SYNTHETIC GENERATION OF FINGERPRINTS

A Thesis Submitted to
the Graduate School of Engineering and Sciences of
İzmir Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in Computer Engineering

by
Emre İRTEM

July 2020
İZMİR

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to *Nesli ERDOĞMUŞ* for her encouragement, guidance, understanding, and patience. Without her supervision, this thesis work would not be realized.

I would like to thank my wife, *Pelin İRTEM* whose love and endless support are always with me.

Finally, I would like to express my gratitude to my mother *Sezer İRTEM*, my father, *Ali İRTEM*, and my brothers, *Görkem İRTEM & Tolga İRTEM* for their unconditional support.

This thesis is partially supported by The Scientific and Technical Research Council of Turkey (TUBITAK) under the grant 217E092.

ABSTRACT

SYNTHETIC GENERATION OF FINGERPRINTS

Fingerprints are unique to each person and they have been widely used and accepted for identification purposes by the society. Fingerprints can be captured by using ink and paper to get a print and then digitizing it or more recently by using specialized sensors. But in both cases, trained specialist supervision is mostly needed. Moreover, since fingerprints are personal information, they are protected by the laws on personal data protection. Therefore, collection/sharing of real fingerprints is difficult and illegal without the consent of their owner. On the otherhand, deep learning systems that are proven to be very successfull in many machine learning task, usually depend on very large training sets to achive high accuracies. In this study, to overcome the data hunger problem for training deep neural networks, synthetic fingerprints are generated by using model-based methods. For this purpose, firstly master fingerprint images are generated and next many impressions are derived from them by applying real-world degradations.

The realism and the usability of synthetic fingerprints are tried and validated using a fingerprint classification system. For which, a deep neural networks are trained with and without the synthetically generated data. As a result of the experiments, it is shown that the generated fingerprint images are realistic enough to positively effect the classification results and that the usage of the synthetically generated fingerprints in training deep systems are promising.

ÖZET

PARMAK İZLERİNİN SENTETİK ÜRETİMİ

Parmak izleri her kişi için benzersizdir ve toplum tarafından kimlik tespiti amacıyla yaygın olarak kullanılmış ve kabul görmüştür. Parmak izleri, kağıt ve mürekkep kullanılarak toplanıp daha sonra digital ortama aktarılabilir ya da daha güncel olarak özel sensörler kullanılarak toplanılabilir. Ancak her iki durumda da özel olarak eğitilmiş uzman denetimine ihtiyaç vardır. Ayrıca parmak izleri kişisel bilgiler olduğu için kişisel verilerin korunmasına ilişkin yasalar ile korunmaktadır. Bu nedenle, kişilerin izni olmadan, gerçek parmak izlerinin toplanması ve paylaşılması zor ve yasadışıdır. Öte yandan, makina öğrenmesi görevlerinde başarılı olduğu kanıtlanan derin öğrenme sistemlerinin yüksek başarımlara ulaşması genellikle çok büyük eğitim kümelerine bağlıdır. Bu çalışmada, derin sistemlerin eğitimindeki veri açlığı sorununun üstesinden gelmek için model tabanlı yöntemler kullanılarak sentetik parmak izleri üretilmiştir. Bu amaçla, ilk olarak ana parmak izi görüntüleri oluşturulmuş ve daha sonra gerçek dünya bozulmaları uygulanarak birçok gösterim elde edilmiştir.

Sentetik parmak izlerinin gerçekçiliği ve kullanılabilirliği, parmak izi sınıflandırma sisteminde denenmiş ve doğrulanmıştır. Bunun için, sentetik olarak üretilen verilerle ve veriler olmadan derin sinir ağları eğitilmiştir. Yapılan deneyler sonucunda, üretilen parmak izi görüntülerinin sınıflandırma sonuçlarını olumlu etkileyecek kadar gerçekçi olduğu ve sentetik olarak üretilen parmak izlerinin derin sistemlerin eğitiminde kullanımının umut verici olduğu gösterilmiştir.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

Figure **Page**

Figure **Page**

Figure **Page**

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

Biometric recognition systems help to determine the identity of the people based on their physiological or behavioral characteristics. Growth of the world population and developments in technology created an increasing demand for the use of biometrics. Various biometric systems are widely integrated into many areas of everyday life including airport security, attendance control, access control, law enforcement, and banking. There are many biometrics traits, including but now limited to face, hand geometry, DNA, iris, and fingerprint, which are based on physiological characteristics of a person, and keystroke, signature, mouse movement, and gait, which are based on his/her behavioral characteristics.

The uniqueness and the permanence of fingerprints were discovered many centuries ago. Today, to use them for biometric identification is widely accepted by the society. Although different techniques are used due to the changing necessities and technological improvements, the importance of fingerprints remained the same. The first modern examples of fingerprint recognition systems appeared in the late 16th century [5]. In 1892, Galton has mentioned the permanence and uniqueness of fingerprints, and separability to subgroups (classes) with respect to the ridge flow [6]. In the early 20th century, fingerprint identification was accepted as a standard identification technique. In 1924, FBI established a bureau for fingerprint identification with 810.000 fingerprint cards [4]. Today, Automated Fingerprint Identification System (AFIS) is widely used by the governments and their forensic agencies.

1.1 Motivation

Each day, hundreds of search requests are sent to huge fingerprint databases, and it is not feasible for fingerprint experts to compare all those fingerprint images one by one. Therefore, automation has become a must and for decades, researchers have come up with different algorithms and systems to overcome this task. Previously, they need large-scale datasets to test the proposed algorithms in a realistic setting. However, lately, large-scale

datasets are not only needed for performance evaluation but also for training deep learning systems.

Fingerprints are not very easy to collect since they need special equipment and furthermore, they are protected by privacy protection laws since they contain personal information. Therefore, collection and distribution of real fingerprint datasets is not a simple task. It is both labor-intensive and legally complicated. An alternative solution to overcome the scarcity of data, for training deep neural networks and testing performance of algorithms in realistic settings, is to generate synthetic fingerprints. There are some previous studies on testing fingerprint identification systems with synthetic data, however, to the best of our knowledge, there are no publications in the literature in which synthetically generated fingerprints are used for training deep neural networks. In order to fill this gap, this thesis focuses on generation of synthetic fingerprints to be used for training deep learning systems in order to increase their performance.

1.1.1 Contributions

In this study, the fundamental steps for generating synthetic fingerprint are inspired by SFinGe method [7] which is a model-based generation technique. However, some intermediate steps are modified in order to obtain visually more realistic synthetic fingerprints and some intermediate methods are developed from scratch to generate the fingerprints.

The contributions in master fingerprint generation can be listed as the following: (I) A model is proposed for generating arch class orientation image using sine function with different amplitudes. Variation of the orientation image is improved by adding an additive noise term to the model. (II) In ridge generation stage, autoleveling operation is applied to the Gabor response images in each iteration in order to obtain a better ridge structure with less number of iterations. (III) A model is proposed to generate the binary initial image so that it has the same number of white pixels as the number of pixels in a single Gabor kernel.

The contributions in impression generation can be as follows (I) scars are modeled by using deformed ellipses. (II) In order to generate ridge perturbations, a method is proposed that focuses on both generation of ridge discontinuities and generation of prominent/dark regions. (III) background textures similar to the backgrounds in NIST SD4 dataset are generated by using coherent noise images and then adding marks, labels, dots, etc.

Meta-data for the synthetically generated fingerprint images is also crucial. Since the fingerprint generation is model-based, class, orientation, and frequency information of

the fingerprint images are determined before ridge patterns are obtained. Thus, this data is already available for the generated fingerprints. Additionally, in this study, a method is proposed to detect minutiae locations during master fingerprint generation stage using the final Gabor response image. In this way, generated fingerprints can be used to develop and analyze minutiae extraction algorithms as well as orientation extraction, and fingerprint classification.

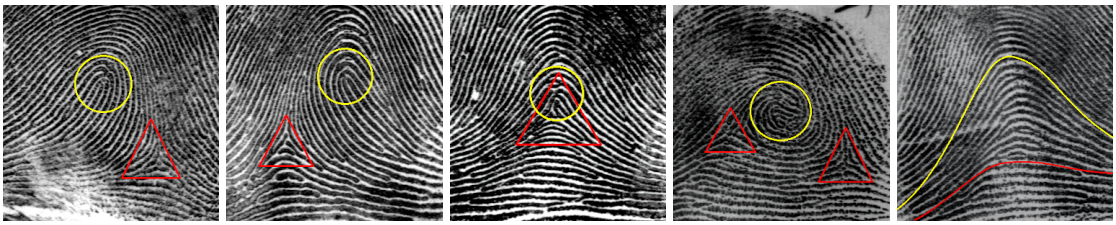
Last but not least, the effectiveness of using synthetic data in training a deep learning system is demonstrated. To evaluate how realistic the generated fingerprints are, having them compared and analyzed by forensic experts would be very informative. Unfortunately, we did not have access to such expertise. But in a sense, this is not the point that this study aims at. We are more interested in validating the fidelity of the generated data by showing its positive impact on accuracy of deep neural networks. Historically, synthetic data usage is common in testing of the developed algorithms. This study shows that synthetic data is able to mimic the real data behavior and come in useful for training deep learning systems, especially when a small amount of real data is available.

1.2 Fingerprint Structure

Fingertip is covered with a crinkled skin tissue which is believed to increase the friction between the fingertip and the surfaces of grabbed objects. Fingerprints are the prints of the ridge pattern created by those crinkles on the finger tip skin. The general assumption is that the fingerprint pattern is created by a random process during embryonic development and it is not entirely determined by genes. This means even identical twins have different fingerprints.

Fingerprints are classified into five main categories according to the pattern in their ridge lines: left loop, right loop, tented arch, whorl, and arch which are shown in Figure 1.1. Fingerprint classes are useful for reducing the search space for fingerprint recognition.

In a fingerprint, the points where the curvature of the ridge lines is high are referred as singular points. They have two types: loop and delta. In the Figure 1.1 singular points are shown for each fingerprint class. Those classes are determined using the relative positioning of these singular points. For instance, left loop class has the loop always located to the left of the delta, and the right loop has the loop always to the right of the delta. The whorl class may have one or two loops between two deltas. Arch class does not have any loop or delta. In addition to pruning the search space, fingerprint matching algorithms usually align the images to the center using core points. Which are

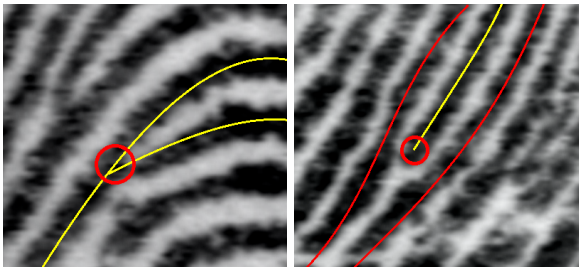


(a) Left Loop (b) Right Loop (c) Tented Arch (d) Whorl (e) Arch

Figure 1.1: Fingerprint classes from NIST SD4 dataset [1]. Loops are shown by yellow circles and deltas are shown by red triangles.

the uppermost loop in the fingerprints. For the arch class core point is selected as where the curvature is highest [8].

Minutiae points are more minuscule details in fingerprints which are defined as the discontinuities on the ridge flow. There two main types of minutiae points: bifurcation and ridge ending, as seen in Figure 1.2. Actually, there are approximately 150 different types of minutiae points but they are actually different combinations of ridge ending and bifurcation [9]. The fingerprint matching algorithms generally use the locations of these two types to find correspondences between two fingerprint images (Figure 1.3).



(a) Bifurcation (b) Ridge Ending

Figure 1.2: Minutiae types from NIST SD4 dataset.

Drilling down into smaller details, ridges are observed to have very small and peculiar structures such as path deviations, ridge shapes, sweat pores, scars, and other permanent variations [10]. Figure 1.4 shows an example of sweat pores appeared as white dots on ridges. These features are highly distinctive however, they are generally visible only in high-quality images.

All those, fingerprint features mentioned above are separated into three different hierarchical levels. Level 1 features (Singular points) are used to determine classes of fingerprints and to align the fingerprint image before matching. Level 2 features (minutiae points) are generally used at the matching step. Level 3 features (sweat pores) provide

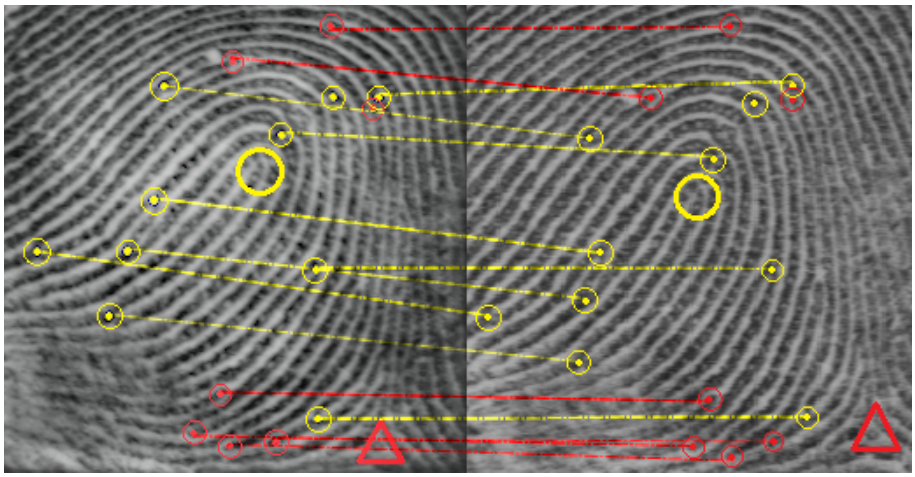


Figure 1.3: An example minutiae correspondence from NIST SD4 dataset.

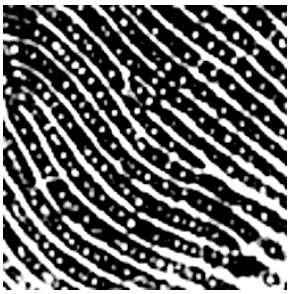


Figure 1.4: An example of sweat pores from FVC2002 DB1 dataset [2].

more accurate matching but require high-quality fingerprint images. This study covers the generation of Level 1 and Level 2 features while generating the synthetic fingerprint images.

1.3 Thesis Outline

This thesis consists of six chapters.

In chapter 2, the literature survey on fingerprint image synthesis is given. The studies are analyzed in two groups: model-based methods and learning-based methods.

In chapter 3, the methodology for generating a master fingerprint image is explained in detail. Orientation image, frequency image, and Ridge pattern generation steps are described.

In chapter 4, our method to derive impressions from the generated master finger-

print is presented. Modeling of the real-world noises that brings visual realism to the fingerprint images is explained in detail.

In chapter 5, the experiments that are conducted to test the realism of synthetically generated fingerprints are described and the results are examined.

In chapter 6, the thesis is concluded and possible directions for future studies are discussed.

CHAPTER 2

RELATED WORK

The approaches to generate synthetic fingerprint images can be mainly categorized into two types: **model-based techniques** and **learning-based techniques**. Each technique has its own advantages and disadvantages. In model-based techniques, too much manual effort is needed. They require to needed to decompose the generation steps into separate tasks and each task brings about a substantial amount of engineering. For instance, in a model-based fingerprint generation pipeline, (I) a fingerprint class is chosen, (II) its respective singular point coordinates and ridge orientations are designated. (III) ridge frequency maps are created and (IV) ridge pattern is generated with the help of filtering, and (V) the impressions of a fingerprint are produced. Additionally, in those techniques, many assumptions are made. For instance, most of them assume independence of ridge orientations and minutiae locations. However, in reality, there should be a dependency between those two. Therefore, model-based techniques generate unrealistic minutiae configurations [11]. Their main advantage is that model-based techniques do not require any training data. On the other hand, Learning-based methods require a large number of training samples to learn fingerprint generation. Moreover, The learning-based techniques result in black-box generators which means that no insight is intrinsically obtained as in model-based approaches and additional effort is needed to obtain the fingerprint meta-data. For instance, it may require to run a minutiae detection algorithm on the generated data for extracting minutiae points. Nevertheless, they can generate more statistically realistic datasets compared to model-based techniques.

2.1 Model-Based methods

In [12] a model-based fingerprint generation technique is proposed which first generates a master fingerprint and then generates many impressions from that. As the first step of master fingerprint generation, global fingerprint shape is created, then ridge orientation map is produced using a zero-pole mathematical model [13] with the loop and delta coordinates. Next, the ridge frequency map is generated based on visual inferences

from several real fingerprints, and the ridge pattern image is obtained through iterative Gabor filtering [14] on a randomly initialized image. As a result of this process, minutiae points appear at random locations in the Gabor response image. After applying a threshold to this image, the binary master fingerprint image is obtained. The next step is generating an impression from the master fingerprint by applying some real-world distortions. The first distortion models wet or dry skin conditions, using morphological operations. The second distortion mimics skin elasticity, employing, Grid-like Non-linear transformations. Finally, ridge discontinuities are modeled by adding many white blobs to the fingerprint image.

In 2004, R. Cappelli [7] introduces the SFinGe approach which can generate more realistic impressions compared to its previous version [12]. SFinGe impression generation includes the following steps: Step 1: Ridge pattern is randomly translated to simulate different placements of a finger in an image. Step 2: Morphological operators are applied to mimic skin wetness or dryness via dilation and erosion, respectively. Step 3: Skin elasticity is modelled by the non-linear distortion model. Step 4: Ridge discontinuities and irregularities are modeled by adding many white blobs with different shapes and sizes. Step 5: Because real fingerprints are not always perfectly centered, rotation and translation operations are applied. Step 6: Using a mathematical model which is based on The Karhunen-Loeve (KL) transform, a realistic background is generated and superimposed with the impression.

In [15] a new coherent noise model (Perlin noise [16]) instead of uniform noise is proposed to be used in the impression generation phase. In this way, better fingerprint impressions are generated with improved variability.

In [11] a method is proposed to generate texture and noise for master fingerprint images obtained using the SFinGe method [7]. First, the ridge features of real fingerprints are modeled using a non-parametric modeling approach. Then obtained features are sampled from that model and master fingerprints and their impressions are obtained. Finally, at the end of the cycle, synthetic fingerprints are validated for realism by comparing their feature densities with the real fingerprints.

In [17] the authors develop a method to overcome the problem of unrealistic minutiae configuration in the generated fingerprints. The method is based on SFinGe [7] and it is only interested in the realism of the generated minutiae configurations. It uses orientation maps that are sampled from real fingerprints. After generating the master fingerprint, realness tests are applied to minutiae templates of the synthetically generated fingerprints. In this way, they iteratively improve the synthetic database. In the end, they obtain a more realistic database in terms of minutiae configurations.

Zhao et al. [18] offers a method that uses statistical models trained on publicly available real-fingerprint datasets, to obtain realistic fingerprint features (such as singular point locations, minutiae points). The method consists of four parts: feature sampling

using a statistical model, generating a master fingerprint, generating many impressions of that master fingerprint, and rendering the fingerprint images. They use AF-FM filters instead of Gabor filters in the ridge generation phase.

2.2 Learning-Based methods

In [19] Bontrager et al. propose a method to generate synthetic fingerprints using deep neural networks to attack the fingerprint matching systems. The method consists of two stages which are training a generator network (a Wasserstein GAN, WGAN) and evolving latent variables of generator network using The Covariance Matrix Adaption Evolutionary Strategy (CMA-ES). This method searches a fingerprint that matches with a large number of other fingerprints, in evolving fashion.

In [20], a method based on Improved-WGAN (I-WGAN) is introduced. The method includes training a Convolutional Autoencoder (CAE) with 512-dimensional latent representation vector and initializing the generator of WGAN with the decoder of the Autoencoder. In this way, fingerprint quality and diversity are improved.

M.Attia et al. [21] presented an approach that generates fingerprints using Variational Autoencoders (VAE). The generated samples have the same distribution with the real fingerprint dataset via the latent vectors. An image x in the space of input images X is mapped to a latent vector z in the latent vector space Z by the encoder. $P_Q(x|z)$ is maximized in the training process, and encoder learns the mapping of latent variable vectors (μ, σ^2). These variables are used to sample the latent vector z. Then, the decoder generates an output image by using z. In the optimization process, Kullback-Leibler divergence between $q_Q(z|x)$, which is modeled by the encoder, and $p_Q(x|z)$, which is modeled by the decoder, and the mean-square error between x, that is the input image, and x', that is the reconstructed image, are used in the objective function. After the training step, randomly generated 32-d vectors are fed into the decoder to generate synthetic fingerprints.

In [22], a method is presented that is capable of generating more realistic fingerprints in terms of minutiae count, minutiae direction, and minutiae spatial distribution. In this method, I-WGAN is initialized the same way as done in [20]. The main contribution is to add an identity loss to the generator's objective during training stage. This forces the generator to generate more unique identities.

