Laporan Pengolahan Citra Digital

Image Histogram, Image Negatives, Image Threshold

Fahrul Firmansyah 20081010099

1. Image Histogram

Histogram is one of data visualization that is commonly used to visualize numerical data distribution. Using histogram we can know the pixel value distribution in our gray image. These are the step to visualize pixel value distribution in matlab:

a. Read the Image

```
rgb_image = imread("sample.png");
```

The code above works as an image reader and saves the image as an array with 3 dimensional shape.

Gambar 1. RGB Image

b. Convert Into Grayscale Format

```
gray_image = rgb2gray(rgb_image);
```

Because the image that has been read is in RGB format, we need to convert that to grayscale format. The code above works as a converter that converts our RGB image to gray scale Image.

Gambar 2. Grayscale Image

c. Create Container Variable

```
list = zeros(1, 256);
```

The container variable will save the number of pixel values that start from 0 to 255. Unfortunately, the matlab index starts from 1 and we need to add 1 space more in the container, so the array size becomes 256 and the initial value for each space is 0.

d. Get Gray Image Size

```
[length, width] = size(gray_image);
```

The size of the image will work as a boundary in our for loop that is used to count the number of the pixel. Size() function will return an array that has 2 items that are length and width of image.

e. Count The Pixel Value

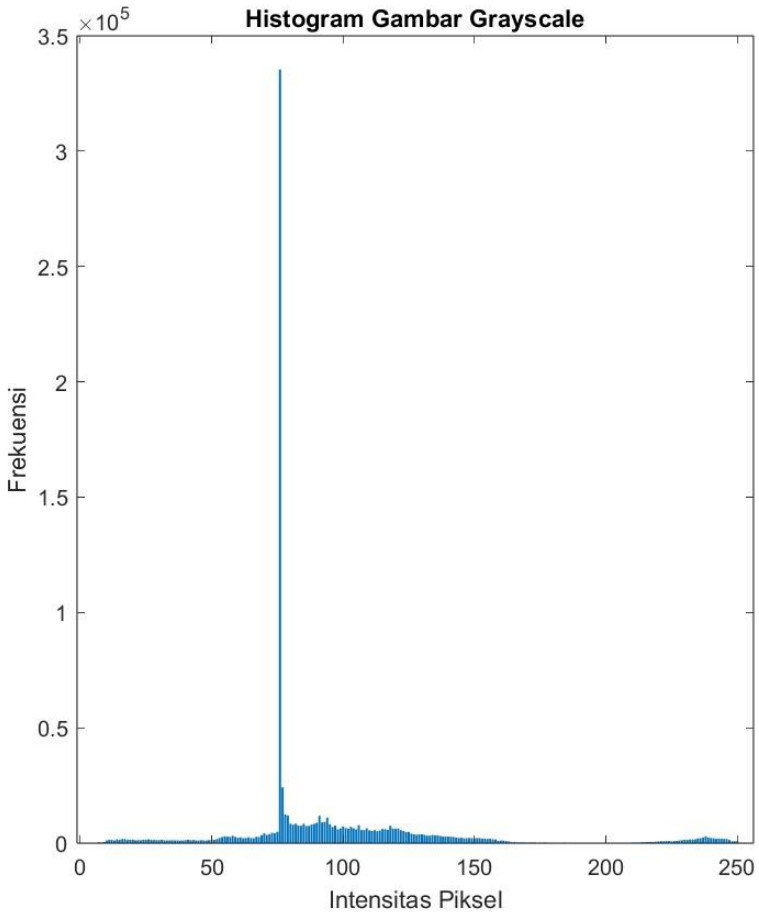
```
for i = 1:length
    for j = 1:width
        intensity = gray_image(i, j);
        list(intensity + 1) = list(intensity + 1) + 1;
    end
end
```

This is the main part of visualizing pixel value distribution, there are 2 loops that are used. First loop works as the indexer in the length image, and the second loop works as the indexer in the width image. Within the second loop, the intensity variable will take the pixel value of image in i & j position. After that, we add 1 to the list value that has an index equal with the intensity.

f. Plot the Image

```
bar(0:255, list);
xlabel('Intensitas Piksel');
ylabel('Frekuensi');
title('Histogram Gambar Grayscale');
```

Finally we can visualize the pixel value distribution that we have saved in the list variable.



