Graph Application in Image Filtering

Ilham Syahid Syamsudin - 13518028[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13518028@std.stei.itb.ac.id

***Abstract*— The graphs have many applications in real life. Including for image processing and analysis. Graph methods can be used in image segmentation, image filtering, and image classification or clustering. This paper will discuss about graph applications in image filtering. Image filtering is popular for image processing. It is a technique for enhancing and modifying an image. This paper also includes several examples for each image filters. The sample program in this paper used in Python 3.7.5 and including extra libraries such as Matplotlib, Numpy, and OpenCV. For the project using Jupyter Notebook. Matplotlib used for showing the image and Numpy used for scientific calculations. The important library is OpenCV, it used for extracting image features that can be manipulated as we want.**

***Keywords*—Graph, Image, Image Filters, OpenCV, Function.**

I. Introduction

In the last decade, image processing is one of the popular topics nowadays. Image contains many things, such as memories, information, history, etc. Another name of image processing is called Computer Vision. It makes the computer knows, imitate, or predict visual data as human representation.

Image regarded as being a real-valued, non-negative function of two real variables; the value of this function at a point will be called the gray-level of the picture at the point[2]. Each image depends on size or dimension that usually called by width and height (e.g. 1280 x 720) called pixel. The image contains a bunch of pixels. From Pixel, the image becomes to have the features component, it called color/shade and the location of the pixel itself. At each position pixel (or at each grid square) we usually represent the gray level value using an integer ranging from 0 for black to 255 for fully white[3]. The value each pixel can be represented to the binary that can be seen by the computer.

Expanded from the image, videos can be represented as collections of the images. Each image not only has property and characteristic itself but also can be processed. Due to the functionality of images, not surprisingly many startups have a business model in image processing.

Image filtering can be used for an enhanced quality image. It can be removing or reducing noise that

This paper will explain about basic graph theory, image filtering and also implement image filtering by OpenCV library including the relation between graph and image, how computer representing the image as digital image as we know familiar.

II. Basic Theory

A. Graph

A graph used to represent the object's district and relations between the objects itself [5]. In mathematics, a graph is a pair $G = (V, E)$ consists of V, a set of non-empty set of vertices or nodes and E, a set of edges. Each edge has either one or two vertices associate with it, called endpoints. An edge spanning two vertices v and w represented as (v, w) and two vertices are adjacent to each other. The set of vertices adjacent to v represented as $N(v)$.

A graph with each edge connects to different verticles and there no edge connects the same verticle or loop is called a simple graph. If at least one edge connecting the same verticle is called a looping graph. Otherwise, is called an unsimple graph that has a pseudo graph and multiple graphs.

There are two types of graphs based on the orientations of edges, directed graphs, and undirected graphs. In a directed graph, every edge has direction $G = (V, E)$, V consists of non-empty set of vertices and E consists of a set of directed edges represent as (u, v) defined start from u to v which u and v are vertices. A weighted graph is a graph that has value in every edge in G. Otherwise, an unweighted graph is graph without value in every edge.

Some special graphs have a lot of applications. The examples are a complete graph and a regular graph. A complete graph is a simple graph that every vertex in the graph has edges with another vertex in the graph. A regular graph is a graph that has the same degree in every vertex in the graf. The degree of vertex $d(v)$ is the sum of edge that edge with that vertex. In the directed graph, the degree of the vertex denoted as $d_{in}(v)$ as in-degree and $d_{out}(v)$ as out-degree.

There is a lot of representation of the graph, such as adjacency matrix, incidency matrix, and adjacency list. Because we will discuss about the image, and the image represents the matrix, adjacency matrix will relevant to the representation.

The adjacency matrix is a matrix that represent the relational between the vertex. In the position (u, v) denoted by 0 for no relation between vertex u and vertex v or there no edge between that vertex. Otherwise, if the position (u, v) denoted by 1 it has relation between vertex u and vertex v or there is an edge between those vertices. The relation between them can be represented as a weighted graph, if and only if the element in position (u, v) denoted as an integer.

The incidency matrix is a matrix that shows the relationship between the vertex has size $n \; x \; m$. The rows denoted as vertex's label, and the columns denoted as edge's label. Call the matrix as M. If the element of M at the position (i, j) have value, then the edge of j is vertex-i with the that value (if weighted graph).

Otherwise, if the element of M at the position (i, j) is zero, no relation between them.

