Fingerprint Verification using Image Processing

Naufal Alexander Suryasumirat / 13519135
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519135@std.stei.itb.ac.id

***Abstract*—Fingerprint verification is a well-established method of biometric identification that is widely used for security purposes. In this paper, the author proposes an approach to fingerprint verification using image processing techniques to process the fingerprint image and extract unique features of fingerprint images. The approach can be useful in a range of fields, including access control, or forensic analysis.**

Keywords—Fingerprint, biometrics, security, image processing, access control;

I. Introduction

In this modern day and age, the topic of security has become a more heated debate than ever before seen. Every person has their own personal data they want to protect from other people, for various reasons. The topic of security involves many other domains or fields such as online security or cyber security, telecommunication security or network security, operations security, physical or environmental security, and many other fields. One of the fields that have been very well-researched and developed for many years is biometrics or biometric security.

Biometrics are biological measurements of physical characteristics that can be used directly to identify individuals. Methods used to identify individuals with biometric technology can vary, ranging from the standard methods that most people already know such as fingerprint mappings, facial recognition, and retina scans, to the lesser-known methods such as the shape of a person's ear, the pattern of how a person walks, and even the veins in a person's hand. All of these methods have the same goal of identifying a person based on their physical characteristics, on their physical body.

The well-known methods to identify someone based on their biometrics mentioned above are so popular now in everybody's lives, especially ones based on fingerprints, that it would be very unusual if a person has a phone without a fingerprint sensor function built-in. The technology used has become so advanced that people start looking for under-the-display fingerprint sensors on phones, a more challenging feat than previously used technologies. People also are starting to become very accustomed to biometrics technology because of the convenience it provides. The amount of time taken to unlock your phone using your fingerprint, and the minute effort you use to put your finger on the sensor to unlock your phone has become a necessity for people.

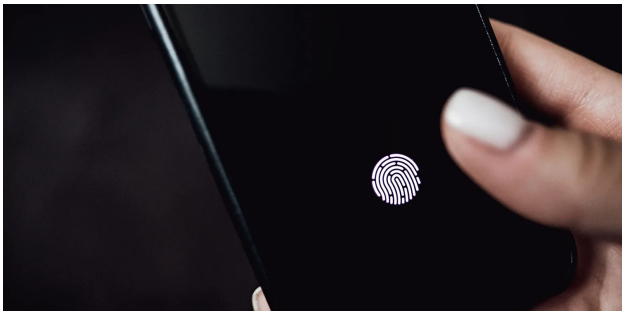

Image 1. Under-display fingerprint sensor used in modern phones.

Fingerprint recognition works by first learning the characteristics of a person's fingerprint through feature extraction and various image processing techniques. Then, storing those characteristics, or in general, storing the "fingerprints" of your fingerprint on the device itself or in a remote server. The data stored would be encrypted for security purposes before storing. Then, when a finger touches the sensor on the device, it then would do the exact same process to the fingerprint, except for the storing part. It would take the characteristics of the fingerprint touching the sensor and try to match it with the stored fingerprints.

As mentioned before, there are various image processing techniques used in recognizing someone's fingerprint, mainly scaling, filtering, segmenting, equalizing, and binarising just to name a few for the pre-processing step of the process. There is also a noise removal step in fingerprint recognition because the image taken from the sensor would not always be ideal for direct feature extraction. For feature extraction of the fingerprint image, there are various algorithms used, focused on morphological features of the fingerprint image.

Fingerprint recognition technology has become a necessity for many people, which is all the more reason to understand the underlying algorithm used to recognize a person's fingerprint while making an attempt to improve it. The reasons mentioned above are what's motivating this paper to be written. The techniques used would be techniques related to image processing, and the result would be an application to identify a person's fingerprint. The results would then be analyzed and evaluated to measure the performance of the experiment done on the process of fingerprint recognition.