2.3 Summary

Learning-based and model-based methods both have some strengths and some weaknesses while generating fingerprints. Model-based methods are able to provide meta-data but they are not as successful in generating realistic fingerprint images. In contrast, learning-based methods generate statistically more realistic fingerprints but they do not provide any extra information about the fingerprint. Additionally, learning-based methods need a huge number of training samples while model-based methods may only need a limited number of real fingerprint images to extract feature statistics such as singular point distribution, minutiae histograms, and ridge orientations, etc. Table 2.1 and 2.2 summarize the details of model-based methods and learning-based methods, respectively.

Table 2.1: Model-based methods.

Method	**Orientation Model**	**Ridge Generation**	**Minutiae Locations**
R. Cappelli et al. [12]	Zero-Pole [13]	Gabor-Filter [14]	Random
SFinGe [7]	Zero-Pole [13]	Gabor-Filter [14]	Random
P.Johnson et al. [11]	Zero-Pole [13]	Gabor-Filter [14]	Statistically Realistic [23]
Q.Zhao et al. [18]	Zero-Pole [13]	AF-FM [24]	Statistically Realistic [23]
C.Imdahl et al. [17]	Zero-Pole [13]	Gabor-Filter [14]	Ellimination by Realness-Test [25]

In this study, a model-based generation technique is adopted due to the following reasons:

- There is a limited number of publicly available fingerprint datasets, so the scarcity of data is a problem for training data-hungry generative deep learning systems.
- It is important to obtain meta-data for synthetically generated fingerprints. The main purpose of this study is to implement a system that generates synthetic data for training deep fingerprint analysis systems developed for different tasks such as minutiae extraction, orientation field estimation, ridge frequency estimation, fingerprint classification, fingerprint centering, and fingerprint matching, etc. Therefore, providing the meta-data is as significant as generating a synthetic fingerprint image

Table 2.2: Learning-based methods.

Method	**Learning Model**
P.Bontrager et al. [19]	Wasserstein GAN (WGAN)
K.Cao and A.K.Jain [20]	IWGAN and Autoencoder
M.Attia et al. [21]	Variational Autoencoder
V.Mistry et al. [22]	IWGAN and Autoencoder with Identity Loss

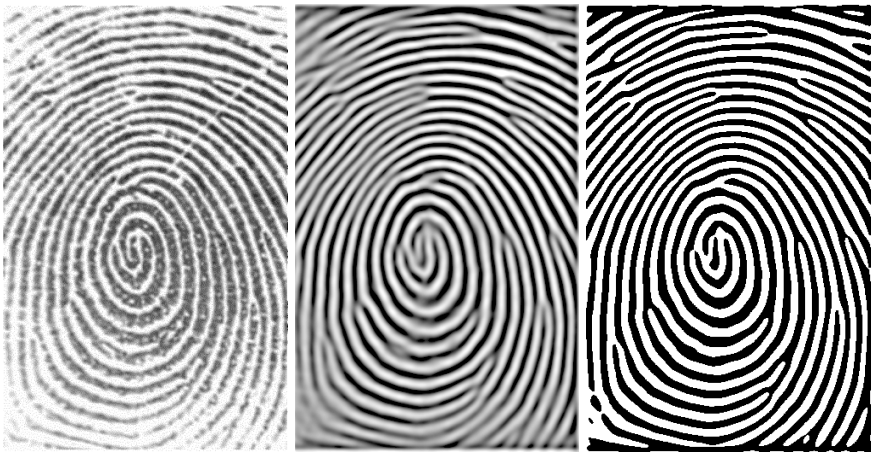
and model-based techniques are capable to do so.

- To train fingerprint matching systems, it is essential to have multiple impressions of the same finger, to teach neural networks the intra-class similarities and interclass-dissimilarities. Model-based approach allow us to generate many impressions with different real-world distortions of the same master fingerprint.

CHAPTER 3

MASTER FINGERPRINT GENERATION

A master fingerprint is a fingerprint that is not affected by any noise or deformations. It can be considered as a perfect fingerprint image from a perfect scanner. In the real world, collecting master fingerprints is impossible, however, it is possible to obtain different realizations of master fingerprints which are impressions. Once an impression is collected, it can be processed to obtain an enhanced image, as an approximation of the master fingerprint. Figure 3.1 shows an example of enhanced image and Figure 3.2 shows an example of master fingerprint.



(a) Noisy image (b) Gabor responses (c) Thresholded image

Figure 3.1: An enhanced image [3] (c) obtained from (a).

In this study, master fingerprint generation method is based-on the SFinGe method [7]. Intermediate steps of the method are inspired by the fingerprint matching process used by the fingerprint experts. This process first segments the fingerprint from the background then estimates the ridge orientations and ridge frequencies. By using the segmented fingerprint region, the estimated ridge frequencies and orientations, the fingerprint image is enhanced and then the minutiae locations are detected. The generation process of master fingerprints is the inverse of this enhancement process. Firstly, the fingerprint area, the orientation image, and the frequency image are generated independently. Then, the

Figure 3.2: A master fingerprint which is generated synthetically.

ridge pattern is obtained by using spatial-filtering techniques. In this study, fingerprint area generation is moved to the impression generation step because it is considered as a real-world distortion since each impression is a projection of the fingertip on a surface with a different shape. Orientation image is generated by using the zero-pole model and To generate the ridge pattern, Gabor-filtering in spatial domain is used. All intermediate steps are discussed in the following sections. Figure 3.3 shows a summary of the implemented master fingerprint generation steps.

3.1 Orientation Image Generation

For ridges in a fingerprint pattern, each location has a principal direction. Close ridges have a tendency to follow similar orientations except for the ones around the singular points. An Orientation image contains these directions for each pixel or pixel group and represents local ridge orientations. Figure 3.4 shows an example partial print with and its orientation field. In this study, the orientation image is generated using the model proposed by Vizcaya and Gerhardt [26], which is an extension of the Sherlock and Monro's model [13]. Both models are capable of generating orientation fields using singular point locations for left loop, right loop, tented arch and whorl classes. For arch class, a different model is used. Sherlock and Monro's model assumes that the orientation field is completely determined by the singular point positions. The Vizcaya and Gerhardt's model introduces more degrees of freedom to Sherlock and Monro's model by adding a piece-wise linear correction function that locally improves the orientations. In the next sub-sections, both approaches are explained with their consecutive stages.

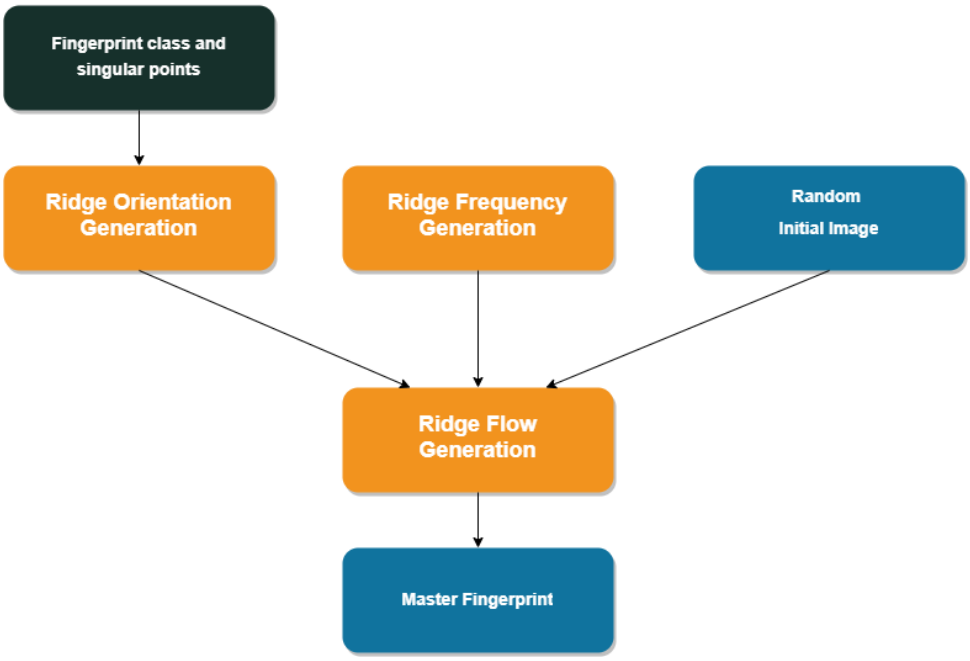


Figure 3.3: Flow chart for the master fingerprint generation steps.

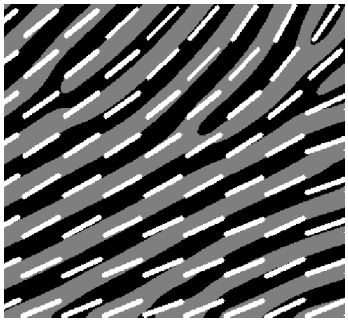
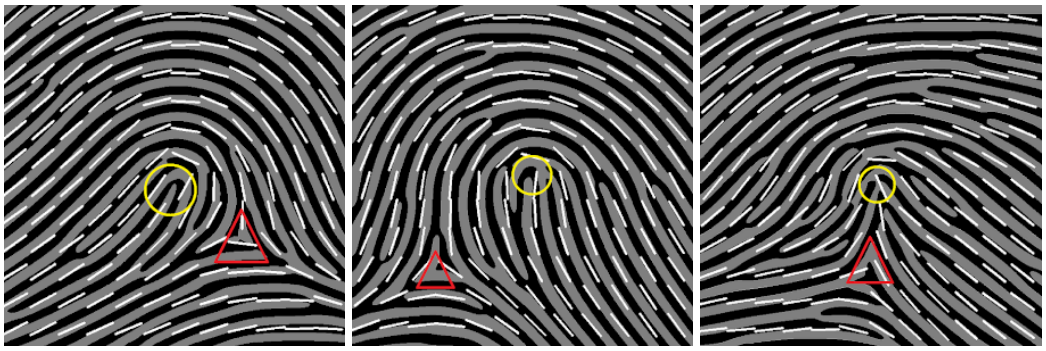


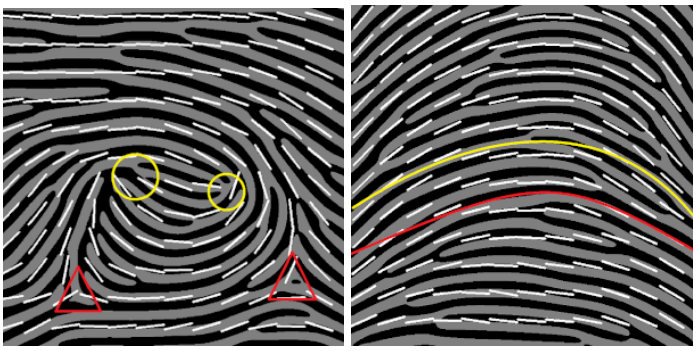
Figure 3.4: The fingerprint detail shows that local ridges follow similar directions.

3.1.1 Sampling Singular Points

Selecting a fingerprint class and sampling singular point locations constitute the first step of generating the orientation image. The fingerprint classes and class-specific singular point constraints are mentioned in Section 1.2. Figure 3.5 demonstrates the relation between orientation field and singular points. Left loop, right loop, and tented arch classes have one loop denoted by S_{loop} and one delta donated by S_{delta}. In these three classes, S_{loop} is considered as the core point of the fingerprint and placed at the center of



(a) Left Loop (b) Right Loop (c) Tented Arch



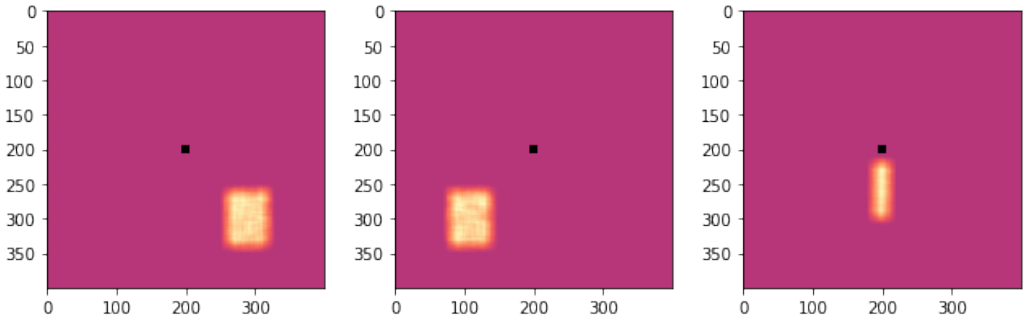
(d) Whorl (e) Arch

Figure 3.5: The white lines represents the orientations of the local ridges.

the master fingerprint image to be generated. S^{c}_{delta}, where c is a fingerprint class (L is left loop, R is right loop, T is tented arch), is uniformly sampled in a rectangular area located with respect to the core point using class-specific constraints, as defined in Equation 3.1. Figure 3.6 shows the heatmap of the $S_{delta_{x,y}}$ for these classes.

$$
\begin{aligned}
S_{loop_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; x = 0.50 \wedge y = 0.50\right\} \\
S^{L}_{delta_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.65 < x < 0.80 \wedge 0.65 < y < 0.85\right\} \\
S^{R}_{delta_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.20 < x < 0.35 \wedge 0.65 < y < 0.85\right\} \\
S^{T}_{delta_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.47 < x < 0.53 \wedge 0.55 < y < 0.75\right\}
\end{aligned}
\tag{3.1}
$$

For the whorl class, the image plane is divided into two regions: left and right. In this way, each region can have one loop and one delta. Singular points in the left and right region are denoted by $S^{W_{left}}_{loop}$, $S^{W_{left}}_{delta}$, $S^{W_{right}}_{loop}$ and $S^{W_{right}}_{delta}$. Each singular point is uniformly



(a) Left Loop (b) Right Loop (c) Tented Arch

Figure 3.6: Heatmap of delta locations for three classes with a $400x400$ image size, calculated using 2000 samples.

sampled by using the Equations 3.2. Figure 3.7 shows the densities of the loops and deltas.

$$
\begin{aligned}
S^{W_{left}}_{loop_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.30 < x < 0.50 \wedge 0.40 < y < 0.60\right\} \\
S^{W_{left}}_{delta_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.15 < x < 0.25 \wedge 0.75 < y < 0.85\right\} \\
S^{W_{right}}_{loop_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.50 < x < 0.70 \wedge 0.40 < y < 0.60\right\} \\
S^{W_{right}}_{delta_{x,y}} &= \left\{(x,y) \in \mathbb{R}^2 \;\middle|\; 0.75 < x < 0.85 \wedge 0.75 < y < 0.85\right\}
\end{aligned}
\tag{3.2}
$$

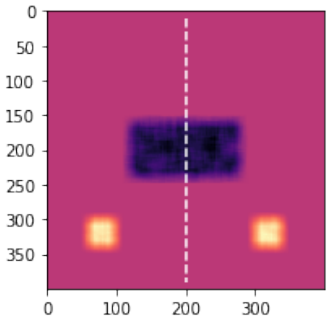


Figure 3.7: The image plane is divided into two regions horizontally. For each region, one loop and one delta is randomly sampled from uniform distribution. Bright pixels denotes the density of the deltas, dark pixels denotes the density of the loops calculated using 2000 samples.

There are no singular points in the arch class, however it has a center point that can be considered as the core point. The center point of arch is denoted by S^{A}_{center}. $S^{A}_{center_{x,y}}$ is sampled from a random uniform distribution, providing the constraint in the Equation

3.3. Figure 3.8 shows the arch center distribution.

$$S^{A}_{center_{x,y}} = \left\{(x, y) \in \mathbb{R}^2 \;\middle|\; 0.40 < x < 0.60 \wedge 0.60 < y < 0.95\right\} \tag{3.3}$$

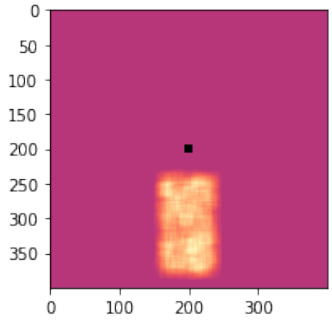


Figure 3.8: The density of the arch center in $400x400$ image size calculated using 2000 samples.

3.1.2 Sherlock and Monro's model

Sherlock and Monro [13] propose a model that is able to compute orientation image using loop and delta locations. The orientation image is computed in a complex plane with a complex rational function (Equation 3.4). Let $S_{loop^i_{x,y}}$, $i = 0..n_{loop}$ and $S_{delta^i_{x,y}}$, $i = 0..n_{delta}$ denote the coordinates of the loops and deltas. The orientation Θ at each point $z = (x, y)$ is calculated as

$$\Theta = \frac{1}{2}\left[\sum_{i}^{n_d} \arg(z - S_{delta^i_{x,y}})) - \sum_{i}^{n_c} \arg(z - S_{loop^i_{x,y}}))\right] \tag{3.4}$$

where $arg(c)$ is the phase angle of the complex number c. Figure 3.9 illustrates the phase angle. The orientation Θ at each point $z = (x, y)$ is the difference between the sum of the phase angles of all deltas and sum of the phase angles of all loops with respect to the z, divided by two.

The outputs produced by this model do not fit very well on real fingerprints as shown in Figure 3.10. The model also suffers from low variability because it assumes that

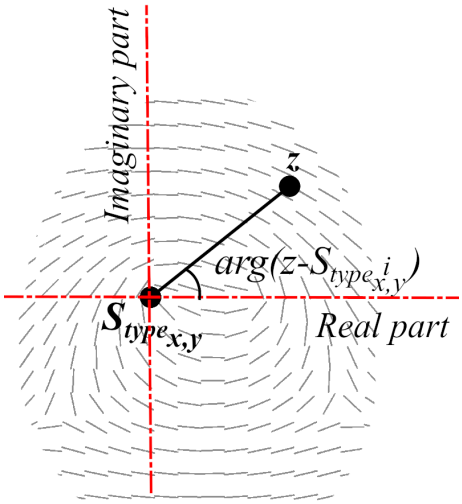
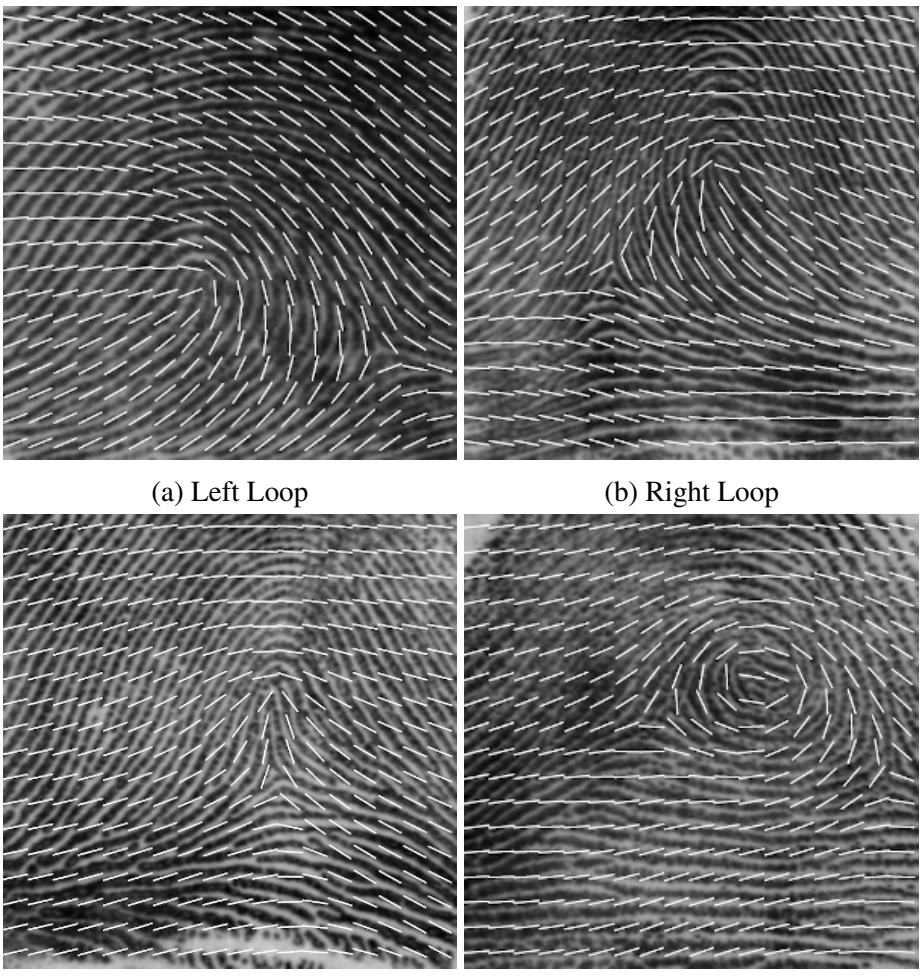


Figure 3.9: The phase angle of $z - S_{delta^i_{x,y}}$ line.



(a) Left Loop (b) Right Loop

(c) Tented Arch (d) Whorl

Figure 3.10: Comparison of orientation image and real image. Orientation images are calculated using the singular points and imposed on real fingerprint images.

the orientations of ridges depend only on the singular point positions. However, there are other factors affecting the ridge orientations.