The adjacency list is a list that every vertex listed with its neighbors. The vertex u is connected with vertex v, if and only if v is listed on u's list and u is listed on v's list. If we want to represent the weighted graph, in adjacency list instead of listed the number, it listed tuple (v, w) where v is the vertex and w is the weight.

A graph has subgraph $G' = (V', E')$ is a subgraph of graph $G = (V, E)$ if and only if V' is a subset of V, and E' is a subset of E [5]. By removing the vertex or the edge in graph G, we can get some subgraphs.

B. Image

For simplify the definition of an image, let's take looks image as a function, f, with $f(x, y)$ gives intensity image at the position (x, y) with range $[0,255]$, look at Figure 1. Value 0 defined as black and value 255 defined as white.

Furthermore, for making colored image function, we can create as a vector function. Take of RGB base for colored image function, we can write as formula below.

$$f(x, y) = [\, R(x, y), G(x, y), B(x, y) \,]$$

Function $R(x, y)$ defined as the intensity of red color, then function $G(x, y)$ defined as the intensity of green color, also function $B(x, y)$ defined as the intensity of blue color. See the illustration below.

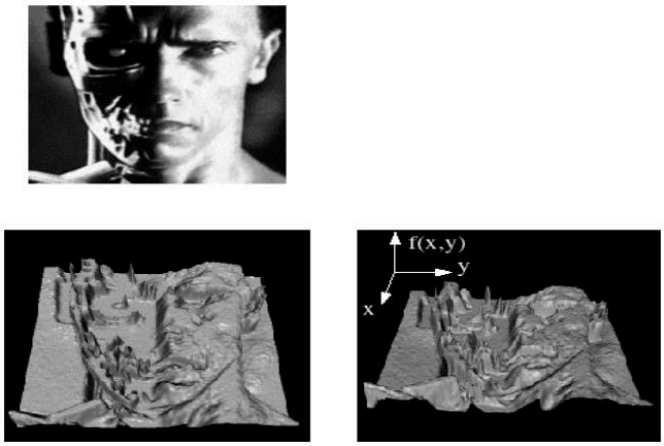


Figure 1. Image as Function. Source: Noah Snavely [10]

C. Digital Image

As mention in the introduction section, digital images are made of picture elements called pixels. Pixels are organized by an ordered rectangular array or matrix. The size of an image is determined by the dimensions of this pixel array. The image width is the number of columns, and the image height is the number of rows in the array. Thus the pixel array is a matrix of M columns x N rows.

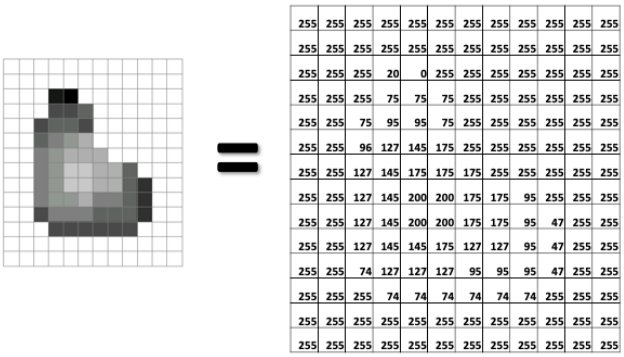


Figure 3. Image as Matrix. Source: Noah Snavely [10]

Reference [6] shows referring to a specific pixel within the matrix, defined to the coordinate at point x, y. The position from the coordinate system in the image matrix defines x as increasing from left to right and y is increasing from top to bottom. If compared with normal mathematic convention, the origin is in the top left corner and the y coordinate is flipped. Basically, digital images were defined in terms of the electron beam scanning pattern of televisions. The beam scanned from left to right and top to bottom. Then, Image filters can be used in ordered to reduce noise or any undesired features of an image.

III. IMAGE FILTERING

Image filters can be used to reduce noise or any undesired features of an image. Image filters also can be creating a better and enhanced an image. Due to features of the image can be manipulated, the image can rearrange as we want to. Before going through, the image has to convert to the matrix and arrange adjacent with a graph using graph theory.

We take G as a graph. As we know earlier, image has several pixels. Each pixel represented as vertex in graph G. Also each pixel has their neighbor which is their adjacent pixels. Each pixel also has the value of their function, Figure 2. If we take the image as a simple graph, we can draw as Figure 3.

