EVALUATION OF PERCEPTUAL HASHING ALGORITHMS ON BIOMETRIC FINGERPRINTS

Rakshitha Nagendrappa

A Capstone Project Submitted to the
University of North Carolina Wilmington in Partial Fulfillment
of the Requirements for the Degree of
Master of Science

Department of Computer Science
Department of Information Systems and Operations Management

University of North Carolina Wilmington

2023

Approved by

Advisory Committee

_____	_____
Dr. Geoffrey Stoker	Dr. Lucas Layman

_____
Dr. Seyedhossien Siadati

_____
Dr. Sima Jafarikhah, Chair

Accepted By

_____
Dean, Graduate School

TABLE OF CONTENTS

ABSTRACT

Evaluation of Perceptual Hashing Algorithms on Biometric Fingerprints. Nagendrappa, Rakshitha, 2023. Capstone Paper, University of North Carolina Wilmington.

Perceptual hash (PH) algorithms are image hashing algorithms that output visual content-based hashes. Unlike cryptographic hashes, where a minute difference in input data has an enormous difference in its hashes, PH algorithms change proportionately to the change in their inputs, i.e., trivial differences between two images result in insignificant differences between their hashes.PH algorithm's application over the years has spiked; for example, in digital forgery detection, data authentication, and preventing nonconsensual data propagation.

This study compares several well-known and publicly available hash algorithms for biometric fingerprint authentication, including Average Hash, Discrete Cosine Transform (DCT) based Hash, Difference Hash, and Wavelet-based Hash. To evaluate the performance of these algorithms, we consider critical factors like **discrimination capability** and **robustness** to variations and distortions applied on a dataset of biometric fingerprints. The outcome of this study will be the best hash algorithm suitable for biometric fingerprints. Our work is to initiate a pipeline to investigate biometric authentication using PH algorithms.

Key Words: Cryptographic Hashes, Perceptual Hashing, Biometric Fingerprints.

LIST OF TABLES

Page

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The saying ‘picture is worth a thousand words’ has never been more suited for this digitalized world where multimedia, electronic devices, etc., play a vital role, especially in people’s communication and lives. Along with it comes the struggle to safeguard people’s personal and sensitive information, which may be anything, including their images and unique features, from being misused, present on the internet, or sent through media. If we consider images, it is easy to tell if two images are visually similar by looking at them as humans have a contextual understanding of the picture. But for a system to do it without contextual knowledge, it is a challenging task.

Many platforms have policies against uploading explicit images of anyone without their consent and will take down such pictures upon an affected user’s request. However, substantial harm may have already occurred when a victim discovers that such an image is online. Ideally, these platforms would prevent such images from being uploaded in the first place. However, this task is a challenging one as a malicious user can apply several types of manipulation over the data to fool the server while their semantic meaning remains the same. Therefore, it is challenging to build a denylist of images using traditional forms of hashing (such as cryptographic collision-resistant hashing), as perceptually similar images will almost certainly produce a completely different hash. To address the problems outlined above, the concept of perceptual hashing has been defined (Jafarikhah).

1.1 Background and Motivation

Hashing, in general terms, is a process of representing information in any form, like images, videos, or text, into a string called hash by the computer. It can be used to detect duplicates, verify identity, etc. The algorithm which does this job is known as the

hash algorithm. The two kinds of hashes are cryptographic hash and perceptual hashes. The former is based on the idea that a minute difference in the input creates an entirely different result: the avalanche effect, for example, SHA256, MD5 (Evans). Conversely, the latter means minor changes in the original image will not affect the overall visual perception of the content; hence minor changes in the hash value will be the result.

The commercial perceptual hash algorithms like Microsoft's PhotoDNA, Apple's NeuralHash, Facebook's PDQ, CSAI match, etc., for detecting Child Sexual Abuse Material (ofcom) are owned by the companies and are kept secret and users must send their raw data to protect their personal information. The companies promise to generate the hash value (tag) of the user's sensitive data, discard the data, and only store the generated tag in their database as a sample registered tag. Clearly, this raises privacy concerns.

More formally, generating the hash described above is defined as follows. The three algorithms that make up the public-key encryption system are the key generation algorithm, the encryption algorithm, and the decryption algorithm. The key generation algorithm takes a security parameter and outputs a public key pair. The encryption algorithm takes a message and the public key and outputs a ciphertext. The decryption algorithm takes a ciphertext and the secret key and outputs the message. Figure 1 illustrates this at a high level.

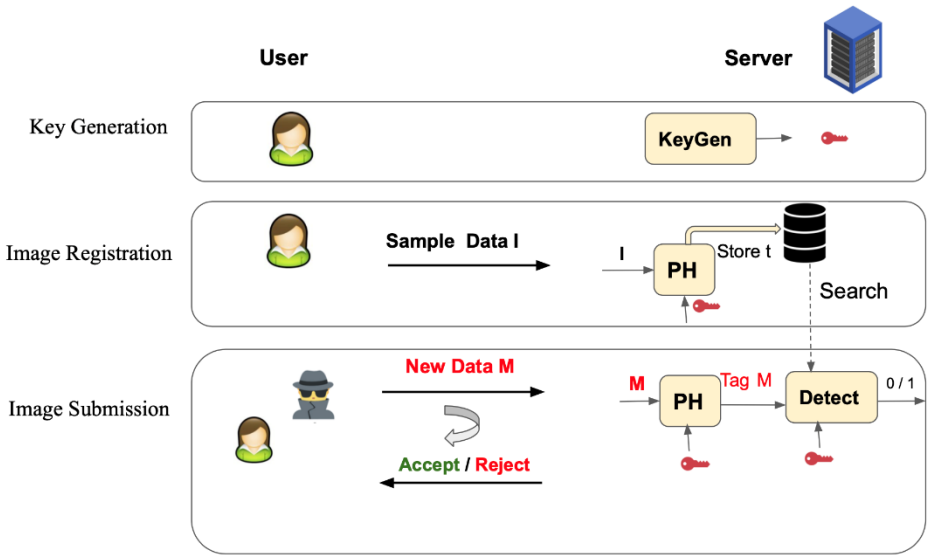

Figure 1. Keyed Perceptual Hash Source (Jafarikhah).

As mentioned in (Gennaro) it is imperative to stop nonconsensual use of inappropriate data and public data sharing. Making users have the raw data, compute the hash on their end, and send it to the companies to store for future purposes would solve the privacy concern which arises with secret perceptual hash. A sufficiently robust perceptual hashing algorithm will detect if two different files represent perceptually the same image, even without an adversary. To achieve this, hash algorithms must first be evaluated, and then apply the publicly evaluable part on top of the hash algorithms. These publicly evaluatable perceptual hash functions allow a user to compute the perceptual hash of an image using a public key, while only the detection algorithm will use the secret key. Since all the excellent computing hash algorithms are being held secretly, we will evaluate the publicly available ones. Some publicly available algorithms include Average Hash, Discrete Cosine Transform (DCT)-based hash, Difference Hash, Discrete Wavelet Transform (DWT)-based hash, Block Hash, etc. (Ramos) whose source code is readily available and open source.

In this project, we will be considering those hash algorithms, and the main idea of this project is to apply biometric fingerprints as input to the perceptual hashes because, with biometric fingerprints, it is even more critical to secure them as they are unique to every individual because of which it is used for identification and authentication purposes. The outcome of this experiment is to find the best perceptual hash algorithm that works better for biometric fingerprints for authentication purposes. PH algorithms in biometric fingerprint authentication address the challenges posed by distortions and deformations that may occur over time. Regardless of the specifically available dataset of fingerprints and the PH algorithms applied, our work initiates a pipeline to explore more on this topic and opens an opportunity in our department to learn and investigate more on the domain of cryptography and applications of PH algorithms to preserve privacy and security.

1.2 Objective

In this work, we will create a pipeline to comprehensively compare and evaluate four open-source algorithms mentioned above. Their performance is evaluated based on parameters like robustness and discrimination property. We will be applying these open-source hash algorithms for biometric fingerprint authentication. The performance of each algorithm is assessed using a dataset of biometric fingerprint images with varying levels of quality and distortion.

1.3 Outline

Following this chapter, the document is divided into four chapters. The second chapter provides a literature review of perceptual hashing and various hash algorithms which serve as the basis for this project. The third chapter provides information about the methods and steps we have planned to use. The fourth chapter describes the results of the

evaluation. Finally, the fifth chapter tells us about the conclusions drawn from the work and future work, along with the limitations faced, if any.

CHAPTER 2: LITERATURE REVIEW AND ANALYSIS

This chapter will provide information about perceptual hashing, different techniques, and how each extract hashes along with a little of the research that happened in this area.

