MODULE 3 – Simple Image Operations – Linear Brightness, Contrast, Inverse, Logarithmic Brightness , and Grayscale Image

A. PURPOSE

- Students can understand and implement Brightness Linear Transformation using Google Colab
- Students can understand and implement Contrast Citra using Google Colab
- Students can understand and implement Inverse Image
- Students can understand and implement Logarithmic Brightness Transformation
- Students can understand and implement the types of Grayscale operations

B. TOOLS AND MATERIALS

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

C. Theoretical Background

- **Image Inverse**

  Image Inverse is a process used to invert the image value by mirroring the center pixel value . The results obtained are known by the name of the image of the negative. This process is also very easy because the operation only subtracts the value of 255 from the red , green , and blue pixel values .

  $g(x) = 255 - f(x)$, where g(x,y) is a negative image, and f(x,y) is the original image

  If the original pixel value is 255, with this operation the result will be 0 (because 255 – 255).

- **Brightness Linear Transforms**

  Linear Brightness Transformation is the simplest digital image processing operation . This operation performs increasing the pixel value if the brightness value is increased , and subtracts the pixel value if the brightness value is decreased.

  $g(x, y) = f(x, y) + b$, where g(x,y) is the pixel value after the transformation, f(x,y) is the original pixel value , and b is the brightness value.

  Due to the simplicity of this operation, the computation is very fast. Although the speed of calculation is high, the transformation has a weakness. Transformation of linear brightness is a one-way transformation, where the image transformed can not be returned to the initial value in doing a reverse linear brightness. For example, if the initial pixel is worth 240, converted by the addition of the value of the brightness 20, then the value of the pixel end is 260. If the program is run, an error will happen because the pixel value is just to be in the range of 0 - 255. In vice versa, subtraction in the result is less than 0 will getting an error as well. To overcome the thing is, usually made function Truncate the duty to prevent the pixel's value from being more substantial than 255 or smaller than 0. If the value after the operation is larger than 255 then the value will be forced (truncated) and becomes 255. If the value after the operation is smaller than 0, the value will be forced (at truncate) to become 0. Using the

truncate function, the value 240 is operated brightness 20 will be 255 (instead of 260). If the value of 255 is operated, brightness -20 will become 235 (not 240). This is what makes the linear transformation of brightness can not be reversed . The truncate function prevents error because the pixel's value is not located outside the range 0-255, but the consequences of pixels already brightnessed can not be returned. This problem can be solved using the Log Brightness transformation, which is also discussed in this module.
The algorithm for truncate can be seen in the following steps :

Steps	Process
1	Start
2	Read f1 pixel value
3	If f1 < 0, then f1 value replaced with 0
4	If f1 > 255, then f1 value replaced 255
5	Stop

- **Contrast Transforms**

Contrast is the degree of spread of pixels - the pixels to a role in the intensity of the color. There are three kinds of contrast: the contrast low, the contrast high, and the normal contrast.

- Image Contrast Low: The image has a low contrasted could happen due to lack of lightness. Image with a low contrast having narrow histogram curve, due to the spread of the intensity of light or the intensity of the dark is not evenly distributed. The darkest image's point does not transform to zero black value, and the lightest does not reach the 255 white value.
- High Contrast Image: is the opposite of the image of the low contrast for having a curve histogram width, the distribution of the intensity of the dark, and the intensity of the light evenly to the entire scale of intensity.
- Normal Contrast Image: Occurs when the width of the curve histogram is not too wide and not too cramped.

Contrast operations can be done by modifying the formula used for linear brightness, namely:

$g(x,y) = a * f(x,y) + b$, where g(x,y) is the pixel value after transformation, f(x,y) is the original pixel value, *a* is the contrast value, and *b* is the brightness value .

In addition to the above formula , contrast transformation can be done by calculating the Contrast Correction Factor using the following formula :

$$F = \frac{259(C + 255)}{255(259 - C)}$$

In order to work , the value of F is stored in a variable of type double and not an integer. The value of C represents the desired level of contrast value .
Stage next is to perform the calculation of the value of contrast in each pixel of R, G, and B.
$R' = F(R - 128) + 128$, where R is the Red pixel value and R' is the final Red pixel value after the Contrast operation. The truncate function is used because the result value of this operation can be outside the range 0-255. Remember back when the function truncates is used, then the process can not be reversed.