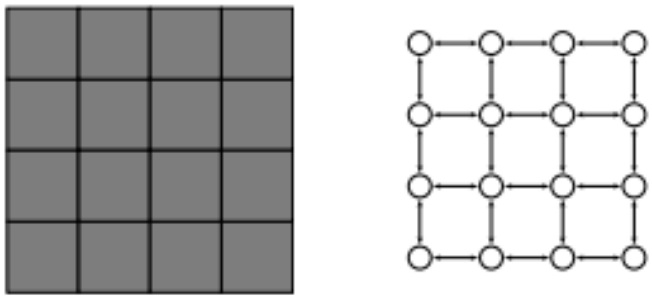


Figure 2. Pixel adjacency graphs. Source: Filip Malmberg [4]

Combined between values and adjacent pixels, we can manipulate the object. Moreover, due to image define as a function, we can apply operators to an image as Figure 4. In that figure, the first image applies to increased image intensity by 20. The second image applies the inverse of x component.

Figure 4. Image transformations. Source: Noah Snavely [10]

There are two types of image filtering: linear filters and non-linear filters. Each filter has several specific purposes to remove noise or enhance an image.

A. Linear Filters

Linear image filtering is the simplest filter. Every pixel in the image can be modified by scalar values. Followed as formula below.

$$G(x,y) = F(x,y) \times K + L \quad (1)$$

- $F(x,y)$ is input image function at the position (x,y).
- $G(x,y)$ is output image function at the position (x,y).
- K, L is scalar constant.

For example, take a look in Figure 4. It is very clear that figure using a linear filter. In the left image, added scalar value (L). As the result, it became white-wide because of white has more gray value as the maximum 255 (fully white). Component of the image can be manipulated as x manipulated by flipped (multiply by -1) in the right image of Figure 4.

Setup the library in Jupyter Notebook below.

```
import cv2
import matplotlib.pyplot as plt
import numpy as np
%matplotlib inline
```

OpenCV library can be used for extracting the features of the image to the matrix of pixel with relating each other. To manipulate the value, Numpy can be an alternative. Also for displaying the image inline in Jupyter Notebook using Matplotlib.

```
def noise_img(path):
    img = cv2.imread(path,0)
    noise = np.zeros(img.shape)
    cv2.randu(noise, 0, 256)
    noisy = img + 0.2*noise
    return noisy
```

A simple implementation for linear filters is a noisy image. Noise can be generated from random value from 0 to 255 in grayscale. For getting linear filters, just add noise to the original image. From the function `noise_img` we can write formula below.

$$noisy(x,y) = img(x,y) + noise(x,y) \times 0.2$$

For displying image, we can write a function below.

```
def show_img(initial, final,
      imsg='Original', msg='Converted'):
    plt.figure(figsize=(10,5))
    plt.subplot(121)
    plt.title("Original")
    plt.imshow(img_ori, cmap='gray')
    plt.subplot(122)
    plt.title(msg)
    plt.imshow(img, cmap='gray')
    plt.savefig(str(msg) + '.jpg')
    plt.show()
```

Fill `initial` with the matrix of the original image. Then, `final` for final result. Then, `msg` for title in column final image with default value 'Converted'. Similar for `imsg`. Here example for calling the function.

```
house = cv2.imread('images/house.jpg',0)
noisy_img = noise_img(house)
show_img(house, noisy_img, msg='Noised')
```

Here is the example result:

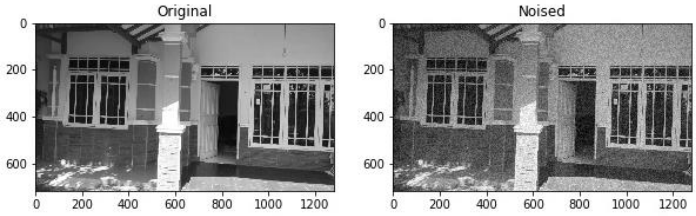


Figure 5. Example of before and after applying noise image filters.

From the example above, we can write down generals function for linear filters. Show in the below.

```
def linear(img, K, L):
    img = np.asarray(img, dtype=np.int)
    img = img*K + L
    img[img > 255] = 255
    img[img < 0] = 0
    return img
```

This function more likely as function (1). We call,

```
linear = linear(house, 2, -100)
show_img(house, linear, msg='Linear')
```

and the result can be seen below.

