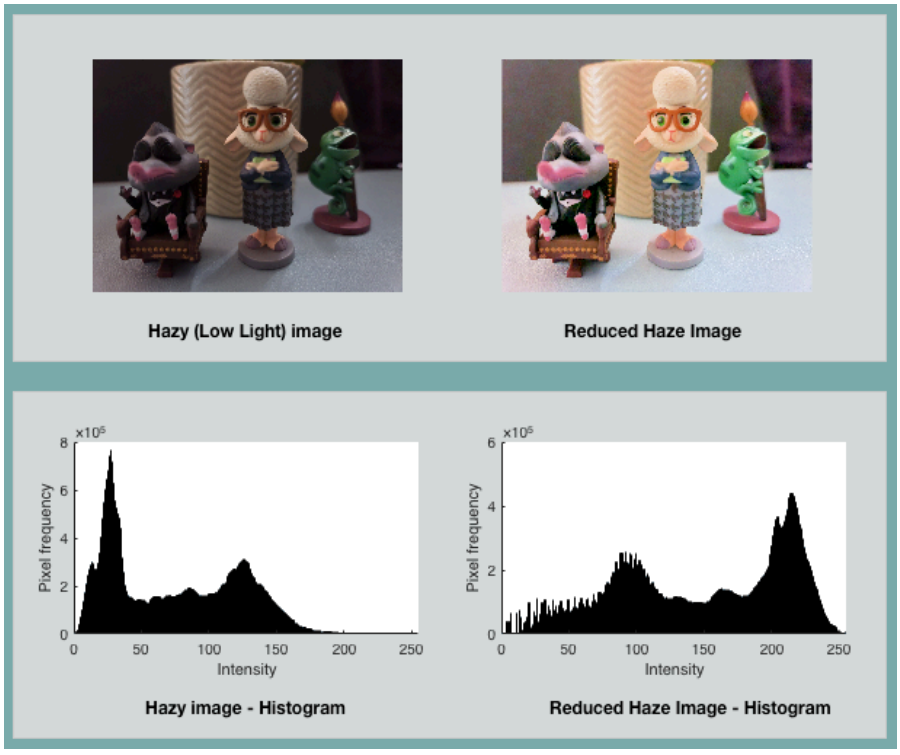

Figure 3. 10 Hazy image improvement

The brightness of an image can change by adding or subtracting values to the image matrix. I designed a slider next to the converted image that moves in the range of 0 to 1. The value of the slider affects the value of image matrixes. By sliding it down, the value gets close to 0, and the

picture becomes darker, and when it slides up, the image turns lighter because the value gets closer to 1. Also, it affects the histogram too. When the image lightness rises up, the number in bins from a range of 0-254 droppers and bin with the value of 255 becomes the longest bin. Figure 3.11 shows the difference between the three images and histograms when the brightness of the image changes.

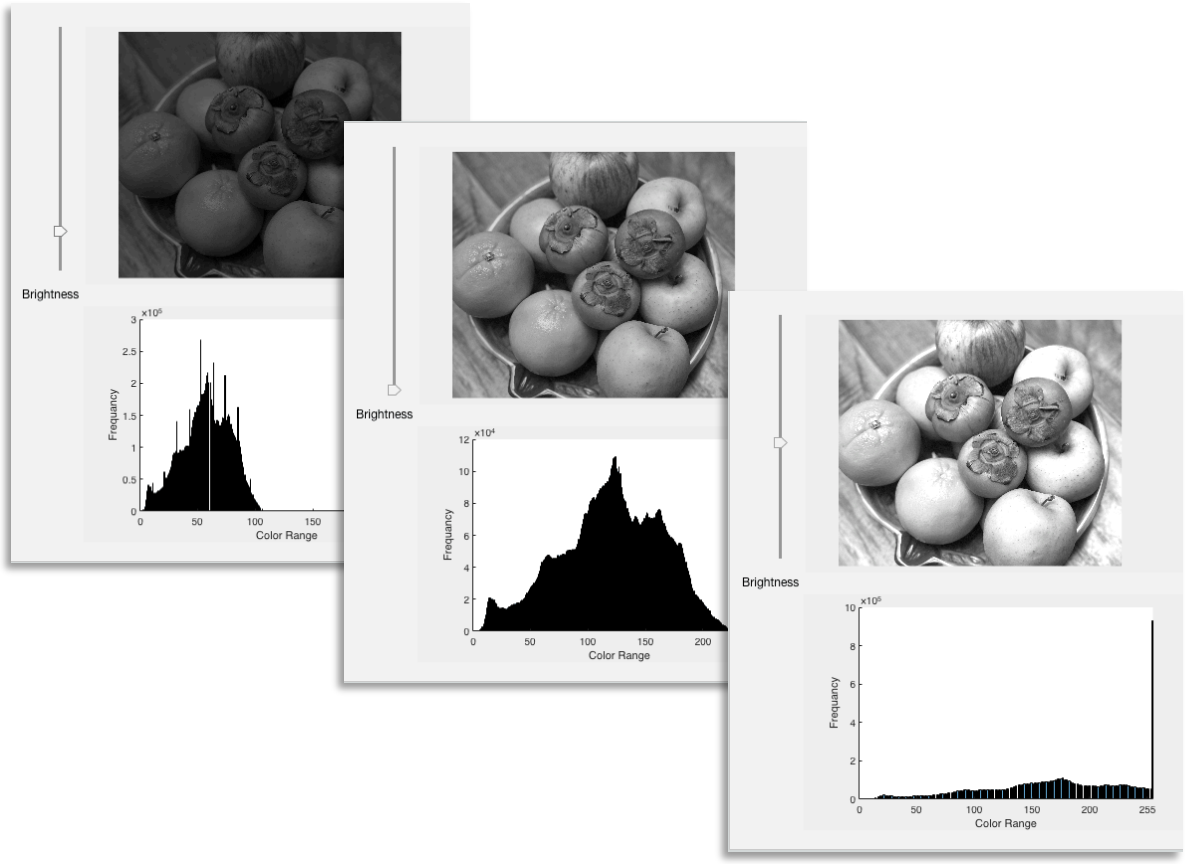

Figure 3. 11 Three-stage of change in brightness and histogram of an image

3.8 Histogram in the Project

When the user converts the image into grayscale or Black and white, the image below of the converted image can see the histogram of the image (Figure 3.10). The black and white image histogram has only two bars because the pixels carry the values of 0 and 1.

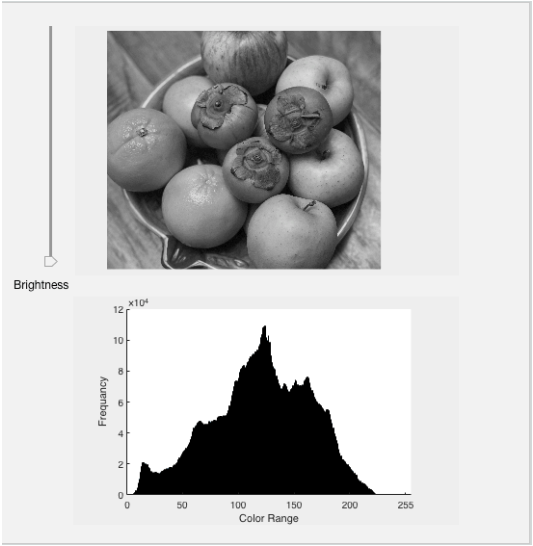

Figure 3. 12 Project's Grayscale image histogram

The histogram(Image) function creates a histogram plot for the downloaded image. This function uses an algorithm that automatically produces bins with a uniform width to show the range of color distribution. The bin shape is rectangular, and the bin's height is the frequency of that pixel in the image.

Each bin in the histogram represents the different values of R, G, and B that are used to create a color. Every number between 0 to 255 carries a different hue, saturation, and brightness. A new color will create when these three colors (R, G, B) with different intensity combine. For example, to create an original Orange color, we need R:255, G:165, B:0, and Halloween orange requires R:230, G:108, B:44, and Tangerine orange R:255, G:132, B:0. If we design a histogram for these three colors, we are going to have figure 3.12. The x-axis numbers are the color range, and the y-axis the pixels' frequency in a specific color range. Bins carry the information of both the x and y-axis.