3.1.3 Vizcaya and Gerhardt's model

Vizcaya and Gerhardt added a piece-wise linear correction function to Sherlock and Monro's model that corrects the phase angle of each singularity. For that purpose, the image plane is divided into L regions (In this study, $L = 8$ is used) at each singular point (Figure 3.11). Those regions are used to compute correction values for phase angles which are linearly interpolated using region boundaries.

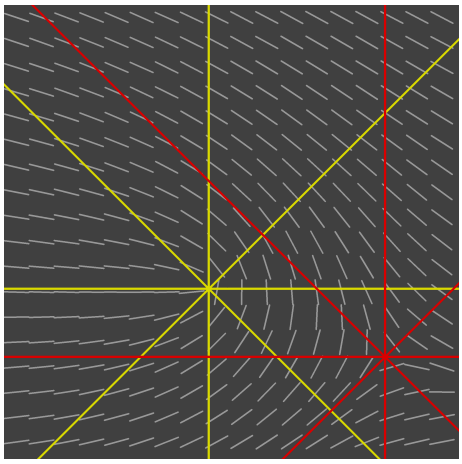


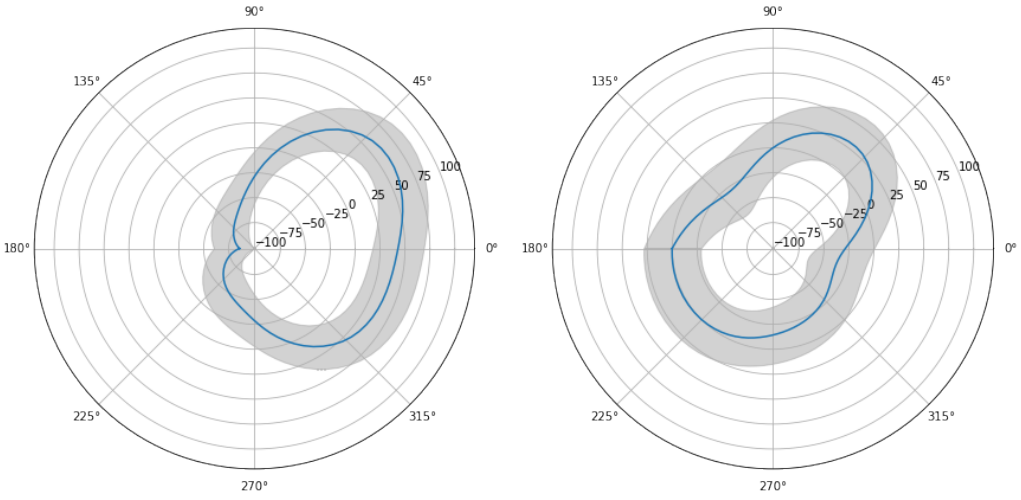
Figure 3.11: An example orientation field of Left loop class and correction axes of loop (yellow) and delta (red). The correction value for the phase angles are linearly interpolated between these lines.

The correction values are stored in a L-dimensional vector for each singular point. In this study, these vectors are uniformly sampled from intervals that are manually inferred by inspecting real fingerprint images. Figures 3.12, 3.13, 3.14, and 3.15 show the visual representations of the vectors for left loop, right loop, tented arch, and whorl, respectively. The angles are shown in degrees (in terms of radian), and the correction vector samples are drawn from gray regions. The values located on perimeters indicate the correction angle with respect to the singular point, and the inner circles represent the correction values. Positive values increase the phase angle whereas negative values decrease it. The sum of the corrected phase angles of loop is subtracted from the sum of corrected phase angles of delta. Half of the difference is the angle for the orientation image for a point $z = (x, y)$. Let

dc_i be a correction vector of delta, lc_i be a correction vector of loop and g be a piece-wise linear correction function. The angle of orientation at each point is calculated using the following equations:

$$\Theta = \frac{1}{2}\left[\sum_{i}^{n_d} g_{dc_i}(\arg(z - S_{delta^i_{x,y}}))) - \sum_{i}^{n_c} g_{lc_i}(\arg(z - S_{loop^i_{x,y}}))\right] \tag{3.5}$$

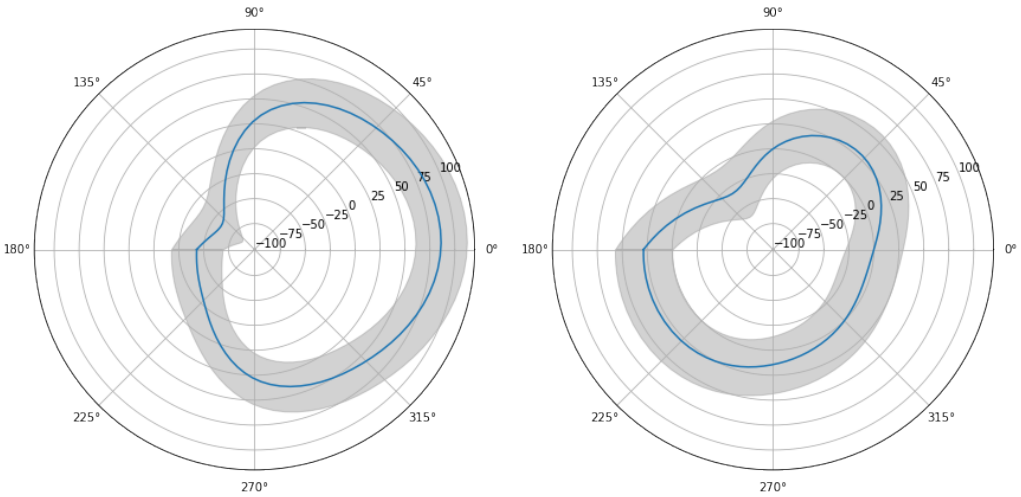
$$\begin{gathered} g_k(a) = k_i(a_i) + \frac{a - a_i}{\frac{2\pi}{L}}(k_i(a_{i+1}) - k_i(a_i)) \\ \text{where } a_i \leq a \leq a_{i+1} \text{ and } a_i = -\pi + \frac{2\pi i}{L} \end{gathered} \tag{3.6}$$



(a) loop (b) delta

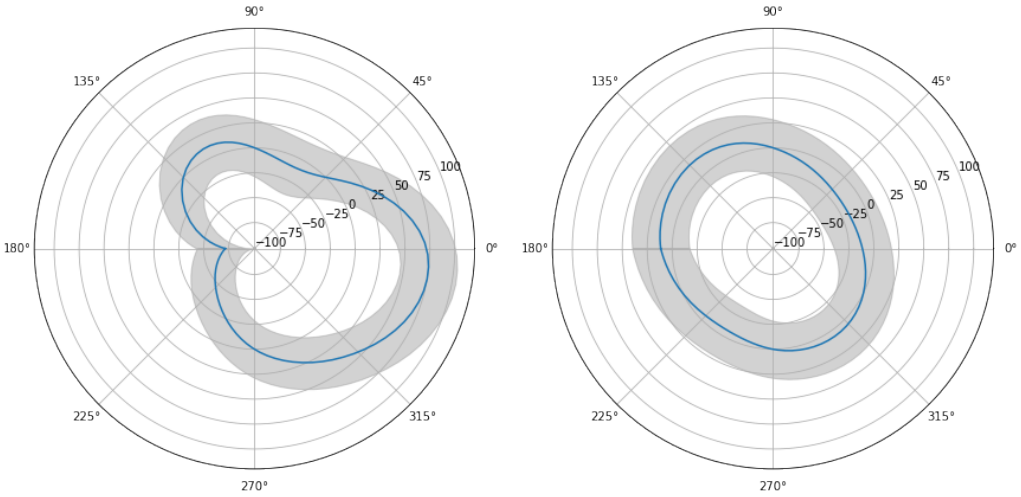
Figure 3.12: A visual representation of left loop correction vectors.

Figure 3.16 shows $k_i(a_i)$ correction value output. The model becomes Sherlock and Monro's model where $k_i(a_i) = a_i$, otherwise it positively or negatively corrects the phase angle. Figure 3.17 shows a sample output generated by the model.



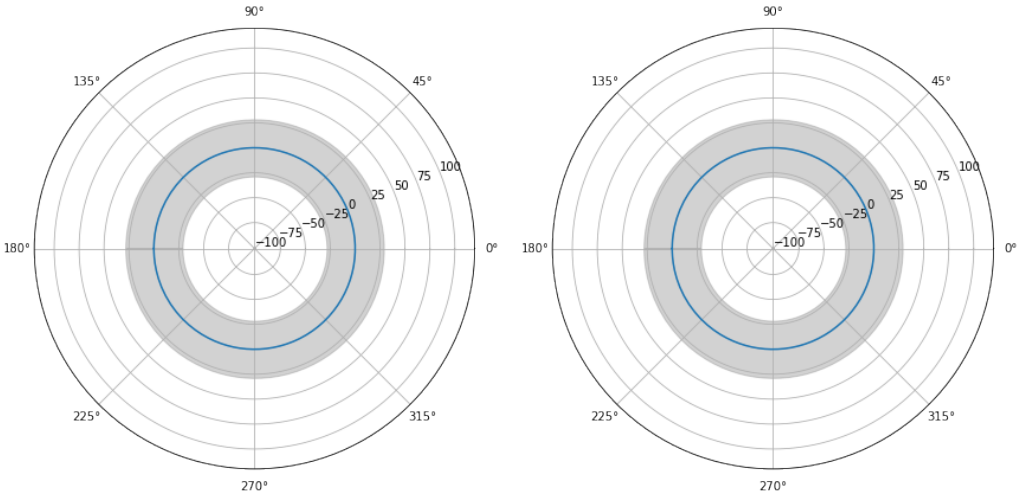
(a) loop (b) delta

Figure 3.13: A visual representation of right loop correction vectors.



(a) loop (b) delta

Figure 3.14: A visual representation of tented arch correction vectors.



(a) loop (b) delta

Figure 3.15: A visual representation of whorl correction vectors.

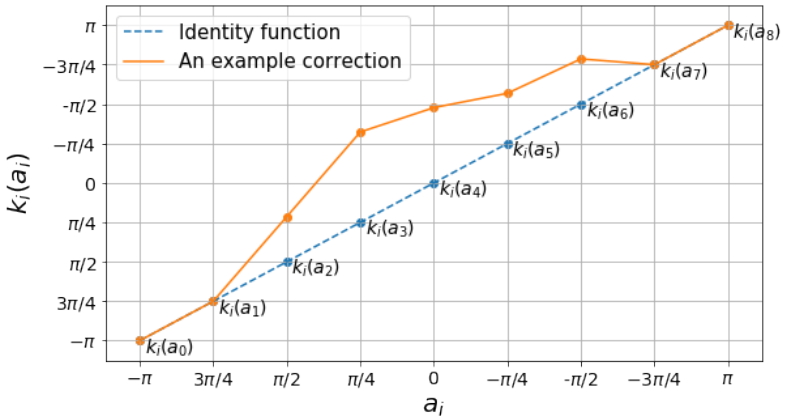


Figure 3.16: $k_i(a_i)$ returns an correction value from a stored array for angle a_i

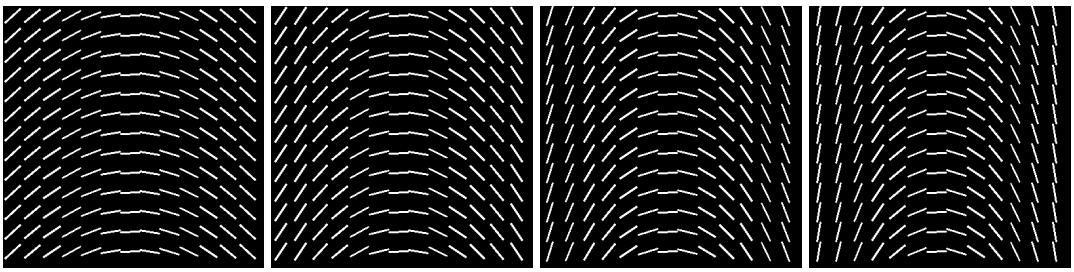
Figure 3.17: An example orientation field calculated using the singular point locations. The ridge orientations look more consistent with the ridges on the real fingerprint.

3.1.4 Generating Arch Class Orientations

The zero-pole model does not generate an orientation image for arch class. In this study, the orientation image of arch class is simulated by sine function. For each pixel, the orientation Θ is calculated as follows:

$$\begin{aligned} \Theta &= \beta \sin(f(x)) \\ \text{where } \frac{\pi}{2} \leq f(x) \leq -\frac{\pi}{2} &\text{ and } 0.75 \leq \beta \leq 1.4 \\ f(x) &= \left[f_{min} + (f_{max} - f_{min}) \frac{(x - x_{min})}{(x_{max} - x_{min})} \right] \end{aligned} \tag{3.7}$$

The function $f(x)$ is the mapping function between x coordinates of the image and the baseline of sine function. The parameter β is used to arrange the amplitude of the sine function. Figure 3.18 shows examples generated using this model with different β values. However, the variance in the generated orientations are observed to be very low. Thus, additive corrections calculated using Vizcaya Gerhardt's method (see Equation 3.6) are generated and added to the orientation image. The additive corrections are . The only difference is that the arch center coordinates are used instead loop and delta.



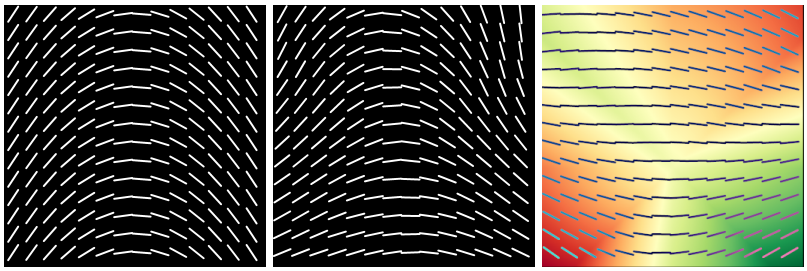
(a) $\beta = 0.75$ (b) $\beta = 1.0$ (c) $\beta = 1.25$ (d) $\beta = 1.40$

Figure 3.18: The effect of different β values.

The corrected orientation angle for arch class is calculated using Equation 3.8. The parameter a is the phase angle of $z = (x, y)$ with respect to the arch center in complex plane and $g_k(a)$ is a linear interpolation function of k that is arch center's correction vector. The γ parameter controls the correction magnitudes, the correction effect is zero at the arch center, and it increases as the point (x, y) goes further away from the arch center.

The correction process is illustrated in Figure 3.19 and Figure 3.20 shows the correction vector of arch center.

$$\Theta_{corrected} = \beta \sin(f(x)) + \gamma g_k(a) \tag{3.8}$$



(a) Θ (b) $\Theta_{corrected}$ (c) $\gamma g_k(a)$

Figure 3.19: The image 3.19a shows orientations generated without correction. The image 3.19b is the corrected version of 3.19a with the additive correction 3.19c. In the Figure 3.19c red, green and yellow regions indicate negative, positive and neutral corrections on the Θ image, respectively.

3.2 Frequency Image Generation

Ridge frequency determines how many ridges appear per unit length along the line that is perpendicular to the ridge lines in a local region. In real fingerprint images, it is usually observed that the upper part of the fingerprints has a lower frequency compared to the rest of the fingerprint and different regions in a fingerprint may have different ridge frequencies. In addition, singular points affect the local frequencies. Thus, it can be concluded that in real fingerprints the ridge frequencies are not random. However, ridge frequencies are randomly generated for all parts of the fingerprint for this study. Generating realistic frequency maps by examining real fingerprint images is left as a future work. Figure 3.21 shows a synthetic sample image for the ridge frequencies. Ridge frequencies have smooth transitions between local neighbor ridges except for areas around the singularities. In other words, close ridges have coherent frequency values. For this reason, coherent noise maps are generated using the bi-cubic interpolation technique. Three uniform noise images are generated randomly at different resolutions ($2x2$, $3x3$, and $5x5$) and then, they are scaled to the same size ($400x400$) by applying the interpolation. The final frequency image is obtained by summing them and applying min-max normalization (In this study, minimum and maximum frequencies are determined as 0.11 and 0.17, empirically.). Figure 3.22 depicts the process.

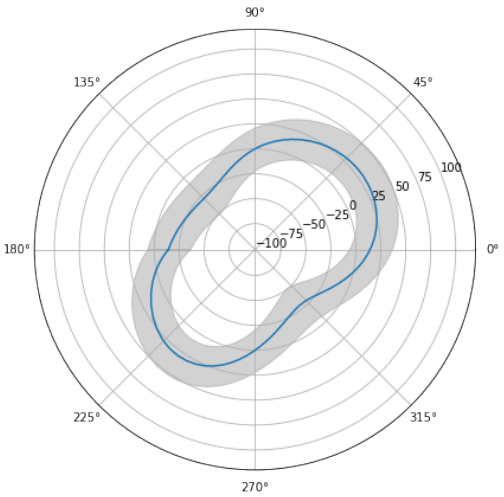


Figure 3.20: The sample region for the correction vector of the arch class.

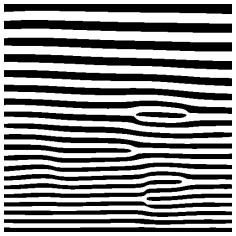


Figure 3.21: An example for ridge frequencies. The upper side of the image has a lower ridge frequency than the lower side.

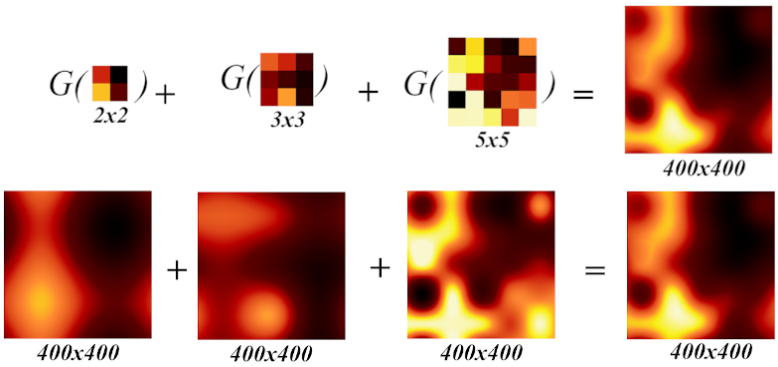


Figure 3.22: An example frequency image generation. Random noise images with resolutions of $2x2$, $3x3$, and $5x5$ are generated, then scaled up to $400x400$ using bi-cubic interpolation and summed up to generate the final frequency image. Bright pixels indicate higher frequencies whereas dark pixels indicate low frequencies.

3.3 Ridge Generation

The orientation and the frequency images contain the ridge directions and ridge densities for each pixel of the fingerprint to be generated. They are widely used in fingerprint enhancement. This enhancement technique can also be used for fingerprint generation. To this end, ridge patterns are generated iteratively by convolving a random initial image with spatial Gabor filters. Finally, the minutiae locations are extracted from the response image. In the following sections, the Gabor filters, ridge generation steps, the impact of the initial image on the resulting fingerprint pattern, and minutiae extraction technique are covered in detail.

3.3.1 Gabor filters

A Gabor filter is a combination of sinusoidal wave and a Gaussian function as shown in Figure 3.23. The Gabor filters are defined as:

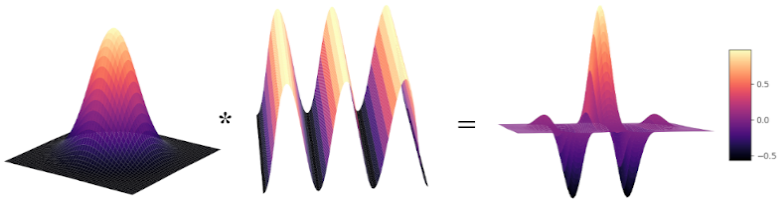


Figure 3.23: A visual description of gabor filters.

$$
\begin{aligned}
g(x, y : \theta, f) &= \exp\left[-\frac{1}{2}\left(\frac{x_\theta^2}{\sigma_x^2} + \frac{y_\theta^2}{\sigma_y^2}\right)\right]\cos(2\pi f x_\theta) \\
\begin{bmatrix} x_\theta \\ y_\theta \end{bmatrix} &= \begin{bmatrix} \sin(\theta) & \cos(\theta) \\ -\cos(\theta) & \sin(\theta) \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix}
\end{aligned}
\tag{3.9}
$$

where θ is the orientation of the filter, (x_θ, y_θ) are the coordinates after rotation applied to coordinates (x, y), f is the frequency of the sinusoidal wave and σ_x, σ_y are standard deviations of the Gaussian function. In this study, σ_x and σ_y are set to be equal.

$\sigma_x = \sigma_y = \sigma$, so the Equation 3.9 is simplified as:

$$g(x, y : \theta, f) = \exp\left[-\frac{x_\theta^2 + y_\theta^2}{2\sigma^2}\right] \cos\left(2\pi f\left(x\sin(\theta) + y\cos(\theta)\right)\right) \tag{3.10}$$

The parameter σ determines the bandwidth of the kernel. In this study, the bandwidth is set to 1, which results in having only three peaks in the Gabor kernel (Figure 3.24, 3.25).