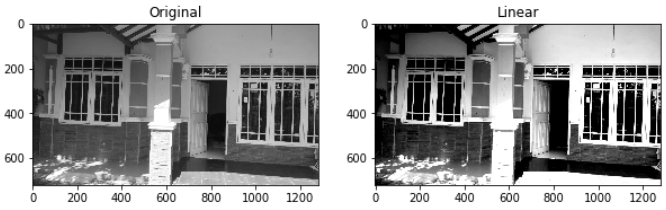


Figure 6. Example of before and after applying general linear function.

The other example is a box filter, or mean filter. The idea is to average the adjacent pixel. By applying this filter, the result is a blurred image. Take look in the below example.

```
mountain = cv2.imread('images/mountain.jpg')
blur5 = cv2.blur(mountain, (5,5))
blur10 = cv2.blur(mountain, (10,10))
blur20 = cv2.blur(mountain, (20,20))
show_img(mountain, blur5, msg='Blur 5x5')
show_img(blur10, blur20, imsg='Blur 10x10',
msg='Blur 20x20')
```

The result can be shown below:

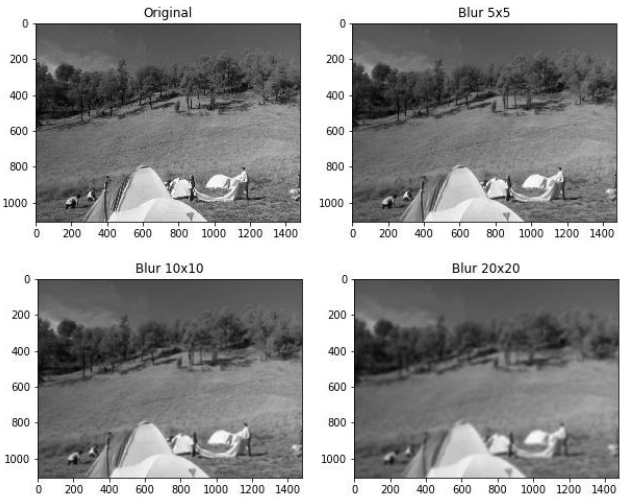


Figure 7. Example box filter.

As an increase in the kernel, the image more gets blurred. If we take a look blur with kernel 5x5, there is no significant change. But for kernel 20x20, the image more blurred than before.

B. Non-linear Filters

Linear filters are usually sufficient for getting the result as we want to. But in the several cases, we want to get significant value. In the way of using it, nonlinear filters can more complicated than linear filters. But, from the complicated calculation, we can get more detail and approach to the significant result.

a. Median Filters

Median filters can be called neighborhood filtering. The idea is to use the median technique. This filter very useful for removing deep or sharp noise such as salt and pepper (not discussed in this paper).

This filter travers around all the image pixels. Then, collect the pixel values and the adjacent value then takes the median of those values. The result will place to the center of the pixel. In OpenCV, itself has median filters function. Look at in the below example.

```
penguin = cv2.imread('images/penguin.png')
median3 = cv2.medianBlur(penguin, 3)
median5 = cv2.medianBlur(penguin, 5)
median7 = cv2.medianBlur(penguin, 7)
show_img(penguin, median3, msg='Median 3x3')
show_img(median5, median7, 'Median 5x5',
'Median 7x7')
```

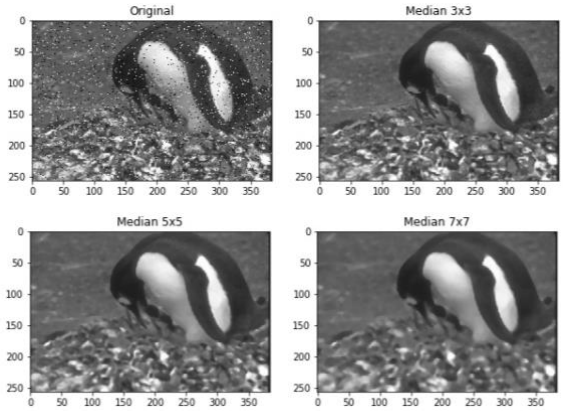


Figure 8. Example for median filters in the different kernel.

