Tel-Aviv University
Raymond and Beverly Sackler Faculty of Exact Sciences
The Blavatnik School of Computer Science

Fingerprints Image Spoof Detection and Classification

by
Tatiana Barsky

The dissertation submitted in partial fulfillment of the requirements for the degree of
Master of Science

Prepared under the supervision of
Prof. Yehezkel Yeshurun

January 2009 Shebat 5769

Acknowledgments

I would like to express my gratitude to prof. Yehezkel Yeshurun for his guidance and help throughout the research.

I am deeply indebted to my parents, Maria and Stanislav, for their stimulating support and their essential help. I would like to give my special thanks to my husband Roman, whose help, patience and love enabled me to complete this work. I thank all my family, especially Malvina, Vladimir, and Ira, for their encouragement and faith. I thank my son, Ethan, for his good behavior in the critical moments of the research.

I also thank all the people who helped me to collect and process the fingerprints.

Abstract

Biometric identification is gaining more and more recognition as a leading technology for identity management and security systems. This inevitably is followed by numerous methods that are aimed at breaking those security measures. In this paper we address the problem of fingerprints spoofing, that is, the usage of counterfeit elastic fingerprints. Based on image features extracted from 2D fingerprints images, we present several novel algorithms that classify fingerprints images as real or fake, and demonstrate their performance on a fingerprints database. We conclude that it is possible to achieve a relatively high level of confidence using our approach.

Contents

List of Figures

List of Tables

List of Algorithms

Abbreviations and Notations

AFAS	—	Automatic Fingerprint Authentication System
AFIS	—	Automatic Fingerprint Identification System
ACL	—	Anti-counterfeit Level
CEA	—	Characteristics Extraction Algorithm
FAR	—	False Positive Rate
FRR	—	False Rejection Rate
Sensitivity	—	$1 - \text{FRR}$. Proportion of correctly classified actual positives
Specificity	—	$1 - \text{FAR}$. Proportion of correctly classified actual negatives
Positive Predictive Value	—	$\frac{\text{Correctly classified actual positives}}{\text{Positive classified samples}}$
Negative Predictive Value	—	$\frac{\text{Correctly classified actual negatives}}{\text{Negative classified samples}}$
Positive Likelihood	—	$\frac{\text{Sensitivity}}{(1 - \text{Specificity})}$
Negative Likelihood	—	$\frac{(1 - \text{Sensitivity})}{\text{Specificity}}$
Prevalence	—	$\frac{\text{Actual positives}}{\text{Total number of samples}}$
Error Rate	—	$\frac{\text{Incorrectly classified samples}}{\text{Total number of samples}}$

CER	—	Crossover Error Rate
Correct Rate	—	$1 -$ Error Rate
Inconclusive Rate	—	$\frac{\text{Non-classified samples}}{\text{Total number of samples}}$
Over-fitting	—	Fitting a statistical model that has too many parameters. An absurd and false model may fit perfectly if the model has enough complexity by comparison to the amount of data available.
Curse of the Dimensionality	—	Exponential growth of hypervolume as a function of dimensionality [1]
DOS	—	Denial of Service
FBI	—	Federal Bureau of Investigation
EER	—	Equal Error Rate
BSI	—	The German Federal Office for Information Security
RBF	—	Radial Basis Function
LOOCV	—	leave-one-out cross-validation

Chapter 1

Introduction

Contemporary achievements in scanning and imaging technology gave a boost to the increasing demands for biometric authentication. In today's world advanced biometric authentication methods gain an exceptional significance. Reliable identity recognition is one of the most challenging and critical to many security, business, and civilian processes. Perhaps the most fundamental problem with all biometrics is the enormous difficulty of preventing an identity theft. In the event that an attacker manages to steal your biometric data, you risk being disenfranchised from that system forever.