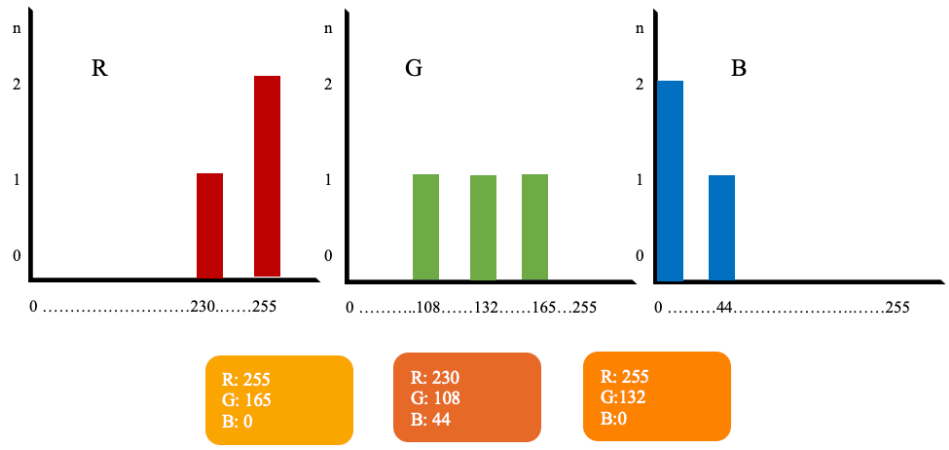

Figure 3. 13 Histogram example of three different orange colors

Using histogram(), I separated three primary colors, Red, Blue, and Green, and designed a histogram for every single one of them (Figure 3.13).

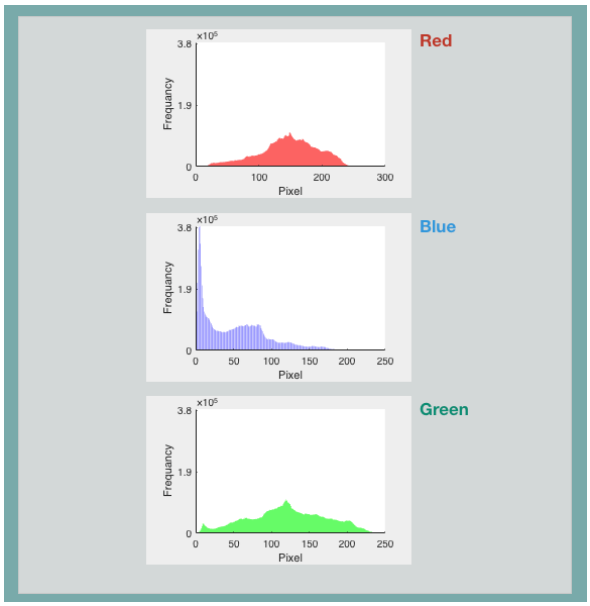

Figure 3. 14 Color histogram for separated color levels

Morphological Operations

During scanning or taking an image, deformation happens, and because of that, we may see two separate objects connected or small unexpected dots appear on top of an object. For example, first image in figure 4.1, a black dot inside the white circle is called false-negative and contains a '0' value. On the black background, a white object with a value of '1' is called false-positive. The second image two objects are connected with few pixels and the false-positive between two separate objects makes our eyes see them as one.

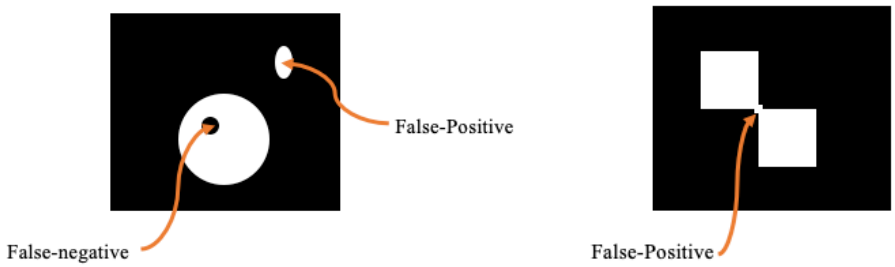

Figure 4. 1 Images with false-negative and false-positive pixels

Morphological processing can help us to solve these problems. The definition of Morphology is to study the shape and structure of an image. Morphological Operations in image processing are nonlinear, and they work on the morphology of features in an image. This method works on binary images with the value '1' as foreground and 0 as background.

In Morphological processing, a set of data is needed, and it is called a structuring element (SE). It is a small binary image and a matrix of pixels in different shapes. The pixels have a value of 0 or 1. Morphological processing uses this structure to probe an image, and can help to fill the holes in the object or clear pixels outside of the foreground that we do not need them. The command to create the shape of the morphological structuring element is strel(shape, parameter). The chosen flat Structuring elements for this project are Disk, Diamond, Sphere, and Square. The

parameter, known as neighborhood, is the radius from the origin, and it creates a matrix of 1 and 0, based on Shape [24]. The Origin is the center of the matrix and it calculates by floor((size(parameter)+1) /2) [24]. Figure 4.3, is a sphere SE made by strel('sphere',3). R is the distance from the origin to the edge of the sphere.

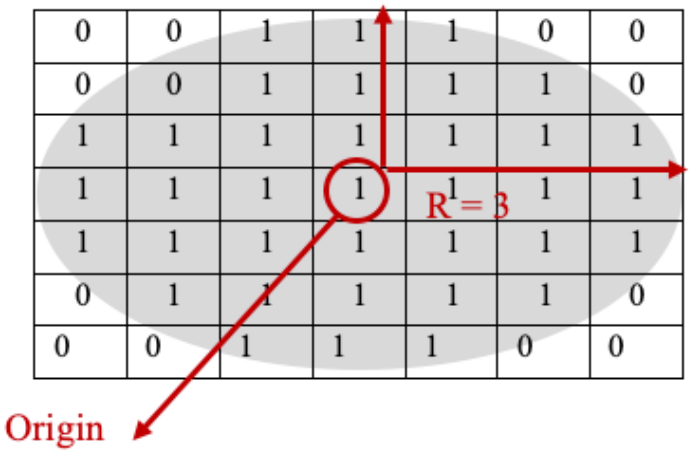

Figure 4. 2 SE in shape of a sphere shape

4.1 Morphological Operation Types

The essential Morphological Operations are Erosion and Dilation. The other operations in the Morphological are based on these two. Other operations collected for this project are Closing and Opening

4.1.1. Dilation

Dilation uses a structuring element to add a pixel to the boundaries in an image. The amount of adding pixels depends on the size and shape of the structuring element. This process makes the object inside of the image tick. The mathematical term for Dilation is: $A \ominus B$. The A is the image and B Structuring element. Suppose the pixelated yellow object below is part of an original image, and the cross shape is the structuring element [25] [26].

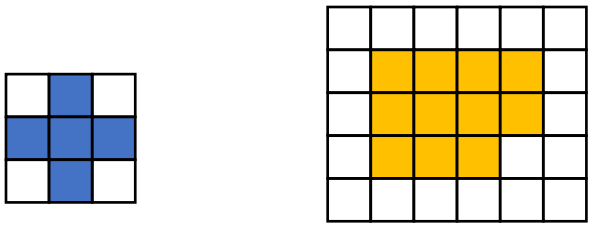

Figure 4. 3 The structuring element and pixel Image

Slide the structuring element (SE) on top of the image. The center pixel of SE, when it hits a pixel with a value of 1 will add another pixel to the object [27].

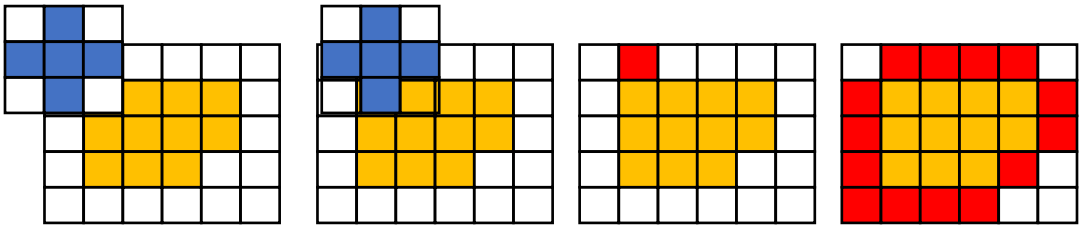

Figure 4. 4 The SE moves on top of the image and removes pixels

4.1.2. Erosion

Erosion uses a structuring element to reduce pixels from the boundaries of an object. The amount of removing pixels from an image depends on the size and shape of the structuring element. After Erosion, the object shrinks or becomes thinner [25]. The mathematical term for Erosion is: $A \ominus B$. In figure 4.5, we will slide the structuring element (SE) on top of the image. The center pixel of SE will move on the object. If SE's shape does not entirely cover the image, it will remove the pixel; otherwise, the image's pixel will remain [27].

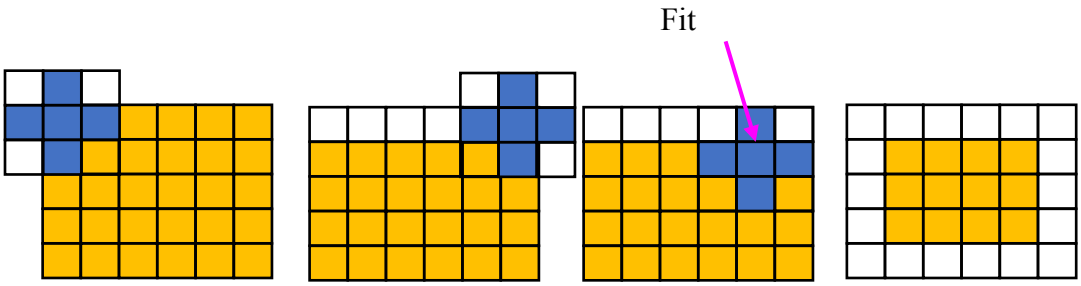

Figure 4. 5 The SE moves on top of the image and adds pixels

4.1.3. Closing

Closing first does Dilation on the image then Erosion. Closing joins or blends the narrow breaks, it closes the gap between objects, and closes the holes. The mathematical term for Closing is: $A * B = (A \oplus B) \ominus B$. In the figure 4.7 The Dilation happens first. The bridge between two shapes becomes smoother, ticker and also the gap is closed. Nest Erosion happens and removes the added pixels from the boundaries [27].