Here is the pseudo-code of the contrast formula above:

```
1  factor = (259 * (contrast + 255)) / (255 * (259 - contrast))
2  colour = GetPixelColour(x, y)
3   redBaru = Truncate(factor * (Red(colour) - 128) + 128)
4  greenBaru = Truncate(factor * (Green(colour) - 128) + 128)
5  blueBaru  = Truncate(factor * (Blue(colour)  - 128) + 128)
6  SetPixelColour(x, y) = RGB(redBaru, greenBaru, blueBaru)
```

- **Logarithmic Brightness Transforms**

  The Log Transform maps an image with a narrow color range to a wider one in the output image. Transformation logarithm useful in the depiction of the graph when the row of values are the difference between the value of the very small and the difference between the value of which is very large, so the difference in value that is very small it will be difficult to see. Range of input that is narrower than the value of gray level low is mapped into the range that the width of the output gray level.

  In general, the form of the log transformation is :

  $$s = c * log(1 + r)$$

  where

  c : constant

  r : Input Image gray-level value

  s : Output Image gray-level value

- **Grayscale Transforms**

  Grayscale is a gray-level image with 8 bits pixel intensity value. Grayscale is usually used to represent the intensity of light on a single band of the electromagnetic spectrum (infrared, light visible, ultraviolet, etc.). Some of the *grayscaling* methods that are often used are *Averaging*, *Lightness*, and *Luminance*. Averaging is determined by averaging the value of each channel R, G, and B. Lightness is determined by averaging the value of the maximum and the value of the minimum entire channel R, G, and B. While the *luminance* (luminosity) determined by giving weightings are adjusted to the visible spectrum exists in the natural image.

  $$Grayscale_{avg} = \frac{(R + G + B)}{3}$$

  $$Grayscale_{Lightness} = \frac{\max[R, G, B] + \min[R, G, B]}{2}$$

  $$Grayscale_{Luminance} = 0.21R + 0.72G + 0.07B$$

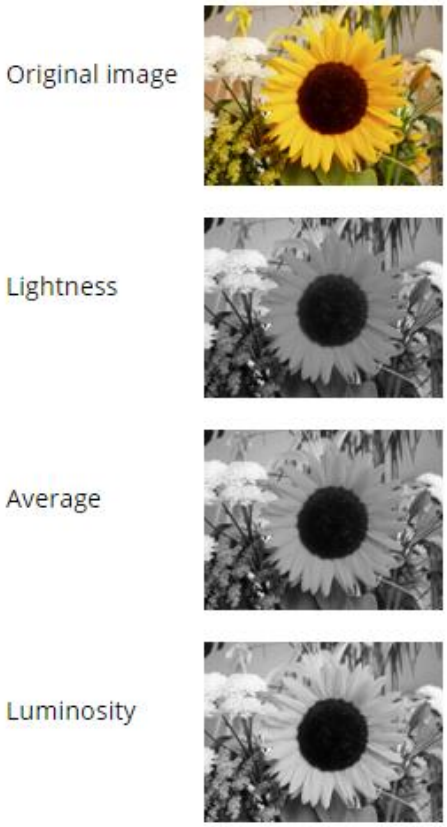

Figure 1. Grayscale Transformation methods

D. PRACTICUM

1. Open https://colab.research.google.com/, select the Github tab and make sure the repository selected is the existing repository used in our first and second week class.