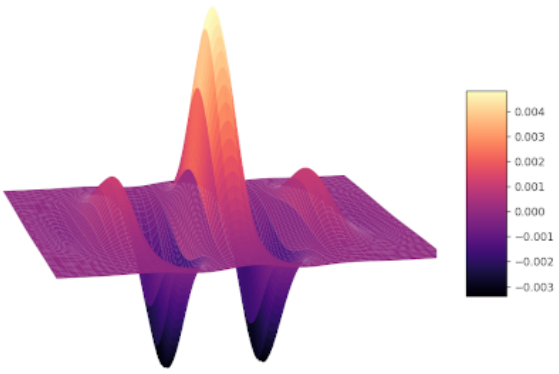


Figure 3.24: The gabor kernel has only 3 peaks with bandwidth set to 1.

3.3.2 Ridge pattern generation

The steps to generate ridge patterns are as the following:

Step 1 An initial image is generated randomly.

Step 2 A convolution operation is applied to the initial image for each pixel using pixel-specific Gabor kernels, for which θ and f values are determined by the previously generated orientation and frequency images. As a result, a response image is obtained.

Step 3 An autoleveling operation, which attenuates high responses and strengthens low responses, is applied to the response image.

Step 4 The response image is assigned as the initial image and the steps are iterated from the Step 2 if $i_{iteration} \leq n_{iteration}$. In this study, the number of iteration ($n_{iteration}$) is set to 20.

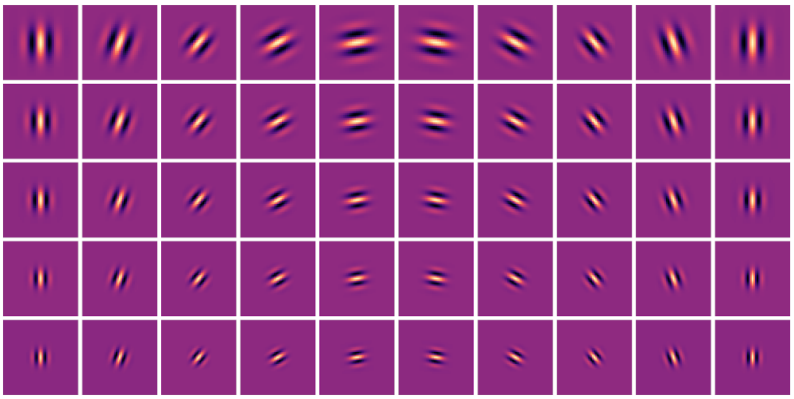


Figure 3.25: Top view of the Gabor kernels for different θ and f values. θ and f increase from left to right and top to down, respectively.

Step 5 After $n_{iteration}$, the ridge patterns are obtained by applying mean thresholding to the final response image.

For efficiency, a discretization operation is applied to the range of the orientation and the frequency images. The orientations are discretized into 20 bins in the interval $0 \leq \theta \leq \pi$ and the frequencies are discretized into 100 bins in the interval $0.11 \leq f \leq 0.17$. The closest bin value is assigned to a pixel for both orientation and the frequency image. In this way, the total number of Gabor kernel size is reduced to $20 * 100 = 2000$.

Figure 3.26 shows the ridge generation process. It can be clearly seen that the ridge patterns get better as the number of iterations increases.

3.3.3 The effect of the initial image on the resulting ridge pattern

The generation process is deterministic i.e. always the same output is obtained with the same orientation, frequency and initial images. However, even with the same orientation and frequency maps different initial images lead to different results, as shown in Figure 3.27.

In addition, the number of the white pixels in the initial image also affects the number of the minutiae points in the generated fingerprint. Figure 3.28 shows the distribution of the minutiae numbers with respect to the number of white pixels in an initial image. In this study, the initial binary image is randomly generated according to the probability of

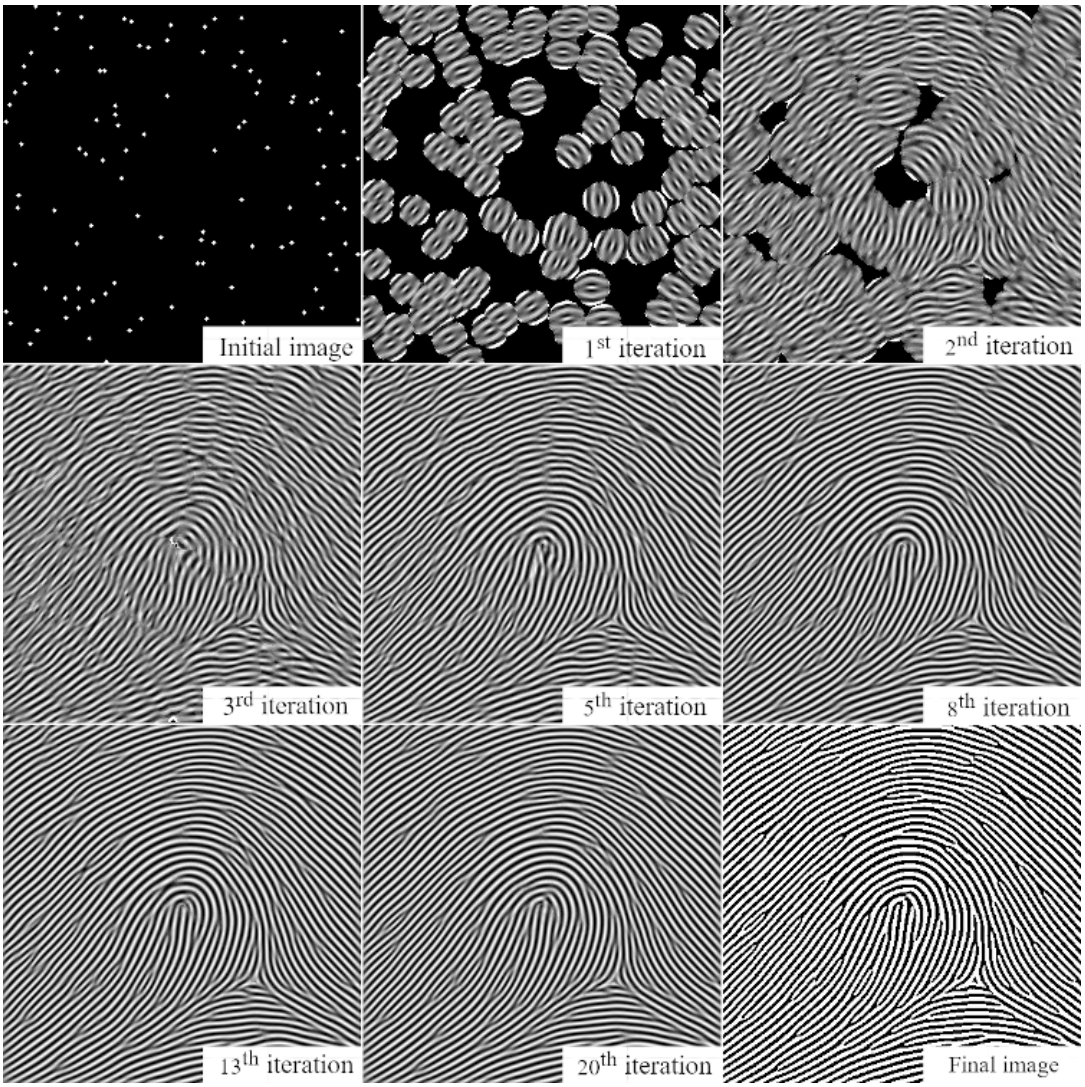


Figure 3.26: An example generation process.

each pixel being 1 or 0 calculated by the following formula:

$$\begin{aligned} P(X=1) &= \frac{K}{I} \\ P(X=0) &= 1 - P(X=1) \end{aligned} \tag{3.11}$$

Here K is the kernel area, I is the image area and X is the pixel to be determined as white or black. The expected total number of the white pixels in an initial image is K.

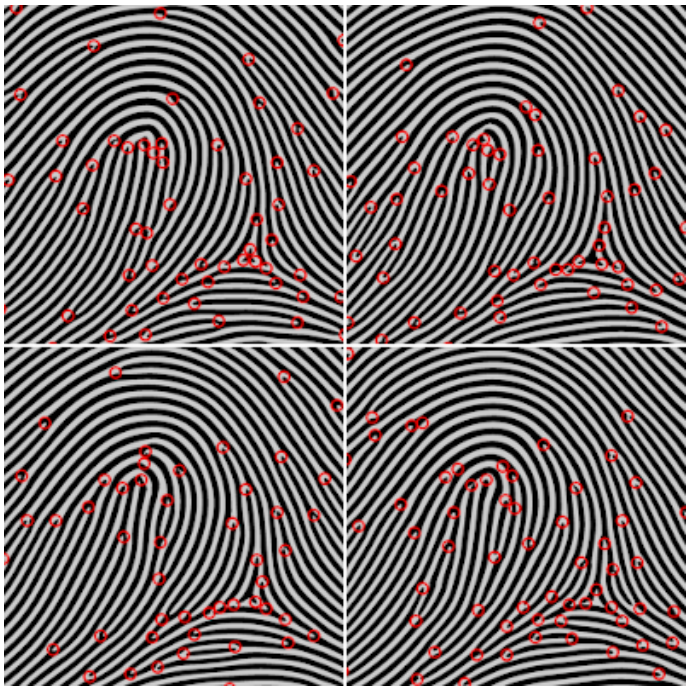


Figure 3.27: All images are generated using the same orientation and frequency images but different initial image. Different initial images lead generation process to result in different fingerprint images.

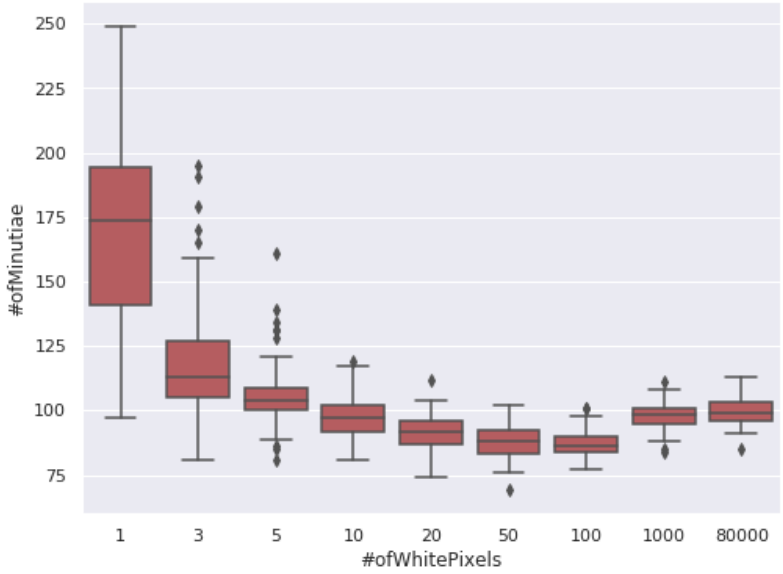


Figure 3.28: The plot shows the minutiae count distribution with respect to the number of white pixels in an initial image for left loop class. For all samples, the same orientation and frequency images are used.

3.3.4 Extraction of minutiae locations

Minutiae locations are extracted from the response image that is obtained after convolution and autoleveling operations. The ridges (black pixels) and the valleys (white pixels) appear at extreme values in this response image and minutiae points appear at the intermediate values. The minutiae type is determined with respected to the threshold to be applied for ridge generation. Figure 3.29 shows local minutiae points in the response image and their thresholded local patch.

Using this insight, the minutiae locations are detected as the following (Figure 3.30):

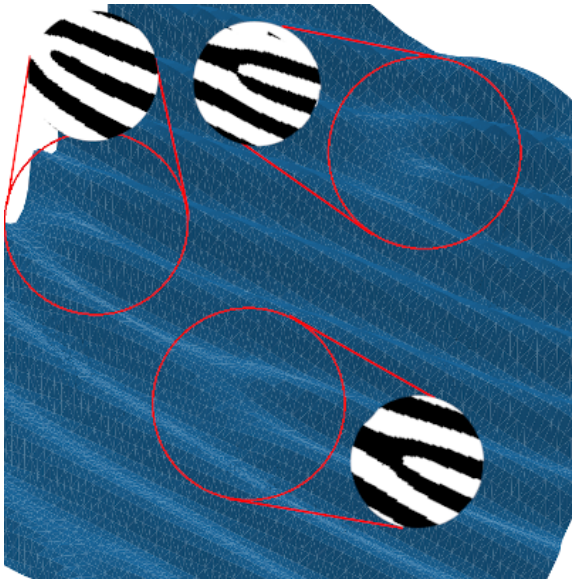


Figure 3.29: A 3D Surface plot for a patch taken from a response image and respective minutiae locations with their thresholded binary images.

Step 1 The range of the response image is scaled into $[-1, 1]$ interval

Step 2 The probability image P that contains probability of being a minutiae for each pixel is calculated using the following formula:

$$P(X) = 1 - \left|\tanh(X)\right| \tag{3.12}$$

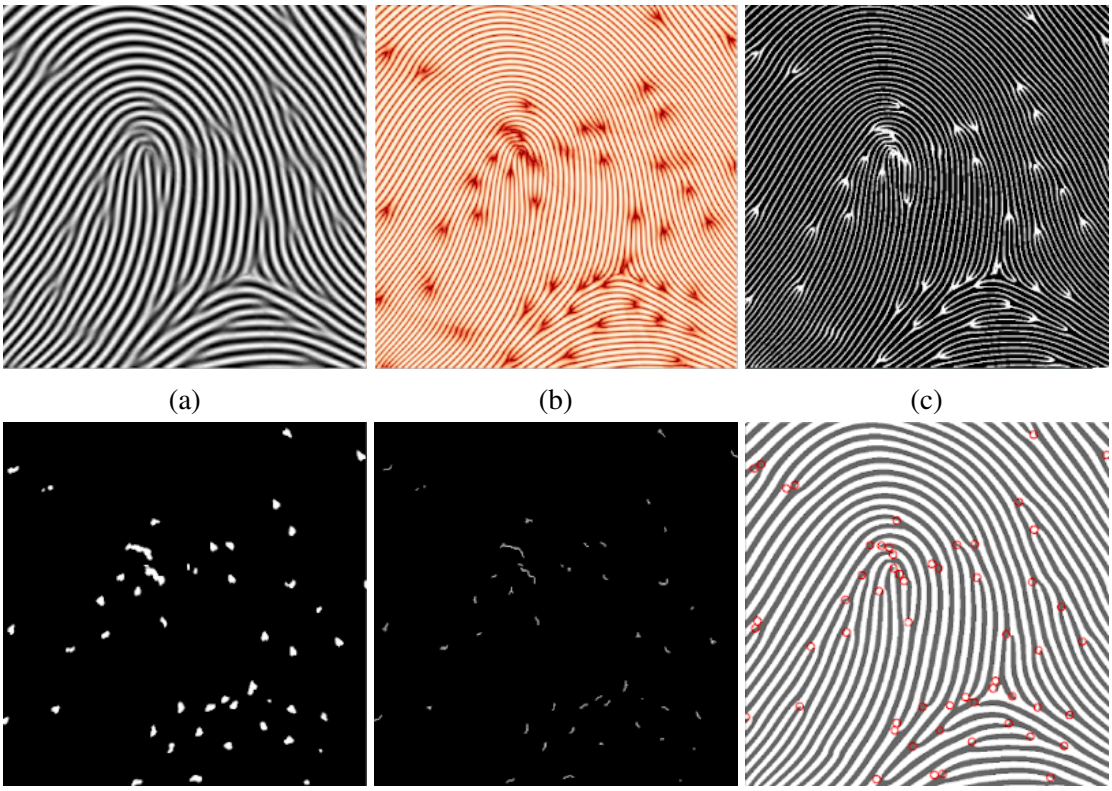


Figure 3.30: (a) response image, (b) minutiae point probability map, (c) thresholded probabilities, (d) denoised minutiae probability map, (e) skeletonized probability map, (f) annotated minutiae locations on fingerprint image

Step 3 A threshold ($P(X) \geq 0.8$) is applied on P.

Step 4 Median filter is applied on the thresholded image for denoising.

Step 5 The resulting image is skeletonized and minutiae locations are detected using maximum suppression algorithm.

3.4 Summary

Most approaches for fingerprint generation start with (I) fingerprint area generation, (II) orientation and (III) frequency image generation. These steps are completed independently from each other. Then, (IV) random initial images are iteratively enhanced by using Gabor filters, with θ and f parameters taken from the orientation and frequency images. After some iterations, Gabor response image is formed and by applying a threshold to it binary ridge patterns are obtained.

In this study, fingerprint area is considered as a real-world distortion and it is moved to the impression generation stage. In order to generate orientation images for fingerprint classes (except for Arch), Vizcaya and Gerhardt's model [26] is used and the correction vectors for the singularities of each class are determined by visually inspecting real fingerprints. For Arch class, the orientation image is generated with the help of the sine function and a correction technique similar to other classes is applied. For frequency image generation, our approach first generates three different random noise images at different resolutions. Then, small noise images are scaled to full resolution by using bi-cubic interpolation and summed up. Finally, min-max scaling is applied to obtain the coherent and pseudo-random frequency images. In the next step, random initial images are convolved iteratively with pixels-specific Gabor filters. And once the Gabor response images are obtained, minutiae locations are extracted. As the final step, master fingerprint images are computed by applying mean thresholding. Figure 3.31 shows example outputs for each fingerprint class.

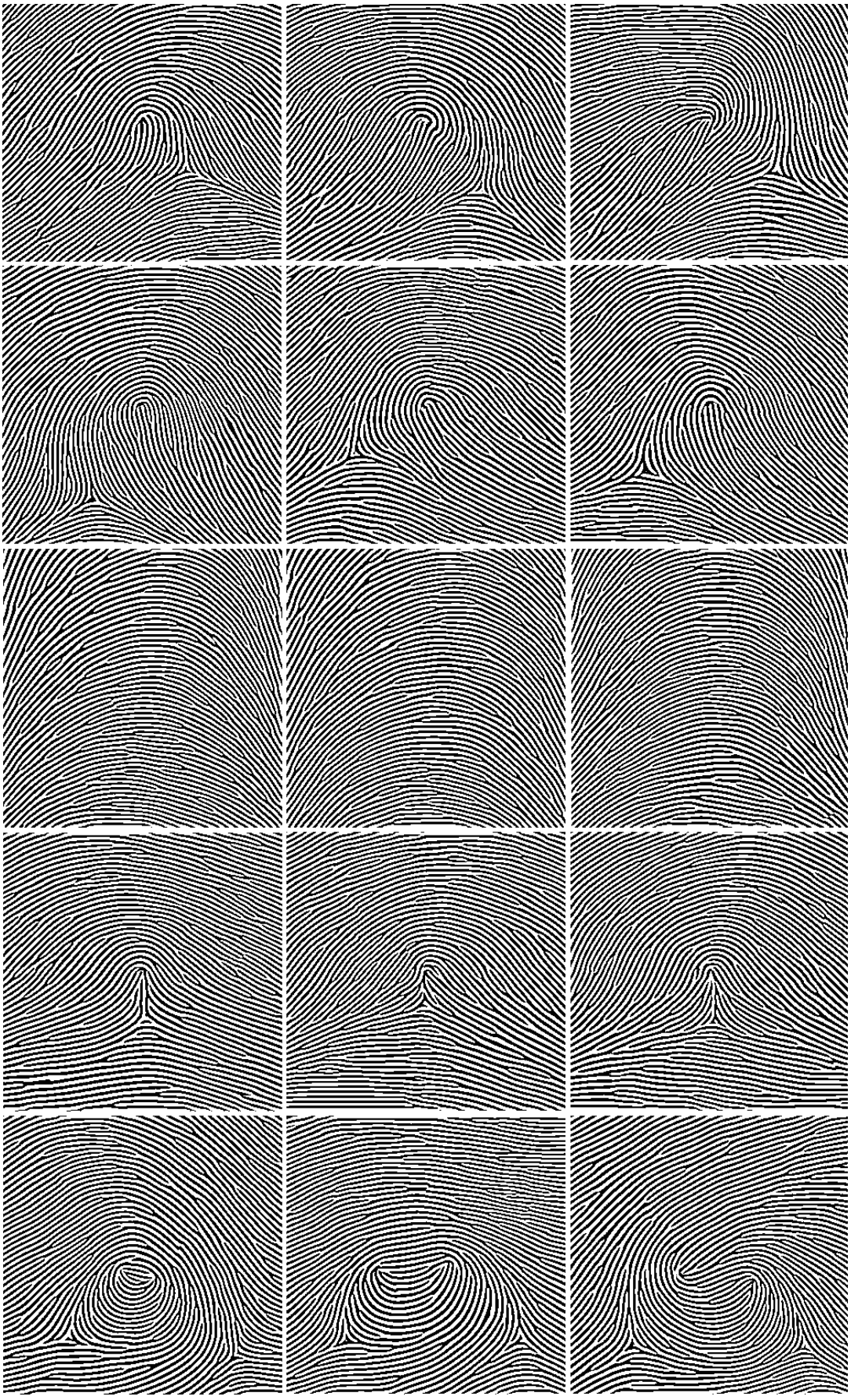


Figure 3.31: Synthetically generated master fingerprints. From top to bottom, sample images from left loop, right loop, arch, tented arch, and whorl.