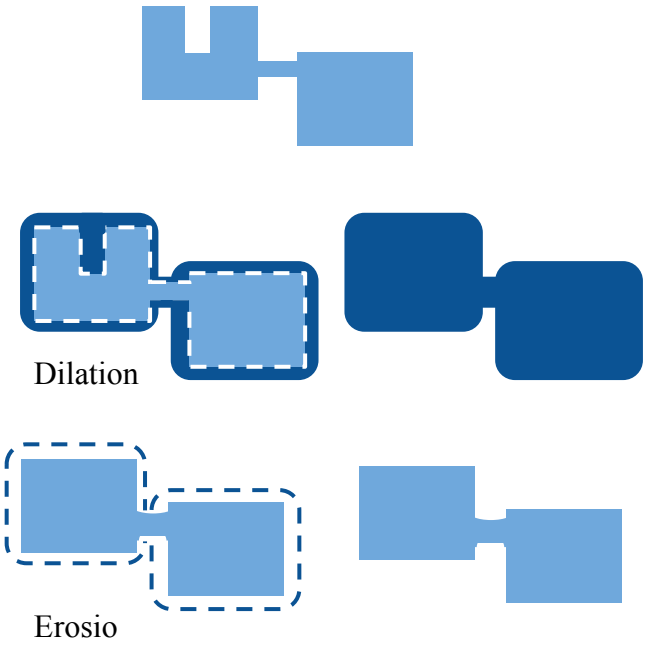

Figure 4. 6 Closing process

4.1.4. Opening

Opening first does the Erosion on the image then Dilation. This method is used to clear small objects from a binary image, separates images that are just touching, and smooths the object's contours. The mathematical term for Closing is: $A \circ B = (A \ominus B) \oplus B$. In figure 4.8, two objects are connected with a bridge. The first step of Opening is going to use the SE to struct the Erosion on the objects to clean the bridge and separate them. However, the size will shrink [27] [26].

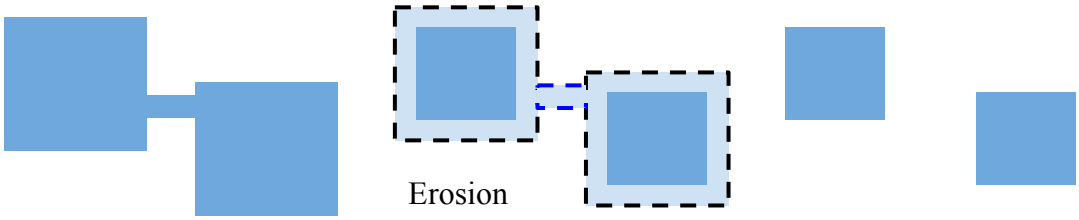

Figure 4. 7 Erosion - First step of the Opening

When the bridge is cleaned on the second step, Dilation uses the same SE to recover the lost pixels. This operation adds pixels to the shrunk object's boundaries. the size and shape may be a bit different from the original connected objects before Opening [27].

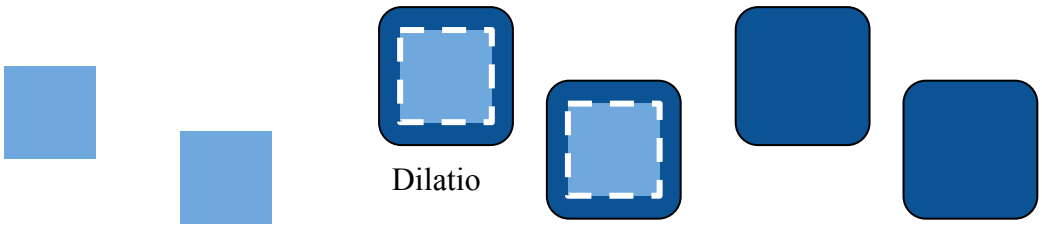

Figure 4. 8 Dilation - Second step of the Opening

Table 4. 1 Summary of four Morphological operations

Morphological Operation	Function (MATLAB)	Description
Dilation	Imdilate(Image,SE)	Background pixels that are a neighbor with foreground with dilation will become part of the foreground.
Erosion	imerode(Image,SE)	Foreground pixels that are a neighbor of SE and cannot cover the SE will become part of the background.
Closing	imclose(Image,SE)	With the same SE, first, dilate the image, then erode it.
Opening	imopen(Image,SE)	With the same SE, first erode the image then dilate it.

4.2 Morphological Processing in the Project

This section of the project requires a binary image with objects in it. For better understanding and analysis, need to load an image that contains shapes; otherwise, the result would not be clear (Figure 4.9). If the user loads a busy image such as a sunset view, the changes will not be recognizable.

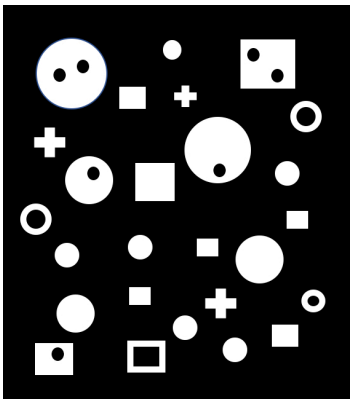

Figure 4. 9 Example of image with object

After uploading the image, application has a panel where the user can choose the shape and size of the structuring element. The structuring element that I chose for this example from the project is ‘Diamond’ with size of 6 (Figure 4.10).

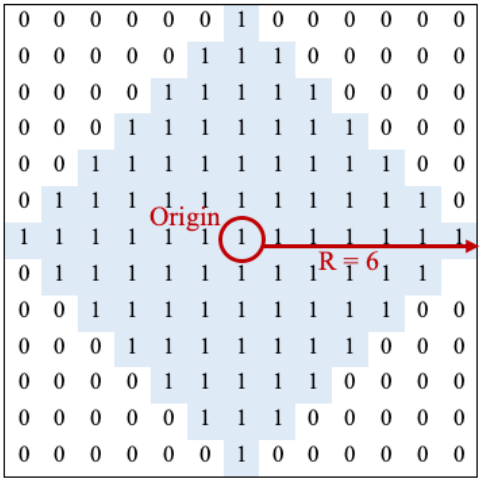

Figure 4. 10 Diamond shape SE in size of 6

In figure 4.11, we can see the MATLAB functions for Morphological operations. The app.se passes the value 6 to the operations. As it shows, ‘imdilate’ uses for dilation, ‘imerode’ for erosion, ‘imclose’ and ‘imopen’ for closing and opening. The imshow will display the result in the specified window.

```
function DisplayButtonPushed(app, event)
    global Image_BandW ;
       app.condition;
     I_dilation = imdilate(Image_BandW, app.se);
     imshow(I_dilation,'Parent',app.DisplayDialation);

     I_erosion = imerode(Image_BandW, app.se);
     imshow(I_erosion,'Parent',app.DisplayErosion);

     I_close = imclose(Image_BandW, app.se);
     imshow(I_close,'Parent',app.Closing);

     I_open = imopen(Image_BandW, app.se);
     imshow(I_open,'Parent',app.Opening);
end
```

Figure 4. 11 SE radio buttons functions

Figure 4.12 is the outcome of choosing ‘Diamond’ SE with a size of 6. And by comparing it with the original image, we can see the differences where it clears some shapes in erosion while reducing the size of them in Opening. Also, in dilation, the outline of the objects appears thicker and it means it added pixels.

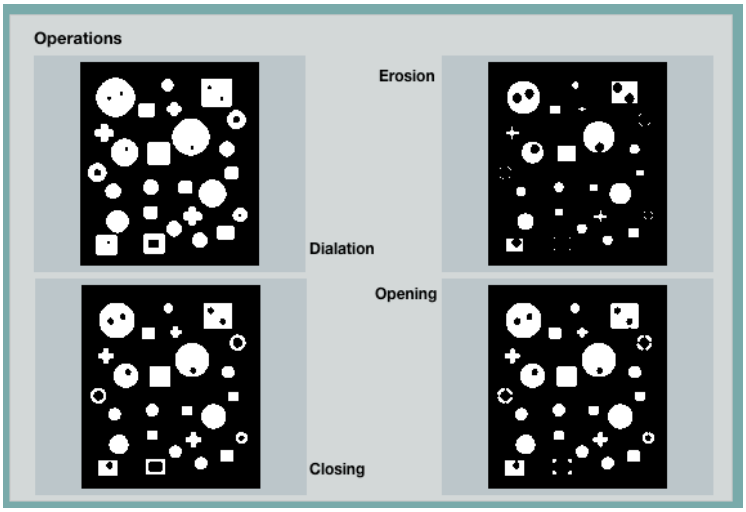

Figure 4. 12 The result of applying ‘Diamond’ SE with size of 6

In Figure 4.13, I added size 18 to the ‘Diamond’ SE and on the Erosion section the objects almost disappeared.

