MODUL 6 – Histogram, Histogram Equalization, Dithering

A. PURPOSE

1. Students understand the image histogram and apply them in a Python program
2. Students understand the histogram equalization and apply them in a Python program
3. Students understand the dithering image and apply it in a Python program

B. Tools and Materials

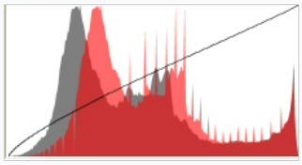
1. PC/LAPTOP
2. Github
3. *Google Colaborator*

1. THEORY REVIEW

C.1 Histogram Image

Image Histogram is a type of histogram that serves as a graphical representation of the distribution of pixel color intensity in digital images. The histogram graph as shown in Figure 1 will display the number of pixels of each pixel color intensity. By looking at the histogram graph, one can quickly assess the pixel color distribution. For color images the histogram represents each available color channel, while for greyscale images it only consists of one histogram. The horizontal axis of the histogram graph represents the variation in pixel values, while the vertical axis represents the number of pixels. The left side of the horizontal axis represents black and dark areas, the center represents gray areas, and the right represents light and white areas.

Sunflower image



Histogram of sunflower image

Figure 1. Example of the Original Image and the Image Histogram

The following is an algorithm for making a histogram from a digital image

1	**Prepare two forms, form 1 contains 1 chart used to display grayscale image histograms, form 2 contains 3 charts used to display R, G, and B histogram images.**
2	**Create a histogram storage variable R, G, and B**

	Variables can be stored in a 2-dimensional array or using a dictionary in Visual Studio. The data type for the horizontal axis can use int or Byte, while on the vertical axis, it is recommended to use the Double data type. This is done because we will normalize the histogram values. In this experiment, our histogram data is normalized because then the image repair process will be carried out using a histogram. We will do an image repair experiment using a histogram in the next jobsheet.
3	**Create a variable of type Bitmap to store the input image value**
4	**Give the initial value for each pixel intensity with a value of 0**
5	**For each image row and column, check the pixel value and add 1 for each examined pixel value**
6	**Check the value of the dictionary, if the histogram of R is exactly the same as G.** // Show form 1
7	**Save the histogram values from the dictionary into a *key* variable *of* type Data List** **Show form 1**
8	**For each key value**
9	**Normalize histogram values**
10	**Add histogram data to the chart**
11	**done**
12	**Otherwise the same between R and G.**
13	// Show form 2 **Do the same steps as when displaying form 1**
14	**done**
15	

C.2 *Histogram Equalization*

Histogram Equalization is an equation that is implemented in a *digital* image in which the distribution of the histogram will be more spread out, in this case, although it cannot be proven that the histogram shape will be uniform, with the *Histogram Equalization it* is certain that the histogram will be more even. The histogram smoothing is obtained by changing the degree of grayness of a pixel (r) with the new degree of gray (s) with a transformation function (T) (Gonzalez & Woods, 2002). *Histogram Equalization is* done by making a transform function from the normalized histogram image. Here is the equation for the transform function:

$$G(z_q) = (L-1)\sum_{1=0}^{q} p_z(z_i)$$

Manual calculation for the *Histogram Equalization* this time uses a 3-bit *image* (L = 8) with a size of 64x64 pixels (MN = 4096) and the intensity distribution will be shown in Table 1, where the intensity level is an integer value at a distance [0, L - 1] = [0, 7].

Figure 2 shows the process of improving digital images using the *Histogram Equalization* .