II. THEORY

A. Digital Image

A digital image is a representation of a two-dimensional image as a set of digital values or a representation of a continuous image using sampling methods. This sampling can be done in the context of space or space and time. If the sampling is done in space only, the result would be a two-dimensional still image, on the other hand, if it is also done in space and time, it would be a digital video consisting of multiple images called *frames*. A digital image is usually rectangular in shape.

Commonly, there are two types of digital images, grayscale images, and colored images. But, most digital images can be represented with digital values of one or multiple two-dimensional matrices. These matrices contain digital values depending on the intensity value at a particular location in the image. The digital values can represent the intensity of the color or the gray level of a particular location in an image.

The digital values in a digital image, also commonly known as pixels, are stored in a computer or other digital device and can be processed and manipulated using image processing algorithms or techniques. For grayscale images, a pixel represents the gray level or intensity level of a particular location in the image, whereas in colored images, commonly RGB images, each pixel represents the intensity of the color it represents, whether it be red, green, or blue. The combination of these three main colors would make other colors such as magenta, yellow, or other colors.

The pixel values or pixels of a digital image are typically stored as integers or floating-point numbers, and the number of bits used to represent each pixel value determines the color depth or bit depth of the image. As an example, an 8-bit image has a color depth of 8 bits per pixel and can represent up to 256 different colors, whereas a 24-bit image has a color depth of 24 bits per pixel and can represent over 16 million colors. In grayscale images, the bits per pixel would represent the variance of gray level it could produce.

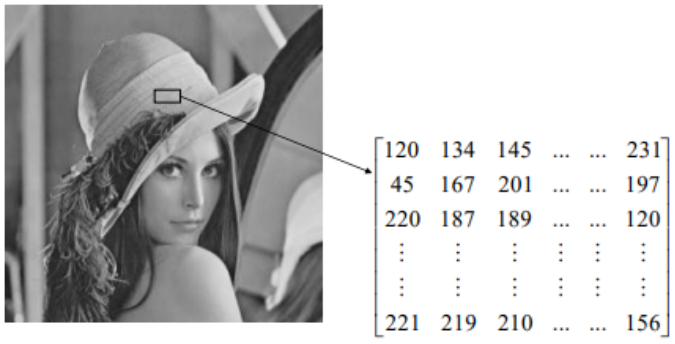

Image II.1. Pixel values of an 8-bit grayscale image

As can be seen in Image II.1, a grayscale image of Lena, the representation of each pixel value is an integer with a maximum of 255. The maximum value or intensity the image could produce is determined by the depth of the image, which in this case is an 8-bit image capable of producing 256 different intensities of a gray level value. In an 8-bit grayscale image, a pixel value of 0 would output the location of the image as black, whereas a pixel value of 255 would output the location of the image as white. Values between white and black in a grayscale image would represent different shades of gray.

A special case of grayscale image would be a binary image, being able to only represent white or black in an image, which would be explained thoroughly later on. Grayscale images could be converted to and from colored images using image processing techniques. An advantage of having a grayscale representation of a colored image would be in the step of image processing. As grayscale images often simplify the process of feature extraction while still maintaining the quality of the features that would be extracted.

B. Image Histogram

Image histograms are a graphical representation of the distribution of pixel intensity values in an image. For an 8-bit image, the values would range from 0 or black to 255 or white. An image histogram is a plot of the number of pixels in an image for each intensity value. For plotting a grayscale image, it would produce a single image histogram representing the number of pixels for each gray level in the image. For a colored image, the image histogram produced would be a plot of color intensities for each color channel in an image, commonly red, green, and blue.

Image histograms provide a quick and easy way to visualize the overall intensity distribution of an image and can be used to identify patterns in the image. It could also be useful for identifying the overall tonal range and contrast of an image. An image with high contrast would have a histogram with a larger range of intensity values, whereas an image with low contrast will have a histogram with a smaller range of intensity values.