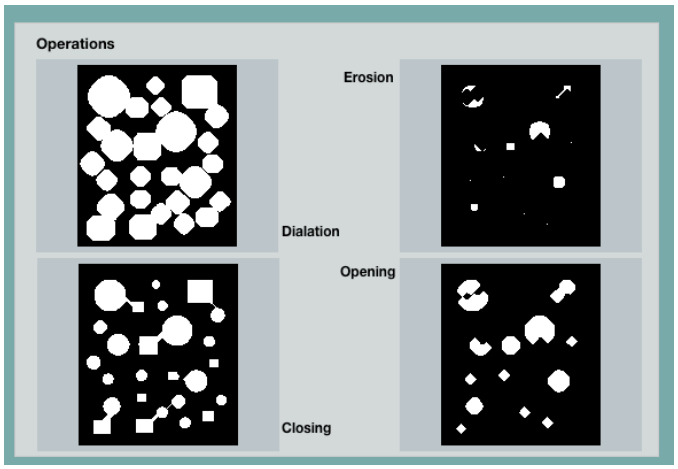

Figure 4. 13 The result of applying ‘Diamond’ SE with size of 18

Edge Detection

5.1 What is edge detection

Each object inside an image has lines around it that separate it from the background or other items; that is why eyes can distinguish the objects from each other. In image processing, the lines around an object are a series of connected pixels that define the objects' boundaries. To find the object, we have to simplify the image by cleaning the noise and filtering the useless data, this method is called edge detection [28].

MATLAB has a function called 'edge.' This function scans an image and finds places where intensity changes rapidly. On the first scan, if the intensity is larger than the threshold, it identifies it as the image's edge. On the second scan, if 'edge' finds lines less than the designed number for the threshold, it will ignore that area and make it zero; means it will become part of the background and not the object. To calculate this, we have to approximate the gradient of the intensity values of the pixels. Image approximating the gradient refers to changing the direction of the intensity inside of an image [28].

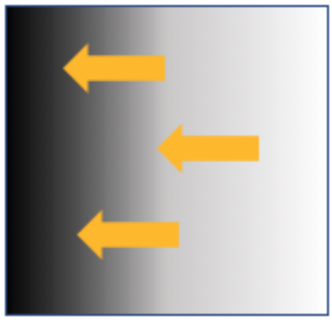

Figure 5. 1 The gradient

To find edges in intensity of an image MATLAB uses:

edge (Image . method);

Image is the name of the input image and the method is the edge detection algorithm. There are different edge detection methods in MATLAB, and for this project, I chose Sobel, Canny, Prewitt, and Robert. Before explaining the edge detection methods, I will discuss about 'convolution mask' which plays an important role in the detection process.

The Kernels used in this example belongs to Sobel. The two x and y-direction kernels convolve with every single pixel in the image to find the regions where the gradient is maximized(change). This technique is called 'convolution mask.' The center pixel is called an anchor [29]. To find magnitude of the edge have to calculate $\sqrt{{G_x}^2 + {G_y}^2}$. where $\mathbf{G_x}$ be the direction of x Kernels and $\mathbf{G_y}$ direction of y, and to find the direction of the gradient $\theta = \text{atan}\left(\frac{G_y}{G_x}\right)$ [29]. To calculate the gradient approximation of the converted grayscale image, we need x and y-direction Kernels. The matrix below is part of the grayscale image and we convolve the selected area with the G_x and G_y [30].

X – direction Kernel

-1	0	1
-2	0	2
-1	0	1

Grayscale image with intensity values [0-255]

150	150	150	183	183	1
150	150	255	210	1	1
150	255	255	1	1	1
250	255	1	1	1	1
210	1	1	1	1	1
1	1	1	1	1	1

y – direction Kernel

-1	-2	-1
0	0	0
1	2	1

Which becomes:

-1*150	0*150	1*150
-2*150	0*150	2*255
-1*150	0*255	1*255

-150	0	-150
-300	0	500
-150	0	255

-1*150	-2*150	-1*150
0*150	0*150	0*255
1*150	2*255	1*255

-150	-300	-150
0	0	0
150	500	255

$| G_x| = (-150 + 0 -150) + (-300 + 0 + 500) + (-150 + 0 + 255) = 315$

$| G_y| = (-150 + -300 - 150) + (0 + 0 + 0) + (150 + 500 + 255) = 315$

Now we have to combine values of G_x and G_y to find the magnitude of gradient at (x, y).

$$\sqrt{315^2 + 315^2} = 445$$

In grayscale intensity matrix, the anchor is the selected gray area, and the value of that cell is 150. Now we replace the new value 445 with old 150. If we repeat this method on the entire matrix, we will find pixels with highest magnitude which are the

part of the edge [30].

5.1.1 Sobel

Sobel operation or Sobel filter is one of the techniques to detect the edges in image processing and includes a pair of three-by-three kernels (matrix). Image approximating the gradient refers to changing the direction of the intensity inside of an image [29].

X – direction Kernel

-1	0	1
-2	0	2
-1	0	1

y – direction Kernel

-1	-2	-1
0	0	0
1	2	1

Figure 5. 2 Sobel Kernels to approximate intensity change in x and y-direction

The algorithm of Sobel needs a binary image; if it is RGB or grayscale will convert it into a binary one. In the next, Sobel defines the its x and y-direction kernel mask then convolve it with the original image to compute the gradient approximation. Then threshold the image to detect the edges (Figure 5.3) [31].

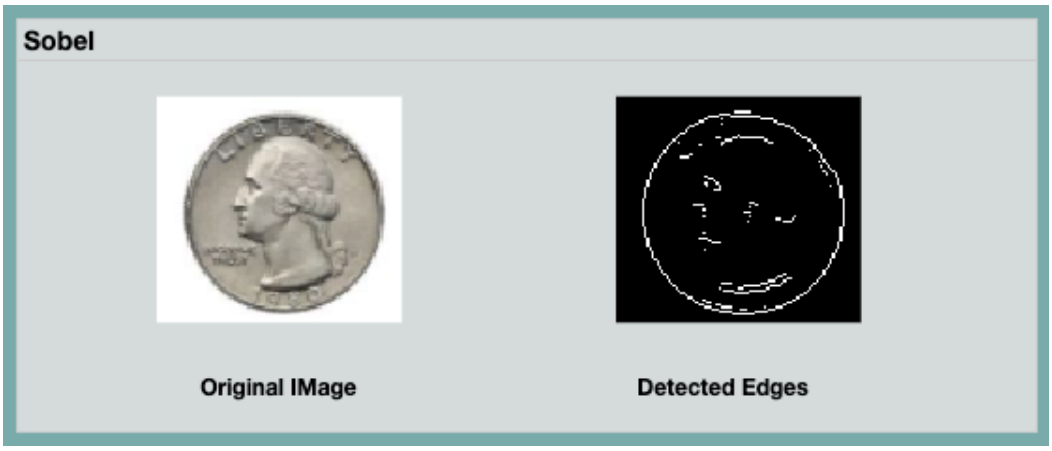

Figure 5. 3 Detected edges using Sobel

5.1.2 Canny

Canny is the most powerful edge detection technique. It uses edge tracking technique called Hysteresis, to detect the edges it uses two types of the threshold for strong and weak edge. It will detect the fragile edges if they are connected to the strong ones. Canny uses pattern recognition technique. This method maximizes the probability of detecting edges for multiply times in this case no failure in detecting edges [29]. The Image blow is the x and y-direction Kernels in Canny.

X – direction Kernel

1	1	1
0	0	0
-1	-1	-1

y – direction Kernel

1	2	1
0	0	0
-1	-2	-1

Figure 5. 4 Canny Kernels to approximate intensity change in x and y-direction

To detect the edges, Canny applies the Gaussian method to the image to make it smoother and removes some noises. Then it works on the gradient intensity of the image to highlight the regions. Algorithm of canny uses the double threshold (hysteresis) on these areas to find both weak and strong regions and if the magnitudes are less than the threshold, put an end to them. If the magnitude is a number between two thresholds it turns to zero too. Unless that pixel is connected to a strong pixel with a gradient above max threshold [29].

Figure 5. 5 Detected edges using Canny

5.1.3 Prewitt

This algorithm of Prewitt works like Sobel, and it is a gradient base operation. The advantage of the Prewitt is that you can easily detect the magnitude and orientation of an image. It is strong to detect horizontal and vertical edges. The Prewitt's magnitude of the coefficients is fixed by default and is unchangeable. And if there are diagonal directions, Prewitt finds trouble to maintain them, so it ignores them and does not show them in the outcome [29].

