DIGITAL IMAGE PROCESSING

ASSIGNMENT 1

Image Histogram, Image Negative, and Image Treshold

Muhammad Sabili Nurilhaq 20081010101

1. Image Histogram

The histogram of an image is graphical representation in the form of a diagram to display the frequency of color intensity of an image. Using histogram we can find out the intensity value of each pixel, which contained in an image. In matlab, we can visualized histogram of an image using “imhist” build-in function on image processing toolbox. However, that function is not allowed to be used in this assignment, so we must use our own algorithm to visualize the histogram of an image. Here are the steps to visualize the histogram of an image on the matlab.

1.1. Read Image

```
img=imread('original.jpg');
```

First of all, we need to read an image using imread function, whose parameter accepts the directive path of the image file to the current matlab working files folder. The code above is the implementation of imread function. In the code, the imread function accepts the image file name only because the image to be used is in the same folder as the current matlab working folder.

Figure 1.1. RGB Image Sample

1.2. Retrieve Rows and Columns of The Image

```
[rows, columns, numberOfColorChannels] = size(img);
```

The above code is used to retrieve the length of rows and columns, which is used in the color intensity frequency generation operation.

1.3. Divide Image Into Each Layer

```
redFormatImage=img(:,:,1);
greenFormatImage=img(:,:,2);
blueFormatImage=img(:,:,3);
```

Images that have been read need to be splited to get each layer. Each layer that has been separated will produce an image with grayscale type which can bee seen in the image below.

Figure 1.2. Red Layer Image

Figure 1.3. Green Layer Image

Figure 1.4. Blue Layer Image

1.4. Create Container

```
xAxis = 1:255;
histRed = zeros(1,256);
histGreen = zeros(1,256);
histBlue = zeros(1, 256);
```

xLevelValue variable is used to be a x-axis containing a range of numbers from 1 to 256. Whereas histRed, histGreen, histBlue variable is used to be a container to store the amount of intensity.

1.5. Operation to Create Color Intensity Frequency

```
for i=1:rows
  for j=1:columns
    histRed(redFormatImage(i,j) + 1) = histRed(redFormatImage(i,j) + 1) + 1;
    histGreen(greenFormatImage(i,j) + 1) = histGreen(greenFormatImage(i,j) + 1) + 1;
    histBlue(blueFormatImage(i,j) + 1) = histBlue(blueFormatImage(i,j) + 1) + 1;
  end
end
```

In this section, writer use his own algorithm to generate color intensity of the image. First loop is used as a row specifier, while the second loop as a column specifier. Each container use the pixel value of each image layer on its indexes which value of each container on its indexes will raised by one.

1.6. Visualization Frequency

After the intensity stored in the each layer container, we can vizualize it into diagram form.

```
figure('name','Red Layer Image Histogram');
tiledlayout(2,1);
```

In order to vizualize it, we need to create figure and layout of its figure first. In the code above we defined figure with name of each layer and layouts with size of 2x1.

```
%top plot
tiled1 = nexttile;
bar(xAxis, histRed);
xlabel(tiled1,'Grey Level');
ylabel(tiled1, 'Numbers of Color');
title(tiled1,"Manual Version Histogram");
```

After defined size of layout, we need to determine the location of the histogram in the layout that has been defined, so that we can modify plot's property like x-axis label, y-axis label, and title of the plot. In this layout we fill it with a bar chart which it is filled with xAxis as x-axis and histRed as y-axis.

```
%bottom plot
tiled2 = nexttile;
bar(xAxis, imhist(redFormatImage(:)));
xlabel(tiled2,'Grey Level');
ylabel(tiled2, 'Numbers of Color');
title(tiled2, "Matlab Version Histogram (imhist function)");
```

We create a new histogram again in the next layout, but using predefined function of the image processing toolbox. This histogram is made for comparison with manual histogram generation. This process is also done for each layer so that it can produce 3 figures output.

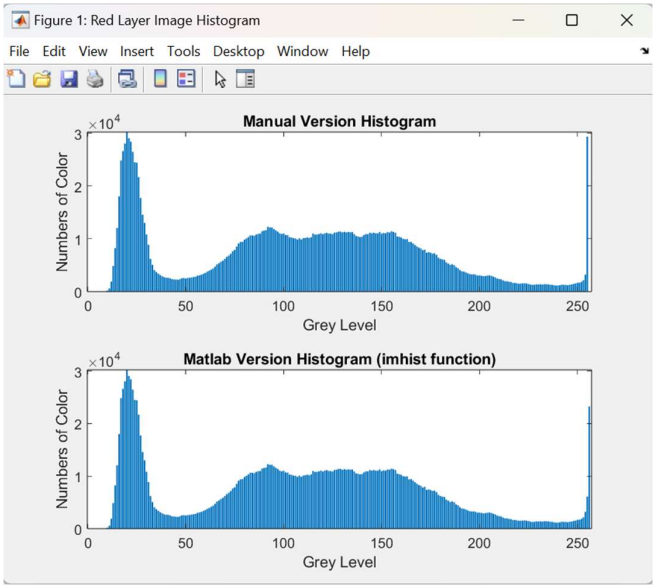

Figure 1.5. Histogram Results (Red Layer)

2. Image Negative

A negative images is an image that results from inverting the value of each pixel. Formula for inverting the value of each pixel is (255 – pixelValue). So that, in matlab we can implemented it to produce the result of negative image. Here are the step to implemented it in matlab.