1.1 Background

Authentication of a person is verifying his identity. Traditionally common forms of security authentication are based on *what you have* (like ID card) and *what you know* (for example PIN or password). Biometric authentication is a relatively new form. Biometrics refers to automatic recognition of an individual based on her behavioral and/or physiological characteristics, such as a fingerprint, iris and face, and, more recently, palm veins and typing rhythms. Companies, producing biometric scanners, claim for user friendliness of biometric identification. Unlike passwords, a fingerprint or iris cannot be lost or forgotten, and the owner does not have to invent a *strong* biometric. However, if a biometric characteristic is stolen or spoofed, it makes much more inconvenience to its owner. He cannot change his iris or fingerprint as easy as he can change a password. Unlike having several passwords for different systems, a biometric characteristic is unique. Once spoofed, all the systems, protected by that characteristic, are compromised. For more drawbacks of biometric methods see [2].

Fingerprints are the oldest biometric sign of identity. The skin on the

inside of a finger is covered with a pattern of ridges and valleys. Every person is believed to have unique fingerprints [3]. This makes fingerprints suitable for verification of the identity of their owner.

There are two types of fingerprint-based biometric systems in terms of their utilization: Automatic Fingerprint Authentication System (AFAS, see Figure 1.2) and Automatic Fingerprint Identification System (AFIS) [4]. In AFAS an input is an identity and a fingerprint image; the output is answer of 'Yes' or 'No' indicating whether the input image belongs to the person whose identity is provided. The system compares the input image with the one addressed by the identity in the database. In AFIS the input is just a fingerprint and the output is a list of identities of persons that can have the given fingerprint and a score for each identity indicating similarity between the two fingerprints (for details see [4]).

Is it secure to use AFAS? Beyond possible network attacks described and researched in [5], an identity falsification using artificial fingerprints is possible—*spoofing*; as a different kind of system abuse, one can deny his authentication claiming that his fingerprint was spoofed, etc. (a comprehensive list of possible attacks is presented by Matsumoto at al. in [6]). Many researches consider the spoofing problem showing various ways to counterfeit the AFAS with artificial finger (for example [6], [7], [8]). Many fingerprint scanners suppliers claim for improbability of spoofing their system (ScienceGL software, The Surround Imager, TST Group). In response, possible solutions to the spoofing problem were proposed from the industry and the academia fields (for example [9]). Many research papers recommend combination of *what you know* and *who you are* methods, i.e. fingerprint authentication with another method such as PIN, password, etc.

One of quickly growing trends is using AFAS in conjunction with smartcards. A *smartcard* is any pocket-sized or smaller card with embedded integrated circuits. The standard perception of a smartcard is a microprocessor card of credit card dimensions with various tamper-resistant properties. A smartcard may store an encrypted digital certificate issued from the Public Key Infrastructure (PKI) along with any other relevant information about the card holder. A smartcard is a *declared identity* in Figure 1.2, identity information stored on a smartcard addresses a previously enrolled fingerprint image stored in the database.

Applying AFAS in conjunction with smartcards gives the highest security level, combining three known tactics of *what you have*, *what you know*, and *who you are*. AFAS pursues reliability in verifying person identity. AFAS usually requires information about card holder and physical fingerprint as its input. Smartcards, storing an encrypted digital signature issued from the PKI along with relevant information about the card holder, augment

reliability in identity recognition with AFAS. Thus combining traditional form of secure authentication (*what you have*) with relatively new form of biometric authentication (*who you are*), the system is claimed to be more secure. But even then an identity falsification, using artificial fingerprints as input to AFAS is possible—spoofing.

1.2 Related work

Various anti-spoofing methods were suggested and widely used in order to overcome the spoofing problem ([10], [11]). One of the techniques is fusing various biometric information sources for more secure authentication. For instance, Jain at al. used multiple biometrics features for more secured identification ([12], [13], [14]). Duc at al. ([15]) used the fusion of face and speech information. However, multibiometric systems involve additional cost and increase in the enrollment and verification times.

Another technique is compounding of several scanners or additional physical equipment supplements ([16], [17]). Yet anther one is multispectral imaging [9].

In response, several researches came up with methods of counterfeiting existing authentication systems ([6], [7], [18], [19], [8]).

Over time more and more sophisticated anti-spoofing methods were developed. They did improve the anti-spoofing capabilities of the biometric systems. However the computational requirements of state-of-the-art anti-spoofing methods are incompatible with existing embedded systems.