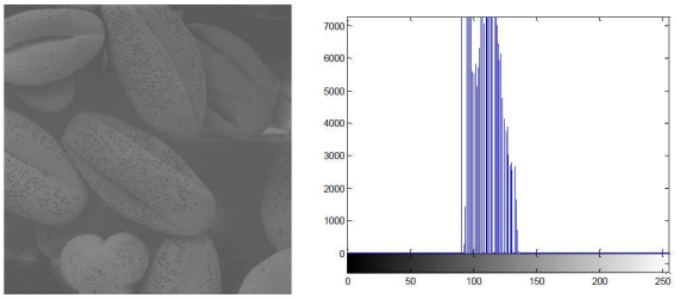

Image II.2. Image histogram of a low-contrast grayscale image

Other than visualizing the intensity distribution of an image, image histograms can be used to adjust the overall contrast and tone of an image. Techniques such as histogram equalization can be used to stretch the intensity range of an image, making the overall image appear to have more contrast because of the wider range of intensity values. Techniques such as histogram matching can also be used to adjust the contrast of an image to match a reference image.

C. *Image Enhancement*

Image enhancement is the process of improving the subjective visual quality of an image. Subjectivity is mentioned because of the nature of image interpretation which could be different depending on the observer. Enhancement of an image could be done through a variety of techniques, such as adjusting the contrast of an image, brightness, or color balance of an image. Noise removal is also one of the techniques in image enhancement which is used to reduce the noise of an image for further processing that would otherwise be badly affected by noise.

Depending on the domain of operations, whether it be spatial or frequency, methods for image enhancement could be categorized into spatial domain and frequency domain. Image enhancement techniques in the spatial domain are done by manipulating the pixels in a two-dimensional digital image directly, whereas techniques in the frequency domain are done by first converting the image into the frequency domain and manipulating the results. Converting a spatial image into the frequency domain is done through *Discrete Cosine Transform* (DCT) or *Fourier Transform* (FT).

The goal of image enhancement is to improve the visual quality of an image as previously mentioned, but sometimes can be mistaken as image restoration which refers to the process of repairing damaged or degraded images. The techniques used in both processes are different depending on the task or goal trying to be achieved.

Image II.3. Image enhancement of a low-contrast image

D. *Histogram Equalization*

Histogram equalization is one of the methods of enhancing an image using the histogram of the image. The goal of histogram equalization is to manipulate the pixel intensities in an image to be uniformly distributed. There are variants of histogram equalization, including the commonly used histogram equalization and adaptive histogram equalization.

The first variant of histogram normalizations or the normally used histogram equalization is a method of image enhancement that adjusts the contrast of an image by redistributing the pixel values. Histogram equalization is based on the idea that the pixel values in an image should be uniformly distributed across all available ranges of values. The contrast of the image would be increased by applying this method.

Histogram equalization is performed by first calculating the histogram of the image, which is then converted into bins. The next step is to calculate the cumulative histogram of the image which is used to map the pixel values of the original image to the new values which will result in a more uniformly distributed histogram. Lastly, the pixel values of the original image are replaced with the new values received from the calculations.

Adaptive histogram equalization is an image enhancement technique that is different from the normally used histogram equalization. It differs by computing several histograms, with each histogram corresponding to a section of the image, which is then used to compute the new values for pixel intensities of the image. It is designed to address the limitations of traditional histogram equalization.

Adaptive histogram equalization is able to preserve the global contrast of an image while still enhancing the local contrast of distinct sections of said image. It is a particularly useful technique for images with varying regions of contrast. It also has many variants of implementations, one of which is *Contrast Limited Adaptive Histogram Equalization* (CLAHE). CLAHE takes care of over-amplification of the contrast and operates on small regions in the image, or tiles.