CHAPTER 4

IMPRESSION GENERATION

The previous chapter explains how to obtain synthetic master fingerprint images.Those images are noise-free and do not look realistic. While capturing real fingerprints, several factors introduce variations that lead to different impressions of the same finger. To obtain more realistic fingerprints, these factors should be modeled and applied to the master fingerprints. In real-world conditions, the fingerprint images have the following variations:

1. Pose variations such as displacement, rotation, and scale.
2. Touch area variations caused by pressing the fingertip on to the capturing medium with different angles and forces
3. Due to the curved shape and elasticity of the finger tips, non-uniform deformations
4. Ridge line variations due to various skin conditions (e.g. wetness) and different pressure intensities
5. Permanent damages on the fingertip such as scars
6. Background and lighting variations

The following sections describe how these factors are modeled to obtain a visually more realistic fingerprints. In this study, the variations are applied to the master fingerprints in the order of (I) scars, (II) ridge thickness variations, (III) fingerprint area, (IV) skin deformations due to elasticity of the skin, (V) pose variations on capturing medium, (VI) ridge perturbations, (VII) background, (VIII) intensity and lightning conditions.

4.1 Scar

In real fingerprint images, it is observed that in some regions, ridge lines are cut by the white traces. This is usually due to permanent damages or skin wrinkle on the finger skin. Scars at the fingertip can be of various size and severity. In this study, only

the simple scars are included. More severe scars created for fingerprint alteration is out of the scope of this thesis. Scar examples of real fingerprint image are shown in Figure 4.1.

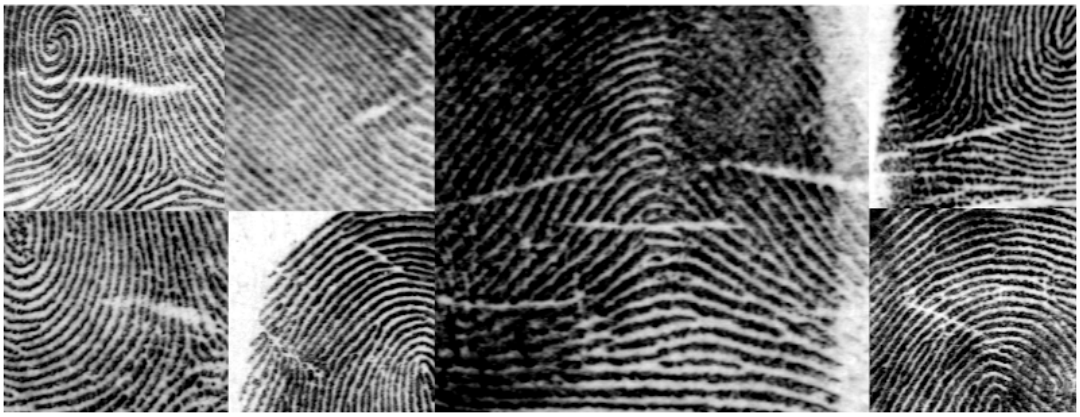


Figure 4.1: Examples of scars from real fingerprints.

A single scar is modeled in the scar's local coordinate space, using three parameters which are:

1. θ : the angle of the scar. Orientations can be horizontal, vertical, or mixed. If the orientation is horizontal or vertical then θ value is uniformly sampled from the interval $[80°, 100°]$ or $[-10°, +10°]$, respectively. If the orientation is mixed, θ is sampled from $[0°, 360°]$.

2. L : the length of the scar which is uniformly sampled from the interval $[40, 200]$.

3. T : the thickness of the scar which is uniformly sampled from the interval $[2, 6]$

For each scar, an ellipse with L as major-axis length and T as its minor-axis length is centered in its object space and rotated θ degrees. Then, the object space is deformed using a piece-wise affine transformation. Example transformations are shown in Figure 4.2. The generated scar is imposed on the master fingerprint image at randomly generated coordinates. In this study, the maximum allowed number of scars is 12 per fingerprint. Figure 4.2 also, shows an example fingerprint with various scars imposed.

4.2 Ridge Thickness

Ridge thickness in the fingerprints get thinner if the finger skin is dry or the applied pressure on the surface is low. and thicker if the finger skin is wet or the applied pressure

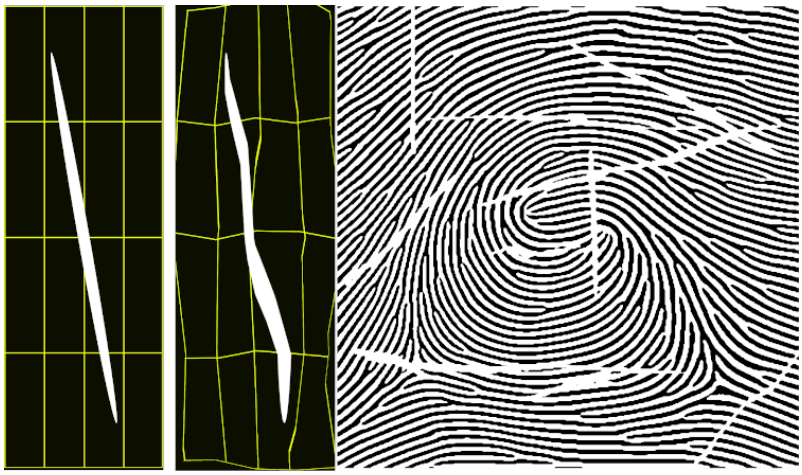


Figure 4.2: The object space of a single scar, on the left and multiple scars added to a master fingerprint, on the right.

is high. To simulate these variations, morphological operations are applied to the master fingerprints which are scaled up by a factor of 4 $(1600x1600)$ in order to achieve higher number of levels in ridge thickness. After the morphological operations, the images are down-scaled to their original size $(400x400)$. The parameter T controls the ridge thickness. Negative T values model the dry finger or low pressure using dilation operation and positive T values model wet finger or high pressure using erosion. Disk element with radius of 1 is used as the structuring element and T value is uniformly sampled from the interval $[-4, +4]$. The magnitude of the T parameter indicates how many times the operation will be applied on the master fingerprint. Figure 4.3 indicates the ridge variations with respect to the different T values.

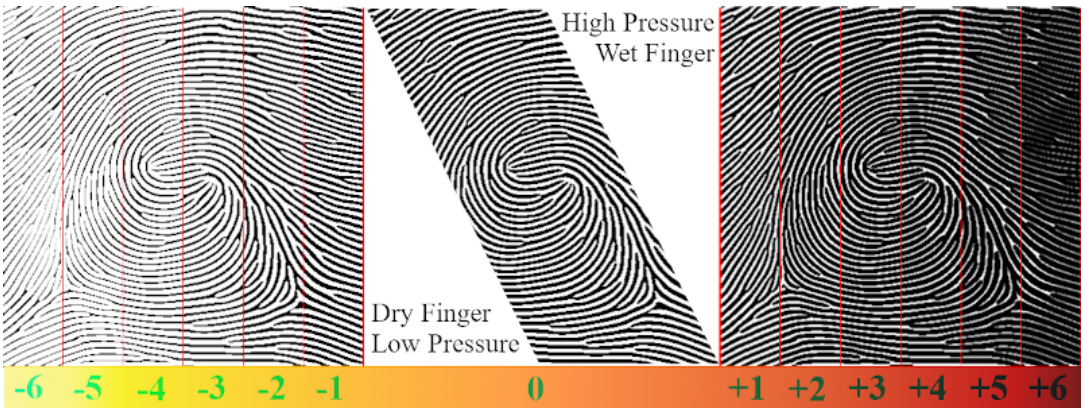


Figure 4.3: Negative T values model dryness or low pressure against the touching surface, positive T values model wetness, or high pressure.

4.3 Fingerprint Area

The fingertip is deformable and has an elliptic shape. Due to the elasticity and the 3D shape of the finger, the imprint of the finger is also ellipse-like. The applied pressure against the surface, the position and the size of the finger are the factors that change the fingerprint shape. In this study, fingerprint shapes are generated using the model proposed in [4] and shown in Figure 4.4.

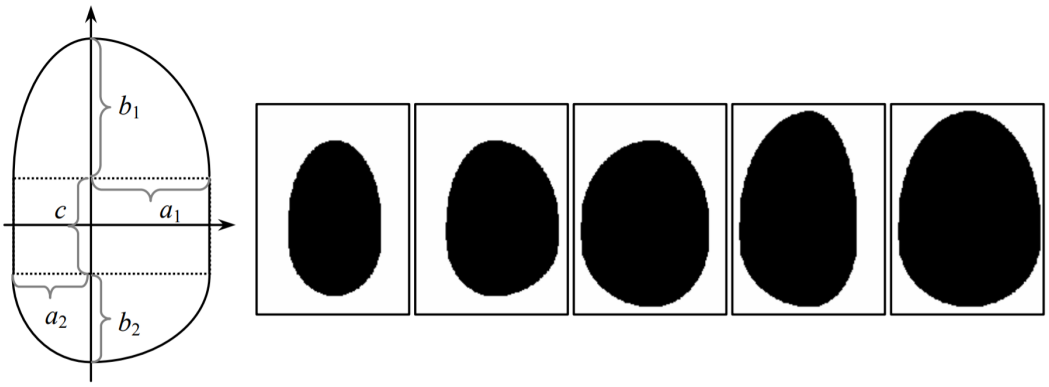


Figure 4.4: The fingerprint shape model [4].

To generate the fingerprint shape, the parameters a_1, a_2, b_1, b_2, and c are used to set the major/minor axes of four ellipses and the width/height of the rectangle in the middle. The Figure 4.5 shows how the fingerprint area is formed using 4 ellipses and a rectangle. After the global shape is obtained, its boundaries are randomly distorted. In this way, sharp edges are eliminated and a smooth transition is provided between the fingerprint and the background. Figure 4.6 shows some examples of generated fingerprint areas.

4.4 Skin Deformation

The elasticity of the fingertips causes non-linear distortions/deformations in fingerprints (Figure 4.7). Two impressions from the same finger may have dissimilar non-linear distortions due to different and pressure placement of the finger against the surface. To model these distortions, a grid of size RxC, where R is the number of rows and C is the

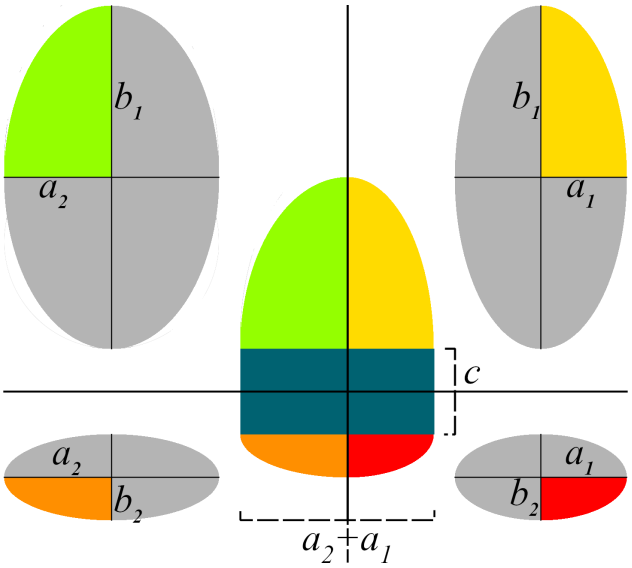


Figure 4.5: Fingerprint area generation.

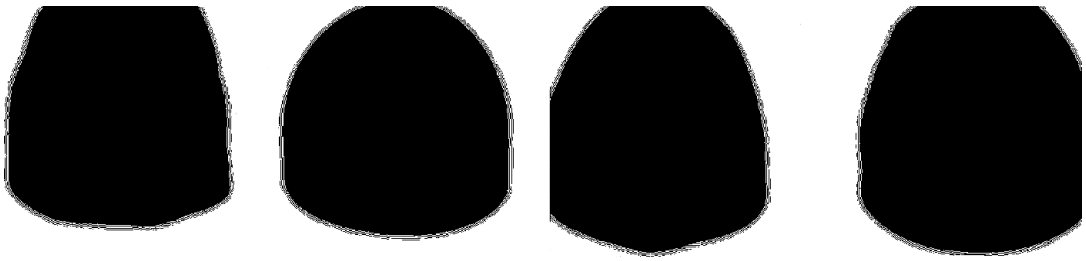


Figure 4.6: Generated fingerprint area examples.

number of columns, is placed on a fingerprint. It is deformed by applying Piece-wise Affine Transformations by M pixel, where M is the movement parameter and randomly sampled for each grid segment from the interval $[-7, +7]$ (Figure 4.8).

4.5 Pose Variation

Several factors such as different rotations/displacements of the fingers to surfaces, scanning quality or camera distance (in offline sensing) may introduce pose variations to the impressions. These variations are modeled by Rotation (R), Translation (T_x, T_y) and Scaling (S) operations. R parameter is randomly sampled from a range of $[-12, 12]$ in degrees, T_x and T_y parameters are randomly sampled from a range of $[-30, +30]$ in pixels

Figure 4.7: Red and green fingerprints are different impressions from the same real finger taken from FVC2002 DB-1A [2] and colorized. They are centered according to their singularities and superimposed (image in the middle). Due to the elasticity of the skin and non-linear distortions, they do not perfectly cover each other.

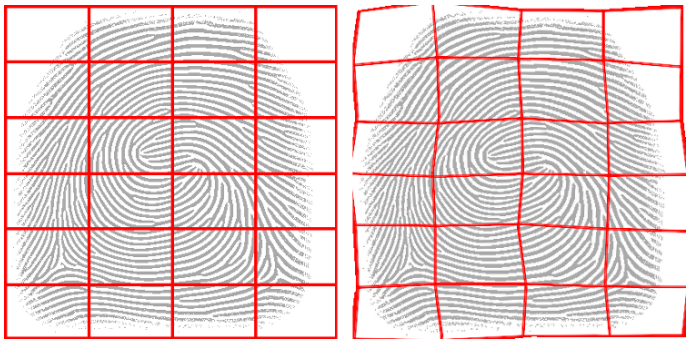


Figure 4.8: Left image shows the grid placed onto the master fingerprint before being deformed. Right image shows the grid after the deformation being performed.

and S parameter is randomly generated from a range of $[0.85, 1.15]$. Figure 4.9 shows some example outcomes.

4.6 Ridge Perturbation

Ridge perturbations are the discontinuities on ridge lines and irregularities in the ink pattern. Example real fingerprint patches with such perturbations are shown in Figure 4.10. Fingerprints may have bright and dark regions that affect the ridge prominence visually. Figure 4.11 shows a real fingerprint and its ink density map. In order to generate

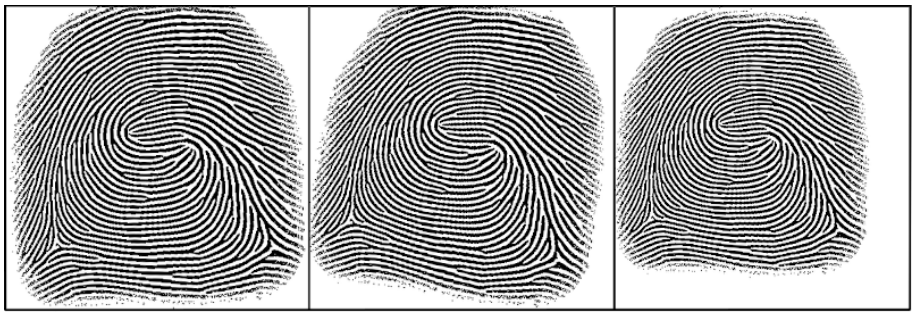


Figure 4.9: Examples of rotation, translation, and scaling operations applied to a fingerprint.

visually realistic, noisy ridge pattern images, those perturbations should be included in the generation process.

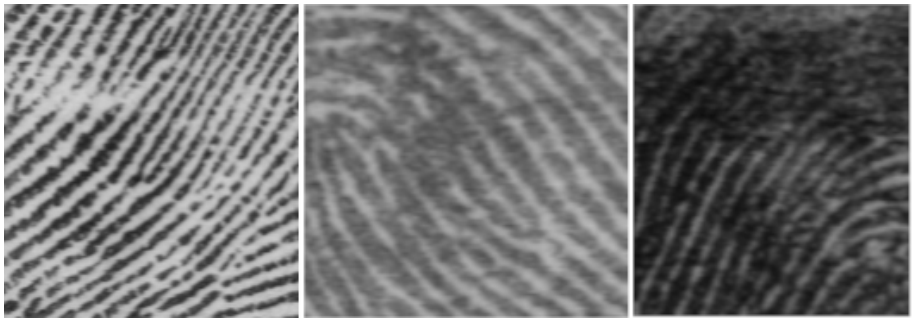


Figure 4.10: The Ridge discontinuities and ink irregularities.

In order to generate perturbations, a function that determines whether the ridge is visible or not in a specific pixel is defined with the following parameters. P is the probability map of being whitefor each pixel, M is the master fingerprint image that stores the ridge information. R is the final noisy ridge pattern image that is calculated using $Perturbate(P, M)$ function:

$$R_{x,y} = Perturbate_{x,y}(P_{x,y}, M_{x,y}) \tag{4.1}$$

$Perturbate(P, M)$ function performs the following operations:

1. For each pixel sample a $score$ uniformly distributed between [0, 1].
2. Initialize R in a way that all pixel values are white.

Figure 4.11: Left image is a real fingerprint image and right image is the median filter applied version of the left image using $(27x27)$ kernel. Some regions are more prominent and some regions are light due to non-uniform distribution of the ink density.

3. Set $R_{x,y}$ values as black where $score_{x,y} \leq P_{x,y}$ and $M_{x,y}$ is black.

4. Apply Gaussian blur to R.

Figure 4.12 shows R image generated using uniform P maps of different levels (same score for all pixels). With uniform P maps the ridge discontinuities are simulated. However, they do not create prominent or light regions. In order to solve this, again, a coherent noise map is generated to be used as P. In other words, the P map is formed by combining noise components $P_1, P_2, .., P_N$, ramdomly generated at different window sizes $w_1, w_2, .., w_N$, where N is randomly sampled from the interval $[1, 30]$ and $w_i = 2i + 2$. After all P_i are generated, they are scaled up to the same size $(400x400)$ using bi-cubic interpolation, and then summed up ($P = \sum_{i=1}^{N} scale(P_i, 400x400)$). Figure 4.13 shows the P map generation process.

Next, min-max scaling is applied to the P map so that it remains in the interval $[p_{min}, p_{max}]$. As p_{min} value gets larger, the total number of black pixels decrease thus the fingerprint ridges get lighter. Additionally, as p_{max} gets smaller, the total number of the black pixels increase so the ridge lines get more prominent. In addition, a small range of P values lead to uniformly distributed ink patterns and a large range of P values lead to irregular prominent and light regions. Figure 4.14 and Figure 4.15 show different outputs for different values of p_{min} and p_{max}. Figure 4.16 shows an example output of the perturbation process using a coherent probability map.

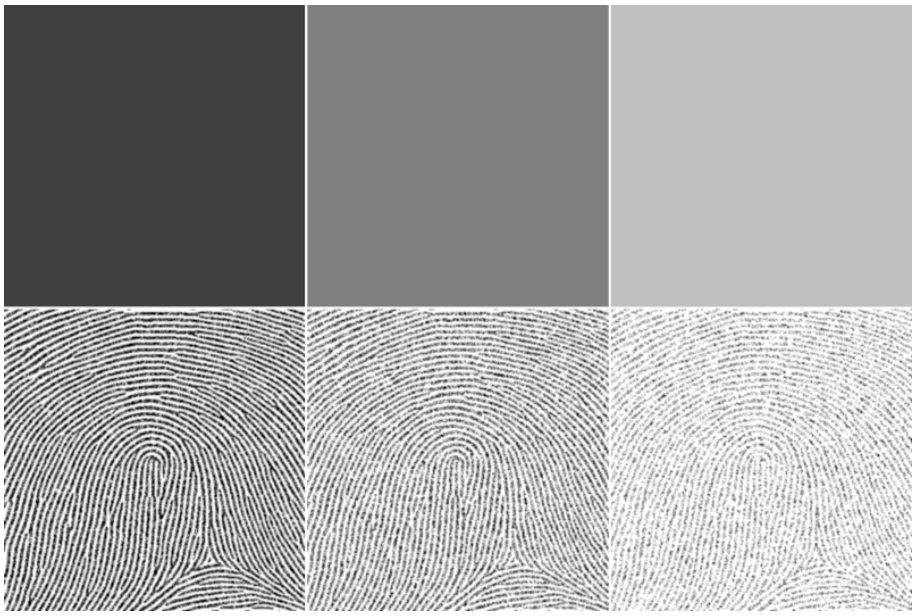


Figure 4.12: Top row images indicates the probability maps which are globally set to constant values, $P = 0.25$, $P = 0.50$, $P = 0.75$, respectively. Bottom row images are the R images which are modeled by the $Perturbate(P, M)$ function.