X – direction Kernel

1	1	1
0	0	0
-1	-1	-1

y – direction Kernel

-1	0	1
-1	0	1
-1	0	1

Figure 5. 6 Prewitt Kernels to approximate intensity change in x and y-direction

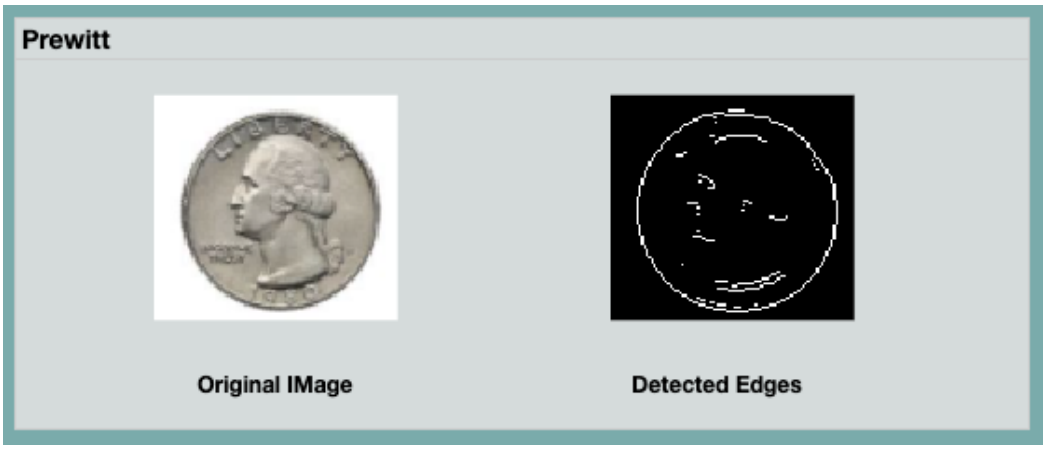

Figure 5. 7 Detected edges using Prewitt

5.1.4 Roberts

Roberts uses a two-by-two kernel mask and is simmer to Prewitt and Sobel. For this method, it is not hard to find the both directions of edges, and also it can follow the diagonal edges. On the other hand, it is very sensitive to the noises; this interferes the accuracy of Roberts in edge detection. The algorithm of starts to reading an image then convert it to double. On the next step defines its two-by-two Kernel mask and starts to compute the magnitude and gradian. Last step is the thresholding the process displays in figure 5.9 [29] [32].

X – direction Kernel

1	0
0	-1

y – direction Kernel

0	-1
1	0

Figure 5. 8 Robert Kernels to approximate intensity change in x and y-direction

Figure 5. 9 Detected edges using Robert

Table 5. 1 Edge detection Operations advantage and limitation

Operator	Advantage	Limitation
Sobel	• Can find smooth edges • Has simple computation	• Diagonal lines are not always retained • Very sensitive to noise • Edge detection result is not very accurate • Cannot detect the thick edges
Canny	• It is fast and accurate (low error rate) • The noise has less effective compare to the other methods • Using Gaussian filter to smooth a noisy image	• Cannot detect edges if Gaussian filter increases the % of the smoothness • The algorithm is expensive
Prewitt	• Works on vertical and horizontal edges	• Fixed and unchangeable magnitude of the coefficient • Cannot follow points on diagonal direction
Robert	• Can follow point on diagonal direction • Works on vertical and horizontal edges	• Noise sensitive • Not accurate to detect edges

5.2 Edge Detection in the Project

After uploading an image, edge detection options are available to choose from the provided list. I decided Sobel for this practice. Assigning the threshold to operation increases the sensitivity of the edge detection. This section aims to detect and edge and display the outside boundaries on top of the image. Each operation has its algorithm to detect the edge base. After checking the contrast of the image, it calculates the gradient. The method applies a three-by-three Sobel Kernel mask.

Figure 5. 10 Detected edges using Sobel

here are gaps inside the outline of the object. Thus, I designed a cross-shaped (two perpendicular linear SE) segmentation element to dilate the objects. The dilation closes the outside line, but there are holes inside of the detected items. After closing them using imfill() function, we will clean the lines attached to images borderline with imclearborder() function. That lines are part of the shape, and they cannot be the whole object to detect. To make the outlines of the detected object smoother, I use Erosion. It can clear extra pixels and make the segmented object more natural. In the final step, to display the outline of the shape over the original image, I used bwperim(), and Segout().

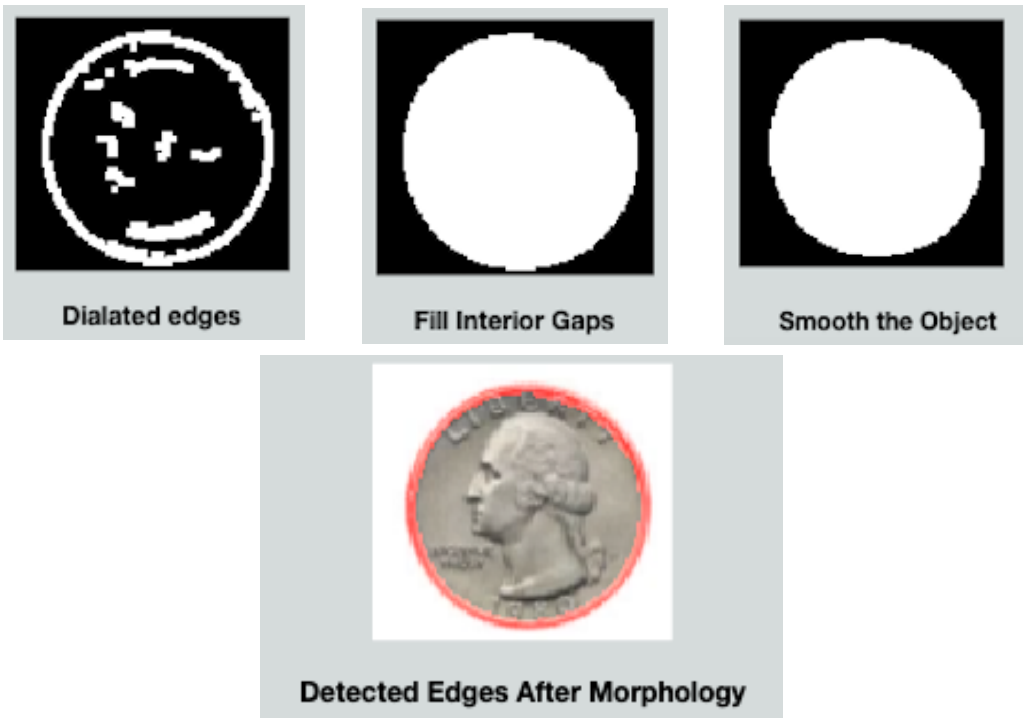

Figure 5. 11 Process of Edge detection and Morphology

Figure 5.12 is compering four edge detection algorithms next to each other. The chosen threshold is 0.150 to raise sensitivity. Canny detected more lines than the rest because it finds week lines with low intensity if connected to strong ones. They detected the coin's outline because the intensity is high at that point, and the coin has high contrast with the background. Because I raised the threshold, they found inside lines intensity less than the designed number, so they ignored that area and made it zero. If I increase the number of the threshold rapidly, the outlines will also disappear (Figure 5.13).