A relatively new approach to anti-spoofing problem is extracting geometric characteristics. It may lead time, space, and energy saving solution. Bulatov et al. [20] applied geometric classifiers in palm recognition research in 2004, and Varchol and Levicky [21] in 2007. The former reports FAR = 2%, FRR = 15%; the latter represents FAR = $0,1812\%$, FRR = $14,583\%$. Yet palm-based recognition and identication is uncommon due to high costs of hardware and insufficient accumulated experience.[1]

A robust method that can be used on existing embedded system is required.

1.3 The Proposed Approach

This paper focuses on solutions for fingerprint spoofing problem in smartcards authentication system. An Image Acquisition (Figure 1.2) is performed with

[1]Including legal issues: the validity of palm-based evidence yet to be prooved.

an optical scanner (for reviewing common fingerprint scanning technologies see [2], [7]). We present a new anti-spoofing approach that significantly increases the AFAS security. It is software based solution, and applicable to embedded systems, that involve smartcards.

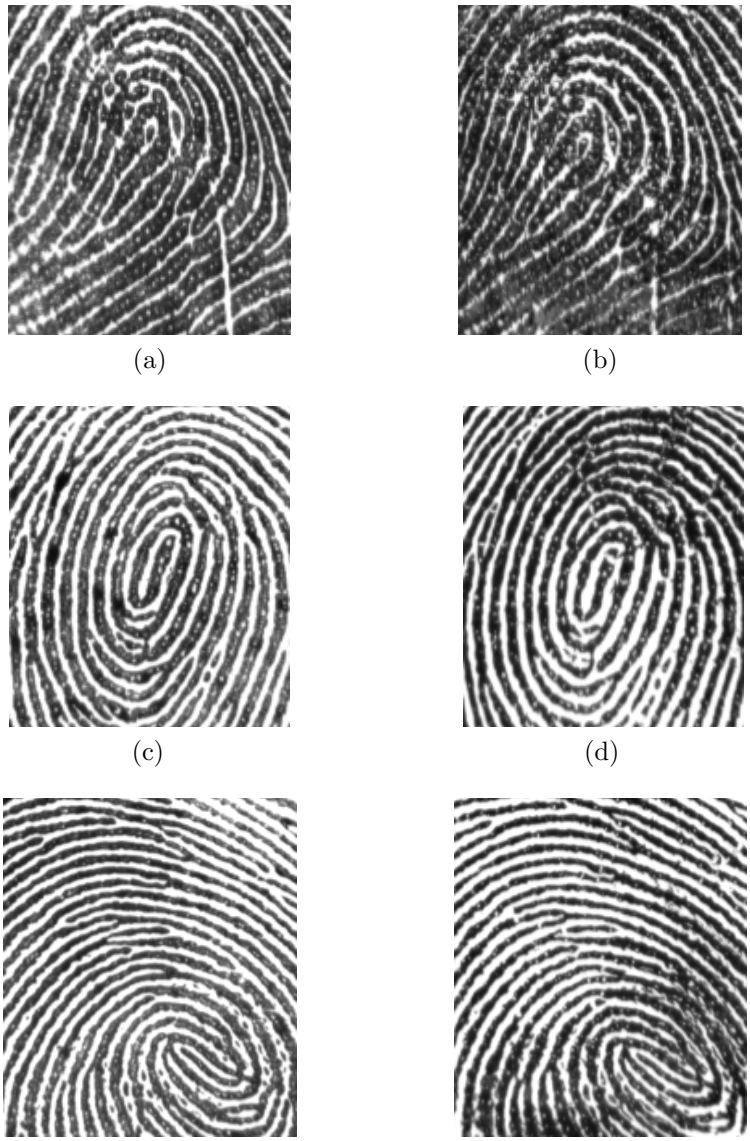
Typical AFAS consists of the following layers: *image acquisition*, *features (minutiae) extraction*, *matching*, and *decision for authentication*, as shown at Figure 1.2. We add a new layer in the initial phase of AFAS, just after an image acquisition. We propose two alternatives of this layer, both have extremely low storage requirements (about 100 bytes).

The first alternative is called an *anti-counterfeit layer* (ACL). ACL solves the following problem: given a fingerprint image, it classifies the image as a fake or as authentic (real). ACL is based on classification of the features extracted from the fingerprint images. The second alternative is called *smart anti-counterfeit layer* (Smart ACL). The Smart ACL solves the same problem that the ACL does. The main difference between ACL and Smart ACL is that in the latter, a classifier is trained on the *ratio* between the features instead of the features themselves (see Chapter 3 for details).