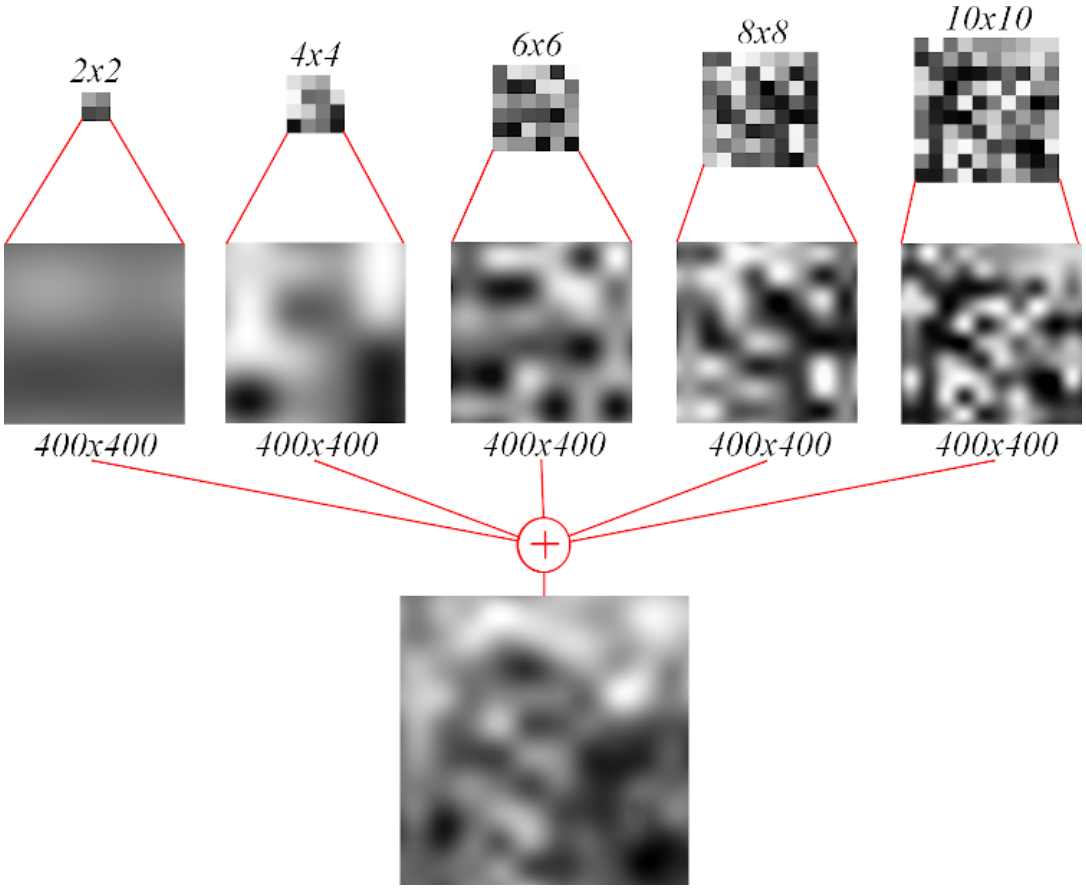


Figure 4.13: Components of the P map where $N = 5$.

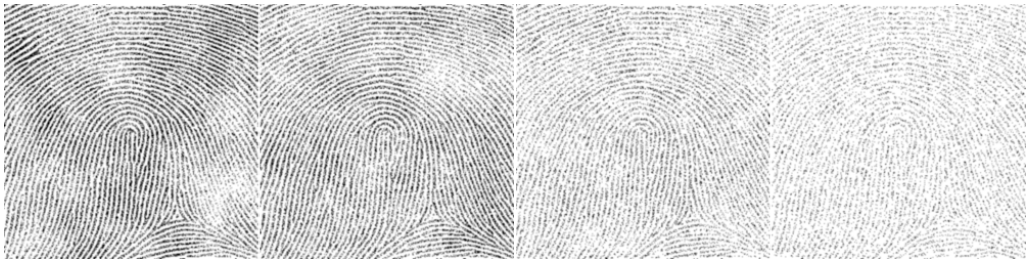


Figure 4.14: Output images for $p_{max} = 0.85$ and p_{min} values in $[0.15, 0.35, 0.65, 0.85]$. When the p_{min} values increase, the prominent and light regions increase and when p_{min} is 0.85, the ink pattern is uniform.

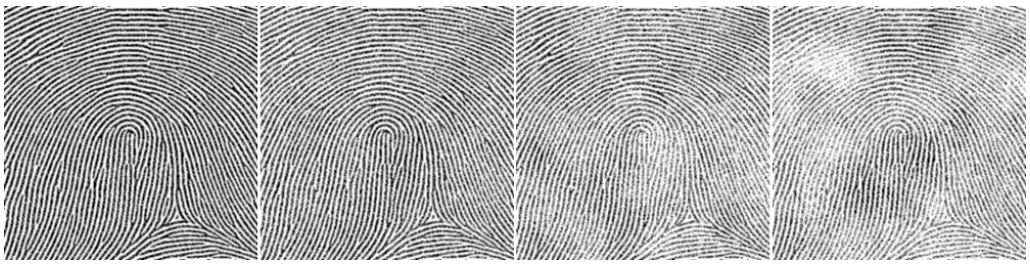


Figure 4.15: Output images for $p_{min} = 0.15$ and p_{max} values in $[0.15, 0.35, 0.65, 0.85]$. When the p_{max} values decrease, the dark regions increase and when p_{max} is 0.15, the ink pattern is uniform.

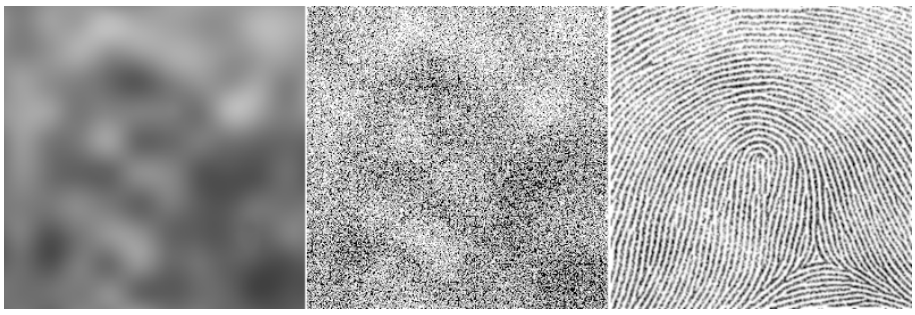


Figure 4.16: Probability map P, ink density map, and generated fingerprint, respectively.

4.7 Background

For this study, NIST SD4 [1] dataset is used which was collected using offline-sensing methods. Fingerprints are imprinted on cards using ink and digitized by scanning those cards. For this reason, the fingerprint images have a paper background with additional horizontal and vertical lines, annotations, marks, and stains. Figure 4.17 shows texamples of different backgrounds from the NIST SD4 dataset. To generate realistic backgrounds, real fingerprint images are visually inspected and some variations are modeled.

Figure 4.17: The fingerprints from NIST SD4 dataset. Backgrounds of the fingerprints have the following variations: annotations, numbers, horizontal and vertical lines.

To generate a paper-like texture, again a coherent noise is generated in the same way as in the frequency image generation (Section 3.2) and perturbation generation (Section 4.6). Then a random noise image is generated and added to the coherent noise image with different weights. Next, marks, and annotations that often exist on the fingerprint images (such as digits, class labels and finger info) are simulated and printed on the background at a random location and random scale. Then, a random number of lines and dots are added to random positions with random angles and sizes. Finally, uniform noise is added to the background image and it is blurred by applying a Gaussian filter. The noisy master fingerprint is imposed on the background with the addition of a coherent noise layer between the background and the noisy master fingerprint. Finally, the brightness level and contrast adjustments (Sigmoid correction) are applied. Figure 4.18 shows some examples of generated backgrounds.

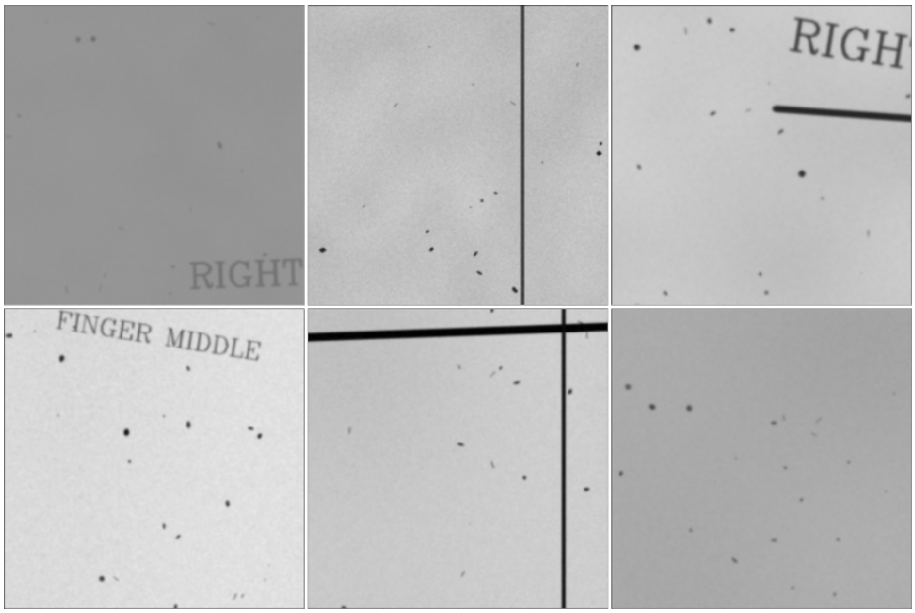


Figure 4.18: Generated background examples.

4.8 Summary

In order to simulate real-world distortions on the generated noise-free master fingerprints, the following degradations are modeled and applied:

1. Scar: In order to model scars, a random number of ellipses with random length, thickness, and angle are added to the fingerprint images.

2. Ridge Thickness: Morphological operations are used to simulate the wetness/dryness of fingerprint or high/low pressure against the surface.

3. Fingerprint Area: The fingerprint shape model proposed in [7] is employed.

4. Skin Deformation and elasticity of the fingers: They are modeled by piecewise affine transformation on a regular grid.

5. Pose Variation: Different displacements of the fingers against the touch surface are modeled by using image rotation, translation, and scale.

6. Ridge Perturbation: ridge discontinuities, ink irregularities, and prominent/light regions on the fingerprints are modeled by generating probability maps using coherent noise. For each ridge pixel, a randomly generated score is assigned and thresholded using the probability map to model whether a pixel is black/ink or white/ink-less.

7. Background texture with marks, lines, and annotations: In order to obtain a similar background with the real dataset (NIST SD4), a paper-like texture is generated and

random annotations, dots, and lines are added. Then, uniform noise is added to the background, the fingerprint is imposed on the background, and the Gaussian blur applied to the image. Finally, image exposure is adjusted using Sigmoid correction.

Figures 4.19, 4.20, 4.21, 4.22, and 4.23 show synthetically generated fingerprint impression examples whose for left loop, right loop, tented arch, arch, and whorl classes respectively.

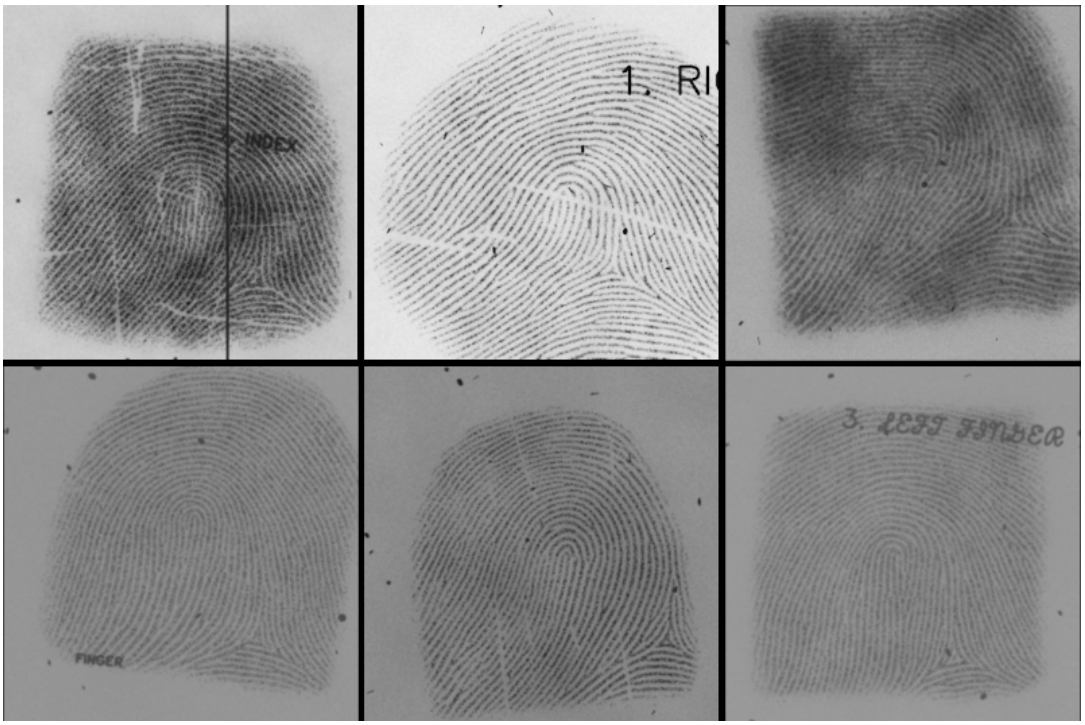


Figure 4.19: Left loop impressions.

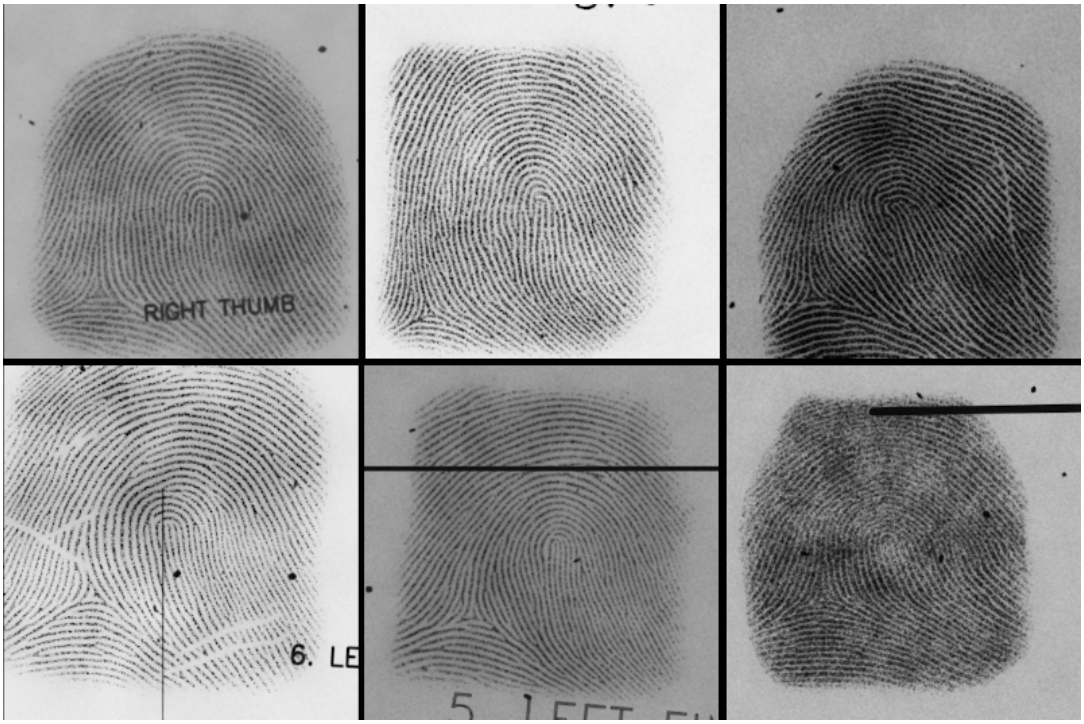


Figure 4.20: Right loop impressions.

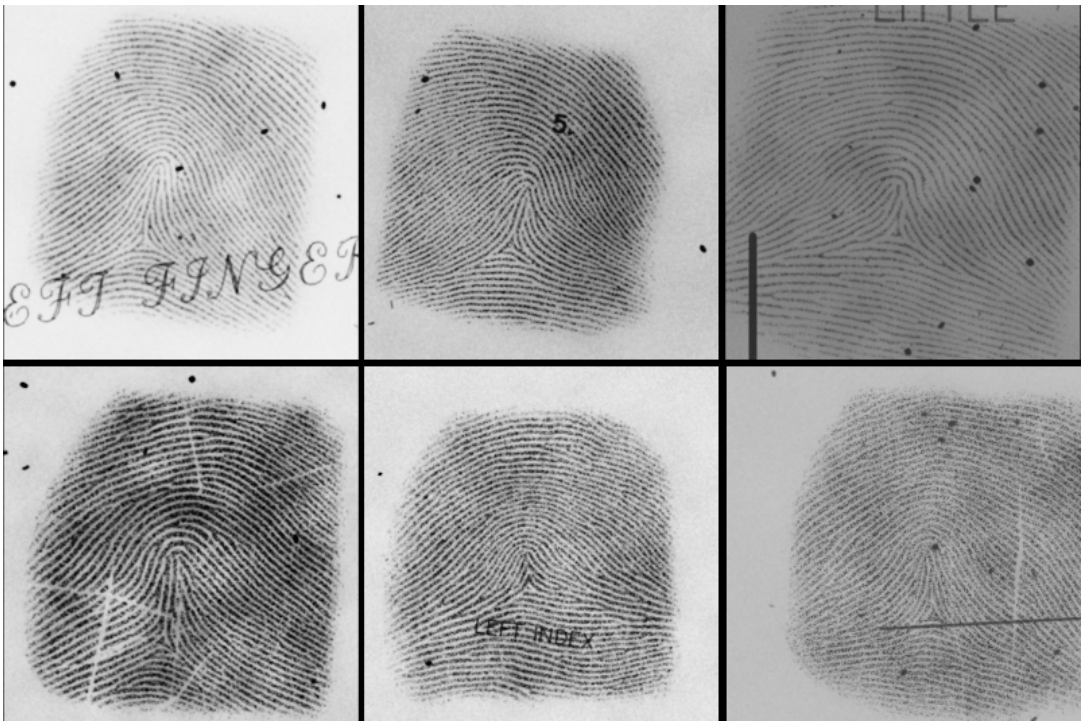


Figure 4.21: Tented arch impressions.

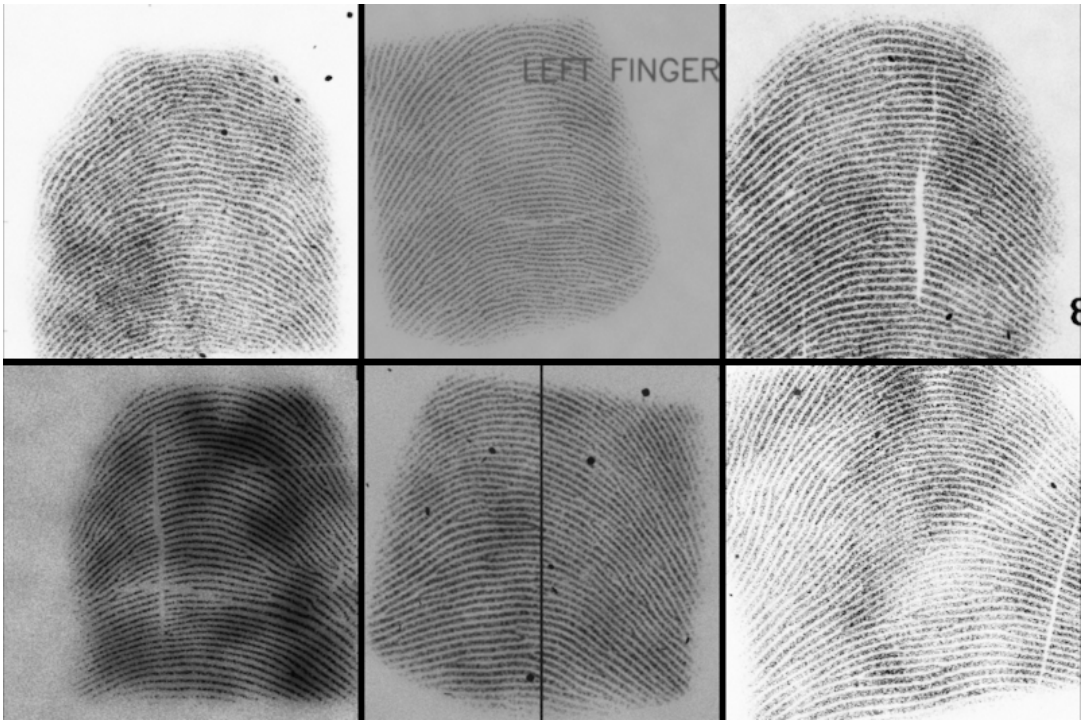


Figure 4.22: Arch impressions.