Figure 2. Flowchart of the path of digital image improvement with HE

In Figure 3 shows before the histogram equalization process chart. Figure 4 shows the results of the transform function. Figure 5 shows the histogram of the calculation results using the HE method. The calculation of the *Histogram Equalization* uses the formula:

$$s_0 = T(r_0) = 7\sum_{j=0}^{1} p_r(r_j) = 7\ p_r\ (r_0) = 1.33$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7\ p_r\ (r_0) + \ 7\ p_r\ (r_1) = 3.08$$

dan $s_2 = 4.55,\ s_3 = 5.67,\ s_4 = 6.23,\ s_5 = 6.65,\ s_6 = 6.86,\ s_7 = 7.00$

Table 1 Distribution of intensity and value histogram for 3-bit *image* , 64 x 64 *digital image*

r_k	n_k	$\mathbf{p_r(r_k) = n_{k/MN}}$
$r_{0=0}$	790	0.19
$r_{1=1}$	1023	0.25
$r_{2=2}$	850	0.21
$r_{3=3}$	656	0.16
$r_{4=4}$	329	0.08
$r_{5=5}$	245	0.06
$r_{6=6}$	122	0.03
$r_{7=7}$	81	0.02

Where n_k = k pixel value, and n is the total number of pixels in an image. Another formula that can be used to calculate the *equalization histogram* is as follows:

$$K_o = round\left(\frac{C_i.(2^k - 1)}{w.h}\right)$$

Dimana :

C_i = Cumulative distribution of pixel values

round = Closest rounding function

K_o = The value of the pixel color intensity of the *Histogram Equalization* result