2.1 Perceptual Hashing

We would like to mention an example of the necessity to discover perceptual hashing before looking into its details. In 2018, a scary image of a woman circulated on the web, and a user duplicated it and started to force small children to do hazardous jobs. It even became popular on YouTube targeting small children. YouTube took out all the content related to the duplicated image, but its variants continue to make children obsessed for months (O'Malley). This can be one of the drawbacks of cryptographic hashes as they could not find similar content and not effectively remove all variants.

Perceptual hashing is a technique in image hashing that can be used to determine if two multimedia files are similar based on the hash produced for both. Some critical applications of perceptual hash include image recognition, authentication, image search, data duplication, digital forensics, and other applications. The main goal of these hashing algorithms is to generate a similar hash for similar inputs.

Some of the properties for a hash algorithm to perform perceptual hashing according to (Farid)includes:

1) *Resilient or Robust:* Should recognize similar or identical content even when there are minute variations between the content as they are meant to be insensitive to minor changes in the content.
2) *Distinct:* Should be able to distinguish between similar but not identical content.
3) *Efficient:* Should have low computational cost and time.

4) *Nonreversible:* Having the hash, it should not be possible to get back the input.
5) *Deterministic:* The hash algorithm must give the same hash for an image regardless of how many times it is inputted.

Perceptual hashing usually has three main steps as follows:

1) *Preprocessing.* involves resizing, converting images to grayscale, etc.
2) *Hash Generating.* the "hash" of the image is created, which is a hexadecimal string of fixed length of the visual content.
3) *Decision making.* The final process is decision making, where the resulting hash of the image will be compared to other hashes to determine the similarity between two hashes using Hamming Distance, where the number of differing bits represents the degree of dissimilarity between the two images. Comparison can be made using attributes like Euclidian Distance, Hamming Distance, Normalized Hamming Distance, Bit Error Rate, etc. Published perceptual hash algorithms will use different distance metrics to determine similarity (Priyanka Samanta).

The publicly available hash algorithms we have chosen and their process of how the hash is generated for comparison is as follows:

1) *Average Hash.* It *i*s also known as mean hash based on representing an image as a hash based on the average value of each pixel. It first converts the image into grayscale, resizes it into 8x8 resolution which can be seen in below Figure 2, then calculates the average value of the 64 pixels in the grayscale image, and based on this average value, the pixels in the grayscale image are either set to 1 (if the pixel value is greater than the average) or 0 (if the pixel value is less than or equal to the average) to form a hash value which is used for comparison by calculating the Hamming Distance between two hashes (hackerfactor.com).

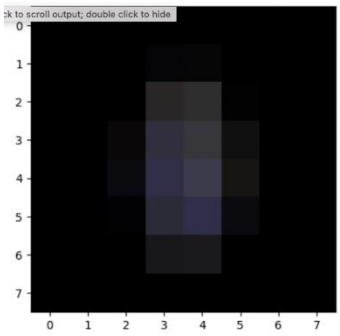

Figure 2. 8X8 Resolution Image.

2) *Difference Hash.* It is like the Average Hash technique, but instead of average pixel color, it considers the difference between neighboring pixels of 8x8 resolution, i.e., gradient. This hash algorithm first converts the image into grayscale and resizes it to a 9x8 resolution image as shown below in Figure 3; then, it calculates the difference between the value of each pair of horizontally neighboring pixels and determines whether the left pixel is brighter than the right pixel. Based on these differences, the pixels in the grayscale image are either set to 1 (if the difference is positive or brighter) or 0 otherwise to form a hash value that can be compared with other hashes for similarity checking (hackerfactor.com).

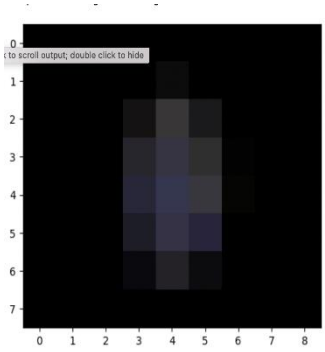

Figure 3: 9X8 Resolution Image.

3) *DCT-based Hash.* It uses Discrete Cosine Transform(DCT) to represent the hash of the image; the idea behind this algorithm is to transform an image into a set of cosine waves of increasing frequency and use the most significant frequency coefficients to generate the hash, to do this it first converts the image into greyscale, smoothens and resizes it to 32x32 resolution as shown below in Figure 4 and selects the significant coefficients and generates the binary value based on whether they are above or below the average value to form a 64-bit hash later which will be used for comparison (hackerfactor.com).

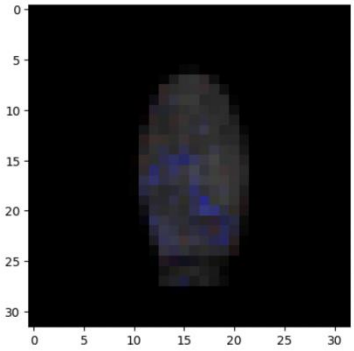

Figure 4: 32X32 Resolution Image.

4) *Wavelet-Based Hash.* It is like DCT based hash, but instead of DCT, it uses Discrete Wavelet Transform to turn the greyscale image into scaled and shifted Haar Wavelets. The rest of the process is the same as the phash (content-blockchain.org).

Earlier researchers worked on symmetric hash algorithms and their variations to secure fingerprint features during identification and verification (Tulyakov) (Kumar). Despite the importance of their work, their schemes are not as robust as it had slight loss in the accuracy of the secure system as compared to the straight version that can be attributed to factors such as reduction in the number of points being matched.

A few research based on perceptual hash algorithms were applied to fingerprint authentication; the paper (Hamadouche) shows various metrics chosen to evaluate the perceptual hash algorithms, like robustness, by imposing content manipulation techniques on the raw fingerprint images, authentication, etc. According to the paper, different PH algorithms showed better performance under other metrics. For example, Average Hash showed better performances for robustness, DCT based hash showed better results for authentication. Normalized Hamming Distance is calculated and used to derive the conclusions. Raw fingerprint images have been used in this research, so the input is considered a normal fingerprint image. In our study, we have also considered every fingerprint's minute details, called minutiae, which include ridges and valleys with terminations and bifurcations.

One of the papers suggests utilizing perceptual hashing to enhance the extraction of minutiae from fingerprint images that can withstand context-preserving procedures. The hash extraction follows wavelet transform and Singular Value Decomposition (SVD). The metrics used to calculate the effectiveness of the algorithms are Structural Similarity Index Measurement (SSIM) and Peak Signal to Noise Ratio (PSNR). The experimental results indicate that the technique resists image-processing operations and geometric attacks (Rani).

The need to develop various perceptual hash algorithms arose because other perceptual hash algorithms were developed based on the different requirements, applications, and drawbacks of older PH algorithms. Some algorithms were developed to be more robust for image manipulations, and some for duplicate image detection (Toropchin), while others were suitable for other applications. So, based on the application and input, a particular PH was chosen.

2.2 Terminology Related to Fingerprints

2.2.1 Minutiae Features. These are the unique features present in every fingerprint for every person. These features include terminations and bifurcations of ridges and valleys in the fingers. Because of these unique features, fingerprints are used for identification. Before using them for this purpose, the authenticity of these fingerprints must be verified. With the widespread use of fingerprints for biometric identification, their authenticity has become a crucial research issue due to the impending fraudulent use of duplicate images. One technique that can be used to verify the authenticity of fingerprints is perceptual hashing.

2.2.2 Orientation Maps. Orientation maps are the type of fingerprint preprocessing to get the orientation of fingerprint images using the degree of angles and inclinations of ridges and valleys of a fingerprint.

CHAPTER 3: METHODOLOGY

This chapter talks about the different steps of the project, how we will be achieving it and the analysis of how the evaluation is done. We will use Jupyter notebook, Visual Studio Code and GitHub as version control systems for the Python code to conduct this evaluation. We will create a pipeline for comparing any perceptual hash algorithms. In our project, we have taken four hash algorithms and a dataset of fingerprints mentioned below. The source code will be available on GitHub along with the outputs. The results will be visualized for a better understanding of the evaluation.

3.1 Fingerprints Dataset

The database chosen for this project is "The Tsinghua Distorted Fingerprint Database," and it has 320 (200 trained and 120 tested) pairs of normal fingerprints and distorted ones from different fingers, as shown in Figure 5 (Xuanbin Si).

Figure 5. Normal and Distorted Fingerprints.

3.2 Minutiae Extraction

For minutiae extraction, the following processes will be followed:

The fingerprints in the above database are the raw form of data, and they cannot be stored in the database in their raw form because of privacy concerns. These fingerprints must be converted by extracting unique features into a fixed-

length string called hash to keep them on the server or with the vendors.