2.1. Read Image

```
img=imread("original.jpg");
```

Same as the previous section, we read image into a variable using imread function with the same image sample which can be seen in figure 1.1.

2.2. Retrieve Rows, Columns, and Color Channels

```
[rows, columns, numberOfColorChannels] = size(img);
```

Same as the previous section, we retrieve rows, columns, and channels of the image using 'size' function.

2.3. Create Image Result Container

```
negativeImageResult=uint8(zeros(rows, columns, numberOfColorChannels));
```

In order for it to be displayed as an image we need to create container with uint8 type. The container contains a 3-dimensional array the same size as the img variable, but its contained zero value inside.

2.4. Inverting Process

```
for row=1:rows
    for col=1:columns
        for chan=1:numberOfColorChannels
            negativeImageResult(row, col, chan)=255 - img(row, col, chan);
        end
    end
end
```

Many consecutive first, second, and third iterations correspond to the length of rows, columns, and numberOf Channels. On each iteration, it will change value each pixel into negative form (previously described formula) to store it in the container.

2.5. Displaying The Image Result

```
imshow(negativeImageResult);
```

So that, we can display it using the imshow predefined function whose results can be seen in figure 2.1.

Figure 2.1. Negative Image Results

3. Image Treshold

Image tresholding is a technique for image segmentation that convert image into a binary image. Pixel's value will be changed into 0 (black) or 255 (white) depending on the location of the pixel value against treshold. E.g. we defined treshold equal to 100, so that if the pixel value greater than 100, pixel value will be changed to 255 (white). Meanwhile, if the pixel less than equal 100, pixel value will be changed to 0 (black). We can implemented it in matlab, then this section we defined the treshold is equal to 80. Here are the step to implement it in matlab.

3.1. Read Image

```
imgReader=imread("original.jpg");
```

Same as the previous section, we read image into a variable using imread function with the same image sample which can be seen in figure 1.1.

3.2. Convert Image

Image that has been retrieved will be converted into greyscale type. In order to converting the image we can use predefined function from matlab in the form of "rgb2gray" as can be seen in the code below.

```
greyscaleImg=rgb2gray(imgReader);
```

The results of the image conversion to grayscale will be displayed in the program output as a comparison with the tresholding image.

3.3. Retrieve Rows and Columns

The grayscale image contains only a 2-dimensional array, so we can only retrieve rows and columns without channels.

```
[rows, columns]=size(greyscaleImg);
```

3.4. Create Container and Treshold Constant

```
tresholdImageContainer=uint8(zeros(rows, columns));
tresholdConst=80;
```

Then, we must also defined container and the treshold constant in order to be used in the operation process.

3.5. Operation of Change Pixel

```
for row=1:rows
    for col=1:columns
        pxlValue=greyscaleImg(row, col);
        if pxlValue>tresholdConst
            tresholdImageContainer(row, col)=255;
        else
            tresholdImageContainer(row, col)=0;
        end
    end
end
```

We can used predefined variables to looping operation. First loop works as a row iteration and the second loop also works as a columns iteration. Each iteration, the pixel value will be checked whether it is greater than 100 or not. So that, we can change it according to the results of checking.

3.6. Displaying The Results of Image

To display results image, we have to defined a figure and layout and in this case the layout used is 1x2.

```
figure("Name", "Image Tresholding Results");
tiledlayout(1,2);
```

Then, we must determine what images will be displayed on the first and second layout. In this case, first layout will be displaying Grayscale Image which is results from conversion. We also give title into first layout which can be seen in the code below.

```
tiled1=nexttile;
imshow(greyscaleImg);
title(tiled1, "Greyscale Image");
```

In the second layout, we will displaying results of the tresholding image which can be seen in the code below.

```
tiled2=nexttile;
imshow(tresholdImageContainer);
title(tiled2, "Image Results (Treshold = "+tresholdConst+")");
```

Result of them can bee seen in the Figure 3.1.

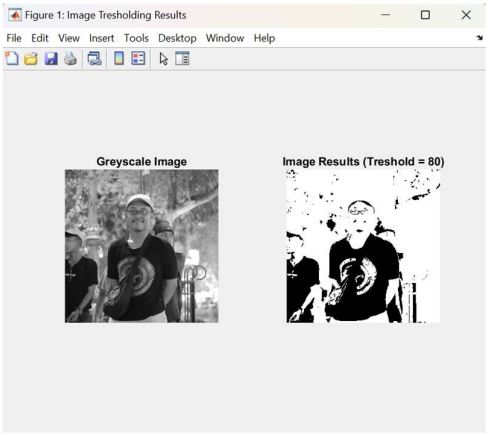

Figure 3.1. Results of Image Tresholding