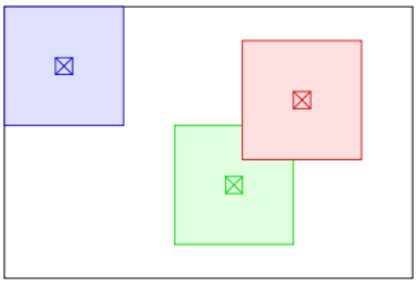

Image II.4. Visualization of the Adaptive Histogram Equalization process

E. *Binary Image*

Binary images are images that consist of only two intensity levels, typically represented as 0 or black and 1 or white. Binary images are used in many image processing techniques such as image segmentation, object recognition, and pattern recognition. It is particularly useful for images that have an object of interest or foreground that is clearly distinct from the background.

To generate a binary image, a threshold value is typically chosen, and all pixels with intensity values greater than the threshold are set to 1 or white, whereas all pixels with intensity values below the threshold are set to 0 or black. The process is often used to segment the foreground of an image from the background of the image.

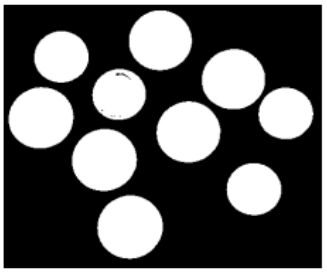

Image II.5. Binary representation of coins image

There are many methods to choose the threshold value for an image to generate a binary image. One of the methods is Otsu's method which is directly named after the inventor of the method. As can be seen from Image II.5, the result of the binarization of an image is only black or white. In the image, a specific threshold value is chosen and the explained method is applied using the specific threshold

F. Image Thinning

Image thinning is a binary image processing technique to reduce regions of an image to its skeleton to approximate the object's line. The goal of image thinning is to remove excess or redundancy of the image and to take only the essence of the image, while still preserving the quality of information for the particular task being performed.

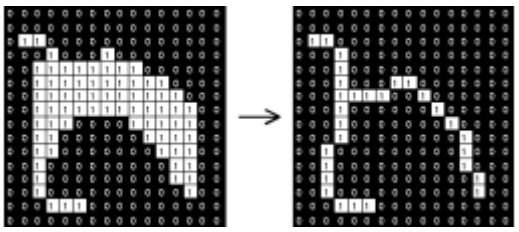

Image II.6. Result of using image thinning to obtain the skeleton of a binary image

The result of the thinned image then could be used in the step of pattern recognition or character recognition. The thinning algorithm used must not shorten the skeleton of an image and must preserve the pixels' relations and not disconnect the object's line. Essentially, the algorithm must preserve the topology and connectivity of the original object or features.

The algorithms used in image thinning use iterative processes to remove pixels from the objects in the image. Image thinning is often used as a preprocessing step in image analysis or recognition tasks and can simplify the processing of an image by reducing the thickness of an object in the image. It is used to make identifying and extracting an object's unique characteristics easier.

G. Image Segmentation

Image segmentation is an image processing operation used to partition a digital image into a collection of pixels in relation to each other. The image is divided into regions, usually separating the foreground image and the background image. By segmenting the image into separate regions, the produced image can be used to process only important parts of the image rather than using the whole image.

The segmentation of an image is based on the brightness intensity of an image, color, texture, or other aspects of an image. Segmentation of an image could use one of those aspects or a combination of multiple aspects of an image. This operation is often used before doing image or object recognition. There are several different approaches to image segmentation, including thresholding, region-based, edge-based, clustering, and deep learning.

The image segmentation approach using thresholding is a simple technique that involves picking a threshold value for pixel intensity and classifying all pixels above or below the value as an object or foreground and the background. The result of image segmentation using thresholding is a binary image.