w = image width

h = image height

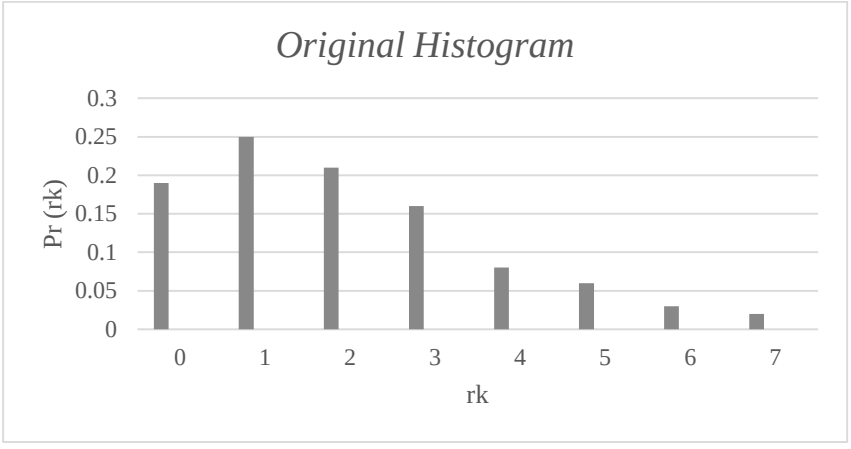


Figure 3. Original histogram before equalization process

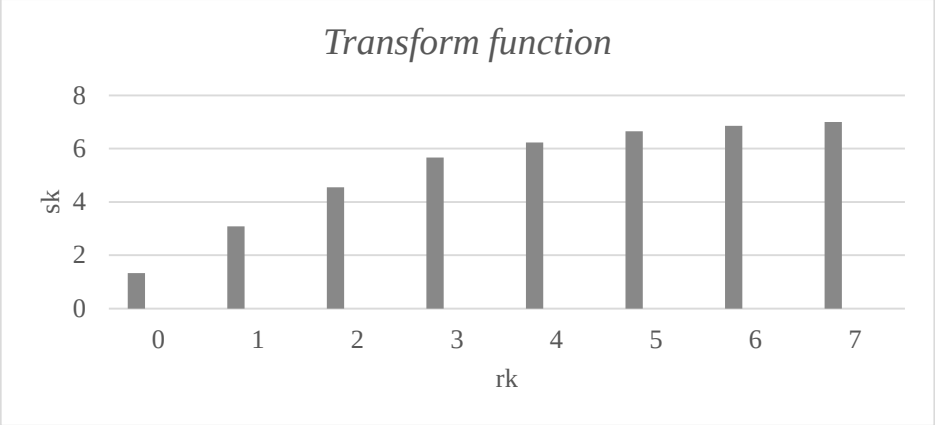


Figure 4. Transform function

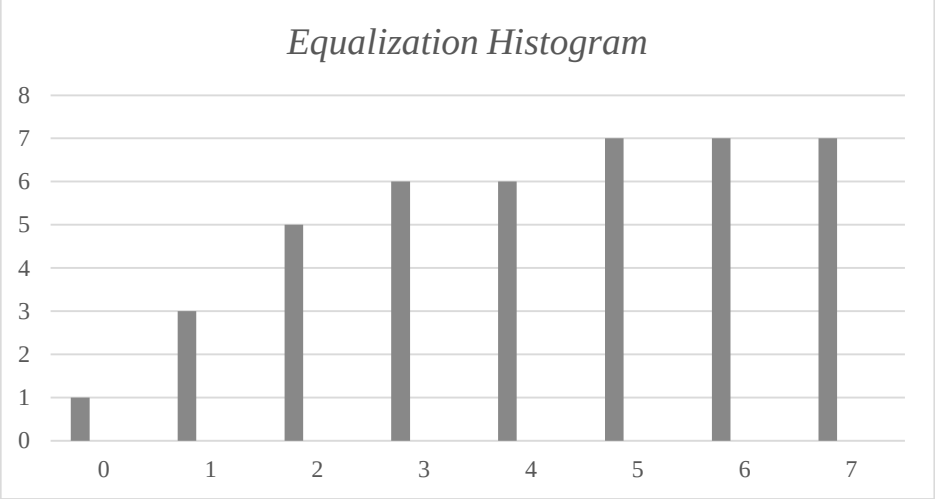


Figure 5. *The calculated* histogram *using the* Equalization Histogram *method*

At this point, s has a fractional value so it is rounded up to the nearest integer value.

s_0 = 1.33 → 1

s_1 = 3.08 → 3

s_2 = 4.55 → 5

s_3 = 5.67 → 6

s_4 = 6.23 → 6

s_5 = 6.65 → 7

6 = 6.86 → 7

s_7 = 7.00 → 7

C.3 Image Dithering (*Error Difussion*)

Error Diffusion is a type of halftoning where the quantization process is distributed to neighboring unprocessed pixels. Its main use is to change the image depth to a smaller one, but the image quality is not that bad. This process is usually used in printing technology, because the number of ink colors is not too much so that the illusion of color is formed from this technique.

Diffusion error is classified as an operation area, because what is done by the diffusion error on one pixel will affect the calculation results of other pixel operations. Error diffusion also has the ability to reinforce the edges of the image. This can make the text in the image more legible than other halftoning techniques.

Some of the Error Diffusion methods include:

1. Floyd and Steinberg

Floyd and Steinberg describe a system for generating error diffusion in digital images using a simple kernel:

$$\frac{1}{16}\begin{bmatrix} - & \# & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

Where - indicates the pixels that have been processed, # shows the pixels that are currently being processed.

Or in the form of a diagram node indicated by G amber 6 below:

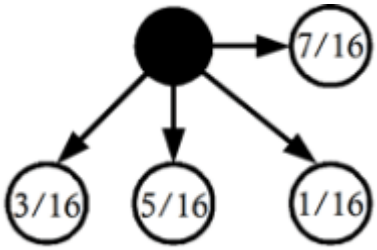


Figure 6. Floyd and Steinberg's Node Dithering Diagram

Floyd and Steinberg's dithering formula has the following equation :

The first formula (R1) → $I_{acc}(i+1,j) = I_{acc}(i+1,j) + \frac{7}{16}e$

The first formula (R2) → $I_{acc}(i+1,j+1) = I_{acc}(i+1,j+1) + \frac{3}{16}e$

The first formula (R3) → $I_{acc}(i,j+1) = I_{acc}(i,j+1) + \frac{5}{16}e$

The first formula (R4) → $I_{acc}(i-1,j+1) = I_{acc}(i-1,j+1) + \frac{1}{16}e$

Where as :

I_{acc} = Pixel target(pixel being processed)
i = row
j = column
e = error

For example, for an image measuring 3 x 3 pixels, dithering is carried out in accordance with the formula steps in accordance with the position to be calculated as shown in Figure 7 below.