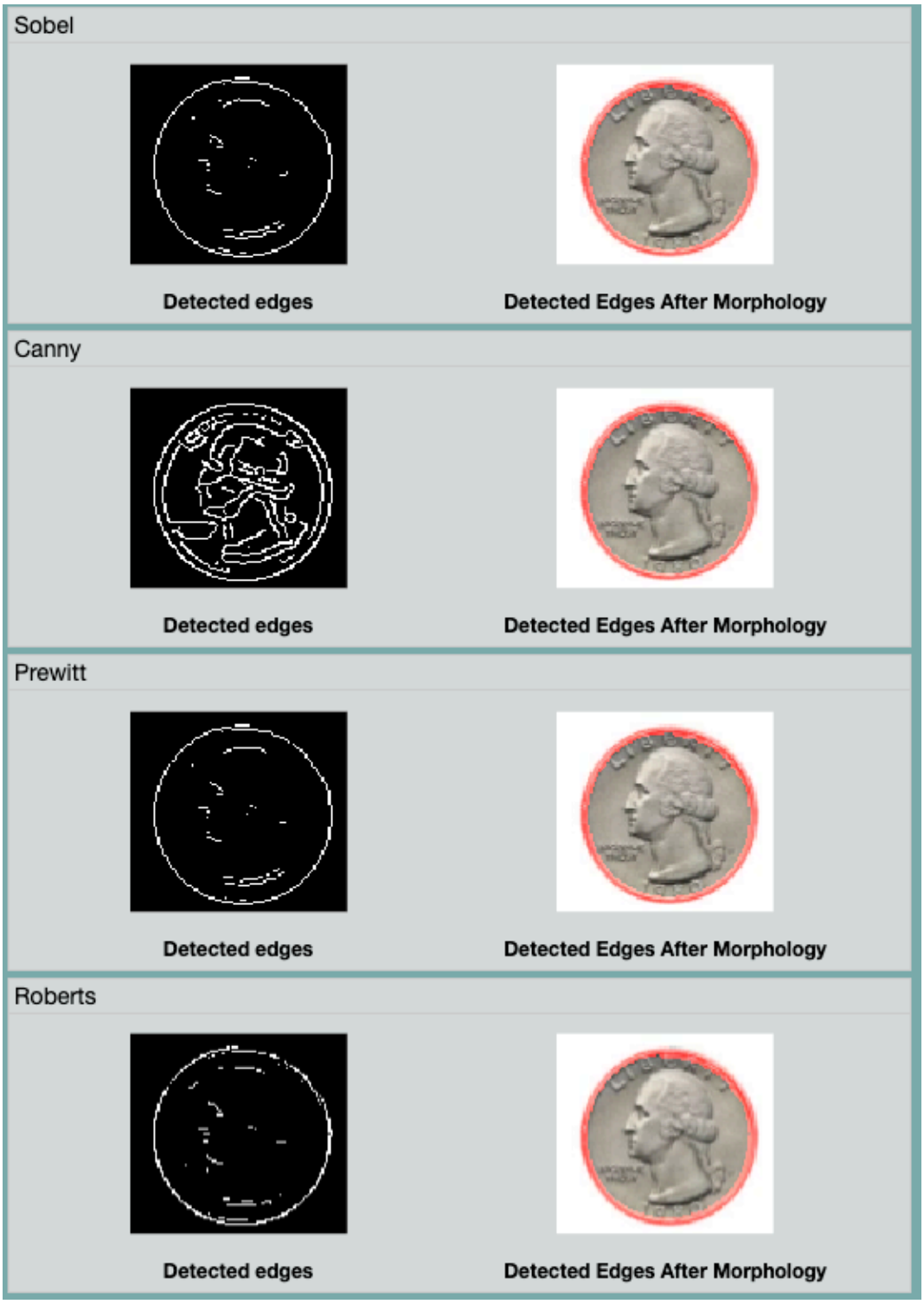

Figure 5. 12 Comparing edge detection algorithms Th = 0.150

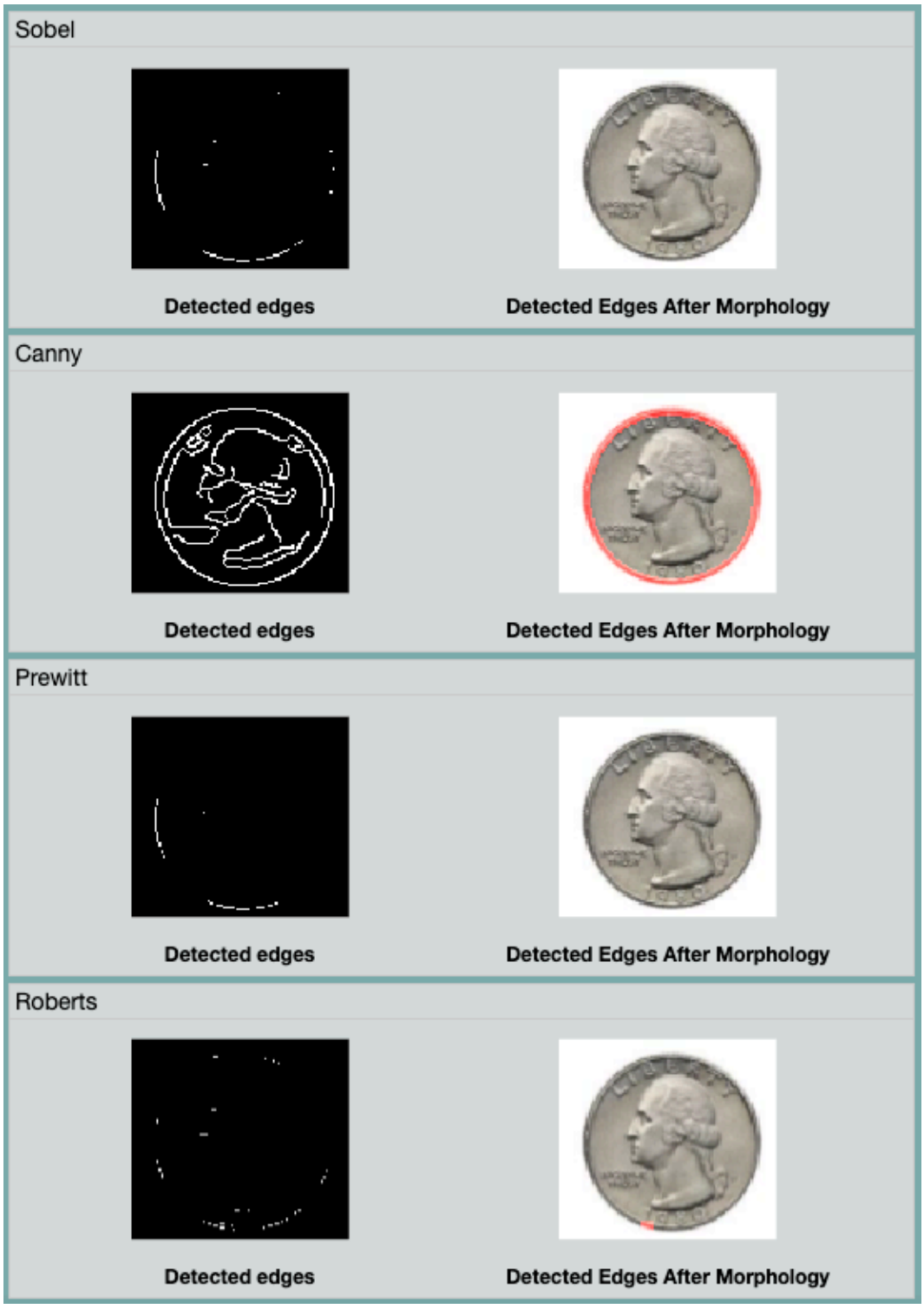

Figure 5. 13 Comparing edge detection algorithms and Morphology Th = 0.250

Noise

The noise in the image means the arbitrary of the brightness in the image constructing time. The noise happens when hands are shaking while taking a picture, during scanning, processing, transmission, etc. The original image loses part of its information because of the noise. To remove the noise, we have to know the type of noise and amount of the damage to find a solution to reconstruct it [33]. Noise creates unwanted effects in digital images, and visually it is not pleasant such as blurring the objects, artifacts that make small cells like pixels. Often noise deforms the background of the image or creates edges [33]. This section focuses on a few models of prevalent noises and techniques to remove them.

MATLAB has a function called imnoise( ) with that we can simulate some noises like Gaussian, Salt and pepper and Motion.

6.1.1 Gaussian

Gaussian is electronic noise and caused by discrete radiation of heat and terminal vibration of atoms [34]. In a digital communication system, 1 turns on the voltage, and 0 does not change it. There is an integrator that collects the energy when it comes in. The duration of transferring signals makes them weak, so we need an amplifier in the system to boost them. Also, we need a power supply, and turning it off and on makes the integrator wait to receive a signal. When the power is on, it heats up. Heat causes electrons to start to move in both amplifier and integrator and affects the voltage. This additive effect causes the Gaussian noise.

A result in mathematics says that if you collect all the moving electrons probability, the overall impact is the probability density function (PDF). Bell-shape graph is the standard form of

PDF is based on Gaussian noise model. Gaussian noise affects grayscale images; that's why its histogram graphed concerning gray value [33].

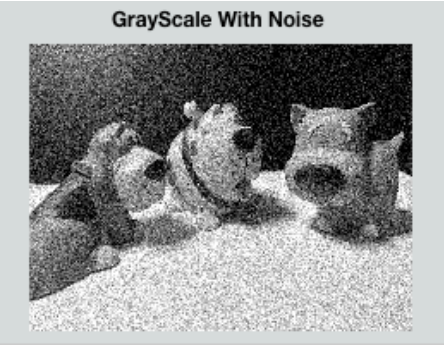

Figure 6. 1 A Grayscale image with Gaussian noise

The function to add Gaussian noise in MATLAB is:

imnoise (I, 'gaussian', m, var_gauss)

in this function, *I* is the grayscale file that you are going to apply the noise, '*gaussian*' is the noise type, *m* is the mean, and *var_gauss* is the variance between [0 1]. If we apply noise without any parameter, imnoise( ) by default uses m = 0 and var_gauss = 0.1 [35].

6.1.2 'Salt & pepper'

'Salt & pepper' noise occur when sudden changes happen in the image signals. Generally, signals damage when the camera's sensor has defected, software or transmission failure, and hardware damage while taking a picture. The effect on the image is black and white dots [20]. Salt noise adds random brightness to an image. The brightness is white noise, so the value of the pixels is 255. Pepper noise adds random dark to an image with a value of 0 [20].