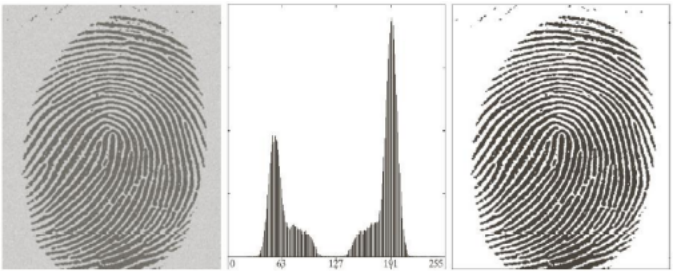

Image II.7. Image segmentation using thresholding on fingerprint image

The method used to pick a threshold value is received through histogram analysis and identifying the peak and valley of the histogram. In Image II.7, it is shown that the optimal threshold value is near 127 to completely separate the two hills of the histogram, and the result of the thresholding is a binary image that can be seen at the right part of the image. It is seen that thresholding separates the fingerprint image from the background, which can be used later for fingerprint recognition.

Techniques used in thresholding are divided into three, global thresholding, local thresholding, and adaptive thresholding. One method of global thresholding is Otsu's method, which is used to find the optimal value for the global threshold. The goal of Otsu's method is to maximize the variance between the foreground and the background. The process of Otsu's method is as follows:

1. Calculate the histogram of an image.
2. Calculate the probability of each intensity level.
3. Calculate the mean intensity of the foreground and background regions for every possible threshold value.
4. Calculate the variance between the foreground and background regions for each threshold value.
5. Choose the threshold that would maximize the variance between the foreground and background regions of the image.

H. Feature Extraction

Feature extraction is the process of identifying and extracting important and unique features from an image. The features can be used to further analyze or manipulate.There are several different approaches for feature extraction, including edge detection, corner detection, *Scale-Invariant Feature Transform* (SIFT), *Speeded-Up Robust Features* (SURF), and *Histogram of Oriented Gradients* (HOG).

SIFT is an algorithm used to detect and match local features in an image. The algorithm first extracts SIFT keypoints of objects in the image and individually compares each feature of the reference image to a new image based on the Euclidean Distance of the feature vectors. The feature

matching algorithm could also be done using *Fast Library for Approximate Nearest Neighbors* (FLANN), which uses the K-Nearest Neighbor (KNN) algorithm.

Image II.8. Keypoints generated by SIFT algorithm for fingerprint matching

As can be seen in Image II.8, the key points generated by SIFT algorithm are plotted on the thinned fingerprint image. The key points would later then be used by the FLANN matcher with a newly received fingerprint image.

III. Implementation

The fingerprint recognition algorithm used in this paper is first done by preprocessing the image. The preprocessing of the fingerprint image is first done by cropping the image to remove unwanted artifacts of the fingerprint image that is consistent throughout all the dataset [Number] used. The cropped grayscale fingerprint image is then histogram equalized using the CLAHE algorithm, an adaptive histogram equalizing algorithm.

The equalized image is then binarized by picking a threshold value using Otsu's method and separated into the foreground and background. The foreground is the fingerprint itself. The binary image is then thinned using the modified fast thinning algorithm [Number], achieving a thinned image to simplify the feature extraction process.

```
def imread(path: str) -> np.ndarray:
  return cv.imread(path)[2:-4, 2:-4, 0]

def imequalize(img: np.ndarray) -> np.ndarray:
  return cv.createCLAHE(clipLimit=2.0,
tileGridSize=(8,8)).apply(img)

def imbinarize(img: np.ndarray) -> np.ndarray:
  _, img_bin = cv.threshold(img, 0, 255,
cv.THRESH_OTSU)
  return img_bin

def fast_thin(img: np.ndarray) -> np.ndarray:
  img = morphology(img)
  img = clean_corners(img)
  img = erase_two_by_twos(img)
  img = erase_ladders(img)
  return img
```