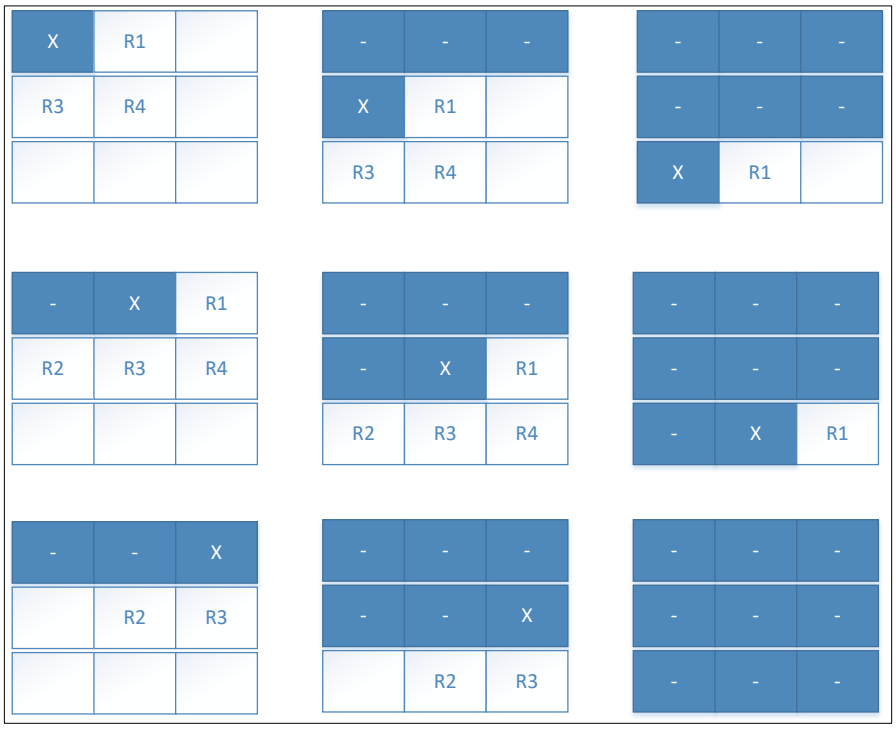


Figure 7. Implementation position of the Dithering Floyd and Steinberg formula

2. Jarvis, Judice dan Ninke dari Bell Labs Jarvis, Judice and Ninke of Bell Labs describe a similar system which they term "minimized average error", using a larger kernel size:

$$\frac{1}{48}\begin{bmatrix} - & - & \# & 7 & 5 \\ 3 & 5 & 7 & 5 & 3 \\ 1 & 3 & 5 & 3 & 1 \end{bmatrix}$$

If you want to simplify the number of colors without diffusion error, then the easiest way is to find the closest color. If the input color is closer to red, then the color will be changed to red. If it is closer to cyan, then the color is changed to cyan, and so on. The results are shown in the picture in the nearest color subsection.

Look at the following 8 images before and after simplified into 8 colors.

Figure 8. Image Color Simplification to 8 Colors

Here's another example:

Figure 9. Another example of simplifying image colors to 8 colors

From the image, the upper left corner is taken and then enlarged as follows:

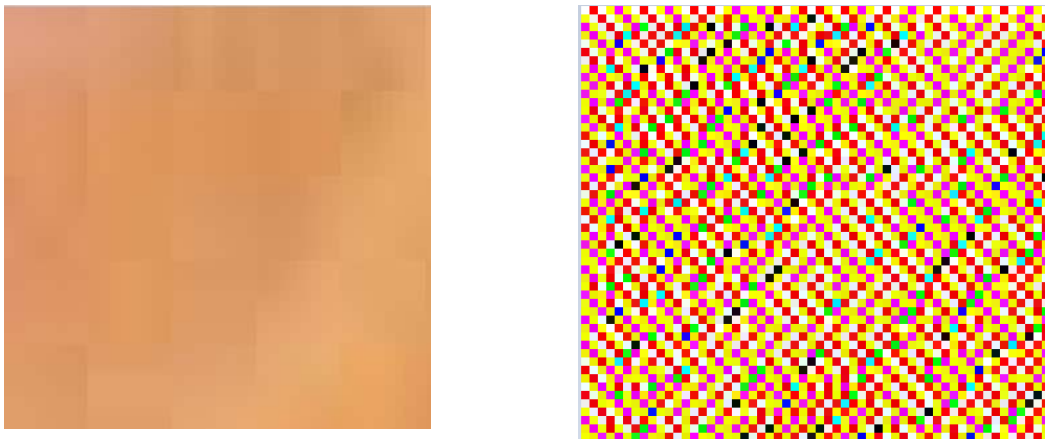


Figure 10. Zoom Results from Simplifying Images to 8 Colors

Error diffusion works by comparing a pixel's original color to the closest specified color and calculating the difference. This difference is called an error. This portion of the error is then distributed to neighboring pixels causing the error to diffuse and is called "Error Diffusion".

The simplest form of error diffusion can be seen in the following figure:

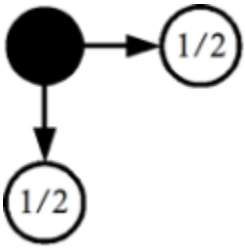


Figure 10. Simple Form of Error Diffusion

With the error diffusion form as above, half the error of the processed pixel (indicated by the black dot) is diffused half to the right pixel and half to the lower pixel. For color diffusion errors, this process must be done on all red, green, and blue channels.

The important part to underline is that the total number of errors diffused should not exceed one. Another important thing to note is that when the error portion diffuses to the neighbor, the neighbor has a pixel value between 0 - 255. If the value is outside 0 - 255, then the final value needs to be truncated.

Here is the pseudo-code for a simple diffusion error that must be performed after a pixel (x, y) has been converted to its nearest color:

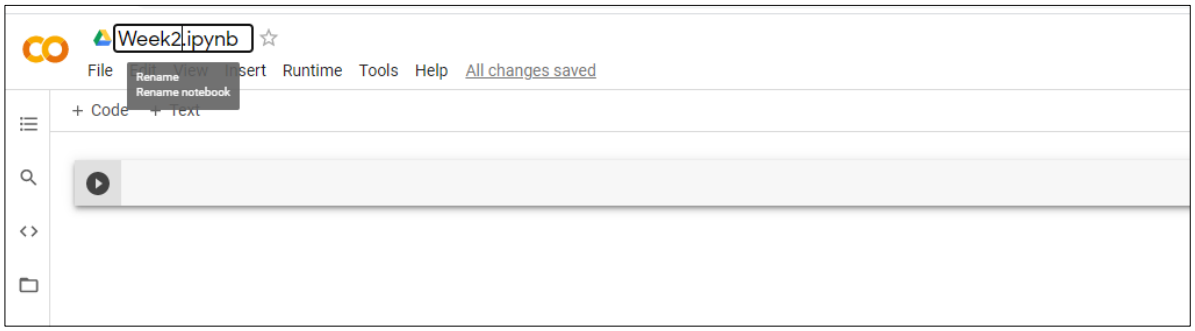
```
1  Error = original color - close color
2  SetPixel (x + 1, y) = Truncate (GetPixel (x + 1, y) + 0.5 * error)
3  SetPixel (x, y + 1) = Truncate (GetPixel (x, y + 1) + 0.5 * error)
```

Here is the pseudo-code for the Floyd and Steinberg diffusion errors that must be done after a pixel (x, y) has been converted to its nearest color:

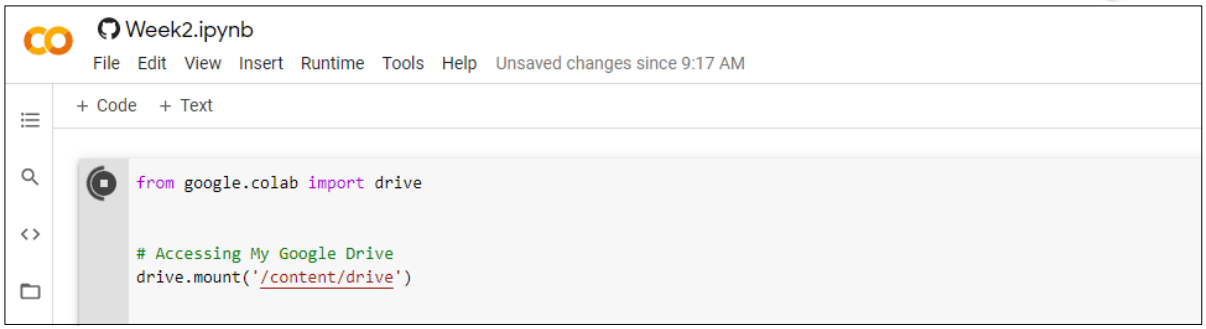
```
1  Error = original color - close color
2  SetPixel(x+1, y  ) = Truncate(GetPixel(x+1, y  ) + 7/16 * error)
3  SetPixel(x-1, y+1) = Truncate(GetPixel(x-1, y+1) + 3/16 * error)
4  SetPixel(x  , y+1) = Truncate(GetPixel(x  , y+1) + 5/16 * error)
5  SetPixel(x+1, y+1) = Truncate(GetPixel(x+1, y+1) + 1/16 * error)
```

C. PRACTICUM

1. o to https://colab.research.google.com/ . After making sure that Google Colab is connected to your Github , continue by selecting the repository that was used in the lab last week , rename the file to "Week 6 .ipynb".



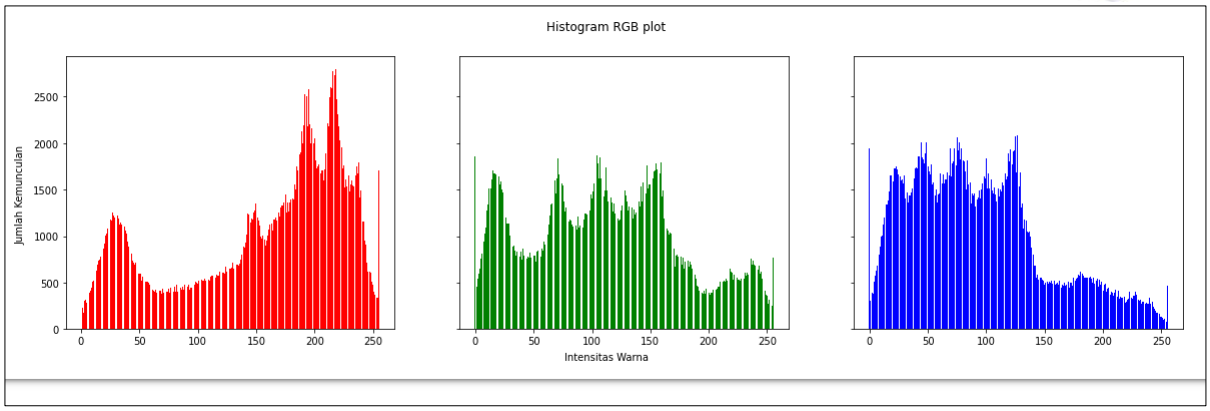
Then import the existing folder on your Drive as follows.