3.2.1 Fingerprint Enhancement. Fingerprints will first go through the enhancement stage, where fingerprint quality is increased for easier identification of unique characteristics. This stage is necessary when the original fingerprint is of mediocre quality. The fingerprint enhancement process typically consists of the following stages:

1. *Image pre-processing.* This stage involves removing any noise from the image. This can be done using techniques such as image smoothing, denoising, and histogram equalization.
2. *Ridge enhancement.* This stage involves increasing the contrast between the ridges and valleys of the fingerprint image. This can be done using techniques such as adaptive filtering, Gabor filtering, or morphological filtering. For this experiment, Gabor filtering is used to filter the ridges.
3. *Ridge orientation enhancement.* This stage involves enhancing the ridge orientation field, which is the direction of the ridges in the fingerprint image. This can be done using techniques such as gradient-based methods, Fourier methods, or Gabor filtering.

The output of this stage will be an enhanced image of fingerprints with ridges and valleys, along with terminations and bifurcations being visible as shown in Figure 6. We will store these outputs in different folders with normal and distorted fingerprints. We will use a Python library called fingerprint_enhancer (Deshmukh, github.com).

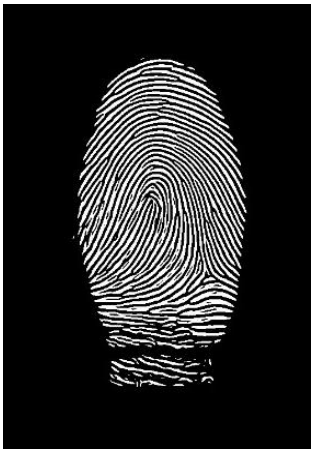

Figure 6. Enhanced Fingerprint.

3.2.2 Fingerprint Feature Extraction. The fingerprint enhancement output will be considered input for this stage. In this stage, each image's distinguishing characteristics will be highlighted to provide output to be robust and accurate images for matching processes.

The process of fingerprint feature extraction typically involves several stages, including image pre-processing, ridge segmentation, ridge orientation field estimation, minutiae detection, and quality control. Each stage of the process aims to remove any noise from the image, highlight the key features of the fingerprint, and create a robust and accurate representation of the fingerprint for matching purposes. The final output of how a feature-extracted fingerprint image can be seen in Figure 7. The fingerprint feature extraction will be performed using the fingerprint_feature_extractor python library (Deshmukh, github.com).

Figure 7. Feature Extracted Fingerprint.

3.3 Orientation Map Extraction

The orientation map of the fingerprint is obtained using gradient analysis by inputting a fingerprint image, converting it to grayscale, calculating the angles of each block, and drawing the calculated angles on an image.

1. *Image enhancement.* To emphasize the ridge and valley features, the input fingerprint image will be first enhanced using Gabor filter.
2. *Calculation of Gradients.* The Sobel operator is used to determine the gradients of the improved image in x and y directions.
3. *Estimating orientation.* The gradient information is used to estimate each pixel's orientation.
4. *Orientation field smoothening.* It is done with a Gaussian filter to minimize noise and irregularities.
5. *Orientation quantization.* To construct an orientation map, the smoothed orientation field is quantized into a small number of angles (usually 8 or 16).
6. *Orientation Refinement.* The orientation consistency check is then used to improve the orientation map. In this stage, it is checked if each pixel's orientation values are constant by comparing them to those of the pixels around it.

The final orientation map fingerprint output can be seen in Figure 8 below. The orientation map extraction will be performed using code available on GitHub (Victor).

Figure 8. Orientation Maps of Fingerprint.

3.4 Metrics Used for Comparison

Hamming Distance. This is derived by measuring two binary hashes, comparing them bit-by-bit (0 and 1), and getting the number of different bits in both hashes. For example, for two binary strings (hashes) X and Y, Hamming Distance can be calculated as follows:

$$\text{Hamming Distance} = \sum |Xi - Yi|$$

Where i is the bit of the string (hash), the higher the Hamming Distance result, the more distinct the strings are.

3.5 Comparison of Hash Algorithms

We will be conducting a comparison based on two diverse kinds of fingerprint data. The comparison that other researchers have already conducted is based on the pixel matrix of fingerprint images directly, i.e., by gray scaling it and reducing its size, but in this project, we have extracted minutiae for the fingerprint image for comparison, which can be seen in Figure 4 above, and in the second part of the comparison we will compare based on the orientation map of the fingerprint as shown in Figure 5 above, which will be explained below along with the metric chosen for the comparison.

3.5.1 Comparison of Robustness. An efficient hash algorithm must be robust and resistant to any kind of manipulation or distortions. This feature of an algorithm can be evaluated using robustness. In this project, we will conduct an experiment to evaluate four hash algorithms. For the minutiae-based comparison, 320 pairs of features extracted images of normal and distorted fingerprint images will be the input. Hashes will be generated according to the type of hash technique provided, and Hamming Distances between each normal fingerprint image and its distorted image will be stored.

And for the orientation map type of comparison, 320 pairs of images of normal and distorted fingerprint images will be given as input, and hashes will be generated according to the type of hash technique provided out of four and their Hamming Distances between each normal fingerprint image, and its distorted image will be stored.

3.5.2 Comparison of Discrimination Probability. For minutiae-based fingerprint image comparison, all the possible pairs of extracted fingerprint images that are visually different will be used for comparison; in our case, it will be 51040 sets (320 images x (320-1)/2, hashes will be generated according to the type of hash technique given and Hamming Distance between both the images will be stored. Likewise, the comparison is done using all possible pairs of orientation map fingerprint images for orientation map type comparison.

3.6 Evaluation and Visualization

Evaluation of the results is done based on the threshold. A series of thresholds will be passed, the least will be the minimum Hamming Distance recorded in this project and the highest of the series will be the maximum Hamming Distance (HD) recorded, the degree of robustness and discrimination capability for each threshold will be calculated.

The specific calculation formulas of Probability of Robustness and Discrimination capability are defined as follows:

$$P_{Robustness}(HD < Threshold) = n1/N1$$

$$P_{Discrimination\ Capability}(HD > Threshold) = n2/N2$$

where n1 and N1, respectively, represent the number of similar images and the total number of images, n2 and N2, respectively, represent the number of dissimilar images and the total number of images.

The results will be visualized by plotting the graphs for each threshold along with robustness and discrimination capability using a line graph. The resultant graph must show that with increasing level of discrimination capability, the hash algorithm loses its robustness. A correlation between robustness and discrimination capability is also plotted. This provides a better understanding of the output.

3.7 Pipeline of the Project

The source code for this project will be made open source, built using Python on Visual Studio and Jupyter Notebook and will be available on GitHub. First, the project will be divided into smaller processes, and then one process's output will be fed as the input to the next consecutive process. The pipeline process flow of the project can be seen in Figure 9. In our project, the small processes involve the following:

1. Fingerprint Enhancement process
2. Fingerprint Feature Extraction process
3. Orientation Maps Extraction process
4. Hash Generation process
5. Hamming Distance Calculation
6. Probability calculation using thresholds

7. Performance Plotting

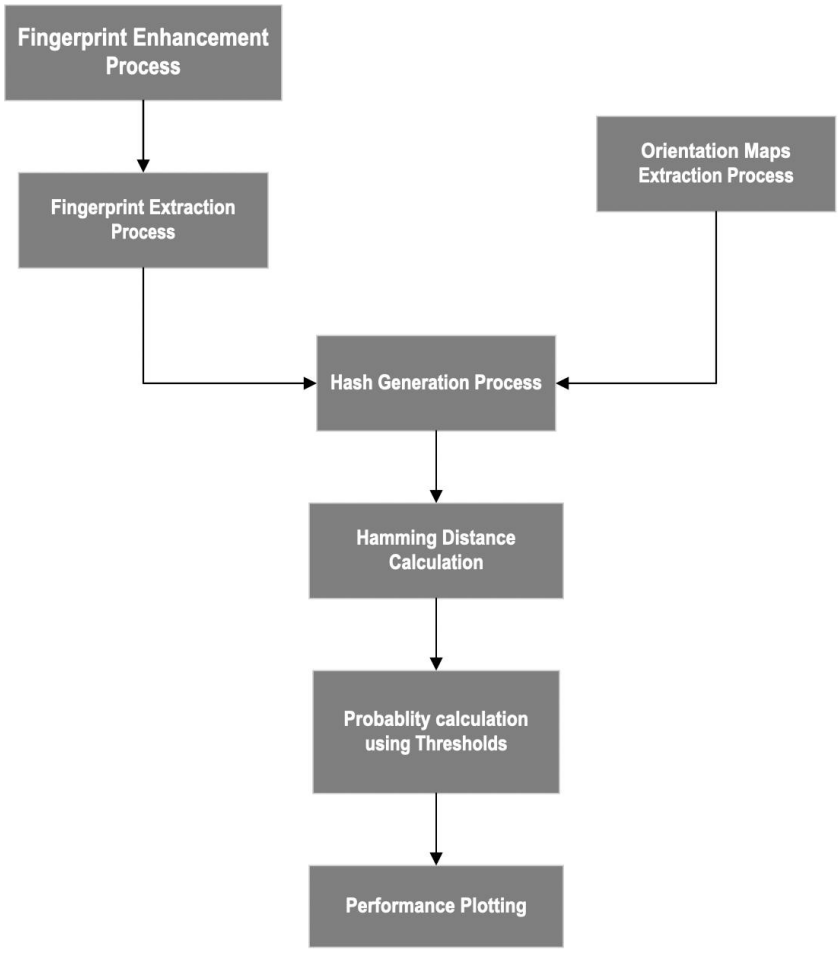

Figure 9. Pipeline Architecture.