The thinned fingerprint image is then applied SIFT algorithm to extract the key points and descriptors of the fingerprint. The key points are then used to match with a new fingerprint image received using the FLANN matching algorithm.

```
def imkeypoints(img: np.ndarray,
                keypoints: Tuple[cv.KeyPoint],
                color: int=0xFF0000) -> np.ndarray:
  return cv.drawKeypoints(img, keypoints,
outImage=None, color=color)

def match_fingerprint(ref: np.ndarray,
                      test: np.ndarray,
                      keyp_ref:
Tuple[cv.KeyPoint]=None, keyp_test:
Tuple[cv.KeyPoint]=None,
                      desc_ref: np.ndarray=None,
desc_test: np.ndarray=None,
                      thresh: float=0.90) ->
Tuple[bool, float, Optional[np.ndarray]]:
  if keyp_ref is None or keyp_test is None:
    keyp_ref, desc_ref = imextract(ref)
    keyp_test, desc_test = imextract(test)
  matches = cv.FlannBasedMatcher(dict(algorithm=1,
trees=10), dict())\
              .knnMatch(desc_ref, desc_test, k=2)
  match_points = [p for p, q in matches if p.distance
< q.distance * 0.1]
  n_matchpoints = len(match_points)
  n_keypoints = min(len(keyp_ref), len(keyp_test))
  match_percentage = n_matchpoints / n_keypoints
  bool_match = match_percentage >= thresh
  match_img = gen_immatch(ref, test, keyp_ref,
keyp_test, match_points)
  return bool_match, match_percentage, match_img
```

Image III.1. Preprocessing step of the image

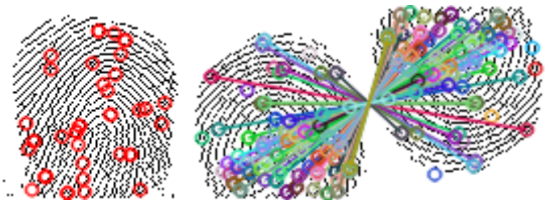

Image III.2. Feature matching of a fingerprint with its vertically-flipped image

As can be seen in Image III.2, the generated SIFT key points are used to match with a new image. In this case, the flipped version of the same fingerprint image. The algorithm succeeds in matching the same fingerprint with a modified image of the same fingerprint. The fingerprint matching algorithm results in a percentage level of matched key points between the two fingerprint images. A threshold is picked to determine if the matching is successful or unsuccessful. In most cases, a threshold value of 90 percent is used to match two fingerprints.

IV. Analysis

From the explanation of the implementation process above, it is shown that the algorithm is successful in matching the same fingerprints that are slightly transformed. The algorithm also achieves 100% accuracy between all the datasets by recognizing only the same fingerprint used as a reference. But, the algorithm does not find the same success in the heavily-altered fingerprint images in the dataset [Number]. The algorithm gives a very low confidence level when matching with heavily-altered fingerprints, but gives a satisfying confidence level for the lightly-altered fingerprints.

```
DATA_PATH = '../data/real/'
files = os.listdir(DATA_PATH)
verified = []
for file in files:
  img_test = imread(DATA_PATH + file)
  _, _, img_test_thin = improc(img_test)
  keyp_test, desc_test = imextract(img_test_thin)
  bool_match, percentage, img =
match_fingerprint(img_thin, img_test_thin, keyp,
    keyp_test, desc, desc_test)
  if bool_match: verified.append((file,
percentage*100, img))

print(f'Accuracy: {((len(files) + 1 - len(verified)) /
len(files) * 100):.2f}')
```

```
Accuracy: 100.00
```

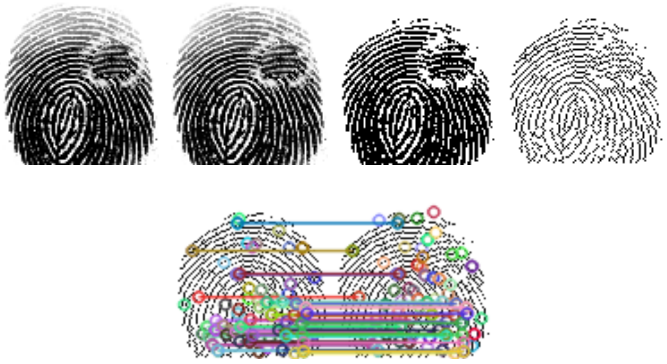

Image IV.1. The algorithm recognizing the lightly-altered fingerprint with 58% confidence