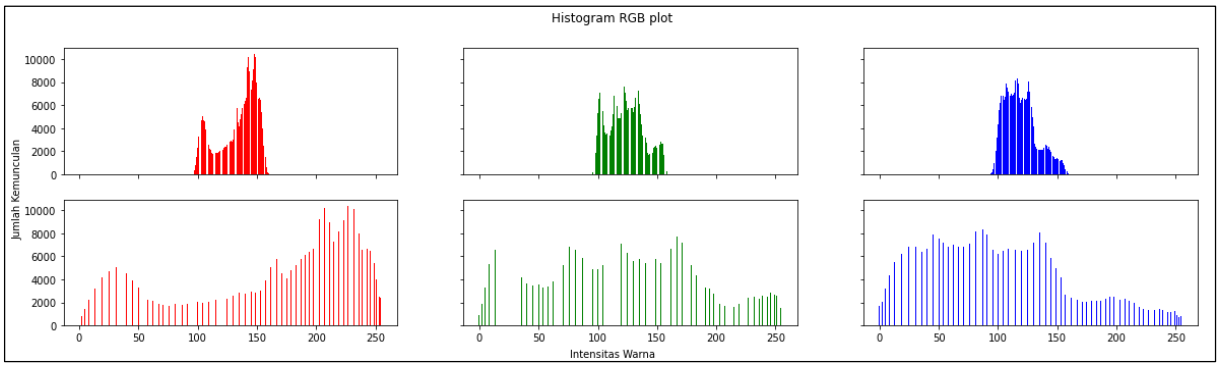
2. Import the following libraries that will be used during the following week 6 practicum trial.

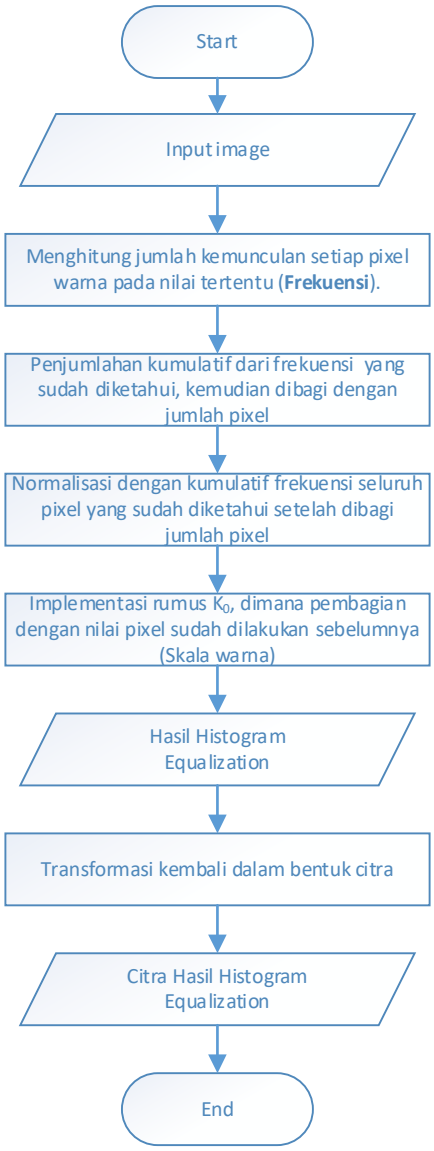
```
import cv2 as cv
from google.colab.patches import cv2_imshow
from skimage import io
import matplotlib.pyplot as plt
import numpy as np
import math
import os
import glob
```

3. Make an image histogram like the following histogram output based on the flowchart below : (Use image lena.jpg)



4. After working on question no. 3, make the same image histogram but use the library that is owned by NumPy, namely "`histogram`". Compare the results. Does the output appear the same?
5. Make an image histogram such as the output *equalization histogram* and also the image display before and after the following *equalization histogram* process based on the flowchart below : (Use image lena_lc .jpg)