We also present a robust implementation of the ACL and Smart ACL. Our algorithms detect 3D fingerprints fakes, prepared using modified Matsumoto receipt (for detail see [6]). This kind of 3D spoof is known in the academy and in the industry as almost impossible to distinguish from the authentic fingerprint (for different spoofs comparing see [18]). The algorithm is easily expandable to other spoof types.

1.4 Paper outline

So far we have presented our approach as a modification to the currently adopted authentication scheme: AFAS. In Chapter 2 we define and describe features and classifiers. Chapter 3 deals with the difference between Smart ACL and ACL, and reviews the use of the identity information for better classification. In Chapter 4 we describe the database. In Chapters 5 and 6 we represent the results of the research, and in Chapter 7 we conclude the research and face open problems.



(a) (b) (c) (d) (e) (f)

Figure 1.1: Examples of fingerprints, used in our research: (a)(c)(e)—original fingerprints, (b)(d)(f)—counterfeited fingerprints. The gummy fingerprints are of a good quality: not only ridges (dark), valleys (light), and scratches are fully replicated, but even the sweat pores are well seen as white dots. Among the spoofing deficiencies: DC levels variation in different areas of a spoof, blunt edges, smoothing artifacts, valleys intensity spectrum difference.

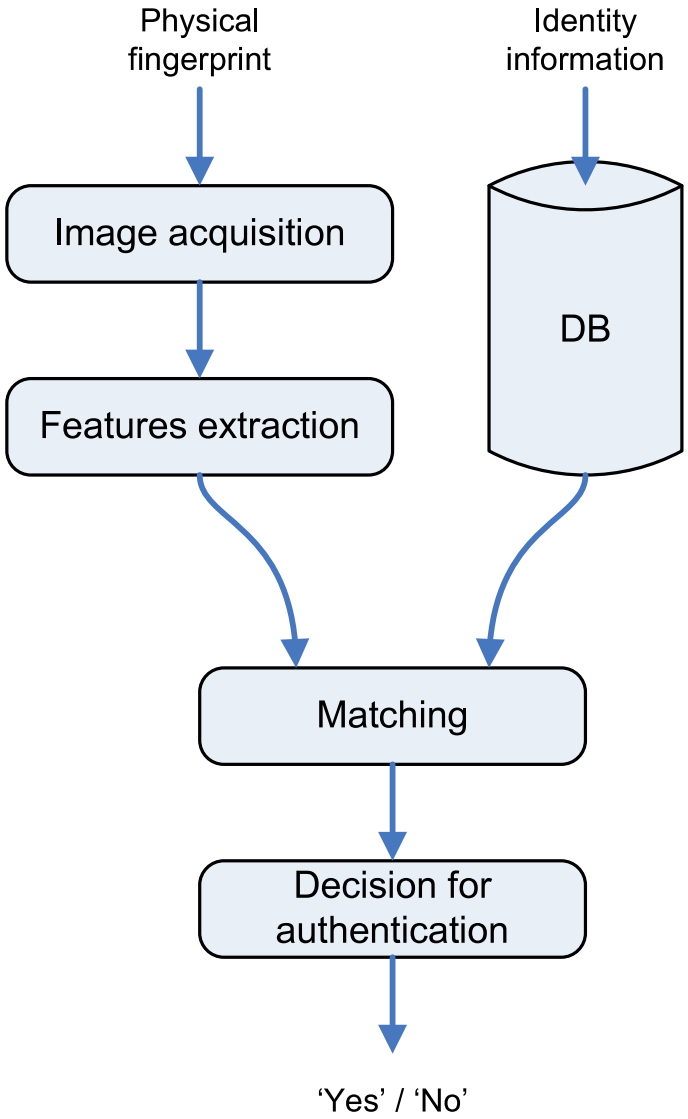


Figure 1.2: AFAS block diagram. Detailed description of each level may be found in [4]

Chapter 2

ACL model

2.1 Design

The *anti-counterfeit layer* (ACL) detects whether a fraudulent fingerprint is given as an input, without referencing the fingerprint owner identity. ACL is applied before *feature extraction layer* in AFAS, right after *image capturing layer* (see Figure 1.2). Detecting a fake fingerprint at such an early stage eliminates the need to access the database, and consequentially prevents a DOS attack.