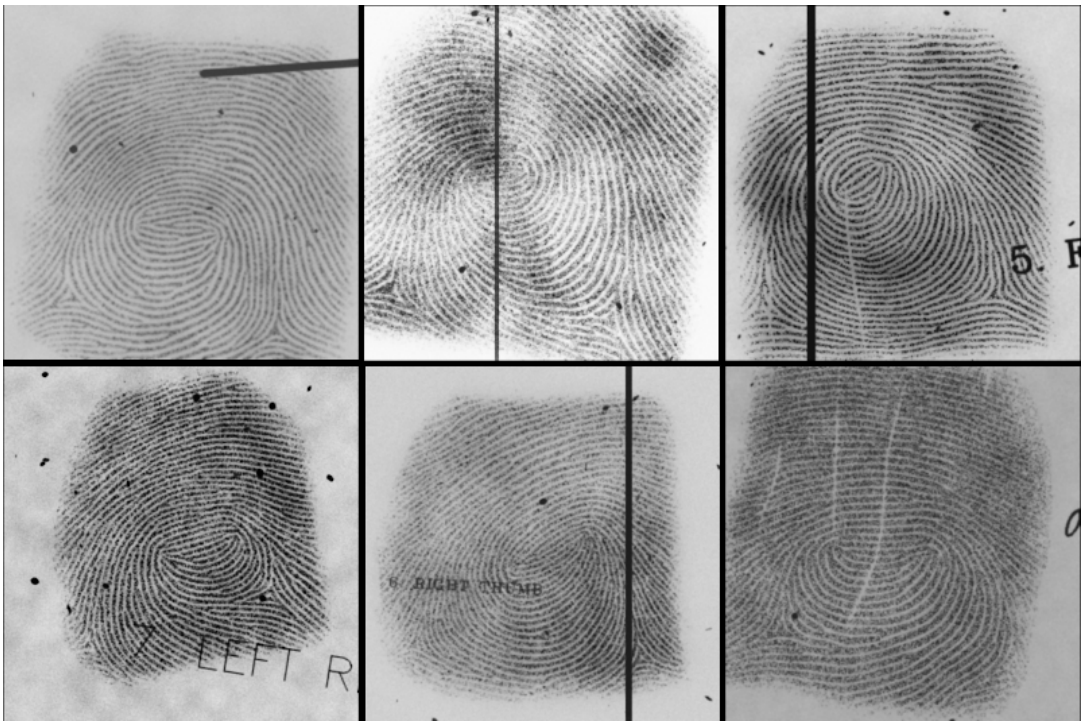


Figure 4.23: Whorl impressions.

CHAPTER 5

EXPERIMENTAL RESULTS

The methodology described in the earlier chapters can be used to generate large synthetic datasets. This chapter focuses on the usability of these datasets using a deep learning based fingerprint classification system. The fingerprint classification task is chosen because it is a fundamental stage for many fingerprint recognition systems. It can decrease the number of required comparisons and reduce the response time of the matching algorithms.

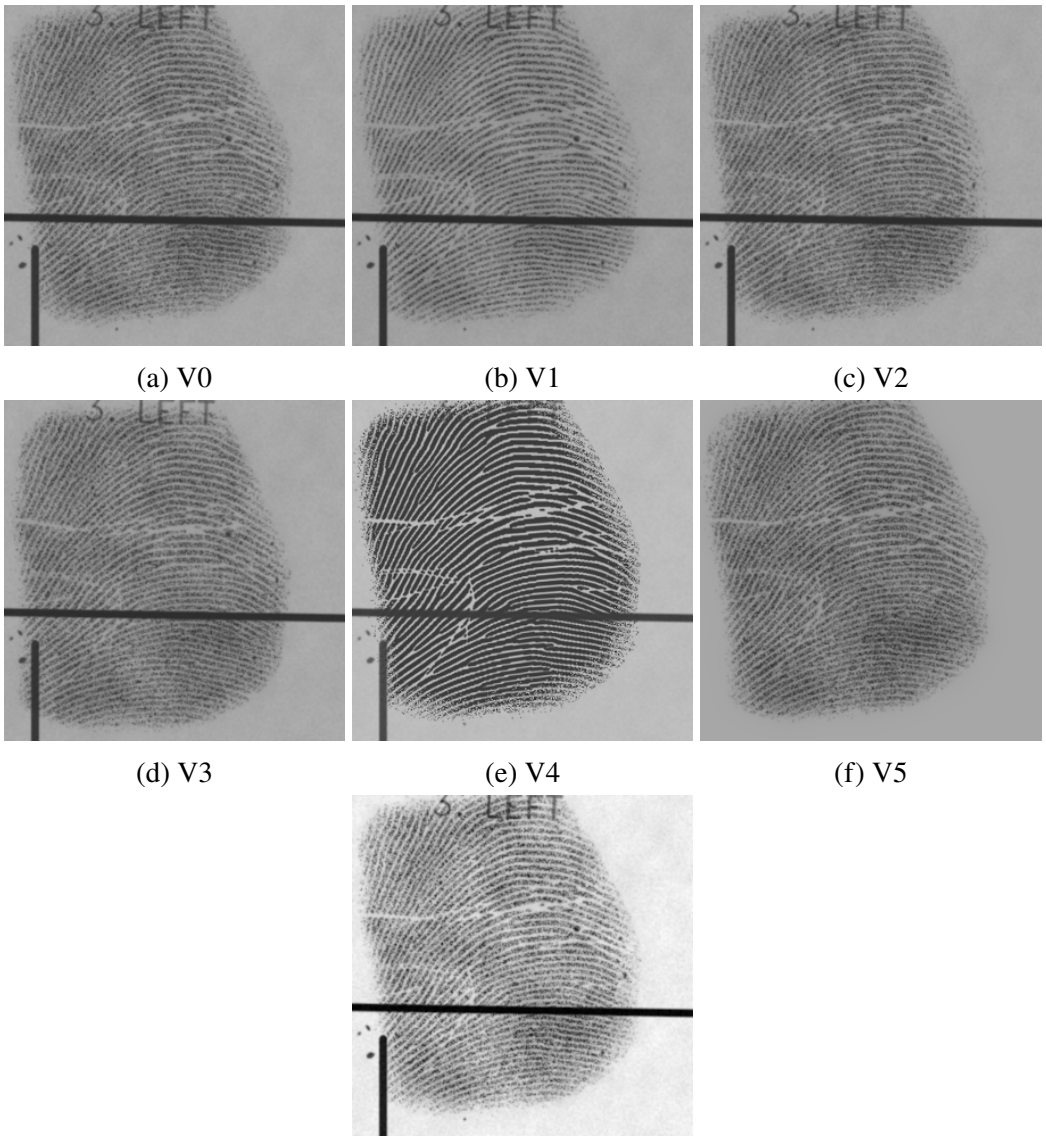
In the experiments, NIST-SD4 is used as the real fingerprint dataset for comparison and assessment of the synthetic dataset. NIST SD4 dataset contains 4000 grayscale fingerprint images of size 512x512 and obtained from 2000 fingers. In the dataset, fingerprint classes are labelled and each class have 400 samples. The main purpose of the dataset is to develop and evaluate automated fingerprint classification algorithms. In the literature, a common approach to divide the NIST SD4 dataset into train and test partitions is that the first 1000 fingers are used for training and the last 1000 fingers are used for testing. In this way, both sets have 2000 fingerprints with two impressions of 1000 fingers.

Additionally, several synthetic datasets similar to NIST-SD4 are generated each with the size of 8000 (1600 samples for each fingerprint class) to be used to train a fingerprint classification system. Natural distribution of fingerprint classes are not uniform (arch:3.7%, tented arch:2.9%, left loop:33.8%, right loop:31.7%, whorl:27.9%) [27]. However, in this study, an equal number of fingerprint images are generated for each class because NIST SD4 dataset has a uniform class distribution. The training set of NIST SD4 and the synthetically generated dataset are used in training the classifiers and all tests are conducted on the test set of NIST SD4. Table 5.1 summarizes the properties of the synthetically generated datasets.

Five experiments are designed to analyze the generalization ability of fingerprint datasets, by testing if they are able to achieve accurate classification results on real data on to improve classifier performances when used as extra training data. The experiments are conducted using ResNet18 [28] and VGG19 [29] deep learning architectures with and without using transfer learning. In the transfer learning case, models that are pre-trained on the ImageNet dataset [30] are used. Each experiment and obtained results are detailed in the following sections.

Table 5.1: Synthetically generated datasets and their descriptions.

Dataset name	**Excluded Distortion Type**
V0	All degradations are applied.
V1	The ridge variation is excluded.
V2	The skin distortion is excluded
V3	The pose variations are excluded
V4	The Ridge perturbation is excluded
V5	The Background is excluded
V6	Post processing (sigmoid correction and brightness adjustment) is excluded.



(a) V0 (b) V1 (c) V2

(d) V3 (e) V4 (f) V5

(g) V6

Figure 5.1: Same fingerprint from different **Vx** databases.

5.1 Experiment 1

The aim of this experiment is to analyze the effect of degradation types that are applied to synthetic data on the classification accuracies. In order to do that, a base dataset (**V0**) that contains all types of degradations is generated and one type of degradation is excluded at a time from the whole set of degradations to obtain different dataset versions. For example in **V1**, the ridge variations are disabled but all other degradations are applied, in **V2** skin distortion is not included while others are applied. In this way, 7 synthetic datasets are generated and used in training the fingerprint classification models. Table 5.1 summarizes the synthetic dataset versions and Figure 5.1 shows an example of different **Vx** versions.

For this experiment, the **Vx** datasets which originally contain 8000 samples are reduced to size 2000 by using the first 400 samples of each class. The classification networks are trained using only synthetic datasets and tested on the real images. For ResNet and VGG-19 networks with and without transfer learning and 7 synthetic datasets, $2x2x7 = 28$ models are trained in total and the results are given in Figure 5.2

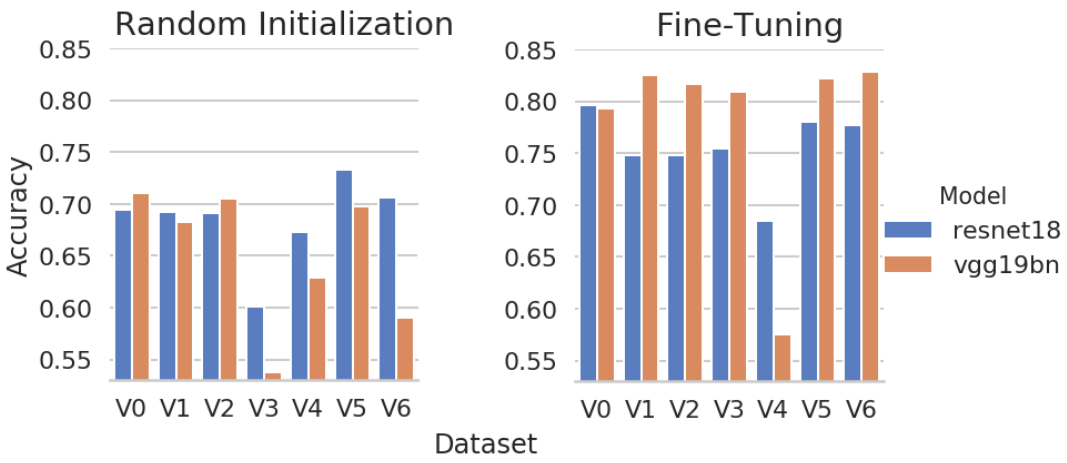


Figure 5.2: Experiment 1 results.

With **V0** dataset and without pre-training, the classification accuracies on real data are 69.45% and 71.00% for ResNet18 and VGG19 networks, respectively. The accuracies of **V1, V2** is not affected significantly by the applied exclusions. It may be a clue that these variations are not very representative for real-world distortions. However, the accuracies **V3** and **V4** sharply decrease for both models. This shows that pose variation and ridge perturbation introduce helpful variations and excluding them causes a loss in the

representation power of synthetic fingerprint datasets. Interestingly, for ResNet without pre-training, **V5** achieves on improved accuracy. The reason for this improvement may be related to the network architecture. ResNet learns from the residuals, and in deeper layers, the background information may propagate and override the fingerprint information. In other words, the model may try to extract information from the background instead of the fingerprint. For this reason, removing the background may let ResNet model to focus on the fingerprint itself and learn better.

Another intriguing result is obtained by **V6** on the VGG model without transfer learning. This setup achieves an accuracy below 60%. **V6** is related to the exposure factor, and normally, post-processing makes the histogram of the synthetic data and real fingerprint data similar. Disabling the post processing step causes synthetic images to have a different range of pixel intensities than the real fingerprints in the NIST dataset and the VGG model may have learned the synthetic intensity distribution which does not generalize well to real fingerprint.

To sum up, pose variation and ridge perturbations are significant for synthetic datasets if no transfer learning is used for training the networks. Additionally, according to model architectures, accuracies are positively or negatively affected by the variations. For instance, VGG model is able to tolerate the lack of background however its performance is negatively affected by the difference in pixel intensity distributions.

with pre-training, ResNet and VGG models trained with **V0** achieves 79.65% and 79.35% accuracies, respectively. ResNet trained with **V0** achieved the best performance among other ResNet models. The ResNet models are pre-trained on the ImageNet dataset whose context is very different from the context of the fingerprints. Thus, it is not surprising to achieve the best performance on the dataset that has the most variance. From the results with **V4**, it is clearly seen that the most important degradation is ridge perturbations for both pre-trained networks. This may be because the ImageNet dataset does not include this kind of noise, thus removing the ridge perturbations from the fingerprint generation process causes a crucial loss of information.

Both with and without transfer learning, it is observed that the most important variation is ridge perturbations because their absence cause sharp decreases in the performances. Additionally, when pose variations are observed to be more important when no pre-training is used. Pre-trained models have learned the rotation-scale-translation variations from the ImageNet dataset thus the performance loss with **V3** is not so critical. **V1, V2** datasets do not have any significant performance loss in configurations. It is suspected that those variations may not be very representative of real world deformations and they should be examined and improved in future work.

5.2 Experiment 2

In this experiment, the classifiers are again trained using **only** synthetic datasets but this time with different number of images $[250, 500, 750, 1000, 2000, 4000]$. The purpose of this experiment is to observe the impact of synthetic dataset sizes. **V0** is selected as the training dataset version and the first N-images are used with equal samples for each class

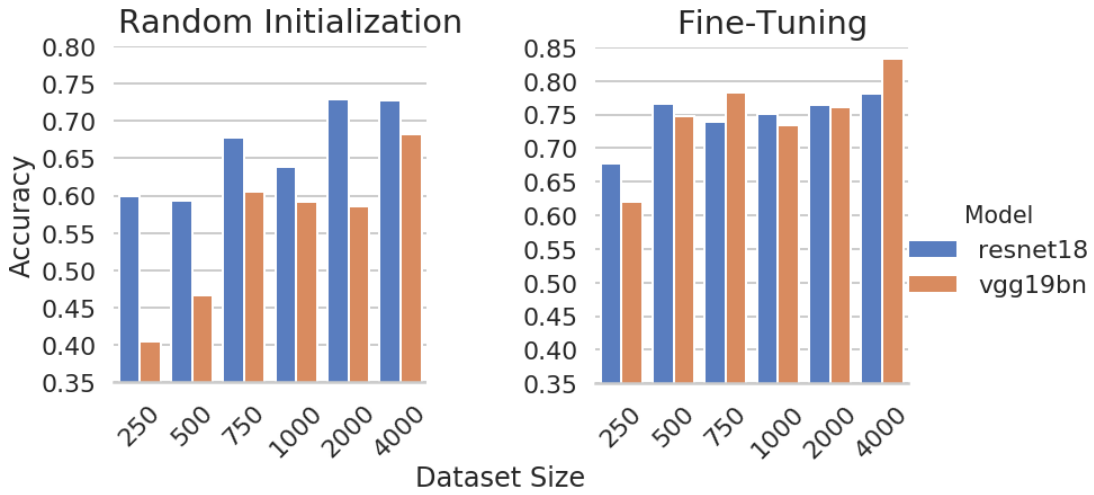


Figure 5.3: Experiment 2 results.

In total, 24 configurations are tested using ResNet and VGG networks. For most cases a strong correlation is observed between the training set size and the accuracies. The real fingerprints are classified better as the number of the synthetic samples increase. This result is expected because more samples carry more variations, and this diversity make the models learn better and generalize well.

For both with and without transfer learning, with dataset of sizes $[1000, 2000]$, accuracies are slightly reduced compared to size 750. This reduction may be explained with the quality of samples between 750^{th} and 2000^{th} in the synthetic dataset. Those images may not include sufficiently good variations that contribute to the generalization ability of the classifiers. However, in general, it can be concluded that when more synthetic data is used in training, the performances get better. Pre-trained VGG network fine-tuned with 4000 synthetic fingerprint images achieves 83.30% accuracy, which is remarkable for a training set wit no real fingerprint data.

5.3 Experiment 3

In the third experiment, the purpose is to compare the classification performances of mixed (real+synthetic) datasets with purely real datasets again in different sizes. The mixed datasets of size 2N are created by using N synthetic data **V0** and N real data. They are compared with the real fingerprint datasets of size 2N. The classifiers are trained with mixed and real datasets and tested on real fingerprint images. N is taken as $[125, 250, 500, 1000, 2000]$

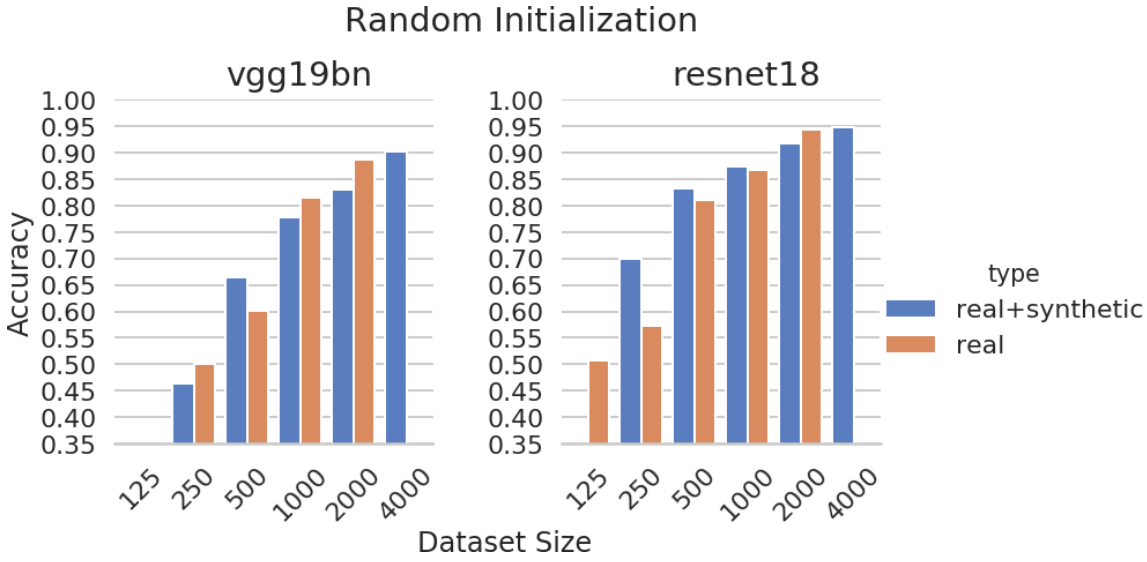


Figure 5.4: Experiment 3 random-initialization results. The results of the real+synthetic dataset with the size of 125 is missing because the models failed to converge. The results of the real dataset with the size of 4000 is missing because the maximum number of available real data is 2000.

In all cases, the performances of mixed datasets are very close to the purely real datasets of equal sizes. Even when half of the training dataset is synthetic, the models can achieve similar results compared to when training is done using only real images. The results without transfer learning even show that mixed data may perform better than real data in some cases. For instance, in ResNet models mixed data outperforms real data for dataset of sizes $[250, 500, 1000]$. This shows that the synthetic data may introduce useful variation contributions especially when the real dataset sizes are not very large. The same outcome is also observed for VGG model with the dataset size of 500.