Image IV.2. The algorithm recognizing the heavily-altered fingerprint with 1% confidence

V. Conclusion

Using image processing techniques including cropping, adaptive histogram equalization, binarizing using Otsu's method, and thinning algorithm [Number] to preprocess a fingerprint image. Then applying the SIFT algorithm to extract distinct or unique characteristics of the thinned fingerprint image and matching with a new fingerprint image using a FLANN-based matching algorithm produces satisfying results for an ideal image and a lightly-altered image of a fingerprint. But, for heavily-altered fingerprint images, the algorithm does not produce a satisfactory result, only achieving a 1% confidence level. While the algorithm is fast in recognizing and identifying fingerprint images, the algorithm still has a lot of vulnerabilities and still, has much room for improvement. But, the algorithm might be a foundation for a better algorithm for future implementations.

Video Link at Youtube

https://youtu.be/vk-QRwVh6SA

Source Code Link

github.com/naufalsuryasumirat/if4073-makalah-13519135

Acknowledgment

The author would like to thank The One Almighty God. While this paper is far from perfection and far from the optimal quality of papers, the paper is written with the best efforts possible from the author. This paper could not have been written without the help of The One Almighty God, the motivation from the author's family, friends, and the honorable lecturer of Image Interpretation and Processing IF4073, Mr. Rinaldi Munir. Many thanks to everyone who helped the author directly or indirectly by motivating the author.

References

[1] Sokoto Coventry Fingerprint Dataset (SOCOFing), https://www.kaggle.com/datasets/ruizgara/socofing?datasetId=38300&sortBy=voteCount, accessed on December 15, 2022.

[2] Fast Thinning Algorithm for Fingerprint Library, https://github.com/Schukuratsu/Python-cv2-fast-thinning-algorithm, accessed on December 15, 2022.

[3] What is Biometrics? How is it Used in Security? https://www.kaspersky.com/resource-center/definitions/biometrics, accessed on December 17, 2022.

[4] Nilar H., Thet N. H. Image Processing Techniques for Fingerprint Identification and Classification - A Review, https://www.ijtsrd.com/papers/ijtsrd26761.pdf, accessed on December 17, 2022.

[5] CLAHE Histogram Equalization - OpenCV, https://www.geeksforgeeks.org/clahe-histogram-eqalization-opencv/, accessed on December 15, 2022.

[6] Feature Matching with FLANN documentation, https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html, accessed on December 15, 2022.

[7] Rinaldi M. Pengantar Interpretasi dan Pengolahan Citra (Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/01-Pengantar-Pengolahan-Citra-Bag1-2022.pdf, accessed on December 17, 2022.

[8] Rinaldi M. Pembentukan Citra dan Digitatlisasi Citra. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/03-Pembentukan-Citra-dan-Digitalisasi-Citra-2022.pdf, accessed on December 17, 2022.

[9] Rinaldi M. Histogram Citra. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2020-2021/06-Image-Histogram-2021.pdf, accessed on December 17, 2022.

[10] Rinaldi M. Image Enhancement. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2022-2023/09-Image-Enhancement-Bagian2-2022.pdf, accessed on December 17, 2022.

[11] Rinaldi M. Citra Biner. https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/20-Citra-Biner-2021.pdf, accessed on December 17, 2022.

[12] Rinaldi M. Segmentasi Citra (Bagian 1). https://informatika.stei.itb.ac.id/~rinaldi.munir/Citra/2021-2022/20-Citra-Biner-2021.pdf, accessed on December 17, 2022.

Statement

I hereby declare that this paper is my original work, not an adaptation or translation of someone else's paper, and not a result of plagiarism.

Bandung, 17 December 2022

Naufal Alexander Suryasumriat
13519135