Compared to AFAS, ACL does not attempt to verify whether an input fingerprint belongs to the declared identity. Compared to AFIS, ACL does not match an input fingerprint against a database of previously enrolled fingerprints. ACL's unique goal is to distinguish an original fingerprint from its spoof.

ACL is comprised of two steps:

1. features extraction,
2. classification.

Features are defined as follows.

Let I be an image of fingerprint, $I \in \Im$. Let's define a *characteristics extraction algorithm* (henceforth CEA), or *feature*, as a function $\text{CEA}(\Im) \Longrightarrow \Re$. Let's denote the number of CEAs in the ACL by n.

The calibration of CEAs is a part of the enrollment process[1]. of fingerprint authentication. In the enrollment step, human's fingerprint is preprocessed

[1]Enrollment process in AFAS is the process of registering subjects with their fingerprint template in the database. Positive and negative enrollments exist. The purpose of the positive enrollment is construction of a centralized or distributed database with eligible members. The purpose of the negative enrollment is construction of a centralized database of members ineligible for some application. For more information see [22]

with each of the involved CEAs. Each CEA yields a scalar v_i, which is aggregated into the resulting *training* vector, $V_{tr} = v_i|_{i=1}^{n}$. V_{tr} is used as an input to the high-level classifier.

During the verification step[2] , the fingerprint in question is preprocessed with the same set of CEAs. A resulting *verification* vector V_{ver}, $|V_{ver}| = n$, is classified. The final answer is *fake* or *real*.

The above definition of CEA, may be easily extended to a weak Bayesian classifier, namely $\overline{\text{CEA}}(\Im) \Longrightarrow \{\text{fake}, \text{real}\}$, by applying a threshold[3] on v_i. This was our initial implementation of the the classifiers, and it served as a basis for the algorithms comparison.

Comparison criteria for $\overline{\text{CEA}}_i$ are FRR and FAR, defined as follows:

$$\text{FRR} = p(S|\hat{G}) = \frac{p(\hat{G}|S)p(S)}{p(\hat{G}|S)p(S) + p(\hat{G}|G)p(G)}, \tag{2.1}$$

$$\text{FAR} = p(G|\hat{S}) = \frac{p(\hat{S}|G)p(G)}{p(\hat{S}|G)p(G) + p(\hat{S}|S)p(S)}, \tag{2.2}$$

where

$p(S)$ a priori probabilities of a real fingerprint in the training set[4],

$p(G)$ a priori probabilities of a fake fingerprint in the training set,

$p(\hat{S})$ a posteriori probabilities of a real fingerprint in the training set,

$p(\hat{G})$ a posteriori probabilities of a fake fingerprint in the training set.

The number of CEAs is not limited. Additional CEA, based on new spoof characteristics, may be easily appended, thus ensuring the ACL adaptability and expandability. Each CEA should answer the layer requirements.

[2]In the verification step the similarity between the enrolled data and the input data is examined. In general, there are three steps involved in the verification process, those are image preprocessing, minutiae extraction, and minutiae matching. ACL comes after the image preprocessing step, before minutiae extraction and matching, thus increasing security by preventing DOS attacks.

[3]The threshold is based on EER measure, which leads to a non robust system. However, $\overline{\text{CEA}}$ serves only as a starting operating point for further classification.

[4]Strictly speaking, $p(S)$ is an a priory probability of a real fingerprint in the *test set*. However, this information is not usually available in a causal system, and thus a priori *train* set probabilities are used, as in [23].

Main ACL requirements are:

1. low computational cost, which yields fast results,

2. storage effectiveness,

3. low equipment cost—CEA works with a single image from an optical scanner.

Currently, seven sets of CEAs have been implemented and tested on the database described in Chapter 4. Each CEA exploits a different artificial spoof characteristic, like smoothness of a spoof, variance of DC levels in the different areas of a spoof, valleys and ridges width ratio, pores frequency, and valleys intensity spectrum.