Additionally, when the mixed dataset performances are compared with the real dataset performances that have same number of real images (N real+N synthetic vs N

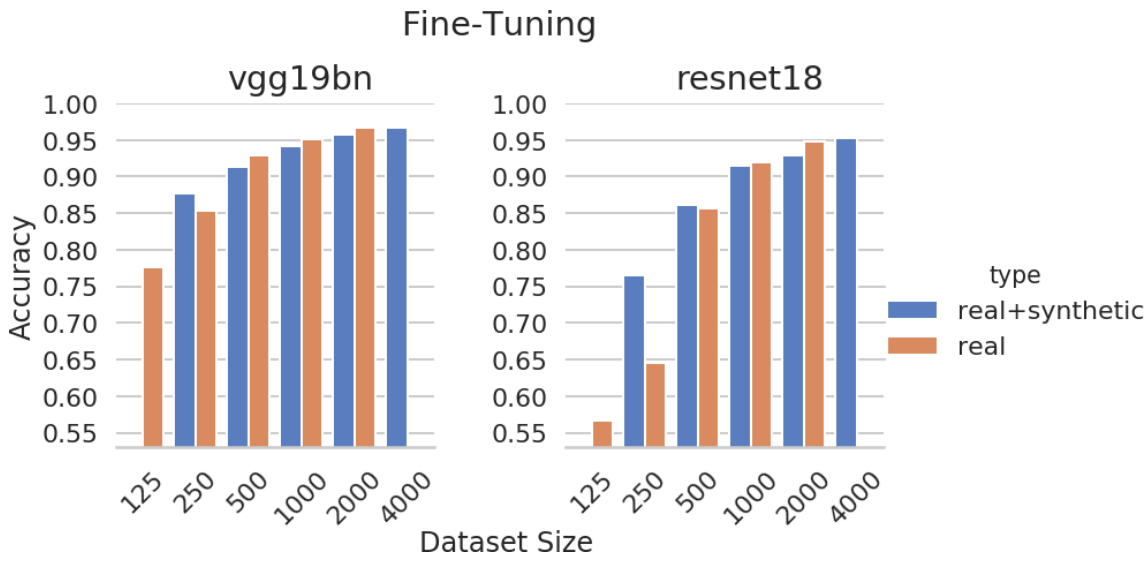


Figure 5.5: Experiment 3 fine-tuning results. The results of the real+synthetic dataset with the size of 125 is missing because the models failed to converge. The results of the real dataset with the size of 4000 is missing because the maximum number of available real data is 2000.

real), the results show that adding synthetic data increases the performances in almost all cases. This shows that when there is a fixed amount of real data, adding synthetic data improves to the performances. However, as expected this improvement decreases as the real dataset size increase.

In the case of pre-trained models, the classifiers can achieve more accurate results than non-pretrained versions because they transfer some knowledge from the ImageNet dataset. However, mixed dataset accuracies are still very close to the real ones and it is clearly seen that adding same number of synthetic data on a real dataset also improves the performances. Pre-trained VGG model fine-tuned with the mixed dataset of size 4000 achieves the best performance with 95.30% accuracy which beats the performance of the same model fine-tuned with 2000 real images, the maximum number of images in NIST SD4 training partition, is 94.80%. This shows that it is possible to improve the fingerprint classification performances by increasing the number of training samples by incorporating synthetic data.

5.4 Experiment 4

The purpose of this experiment is to observe the potential of the synthetic data in improving accuracies by adding different sizes of synthetic data to a constant sized real

data. To do that, 1000 real samples are taken from NIST-train as the base training set. Then, synthetic data **V0** is incrementally added to that set. The increments are [0, 250, 500, 1000, 2000] resulting in the dataset sizes of [1000R+0S=**1000**, 1000R+250S=**1250**, 1000R+500S=**1500**, 1000R+1000S=**2000**, 1000d+2000S=**3000**]. All tests are again conducted on real fingerprint images in the test partition of NIST SD4 database.

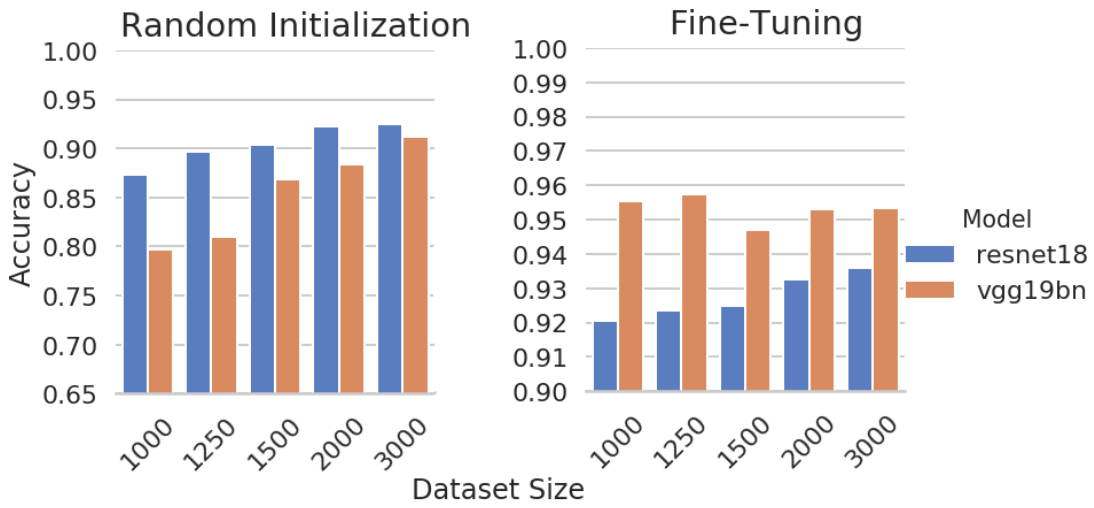


Figure 5.6: Experiment 4 results.

As seen in Figure 5.6, a strong correlation is observed between the increasing size of synthetic data and the accuracies for all cases especially when no pre-training is used. In general, results prove that increasing amounts of synthetic data is capable of improving the performance of the classifiers. As an exception, the pre-trained VGG network is not always positively affected by the increasing number of synthetic data. Our justification for that result is that since the classification of fingerprints is not a very complex problem and the pre-trained VGG have learned main features from the ImageNet dataset, 1000 real fingerprint images are already sufficient to train the classifier, and thus, the synthetic dataset does not introduce any improvements. Another possibility is that the synthetic dataset variations are not representative enough to improve the accuracy of pre-trained networks. Unfortunately, more real data is needed in order to make such analysis.

5.5 Experiment 5

In this experiment, the generated synthetic dataset is tested on a discriminative network. Equal number of real images in the training set of NIST SD4 and synthetic

image in **V0** are mixed with different dataset sizes [250, 500, 1000, 1500, 2000]. For this experiment, VGG model is used with randomly initialized weights and the systems are tested on a new test set which is created using NIST-test (2000 real fingerprint image) and synthetic dataset of size 2000 which is not used in training stage. Figure 5.7 shows the real/synthetic classification accuracies.

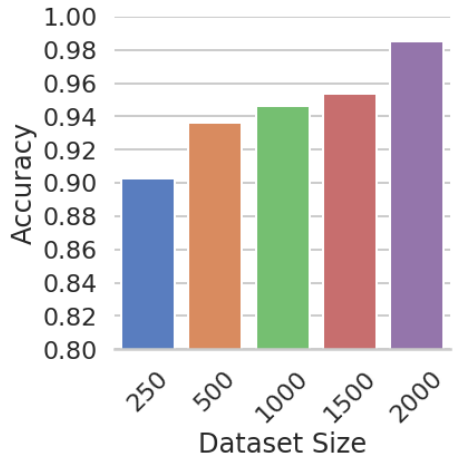
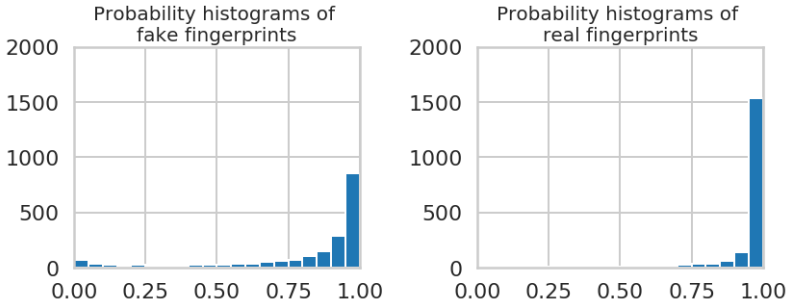
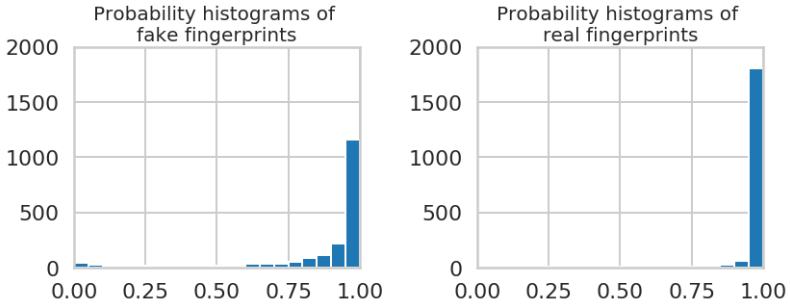


Figure 5.7: Experiment 5 results.

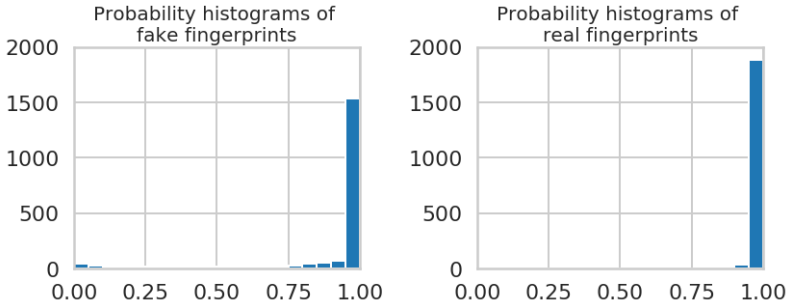
The results indicate that the discriminator network can detect the synthetic fingerprints which are generated by the proposed approach, with high accuracy. The discriminator networks may be successful due to the small details in real fingerprints or the unrealistic statistical feature distribution in the fake fingerprints (e.g. absence of level 3 features, random minutiae configuration, etc), there are some fingerprints which can fool the discriminator network. This is a promising observation, and this insight can be used to obtain a more realistic synthetic fingerprint dataset. By using the synthetic images that can spoof the network and eliminating the others, a better quality dataset can be obtained and used to train the classifiers. Another observation is that as the training set size increases, the discriminator accuracies increase as well. It is also possible to achieve higher accuracies by using transfer learning. Figure 5.8 shows the probability histograms of real and synthetic fingerprints with respect to dataset size. When the dataset sizes increase, the network can detect, synthetic fingerprints more confidently, but the confidence for real class is very high in almost all cases.



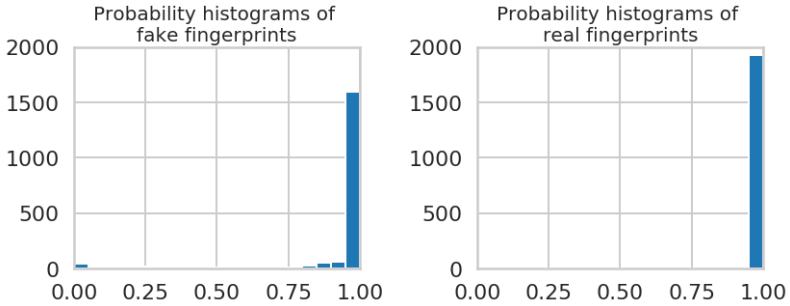
(a) Training Set Size = 250



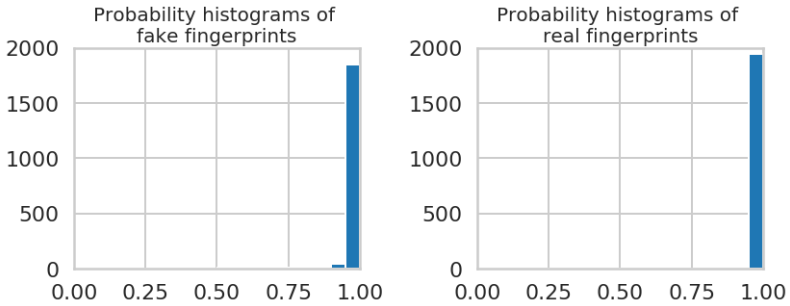
(b) Training Set Size = 500



(c) Training Set Size = 1000



(d) Training Set Size = 1500



(e) Training Set Size = 2000

Figure 5.8: Probabilty histograms of real and synthetic fingerprints for different training set sizes.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this study, synthetic fingerprint images are generated with a high level of visual realism. Because publicly available databases are limited for the fingerprint domain, the main purpose of this work is to generate synthetic fingerprints and test if they can be used for training deep learning systems.

For this purpose, a model-based technique is developed to synthesize fingerprint images. In master fingerprint generation step, orientation images are generated using Vizcaya and Gerhardt's method for left loop, right loop, tented arch and whorl classes, a proposed model is used for generating arch classes. Frequency images are obtained with the help of bicubic interpolation applied on randomly generated small images. The orientation and frequency images are used to generate Gabor filters and the master fingerprint is obtained by applying these Gabor filters on randomly initialized images iteratively.

For impression generation from the master fingerprints, some real-world degradations are simulated in order to introduce more realism and variation to the fingerprint images. Pose variations, fingerprint area, non-linear distortions due to skin elasticity, ridge variations, scars, background and lightning conditions are simulated and applied to master fingerprint images.

In order to test the representative power of the synthetic fingerprints, five sets of experiments are conducted. Fingerprint classification is chosen as the test case. The classifiers are trained with different configurations using synthetic data and tested on real fingerprint images. In the experiments, the impact of the simulated distortions and the effectiveness of synthetic data in training deep neural networks are analyzed. Additionally, a discriminative network that classifies real and synthetic data is trained and tested to see whether it is possible to deceive a deep learning based classifier. In this way, the ability of the synthetic fingerprint images to imitate real ones is also analyzed.

The results show that the generated synthetic images can be used in training deep neural networks and help them achieve promising results. Using only synthetic data for training, the classifiers can achieve nearly 85% accuracy on the test set of NIST SD4.

This result implies the possibility of training deep learning systems with no real data. Synthetic data is added to real data for training, it improves the performances significantly, especially when there is a small number of real data and no pre-training. This is also an important outcome because the main focus of this study is to overcome the problem of the small number of training samples.

When the performances of real and mixed training sets of same size, it is observed that are compared mixed data achieves similar results. This result helps us to asses the realism of the synthetic data. With a different approach, this realism is also put to the test by constructing a discriminative network. Despite the fact that discriminative network can detect synthetic data well, the results are promising because some synthetic fingerprints are able to spoof the network.

All in all, the variation and representative power of the generated synthetic data seems sufficient for improving the performances of the networks when the training set size is small. However, there are some cases where the contributions are negative. Especially, for increasing the performances when large real training sets are available, the synthetic dataset needs improvement.

6.2 Future work

Various parts of the generation process can be improved. For instance: in master fingerprint generation, singular points are sampled randomly from predefined rectangle areas. Thus, the distribution of the singular points are not be realistic. By analyzing real fingerprint images, a non-parametric model can be constructed and these singular points can be sampled from the model. In this way, the sampling distribution of the singular points can be statistically realistic. The orientation images and the frequency images are generated independently from each other. This assumption may cause the model to generate statistically unrealistic fingerprint images in terms of minutiae configurations. A learning-based generation of orientation and frequency image may improve the realness of the synthetic images. The generation of level 3 features is not covered in master fingerprint generation phase. This addition may advance the realism of the fingerprints. In the impression generation part, non-linear deformations of the skin elasticity are modeled by Piece-wise Affine Transformation on a randomly generated grid. By using more physically relevant models, skin distortions can be more realistic. In ridge perturbation, the used model sometimes generates unsightly samples, these can also be improved.

Additional experiments can also be conducted, such as since the discriminator network is not able to detect all synthetic images, a more robust synthetic dataset can be

created and tested to see if accuracies improve further. In other words, a database may be generated which contains only synthetic fingerprint images that can fool the discriminator. it is expected to have higher positive impact on classifier performances.

Additionally, the effect of synthetic data on training deep systems is analyzed on fingerprint classification in this study. It can also be analyzed in other stages of the fingerprint recognition pipeline such as minutiae extraction, fingerprint matching, identification, etc.

REFERENCES

[1] C. I. Watson and C. L. Wilson, "Nist special database 4," *Fingerprint Database, National Institute of Standards and Technology*, vol. 17, no. 77, p. 5, 1992.

[2] D. Maio, D. Maltoni, R. Cappelli, J. L. Wayman, and A. K. Jain, "Fvc2002: Second fingerprint verification competition," in *Object recognition supported by user interaction for service robots*, vol. 3, pp. 811–814, IEEE, 2002.

[3] Y. Hu, X. Jing, B. Zhang, and X. Zhu, "Low quality fingerprint image enhancement based on gabor filter," in *2010 2nd International Conference on Advanced Computer Control*, vol. 2, pp. 195–199, IEEE, 2010.

[4] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.

[5] R. Ramotowski, *Lee and Gaensslen's advances in fingerprint technology*. CRC press, 2012.

[6] F. Galton, *Finger prints: With a new introduction to the Da Capo edition by Dr. Harold Cummins*. Da Capo Press, 1965.

[7] R. Cappelli, D. Maio, and D. Maltoni, "Sfinge: an approach to synthetic fingerprint generation," in *International Workshop on Biometric Technologies (BT2004)*, pp. 147–154, 2004.

[8] D. Maltoni and R. Cappelli, "Advances in fingerprint modeling," *Image and Vision Computing*, vol. 27, no. 3, pp. 258–268, 2009.

[9] J. Zhou, Y. Wang, Z. Sun, Y. Xu, L. Shen, J. Feng, S. Shan, Y. Qiao, Z. Guo, and S. Yu, *Biometric Recognition: 12th Chinese Conference, CCBR 2017, Shenzhen, China, October 28-29, 2017, Proceedings*, vol. 10568. Springer, 2017.

[10] A. K. Jain, Y. Chen, and M. Demirkus, "Pores and ridges: High-resolution fingerprint matching using level 3 features," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 1, pp. 15–27, 2006.

[11] P. Johnson, F. Hua, and S. Schuckers, "Texture modeling for synthetic fingerprint generation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 154–159, 2013.

[12] R. Cappelli, D. Maio, and D. Maltoni, "Synthetic fingerprint-database generation," in *Object recognition supported by user interaction for service robots*, vol. 3, pp. 744–747, IEEE, 2002.

[13] B. G. Sherlock and D. M. Monro, "A model for interpreting fingerprint topology," *Pattern recognition*, vol. 26, no. 7, pp. 1047–1055, 1993.

[14] I. Fogel and D. Sagi, "Gabor filters as texture discriminator," *Biological cybernetics*, vol. 61, no. 2, pp. 103–113, 1989.

[15] R. Cappelli, D. Maio, and D. Maltoni, "An improved noise model for the generation of synthetic fingerprints," in *ICARCV 2004 8th Control, Automation, Robotics and Vision Conference, 2004.*, vol. 2, pp. 1250–1255, IEEE, 2004.

[16] K. Perlin, "An image synthesizer," *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[17] C. Imdahl, S. Huckemann, and C. Gottschlich, "Towards generating realistic synthetic fingerprint images," in *2015 9th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pp. 78–82, IEEE, 2015.

[18] Q. Zhao, A. K. Jain, N. G. Paulter, and M. Taylor, "Fingerprint image synthesis based on statistical feature models," in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pp. 23–30, IEEE, 2012.

[19] P. Bontrager, A. Roy, J. Togelius, N. Memon, and A. Ross, "Deepmasterprints: Generating masterprints for dictionary attacks via latent variable evolution," in *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pp. 1–9, IEEE, 2018.

[20] K. Cao and A. Jain, "Fingerprint synthesis: Evaluating fingerprint search at scale," in *2018 International Conference on Biometrics (ICB)*, pp. 31–38, IEEE, 2018.

[21] M. Attia, M. H. Attia, J. Iskander, K. Saleh, D. Nahavandi, A. Abobakr, M. Hossny, and S. Nahavandi, "Fingerprint synthesis via latent space representation," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pp. 1855–1861, IEEE, 2019.

[22] V. Mistry, J. J. Engelsma, and A. K. Jain, "Fingerprint synthesis: Search with 100 million prints," *arXiv preprint arXiv:1912.07195*, 2019.

[23] Y. Chen and A. K. Jain, "Beyond minutiae: A fingerprint individuality model with pattern, ridge and pore features," in *International Conference on Biometrics*, pp. 523–533, Springer, 2009.

[24] K. G. Larkin and P. A. Fletcher, "A coherent framework for fingerprint analysis: are fingerprints holograms?," *Optics Express*, vol. 15, no. 14, pp. 8667–8677, 2007.

[25] C. Gottschlich and S. Huckemann, "Separating the real from the synthetic: minutiae histograms as fingerprints of fingerprints," *IET Biometrics*, vol. 3, no. 4, pp. 291–301, 2014.

[26] P. R. Vizcaya and L. A. Gerhardt, "A nonlinear orientation model for global description of fingerprints," *Pattern Recognition*, vol. 29, no. 7, pp. 1221–1231, 1996.

[27] C. L. Wilson, G. T. Candela, and C. I. Watson, "Neural network fingerprint classification," *Journal of Artificial Neural Networks*, vol. 1, no. 2, pp. 203–228, 1994.

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[29] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.