CHAPTER 4: OUTLINE OF COMPLETED PROJECT

This chapter outlines the experimental results of the project. The entire code for this project can be found on GitHub repository. The results got for the evaluation are very interesting and before looking at the results first let us see the outputs of each process of the pipeline. It is very important to know each process's outputs before the evaluation results.

4.1 Fingerprint Enhancement Process

The input we provided to this process was the raw images folder we got from the dataset; the output of this process is a folder with enhanced fingerprints as shown in Figure 10 below.

Figure 10. Raw Fingerprint and Enhanced Normal Fingerprint.

4.2 Fingerprint Feature Extraction Process

We gave enhanced images got from the above step as the input for this process, and the output got is a folder of feature-extracted fingerprints, an example of how the feature-extracted fingerprints look is shown in Figure 11 below. Like before, these images will be stored in extracted folders separately for normal and distorted fingerprints.

Figure 11. Enhance Fingerprint and Minutiae Feature of an Extracted Normal Fingerprint.

4.3 Orientation Maps Extraction Process

The input we gave for this process was again a raw image folder as this is the second type of fingerprint representation, and the output we got was a folder of orientation map of fingerprints which will look as shown in Figure 12. Same as above, the files are stored in normal and distorted folders accordingly.

Figure 12. Raw Fingerprint and Orientation Map of the Fingerprint

4.4 Hash Generation Process

This process takes the extracted or orientation map fingerprint image of normal and/or distorted fingerprint image folder from the previous processes to generate hashes for each image with respect to what hash algorithm is inputted, we store the results in the Hashes folder as a CSV file. Hash is generated using a library called ImageHash

(pypi.org). This image hash library includes all the hash algorithms which we have considered along with HSV color hashing (color hash) and Crop-resistant hashing. The image hash library involves methods to pre-process the images before generating the hash values.

A few examples of the hashes generated using Average Hash are shown in Table 1.

Table 1

Example of Hashes Generated for Fingerprint Images

F_id	Hash
f_63	00081c1c1c1c0c0c
f_189	001c1c1c1c081818
f_77	00181c3c3c1c1c0c
f_162	0018383c3c3c0000

4.5 Hamming Distance Calculation

Once the hashes are generated, the CSV file having hashes were fed as input for this process. For checking for robustness, a CSV file of normal and distorted hashes of a particular hash algorithm was given as input, and for discrimination capability, a single CSV file was given as input as it calculates Hamming Distance with all other different fingerprint hashes present. A few examples of Hamming Distance values for robustness and discrimination of Average Hash are given in Tables 2 and 3.

Table 2

Example of Hamming Distance for Robustness

F_id	Hamming Distance
f_63	10
f_189	12
f_77	16
f_162	6

Table 3

Example of Hamming Distance for Discrimination Capability

F_id1	F_id2	Hamming Distance
f_63	f_189	4
f_63	f_77	10
f_63	f_162	6
f_63	f_176	5

4.6 Probability Calculation Using Thresholds

The Hamming Distances of the hashes got in the previous step were given as input for this process. This process gives, for a particular threshold, the probability of similar images and dissimilar images for a given hash algorithm as shown in Tables 4 and 5.

Table 4

Example of Similarity Calculations for Average Hash

Threshold	Probability of similar fingerprints
3	0.1187
6	0.4125
9	0.7031
12	0.8906
15	0.9656

Table 5

Example of Dissimilarity Calculations for Average Hash

Threshold	Probability of dissimilar fingerprints
3	0.9499
6	0.7129
9	0.3727
12	0.1333
15	0.0402
18	0.0122
21	0.0049
24	0.0016
27	0.0002

4.7 Performance Plotting

Once the calculations for similarity and dissimilarity were obtained, they were fed into a data frame, and a graph was plotted which shows both robustness and discrimination capability for a series of thresholds. Once the graph is plotted, the interaction between the robustness and discrimination capability was plotted to see which algorithm gave a result that is best of all other algorithms.

4.8 Evaluation of Hash Algorithms

4.8.1 Minutiae Feature-Based Comparison

4.8.1.1 Average Hash Results. Figure 13 below depicts the robustness and discriminations scores for Average Hash. From the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold value is between 6-9 (7.47) and we can consider the nearest value 9 from the threshold series.

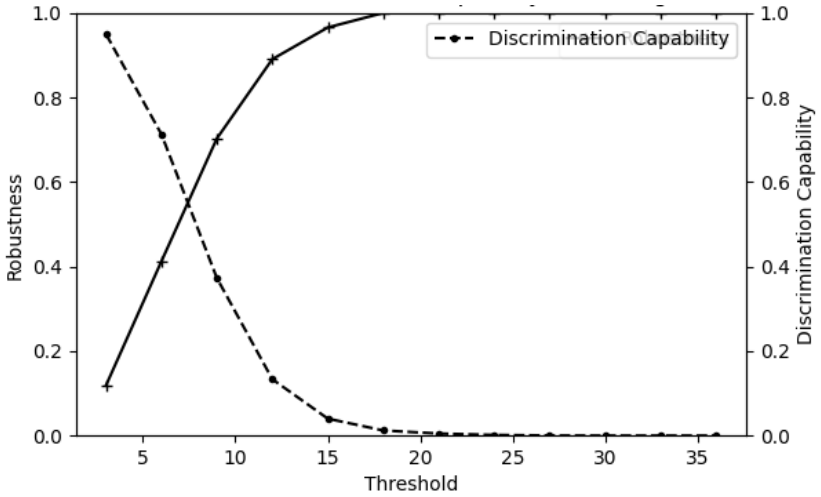

Figure 13. Robustness and Discrimination Capability for Average Hash.

The area under the curve shown in Figure 14 is 0.456 per unit of square.

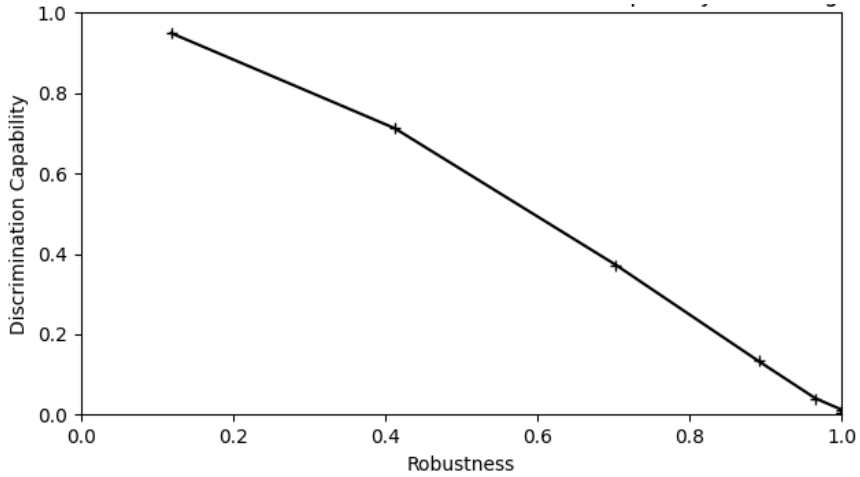

Figure 14. Interaction between Robustness and Discrimination Capability for Average Hash.

4.8.1.2 Difference Hash Results. Figure 15 below depicts the robustness and discriminations scores for Difference Hash. From the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold is 12 in series.