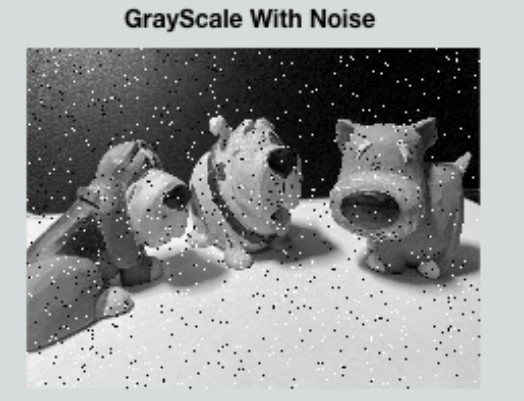

Figure 6. 2 Salt & pepper noise effect on a grayscale image

The function to add ‘salt & pepper’ noise in MATLAB is:

imnoise (I, ‘salt & pepper’, d)

in this function, *I* is the image file that you are going to apply the noise, ‘*salt & pepper*’ is the noise type, and *d* is the density. If we apply noise without any parameter, imnoise( ) by default uses d = 0.05. when we assign a value to *d*, it changes nearly *d*×numel(I) pixels [35].

Function *a* = numel(I) saves number of elements in matrix *I* into *a*. for example if *I* is 8×8 matrix numel(I) will return 64.

6.1 Remove Noise

6.1.3 Median

The median method uses to remove the 'salt & pepper' noise. Before applying the Median to a noisy image, we have to define a size for the filter window (n × m neighborhood). Then Median will start the process by sliding that over the matrix. It collects data in the filtered area, sorts them, and find the median. Next, it replaces the first matrix cell with the founded median value and will start the process on the neighbor cell [20]. Figure (6.3) is an example that shows the process of cleaning the noise.

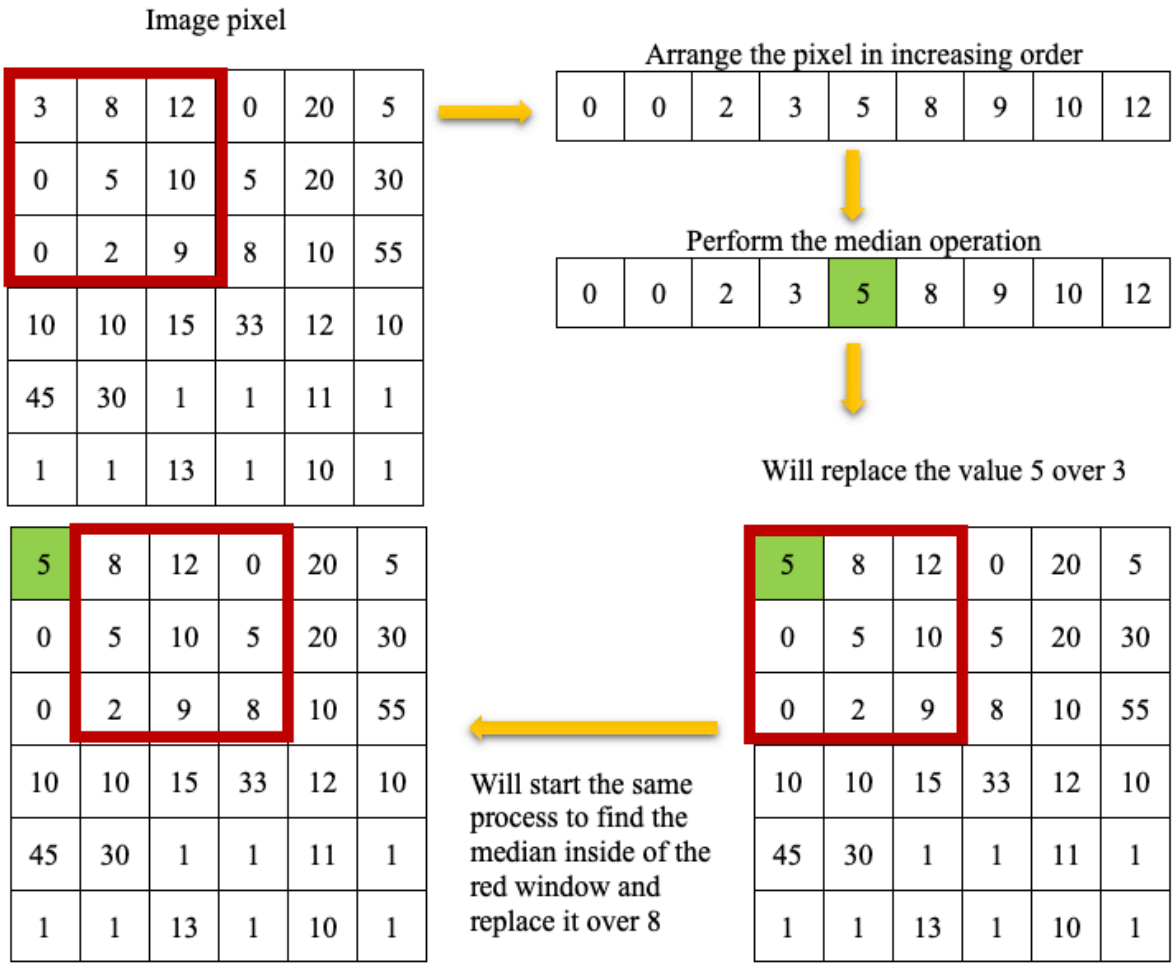

Figure 6. 3Median removing noise process

The function to remove 'salt and pepper' noise in MATLAB is:

medfilt2 (I, [m n])

in this function, *I* is the image file that we are going to remove the noise from it, the *m* and *n* are m-by-n filtering window size. If we apply Median without any parameter, medfilt2 ( ) by default will use 3 × 3 window size [36].

6.1.4 Wiener

Wiener's purpose is to remove noise, and its best use is for Gaussian noise. The other filters are designed to accept frequency to remove the noise; however, Wieners does the noise filtering differently. It works base on estimation and statically approach [37]. Given an example of seeing a blurred flower picture, we guess the image based on our previous knowledge. Wieners to filter a noisy signal, it compares the noisy signals with its desirable signals [37].

Wiener function in MATLAB creates an m-by-n window around neighbor cells to estimate the mean and standard deviation. Wiener adapts the local image variance, and If the area's variance is high, Wiener makes that part less smooth; otherwise, it will do the opposite. There are no designed takes for this function; it does the calculation based on estimating the entered noisy image [38].

The function to remove 'Gaussian' noise in MATLAB is:

Wiener2 (I, [m n] , noise)

in this function, *I* is the grayscale image that we are going to remove the noise from it, the *m* and *n* are m-by-n filtering window size [38].

6.2 SNR and PSNR

We have two images one is original and the second one the noisy one, PSNR compares the ratio of the signal-to-peak between two images. And if the number is higher the quality will be better. SNR is a signal-to-noise ratio. It computing the ratio of noise that added to the original image [20].

6.3 Noises in the project

In this project, the user uploads an image, and from the ‘Apply Noise’ panel, chose the type of noise. There are ‘salt & pepper’ and ‘Gaussian’ noise, and each one of them has its own parameters to enter, and noise will apply based on that. Figure (6.4) is an example of ‘salt and pepper noise’ with a density of 0.5.

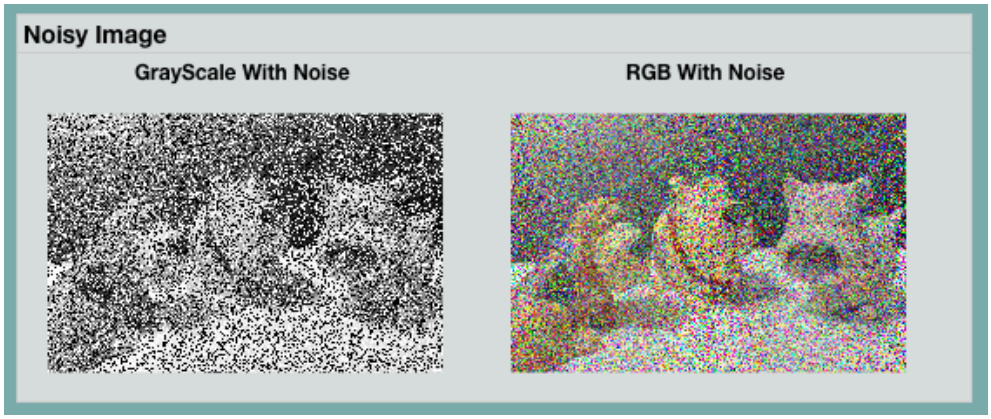

Figure 6. 4 RGB and Grayscale image effected by ‘salt & pepper’ noise d = 0.5

The filter applies to both RGB and Grayscale images. We are able to remove noises from any of them. Removing noise from the grayscale is simply followed by the MATLAB function medfilt2(). The challenging section is removing the noise from the RGB image. First had to get the dimension of the image, such as a row, column, and color band (amount of R, G, and B). then extract color channels and apply Median on each of them individually, at the end concatenate them together and display as a noiseless image.