We have to specify the kernel (or the second argument of the function). It means we take a new square matrix with a size of kernel (i.e 3x3). The important thing that we have to look at, is kernel value must be odd. Because of the median value have to place in the center of the matrix.

b. Gaussian filter

This filter basically for smoothing around the edges. It enhances and reduces the effect of the pixel. By applying this gaussian filter, it removes strong edges and hence to blurred image [11].

In the mathematic formula, Gaussian function is written as:

$$f(x) = \frac{-1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)$$

where σ is variance and μ is mean.

OpenCV also has Gaussian function, look at the below example:

```
gaussian3 = cv2.GaussianBlur(penguin,
            (3,3),0)
gaussian5 = cv2.GaussianBlur(penguin,
            (5,5),0)
gaussian7 = cv2.GaussianBlur(penguin,
            (7,7),0)
show_img(penguin, gaussian3, msg='Gaussian
3x3')
show_img(gaussian5, gaussian7,
imsg='Gaussian 5x5', msg='Gaussian 7x7')
```

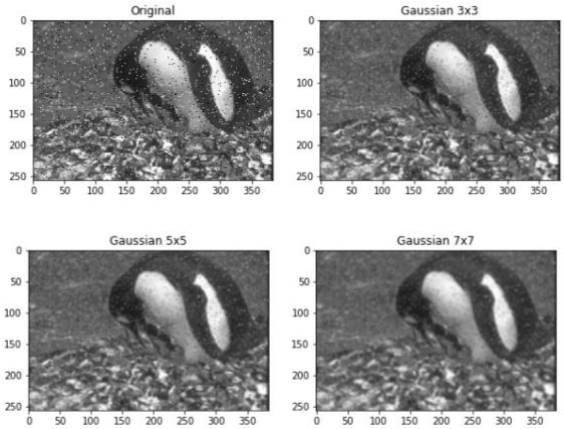


Figure 9. Example Gaussian Filter.

Similar to the median filter, Gaussian also needs a square matrix. Not the best result for reducing salt and pepper noise than the median filter. The best functionality of the Gaussian filter is for smoothing blur that resembles viewing the image through a translucent screen, bokeh effect, enhance image structures at different scales.

c. Histogram Equalization

We have seen in the earlier example, the linear filter function has manually specified K or L to improve image quality. But for histogram equalization, it applies an algorithm and improving image quality. This filter sets the brightness pixel to white and the darkest to the black. The remaining pixel also rescaled, transforming intensity distribution. [11]

The earlier example picture:

Figure 10. Input to histogram Filter

We can see the histogram converted by online converter[1]:

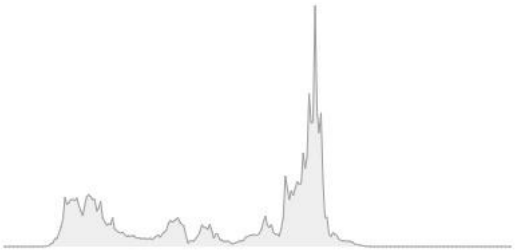


Figure 11. Histogram of the input image

Again, we can use the histogram function from OpenCV. Lets take look in the below example:

```
equ = cv2.equalizeHist(house)
show_img(house, equ)
```

These codes convert image to:

Figure 12. Output image after applying histogram filter

With the same technique, we convert it to histogram:

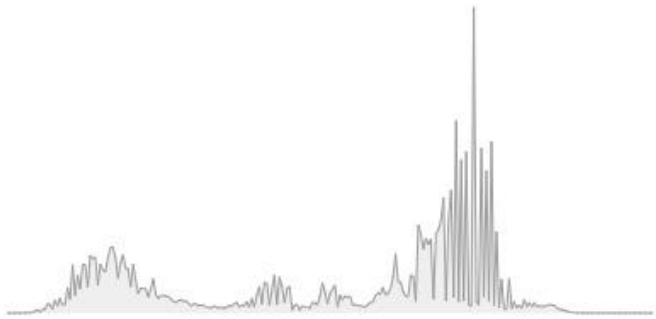


Figure 13. Histogram of the output image

As you can see, the output histogram more widely than the input and the contrast increases significantly.