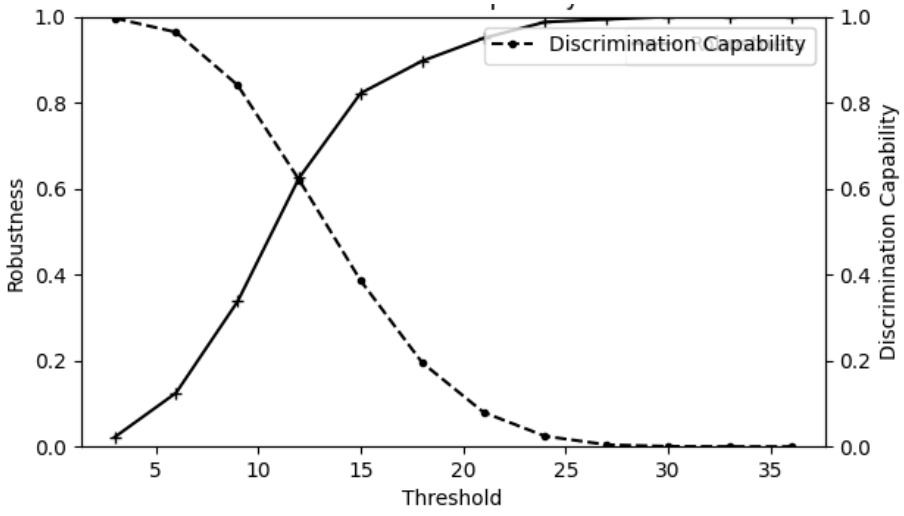

Figure 15. Robustness and Discrimination Capability for Difference Hash.

The area under the curve shown in Figure 16 is 0.533 per unit square.

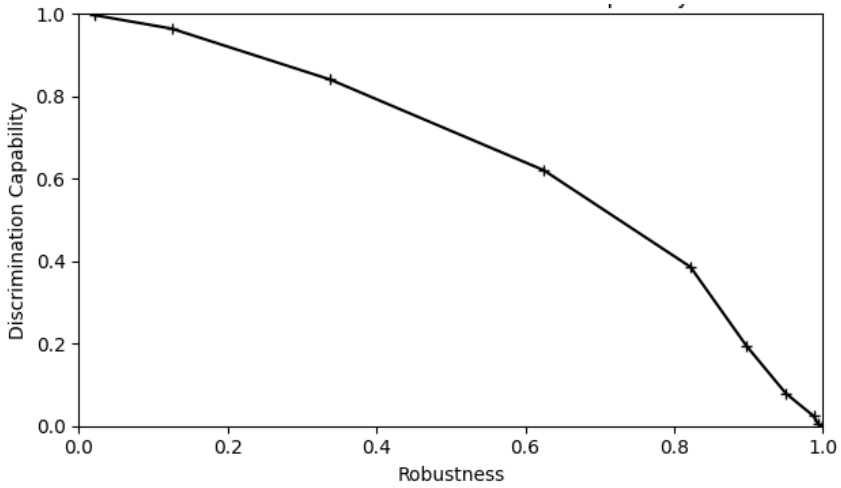

Figure 16. Interaction between Robustness and Discrimination Capability for Difference Hash.

4.8.1.3 DCT-based Hash Results. Figure 17 depicts the robustness and discriminations scores for Average Hash. From the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold value is 27 from the threshold series.

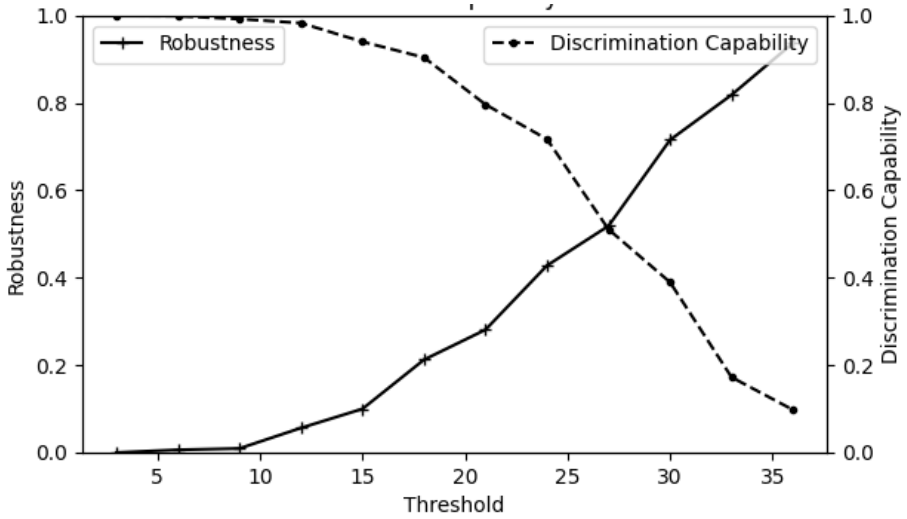

Figure 17. Robustness and Discrimination Capability for DCT-based Hash.

The area under the curve shown in Figure 18 is 0.502 per unit square.

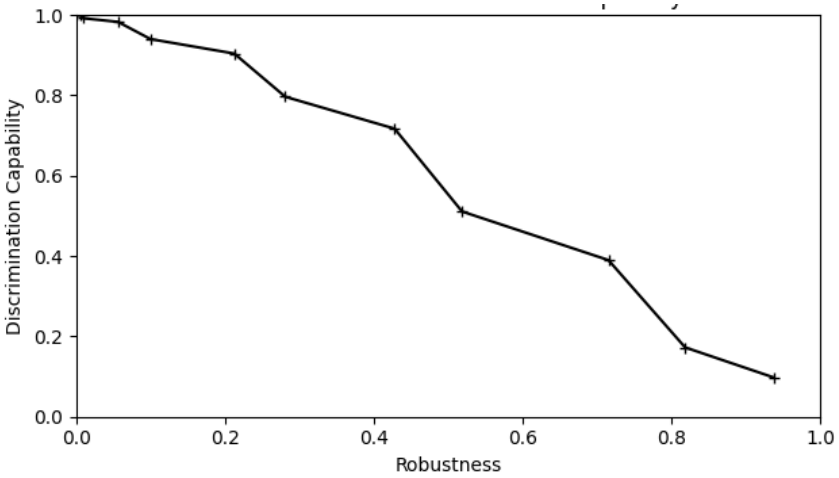

Figure 18. Interaction between Robustness and Discrimination Capability for DCT-based Hash.

4.8.1.4 DWT-based Hash Results. Figure 19 below depicts the robustness and discriminations scores for Difference Hash, from the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold is between 9-12(9.37), we can consider the nearest value 9 from the threshold series.

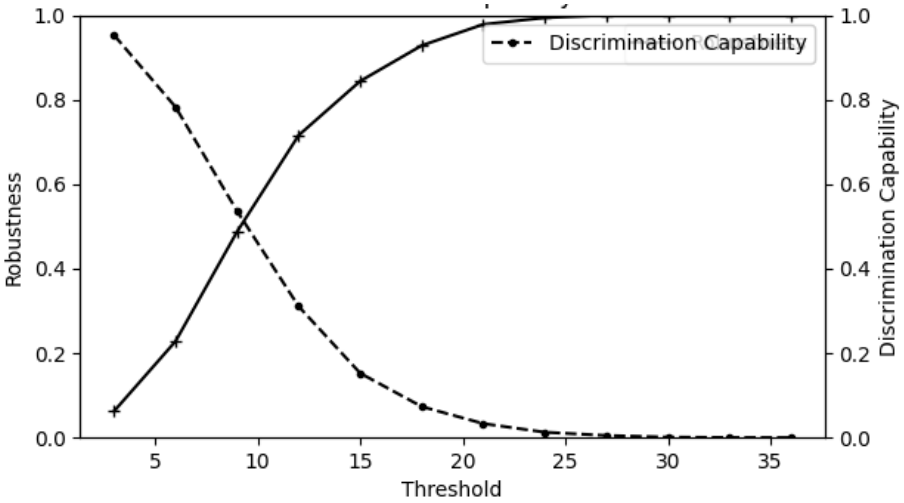

Figure 19. Robustness and Discrimination Capability for DWT-based Hash

The area under the curve shown in Figure 20 is 0.453 per unit square.

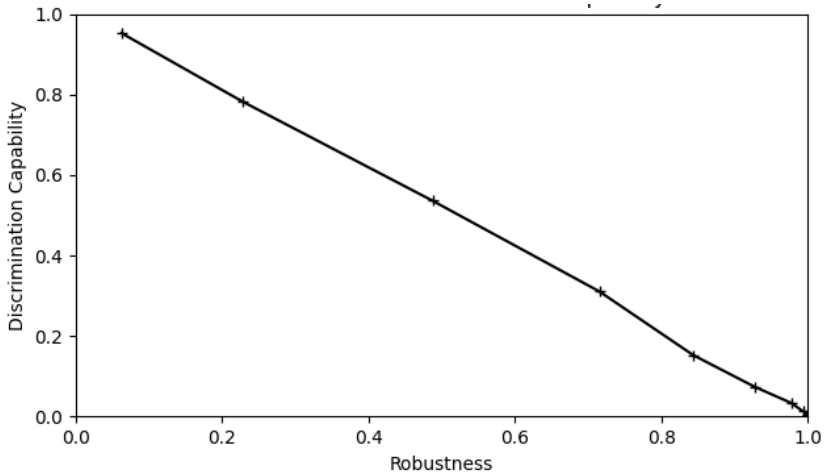

Figure 20. Interaction between Robustness and Discrimination Capability for DWT-based Hash.