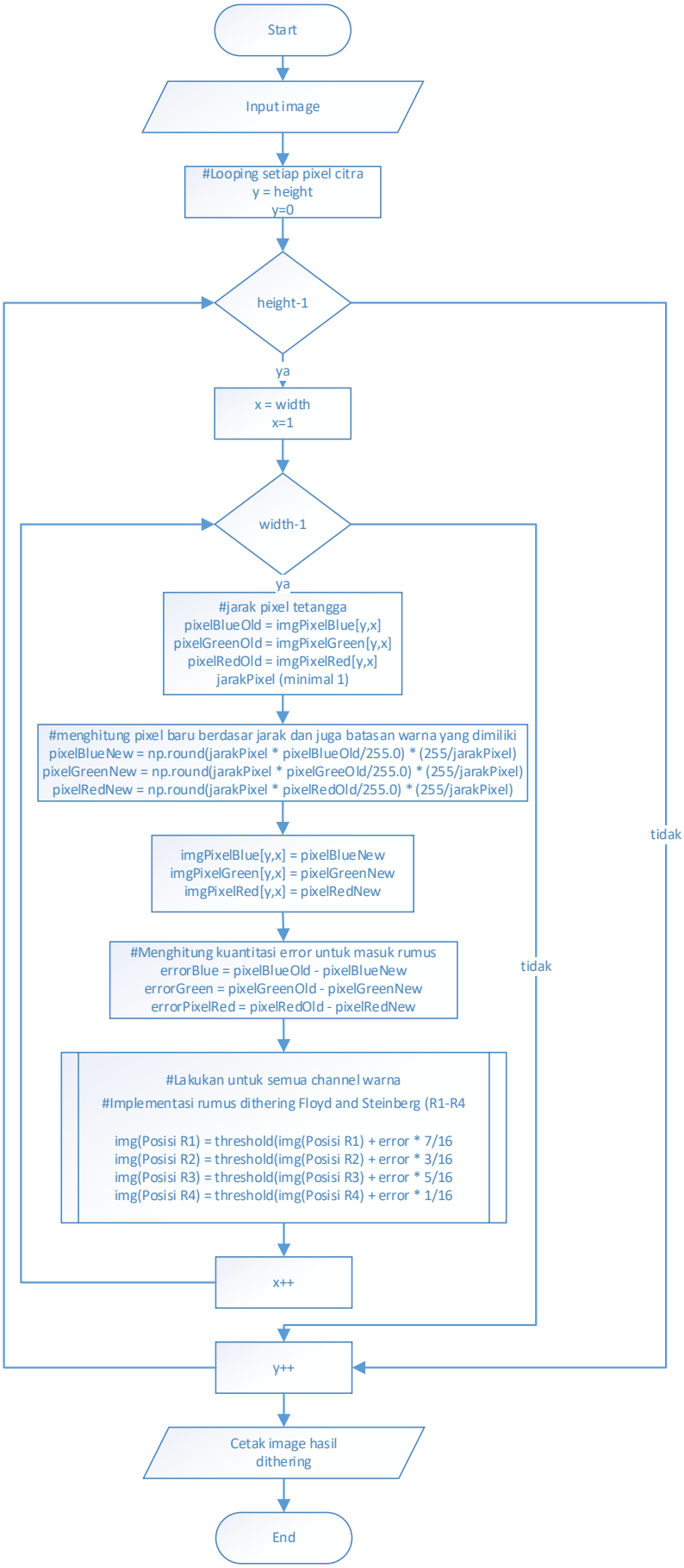


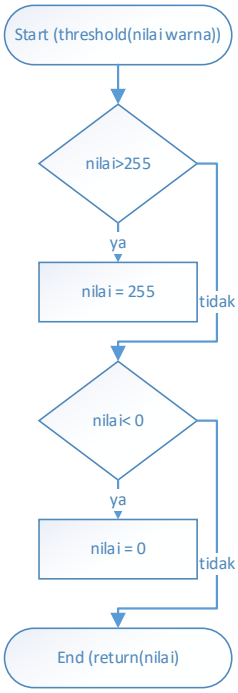
6. After working on question no. 5, make the same image histogram but use the library owned by CV2, namely " `equalizeHist` ". Compare the results. Does the output appear the same?
7. Do the mapping process of 16 million RGB colors into just 8 colors, namely black, green, yellow, blue, cyan, magenta, white as shown in the following output based on the flowchart below! (Use the image testlena .jpg)

8. Do the Floyd and Steinberg dithering process like the following output (initial image display, and after dithering display) based on the flowchart below! (Use wiki.jpg image)

Original Image

Dithering results using Floyd Steinberg





--- SELAMAT BELAJAR ---