Gambar 3. Pixel Value Distribution

2. Image Negative

A negative image, also known as an inverse image or inverted image, is an image where the colors or brightness levels are inverted relative to a standard positive image. In a negative image, the dark areas of the original image appear bright, and the bright areas appear dark. This reversal of colors or intensities creates a striking and often artistic effect. These are the step to inverse image in matlab:

a. Read the Image

```
rgb_image = imread("sample.png");
```

The code above works as an image reader and saves the image as an array with 3 dimensional shape.

Gambar 4. RGB Image

b. Get Image Size

```
[length, width, channels] = size(image);
```

The size of the image will work as a boundary in our for loop that is used to count the number of the pixel. Size() function will return an array that has 3 items that are length, width, and channel of the image.

c. Create Container Variable

```
negative_image = zeros(length, width, channels);
```

The container variable will save the new pixel of the negative image. This container variable shape is equal with the original image.

d. Inverse the Image

```
for i = 1:length
    for j = 1:width
        for k = 1:channels
            intensity = image(i, j, k);
            negative_intensity = 255 - intensity;
            negative_image(i, j, k) = negative_intensity;
        end
    end
end
```

This is the main part of inverting an image, there are 3 loops used. First loop works as the indexer in the length image, the second loop works as the indexer in the width image, the third loop works as the indexer in the channel image. Within the third loop, the intensity variable will take the pixel value of the image in i,j, & k position. After that, we add the pixel value - 255 to the container variable in the same position (i,j,k).

e. Show the Negative Image

```
imshow(uint8(negative_image));
```

Since the range value of the pixel is 0 - 255, we need to cast the negative image variable to unsigned int.

Gambar 5. Negative Image

3. Image Threshold

Image threshold is a technique in image processing to segment an image into an object of interest by dividing it into few groups based on pixel intensity values. These are the step to thresholding image in matlab:

a. Read the Image

```
rgb_image = imread("sample.png");
```

The code above works as an image reader and saves the image as an array with 3 dimensional shape.

Gambar 6. RGB Image

b. Convert Into Grayscale Format

```
gray_image = rgb2gray(rgb_image);
```

Because the image that has been read is in RGB format, we need to convert that to grayscale format. The code above works as a converter that converts our RGB image to gray scale Image.

Gambar 7. Grayscale Image

c. Get Gray Image Size

```
[length, width] = size(gray_image);
```

The size of the image will work as a boundary in our for loop that is used to count the number of the pixel. Size() function will return an array that has 2 items that are length and width of image.

d. Create Container Variable

```
threshold_result = zeros(rows, cols);
```

The container variable will save the new pixel of the threshold image. This container variable shape is equal with the original image.

e. Threshold the Pixel Intensity Value

```
for i = 1:rows
    for j = 1:cols
        intensity = gray_image(i, j);
        if intensity <  75
            threshold_result(i, j) = 0;
        else
            threshold_result(i, j) = 255;
        end
    end
end
```

This is the main part of the thresholding image, there are 2 loops that are used. First loop works as the indexer in the length image, and the second loop works as

the indexer in the width image. Within the second loop, the intensity variable will take the pixel value of image in i & j position. After that, we determine the value will change into totally dark (0) or totally bright (255) based on the threshold that we choose. Finally, the result will be saved in the container variable.

f. Show the Threshold Image

```
imshow(threshold_result);
```

Finally we can plot the threshold result that we have saved in the threshold result variable.

Gambar 8. Threshold Image