4.8.2 Orientation Maps Comparison. The performance of each algorithm for minutiae-based comparison with respect to robustness and discrimination is visualized as follows.

4.8.2.1 Average Hash Results. Figure 21 depicts the robustness and discriminations scores for Average Hash. From the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold value is between 6-9(7.78) and we can consider the nearest value 9 from the threshold series.

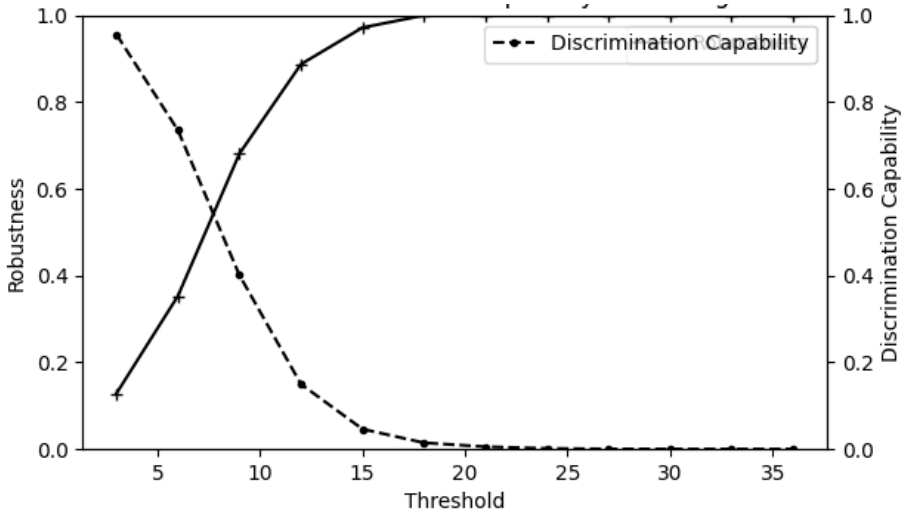

Figure 21. Robustness and Discrimination Capability for Average Hash.

The area under the curve shown in Figure 22 is 0.442 per unit square.

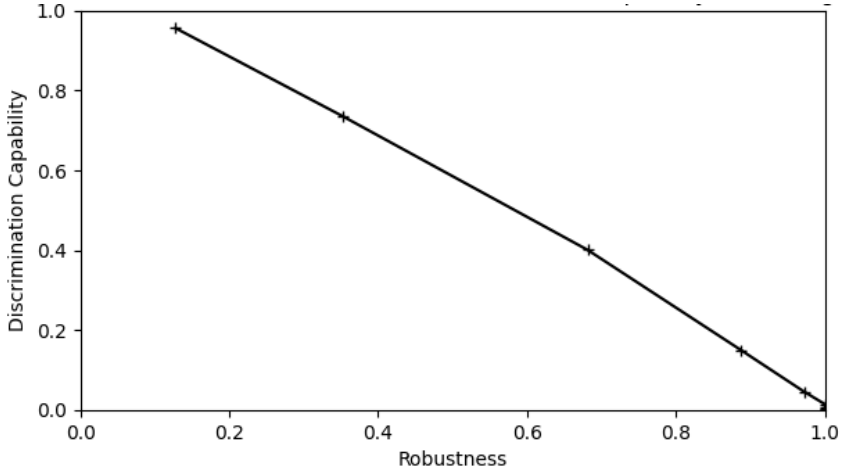

Figure 22. Interaction between Robustness and Discrimination Capability for Average Hash.

4.8.2.2 Difference Hash Results. Figure 23 depicts the robustness and discriminations scores for Difference Hash, from the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold value is between 6-9(8.72) and we can consider the nearest value 9 from the threshold series.

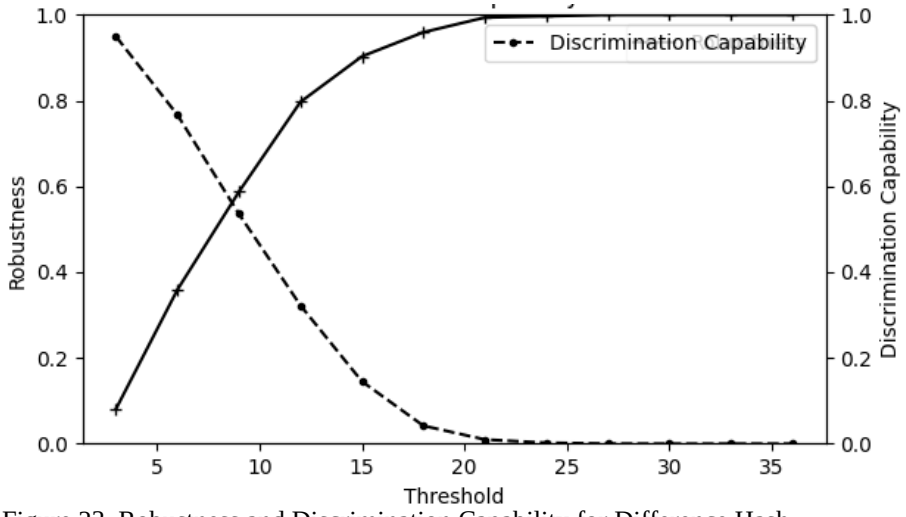

Figure 23. Robustness and Discrimination Capability for Difference Hash.

The area under the curve shown in Figure 24 is 0.539 per unit square.

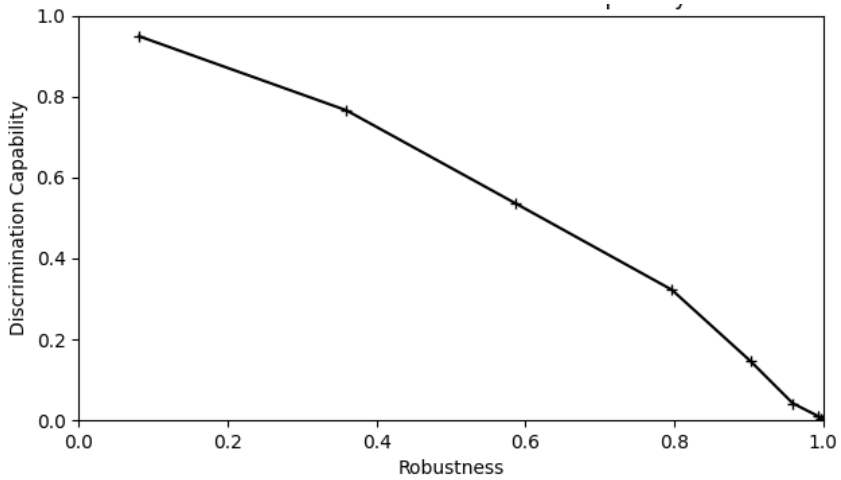

Figure 24. Interaction between Robustness and Discrimination Capability for Difference Hash.

4.8.2.3 DCT-based Hash Results. Figure 25 depicts the robustness and discriminations scores for Average Hash. From the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold value is between 27-30 (28.43) and we can consider the nearest value 27 from the threshold series.

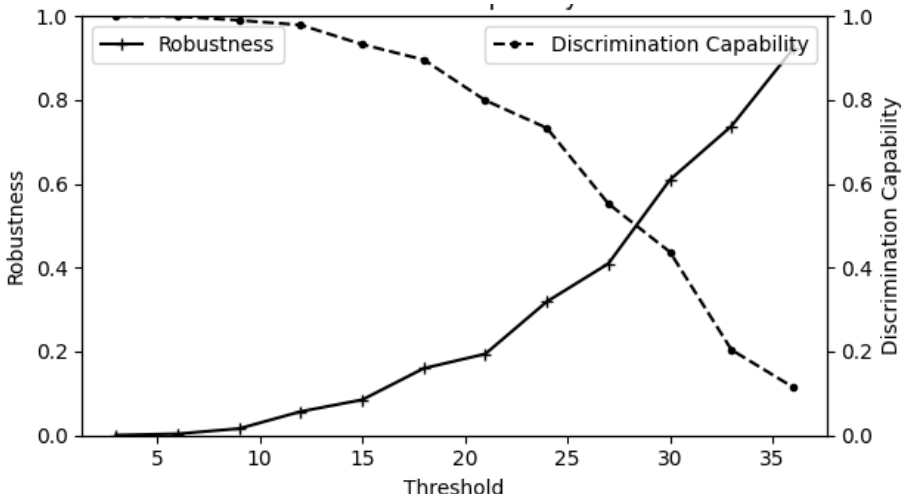

Figure 25. Robustness and Discrimination Capability for DCT-based Hash.