To remove the Gaussian noise, I used Wiener. Users can apply different parameters the see the effect of the Gaussian noise and how Wiener works to remove it.

I designed a section to display information about SNR and PSNR using the MATLAB function:

[peaksnr, snr] = psnr ()

To find similarity between to original image and noise removed one I used MATLAB function:

ssim (Original_image, Filtered_image)

Figure 6. 5 Displaying filtered image, SNR, PSNR, and similarity

Future Work

Image processing is a vast topic, and my project is a small extraction from it. Due to time constraints, I was able to add the most useful and common features to the project.

In the image converting section plus RGB, grayscale, and binary, there are few other types of color images in future work; I will add them. I did not include those types in the project because they have a different pattern and use. Also, it can add some functions to work with high dynamic range images (HDR).

Morphological operations are limited to four flat Structuring elements in this application and would like to add the dimensional ones. Also, Morphology works on images that contain shapes, and in the future, I would like to add more functions to work with busy images.

I like to add filtering to this project to apply effects like smoothing them because, in image processing, these techniques are very useful. Like the Canny edge, detection uses a Gaussian filter to smooth the image and improve the quality of detection.

.

Conclusion

A digital image is a two dimensions file filled with values called pixels or picture elements. Digital image processing focuses on improving image information, and it plays a vital role in today's technology. You can see the application in medical, agricultural, robotic fields, etc. For instance, in the medical field, they use this technique to fix the taken images by X-ray or any other devices because the patient moves while taking the X-ray.

This thesis focuses on some aspects of image processing, and it is for CSUN students who are interested in work on image processing. For this purpose, I design an application in MATLAB. This thesis goal is to help students understand the image and the processes that can apply to it to change an image matrix for different purposes. Moreover, it is a general idea to display how image processing functions.

References

[1] M. A. Joshi, "Background and Applications," in *DIGITAL IMAGE PROCESSING, An Algorithmic Approach*, Delhi, India, PHI Learning Private Limited, 2018.

[2] G. Anbarjafari, "Digital Image Processing," University of Tartu, January 2014. [Online]. Available: https://sisu.ut.ee/imageprocessing/book/1. [Accessed Nov. 2020].

[3] I. Pitas, "Digital image processing fundamentals," in *Digital Image Processing Algorithms*, Cambridge, Pretice Hall, 2000, pp. 1-47.

[4] J. S. Sidhu and A. Singh, "Super Resolution Applications in Modern Digital Image Processing," *International Journal of Computer Applications (0975 – 8887),* vol. 150, no. 2, p. 8, 2016.

[5] MathWorks, "What is Matlab?," The MathWorks, Inc, 1994-2020. [Online]. Available: https://www.mathworks.com/discovery/what-is-matlab.html. [Accessed September 2020].

[6] C. Moler, "Cleve's Corner: Cleve Moler on Mathematics and Computing," 23 January 2018. [Online]. Available: https://blogs.mathworks.com/cleve/2018/01/23/linpack-linear-equation-package/. [Accessed November 2020].

[7] R. Schreiber, "MATLAB," 2007. [Online]. Available: http://www.scholarpedia.org/article/MATLAB. [Accessed November 2020].

[8] "VAREX Imaging," Varex Imaging Corporation, 2019. [Online]. Available: https://www.vareximaging.com/. [Accessed October 2020].

[9] M. Bhat, "Digital Images Processing," *International Journal Of Scientific & Technology,* vol. 3, no. 1, pp. 272-276, 16 October 2014.

[10] C. Solomon and T. Breckon, "Representation," in *in Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab*, Sussex, John Wiley & Sons, Ltd., 2011, pp. 1-14.

[11] K. Sheets, "Learn ImageJ," 2013. [Online]. Available: https://sites.google.com/site/learnimagej/image-processing/what-is-a-digital-image. [Accessed November 2020].

[12] C. Solomon and T. Breckon, "Pixels," in *in Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab*, Sussex, John Wiley & Sons, Ltd., 2011, pp. 49-105.

[13] MathWorks, "Image," The MathWorks, Inc., 1994-2020. [Online]. Available: https://www.mathworks.com/help/matlab/ref/image.html. [Accessed September 2020].

[14] K. W. Watson, "All About Digital Photo," 2003-2020. [Online]. Available: www.rideau-info.com/photos/whatis.html. [Accessed October 2020].

[15] J. d. S. Ignacio, S. J. Buso and W. A. Monteiro, "Processing And Analysis Of Digital Images: How To Ensure The Quality Of Data Captured?," *International Journal of Recent advances in Mechanical Engineering (IJMECH),* vol. 2, no. 2, pp. 1-10, 10 June 2013.

[16] A. Hanbury, "The Taming of the Hue, Saturation and Brightness Colour Space," [Online]. Available: http://www.x-infinity.com/files/xm/HSY.pdf. [Accessed October 2020].

[17] "learn.," Digital Art, Design, and Communication Education, [Online]. Available: http://learn.leighcotnoir.com/artspeak/elements-color/hue-value-saturation/. [Accessed October 2020].

[18] S. Bradley, "The Fundamentals of Color: Hue, Saturation, And Lightness," Vanseo Design, 20 May 2013. [Online]. Available: https://vanseodesign.com/web-design/hue-saturation-and-lightness/. [Accessed October 2020].

[19] S. W. Smith, "Image Formation & Display," in *The Scientist and Engineer's Guide to Digital Signal Processing*, California Technical Publishing, 1997, pp. 373-390.

[20] J. Al-Azzeh, B. Zahran and Z. Alqadi, "Salt and Pepper Noise: Effects and Removal," *International Journal on Electrical Engineering and Informatics,* vol. 2, no. 4, 2018.

[21] MathWork, "Image Types in the Toolbox," The MathWorks, Inc., 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/image-types-in-the-toolbox.html?s_tid=srchtitle. [Accessed September 2020].

[22] GeeksforGeeks, "MATLAB | RGB image to grayscale image conversion," [Online]. Available: https://www.geeksforgeeks.org/matlab-rgb-image-to-grayscale-image-conversion/?ref=rp. [Accessed October 2020].

[23] MathWorks, "im2bw," The MathWorks, Inc., 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/ref/im2bw.html. [Accessed September 2020].

[24] MathWorks, "strel," The MathWorks, Inc., 1994-2005. [Online]. Available: http://matlab.izmiran.ru/help/toolbox/images/strel.html. [Accessed November 2020].

[25] R. Srisha and A. Khan, "Morphological Operations for Image Processing : Understanding and its Applications," in *National Conference on VLSI, Signal processing & Communications* , 2013.

[26] MathWorks, "Types of Morphological Operations," The MathWorks, Inc., 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html. [Accessed November 2020].

[27] C. Solomon and T. Breckon, "Morphological processing," in *in Fundamentals of Digital Image Processing A Practical Approach with Examples in Matlab*, UK, John Wiley & Sons, Ltd., 2011, pp. 197-132.

[28] MathWorks, "Edge Detection," The MathWorks, Inc, 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/edge-detection.html. [Accessed November 2020].

[29] M. Joshi and A. Vyas, "Comparison of Canny edge detector with Sobel and Prewitt edge detector using different image formats," *International Journal of Engineering Research & Technology(IJERT),* vol. 2, no. 3, 2014.

[30] A. Sears-Collins, "How the Sobel Operator Works," Automatic Addison, 17 December 2019. [Online]. Available: https://automaticaddison.com/how-the-sobel-operator-works/. [Accessed November 2020].

[31] O. R. Vincent and O. Folorunso, "A Descriptive Algorithm for Sobel Image Edge Detection," in *Proceedings of Informing Science & IT Education Conference (InSITE)*, 2009.

[32] A. Jose, D. M. Dixon K, N. Joseph, S. George E and A. V, "Performance Study of Edge Detection Operators," in *International Conference on Embedded System*, 2014.

[33] A. K. Boyat and B. K. Joshi, "A Review Paper: Noise Models In Digital Image Processing," *Signal & Image Processing,* vol. 6, no. 2, pp. 63-75, 2015.

[34] A. M. Abd-Alsalam Selami and A. F. Fadhil, "A Study of the Effects of Gaussian Noise on Image Features," *Kirkuk University Journal /Scientific Studies,* vol. 11, no. 3, pp. 152-169, 2016.

[35] MathWork, "imnoise," The MathWorks, Inc., 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/ref/imnoise.html#d122e142496. [Accessed November 2020].

[36] MathWorks, "medfilt2," The MathWorks, Inc., 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/ref/medfilt2.html. [Accessed November 2020].

[37] P. Patidar and S. Srivastava, "Image De-noising by Various Filters for Different Noise," *International Journal of Computer Applications,* vol. 9, no. 4, pp. 45-50, 2010.

[38] MathWorks, "wiener2," The MathWorks, Inc, 1994-2020. [Online]. Available: https://www.mathworks.com/help/images/ref/wiener2.html. [Accessed November 2020].