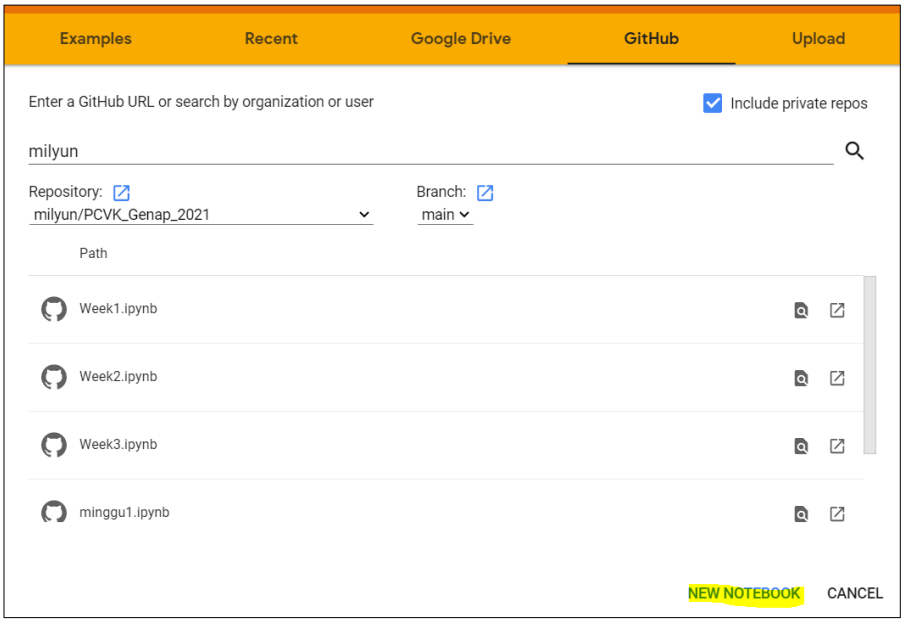

Figure 2. New Notebook in Google Collab

Continue by creating a new notebook and rename the file to "Week3.ipynb". **Caution**: Do not forget to save a copy to Github after doing changes/when you 've finished doing practicum.

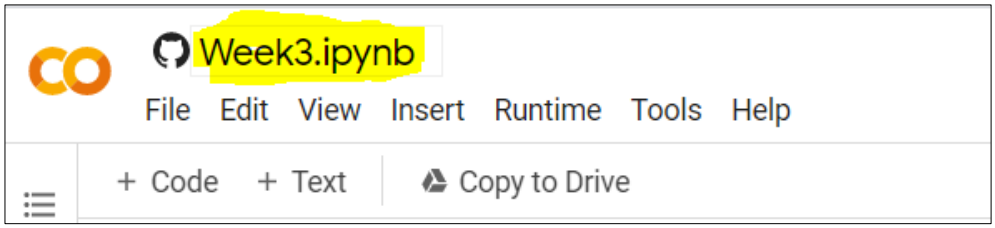

Figure 3. File Naming in Google Collab

2. Access the image folder on your Google Drive with the following code:

```
[ ]  from google.colab import drive

     drive.mount('/content/drive')

     Mounted at /content/drive
```

Follow the authorization process until a message appears: "*Mounted at /content/drive*".

3. Performs a linear transformation of brightness by entering a certain constant value and produces a color . As that has been discussed in the review theory , the formula for doing transformation linear brightness is as follows :

$g(x, y) = f(x, y) + b$

where g(x,y) is the pixel value after the transformation , f(x,y) is the original pixel value , and b is the brightness value .

Write the following code snippet to implement linear brightness in google colab :

Transformasi Linier Brightness

Formula: *g(x,y)=f(x,y)+b*

g(x,y) adalah nilai pixel setelah transformasi, *f(x,y)* adalah nilai pixel asli, *b* adalah nilai brightness

```
print(' Mengubah tingkat kecerahan citra ')
print('---------------------------------')
try:
    brightness = int(input('Masukkan nilai kecerahan: '))
except ValueError:
    print('Error, not a number')

original = cv.imread('/content/drive/MyDrive/images/female.tiff')
brightness_image = np.zeros(original.shape, original.dtype)

#akses per piksel
for y in range(original.shape[0]):
    for x in range(original.shape[1]):
        for c in range(original.shape[2]):
            brightness_image[y,x,c] = np.clip(original[y,x,c] + brightness, 0, 255)

#cara simple tanpa for loop
#brightness_image = cv.convertScaleAbs(original, beta=brightness)

final_frame = cv.hconcat((original, brightness_image))
cv2_imshow(final_frame)
```

The system will display the text field to enter the brightness value from the code pieces above. Next, determine the image on the drive to be processed and stored in the original variable. The next step is to access the input image pixels with three iterations. The first iteration is done on shape[0] for the image height, the second iteration is done on shape[1] for the width of the image,

and the third iteration is done on shape[2], which is the color channel in the image. After making three loops, performed the transformation of linear brightness by adding the value of the brightness in the image input. An example of the result of the program code above is as follows.