The area under the curve shown in Figure 26 is 0.533 per unit square.

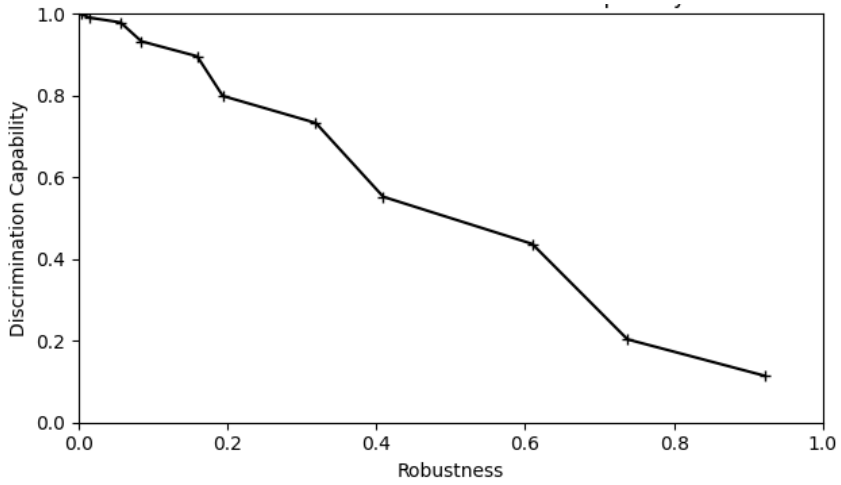

Figure 26. Interaction between Robustness and Discrimination Capability for DCT-based Hash.

4.8.2.4 DWT-based Hash Results. Figure 27 depicts the robustness and discriminations scores for Average Hash. From the figures below it can be seen the robustness of the image's increases, the algorithm loses its capability to discriminate between two similar images. One point of interest in this diagram is where the two trends meet, for which the threshold value is between 21-24(22.91) and we can consider the nearest value 24 from the threshold series.

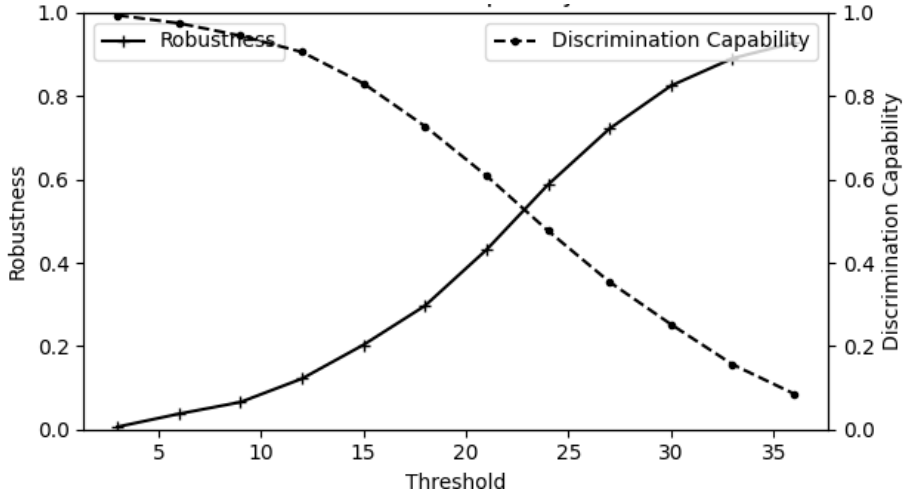

Figure 27. Robustness and Discrimination Capability for DWT-based Hash.

The area under the curve shown in Figure 27 is 0.532 per unit square.

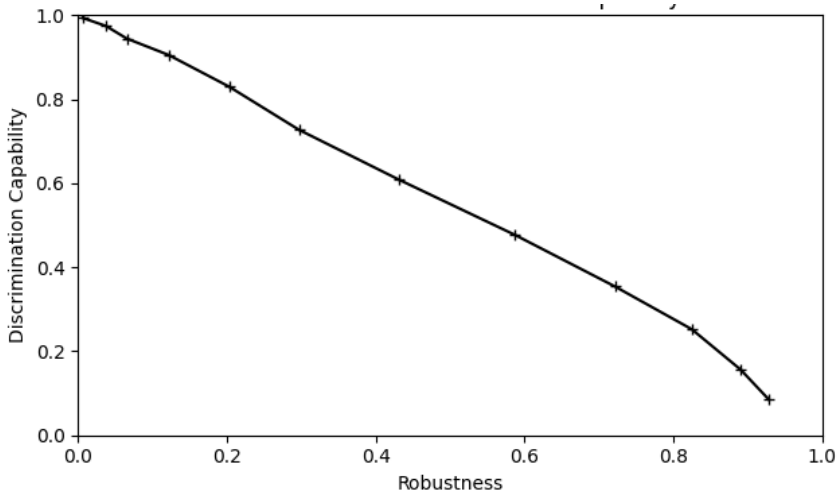

Figure 28. Interaction between Robustness and Discrimination Capability for DWT-based Hash.

4.9 Analysis of Results

The main goal of perceptual hash algorithms is to create a hash that captures the distinctive visual qualities and features of a fingerprint image in a way that is resilient to common image distortions. For an algorithm to be 100 percent robust and discriminative, the points on the graph shown below will be at the right most corner of the graph, the algorithms we considered gave the following results as shown in figure 29 and 30.

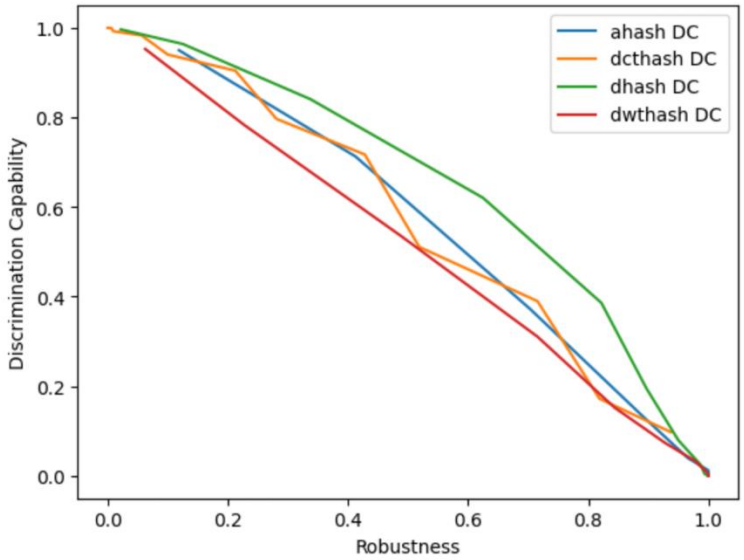

Figure 29. Interaction between Robustness and Discrimination Capability for Minutiae Feature-based Comparison

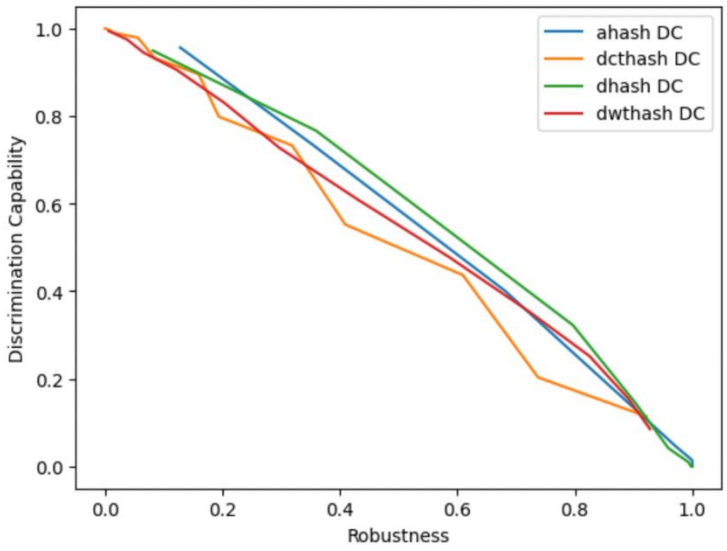

Figure 30. Interaction between Robustness and Discrimination Capability for Orientation Map-based comparison

According to our result, the Average Hash has the highest probability of similar images to total images in Minutiae and orientation type of comparison. But it lacks the ability to differentiate different images, and when we consider the resizing of the image in its preprocessing stage, 8X8 would not have most of the key features of fingerprints. The area under the curve got by plotting robustness against discrimination capability showed least performance with Average Hash.