Here is the compared image:

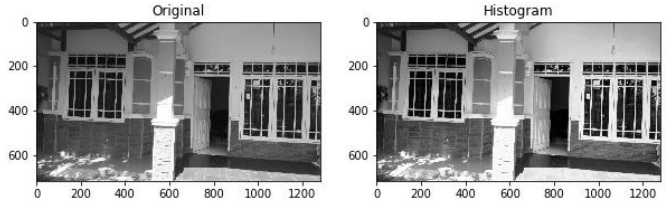


Figure 14. Example of histogram filter

[1] Available: https://pinetools.com/image-histogram

IV. The Other Application

If we talk about image filtering, there a lot of technique that we have already known. This paper gives a simple and common filter and impossible to discuss all. Image filtering can be used for getting information by modifying or enhance the image. Moreover, image filtering can be used for editing the color scheme of the image, adjusting image brightness, contrast, changing the texture of the image or adding other effects to the image.

The other filter that does not explain in this paper is the sobel filter. Basically, sobel filter computes by 3x3 convolution kernels.

We are not talking about it further, just take look in the below example:

Figure 15. Example sobel filter[2].

And many more techniques for image filtering. For graph itself, has a lot of applications, such as image segmentation, image recognition, compression file, and many more.

V. Conclusion

The graph can be used for many applications, including image filtering. By extracting image features and combined with several techniques or algorithms, we can manipulate the image as we want. One of the useful library is OpenCV and the best programming language to interact with it is Python.

VI. Appendix

The author's implementation of the algorithm discussed in this paper can be found in the author's Github repository (https://github.com/ilhamsyahids/Image-Filtering) written in Python 3.7.5 with Matplotlib, Numpy, OpenCV modules. The sample image such as house and mountain were captured by the author. But for penguin image can be found in this URL https://www.fit.vutbr.cz/~vasicek/imagedb/img_corrupted/impnoise_005/106020.png. The other image in this paper have been written with the source either in the footnote or in references except for the image from the author's implementation in Jupyter Notebook.

VII. Acknowledgment

The author would like to thank to Allah for guidance to finish this paper. The author also would like to thank to Dr. Rinaldi Munir as lecturer for the Discrete Mathematics IF2120 class in the third semester and thank you for the guidance in Bandung Institute of Technology. The author also thanks to the developer and contributors of Python, OpenCV, Matplotlib, Numpy, and Jupyter, the author can make first paper about one of graph application. Also thanks to everyone who helped me arrange this paper and the readers may be inspired after reading this paper.

References

[1] MathWorks Documentation. *Image Filtering* [Online]. Available : https://www.mathworks.com/help/images/linear-filtering.html (Accessed 12 November 2019)

[2] Rosenfeld, *Picture Processing by Computer*, ACM Computing Surveys, 1969.

[3] Bebis, George. *Data Structures* [Online]. Available : https://www.cse.unr.edu/~bebis/CS302/image_info.html (Accessed 12 November 2019)

[4] Malmberg, Filip. Image Processing Using Graphs [Online]. Available : http://www.cb.uu.se/~filip/ImageProcessingUsingGraphs/ (Accessed 12 November 2019)

[5] Munir, Rinaldi. *Matematika Diskrit Ed. 6*. Bandung: Informatika, 2016.

[6] Sheets, Kristopher, *What is a digital Image?* [Online]. Available : https://sites.google.com/site/learnimagej/image-processing/what-is-a-digital-image (Accessed 17 November 2019)

[7] Abderrahman, *Image Filtering in Python* [Online]. Available : https://code.tutsplus.com/tutorials/image-filtering-in-python--cms-29202 (Accessed 17 November 2019)

[8] Corso, Jason. *Foundations of Computer Vision* [Online]. Available : https://web.eecs.umich.edu/~jjcorso/t/598F14/ (Accessed 23 November 2019).

[9] Szeliski , Richard, "*Computer Vision: Algorithms and Applications*" Microsoft Research, 2010.

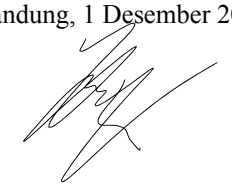
[10] Snavely, Noah, Introduction to Computer Vision [Online]. Available : https://www.cs.cornell.edu/courses/cs5670/2019sp/ (Accessed 24 November 2019).

[11] Boricha, Vijin. Image filtering techniques in OpenCV [Online]. Available: https://hub.packtpub.com/image-filtering-techniques-opencv/ (Accessed 12 November 2019).

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 1 Desember 2019



İlham Syahid Syamsudin
13518028

[2] Source: https://fiveko.com/tutorials/image-processing/sobel-filter/