Figure 4. Linear Brigthness Transformation result

TASK

1. Implement image inverse in your code using formula shown in the theoretical background subsection above, giving the output shown in the following figure:

2. Implement contrast transformation in Google Collaboratory using the formula written in the Theoretical background section, giving the output shown in the following figure:

Figure 5. Contrast Image Transformation result

3. Implement transformation of logarithmic brightness on Google Colaboratory using the formula written in the Theoretical Background Reviews, generating output shown in the following figure:

Figure 6. Log Transformation result

4. Implement grayscale transformation using the averaging, lightness, and luminance method on Google Colaboratory using the formula written in the Theoretical Background Reviews, generating output as follows :
    a. Averaging

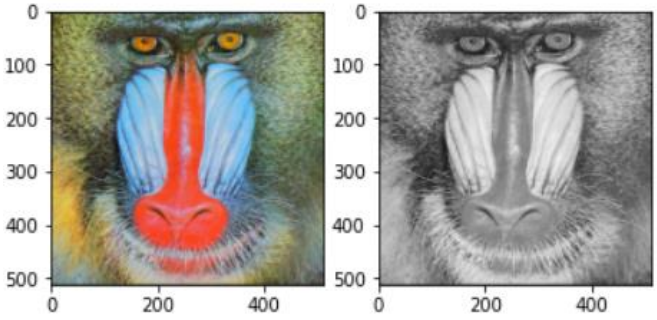

Figure 7. Grayscale Image Averaging method result

    b. Lightness

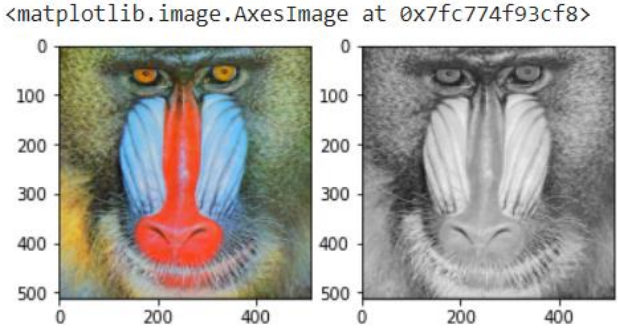

Figure 8. Grayscale Image Lightness method result

    c. Luminance

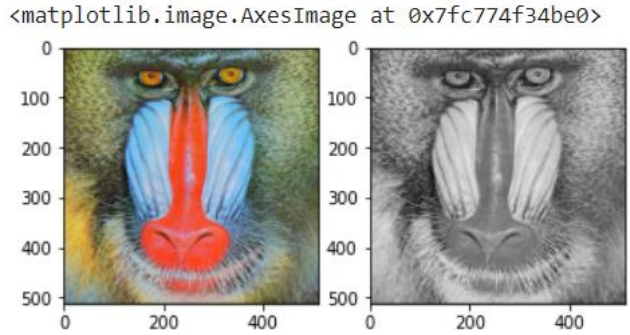

Figure 9. Grayscale Image Luminance method result

5. Show certain colors in the image , and change other colors to grayscale. For example , show the blue color in the image input and change the parts other which is not colored blue into grayscale as in the example below :

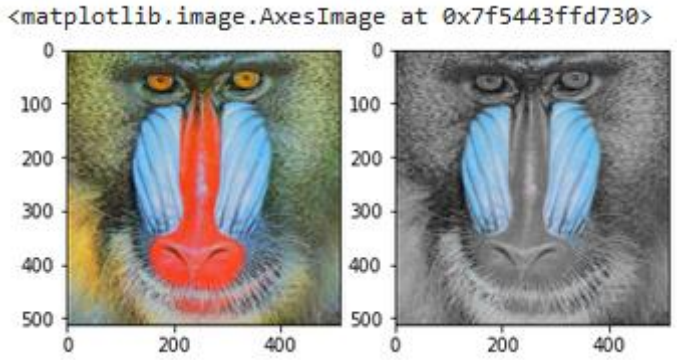

Figure 10. Image transform with custom color selection

--- GOOD LUCK ---