With Difference Hash also, even though the probabilities of similar and dissimilar images to total images are considerable, with resizing of the image in its preprocessing stage to 9X8, it loses most of the vital information just like Average Hash. When robustness was plotted against discrimination capability, the area under the curve was better than all other hashes.

When DCT-hash under minutiae-based comparison is considered assuming the resized 32X32 image will have the most required information compared to others, and the probabilities of both similar and dissimilar images to the total number of images has

given a fair value. When robustness and discrimination is plotted against each other, the area under the curve got with DCT based hash in both minutiae and orientation map extraction are considerable compared to others.

DWT-hash lacks the ability to find perceptually similar images in both minutiae and orientation maps comparison but showed a good result in finding dissimilar images with orientation maps. When robustness and discrimination is considered and plotted, it gave varying results with minutiae and orientation map extraction.

Based on the robustness and discrimination capability scores of each algorithm evaluated in this project, DCT-based hash showed better results for robustness and Difference hash showed better results for Discrimination Capability.

CHAPTER 5: CONCLUSIONS AND FUTURE WORK

5.1 Conclusion

In this work, we studied, created a pipeline to compare the hash algorithms and evaluated the performance of four image perceptual hash algorithms when considering biometric fingerprints. The algorithms considered in this work include Average Hash, Difference Hash, DCT-based Hash, and DWT-based Hash. Two kinds of image preprocessing were done before considering the fingerprints as inputs to the algorithms, which include Minutiae feature extraction and orientation maps of fingerprints extraction. We focused on evaluating based on two major properties of perceptual Hash: Robustness and Discrimination Capability. Even though Average Hash and Difference Hash gave promising results, they are not suitable for biometric fingerprints. DWT-based Hash failed to recognize similar images even with the input of higher resolution images compared to Average Hash and Difference Hash, and DCT-hash, when considering its minutiae features gave acceptable results. But, for biometric fingerprints, considering the importance of their usage, the algorithms we have considered have made less impact and this opens the door to research on evaluating other more robust hash algorithms with the pipeline created by us.

5.2 Future work

1) In this work, we initiate a pipeline to apply a set of fingerprint datasets on a few available PH algorithms. The scope of the work can be extended with a more rigorous fingerprint dataset, and different PH algorithms. The outcome can be deployed to evaluate, compare, and produce the most robust PH algorithm.
2) Kirchhoff's principle states that keeping every aspect of the cryptographic system public except its key is important. The main concern about these publicly

available algorithms used in the project is that even though they are known publicly, in real life, all the computation will be done on the server side of the algorithm, and whatever that has been done in this project is kind of a symmetric key encryption where the same key will be used by both the parties, which will again have privacy concerns as the users will have to trust the system with their sensitive data. This privacy issue can be resolved by generating hash at the user side as shown in figure 31 below.

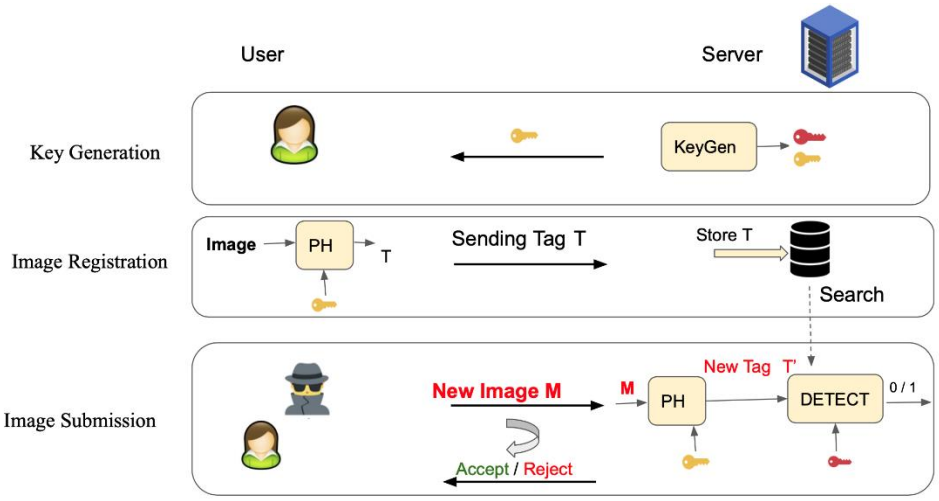

Figure 31. Publicly Evaluatable Perceptual Hash.

3) Applying a publicly evaluable version of PH mentioned in the set of biometric fingerprints is another important and interesting step of this project. This can be achieved theoretically by fully homomorphic encryption (FHE). Intuitively, FHE allows computations on encrypted data which makes delegating computation without compromising the user's privacy possible. However, FHE is a powerful cryptographic primitive that is not efficient and fast. Hence, in (Gennaro) authors proposed an applied protocol that applies Additively Homomorphic encryption as the cryptographic primitives on top of the keyed PH algorithm. Additively

homomorphic encryption such as Pillier encryption allows performing computation by adding two ciphertexts together that produces the same result as encrypting the sum of the two plaintexts.

4) Exploration of fingerprint biometrics with the mentioned publicly evalutable PH can be another avenue of research.

REFERENCES

content-blockchain.org. n.d. <https://content-blockchain.org/research/testing-different-image-hash-functions/>.

Deshmukh, Utkarsh. *github.com*. n.d. <https://github.com/Utkarsh-Deshmukh/Fingerprint-Enhancement-Python>.

—. *github.com*. n.d. <https://github.com/Utkarsh-Deshmukh/Fingerprint-Feature-Extraction>.

en.wikipedia.org. n.d. <https://en.wikipedia.org/wiki/Perceptual_hashing>.

Evans, V. Monga and B. L. "Perceptual Image Hashing Via Feature Points: Performance Evaluation and Tradeoffs." *IEEE Transactions on Image Processing*. IEEE, 2006. 3452-3465.

Farid, Hany. "An Overview of Perceptual Hashing." *tsjournal* 1.1 (2021).

Gennaro, R., Hadaller, D., Jafarikhah, T., Liu, Z., Skeith, W.E., Timashova, A. "Publicly Evaluatable Perceptual Hashing." *Springer, Cham*. 2020.

hackerfactor.com. n.d. <http://www.hackerfactor.com/blog/index.php?/archives/432-Looks-Like-It.html>.

hackerfactor.com. n.d. <http://www.hackerfactor.com/blog/index.php?/archives/529-Kind-of-Like-That.html>.

Hamadouche, Maamar & Zebbiche, K. & Guerroumi, Mohamed & Hanane, Tebbi & ZAFOUNE, Youcef. "A comparative study of perceptual hashing algorithms: Application on fingerprint images." 2021.

Jafarikhah, Tahereh. *Efficient Protocols for Multi-Party Computation"* . 2021. <https://academicworks.cuny.edu/gc_etds/4381>.

Kumar, G., S. Tulyakov and V. Govindaraju. "Combination of Symmetric Hash Functions for Secure Fingerprint Matching." *International Conference on Pattern Recognition*. Istanbul, 2010. 890-893.

"ofcom." Nov 2022. <https://www.ofcom.org.uk/__data/assets/pdf_file/0036/247977/Perceptual-hashing-technology.pdf>.

O'Malley, Katie. .*independent.co.uk*. 27 February 2019. <https://www.independent.co.uk/tech/momo-challenge-youtube-fortnite-peppa-pig-video-parents-a8799776.html>.

Priyanka Samanta, Shweta Jain. " Analysis of Perceptual Hashing Algorithms in Image Manipulation Detection." *Procedia Computer Science*. 2021.

pypi.org. n.d. <https://pypi.org/project/ImageHash/>.

Ramos, Alcides. *apiumhub.com*. 07 July 2022. <https://apiumhub.com/tech-blog-barcelona/introduction-perceptual-hashes-measuring-similarity/>.

Rani, R. Muthu and C. "Perceptual hashing for efficient fingerprint based identification." *4th International Conference on Advanced Computing and Communication Systems (ICACCS)*. Coimbatore, 2017 .

Toropchin, N. I. Korsunov and D. A. "Recognition method of near-duplicate images based on the perceptual hash and image key points using." *IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*. Warsaw, 2015.

Tulyakov, Sergey, Faisal Farooq, Praveer Mansukhani, Venu Govindaraju. " Symmetric hash functions for secure fingerprint biometric systems." *Pattern Recognition Letter* 28.16 (2007): 2427-2436.

Victor, Rayron. n.d. https://github.com/rayronvictor/Fingerprint-Features-Extraction/blob/master/orientation.py.

Xuanbin Si, Jianjiang Feng, Jie Zhou, Yuxuan Luo. "Detection and rectification distorted fingerprints." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3 (2015): 555-568.

APPENDIX A

https://github.com/rn1357/PerceptualHashTechniques