However, running a single $\overline{\text{CEA}}_i$ classifier may not yield satisfactory results for high-security systems. Being computationally and storage effective, $\overline{\text{CEA}}_i$ suffers from relatively high FRR and FAR values, which leads to overall low separability. As a measure to increase the former we denote a high level classifier, which effectively combines the $\overline{\text{CEA}}_i$ and produces a single output based on the results of CEAs. By CEAs combining, we expect and increase probability of the correct classification to the *fake* or *real* clusters.

We experimented with four classifiers: Support Vector Machine (SVM), k-NN learning (k-NN), Decision Tree learning, and Linear Discriminant Analysis (LDA). All supervised learning techniques, besides the k-NN, may be trained offline, thus the ACL assures the efficiency: the enrollment process remains fast; and a classifier's update may be done after enrolling any number of new users. Detailed analysis of classifiers error rates can be found in Chapter 5.

2.2 CEA Description

Most of the CEAs work on some part of an input image, thus boosting performance. Partial image is either a *patch*—2D image part, or a *grid*—a set of 1D lines extracted from the input image. There are several ways to create a grid: random rows and columns, random rows, random columns, random lines. All images were adjusted so that their size and DC level will not affect the results.

Algorithm 1 shows the CEA_1 outline. The algorithm objective is to evaluate the dissimilarities in the gray-levels of ridges and valleys in a real fingerprint compared to those of a spoof. The algorithm randomly chooses vertical

and horizontal lines from an image, yielding a grid. Then it sums the number of gradients of the neighbour pixels in each grid line that are above the empirical value based on training on the whole finger images database.

Algorithm 2 shows CEA_2. The algorithm partitions an image into overlapping square patches of a constant size with small joint bars. Then, for each patch, the difference between the value of a pixel with maximal gray-level in the patch and the pixel with minimal one is calculated, yielding vector V. The output is a mean of the vector V.

Algorithm 3 shows CEA_3. The algorithm evaluates the ratio between the valleys and ridges widths in an image. The valleys in the images of original fingerprints are perceived wider by a human eye. The pixel intensity above the threshold t represents a valley, the intensity below the threshold—a ridge. The output is the valleys pixel count ratio.

Algorithm 4 and Algorithm 5 show CEA_{4A} and CEA_{4B}. The algorithms intend to detect bright regions (*stains*) in a spoof image. They apply *reduce operator* of Gaussian pyramids[5] up to hierarchy level 4, yielding a small image with gray-levels corresponding to lower frequencies. CEA_{4A} outputs a standard deviation of the image. CEA_{4B} calculates a pixels neighborhood difference matrix and outputs its standard deviation.

Algorithm 6 shows CEA_5. The algorithm objective is detection of the valley width homogeneousness. Valleys of spoofs are less continuous then valleys of the real fingerprints: the discontinuities are artifacts originated in the process of spoof making. The algorithm transforms an image to a frequency space. The frequencies above f_1 are high enough to designate sharp transitions.

Algorithm 7 and Algorithm 9 show CEA_{7AA} and CEA_{7CA}. The algorithms focus on the following observation: gray-level distribution histograms may be *sliced* to several ranges, while with the real fingerprints some slices are more concentrated than the others. The reason is the difference in pressure of the real finger on the scanner platen compared to the artificial finger pressure. The pixels with the gray-level above a_p are dominant in the image. a_p differs for each subject in the database.

CEA_{7AA} is applied to the complete image while CEA_{7CA} uses similar technique but is applied to a rows subset of the image. Table 5.1 and Table 6.3 show that performance of these two $\overline{\text{CEA}}$ is similar, while the latter is less computationally intensive. The same is true for CEA_{7AB} (Algorithm 8) and CEA_{7CB} (Algorithm 10).

[5]Each iteration of *reduce operator* takes as its input an image of the size $[k, k]$, applies a Gaussian filter ω with the parameters n and σ, and reduces it to be of the size $\left[\frac{k}{2}, \frac{k}{2}\right]$. Both n and σ are the result of training on the whole finger images database.

Require: Image I_p of person p, constant C_p, factor f_1.

1: **let** $\mathrm{f}(x) = \begin{cases} 1 & \text{if } x > C_p f_1 \\ 0 & \text{otherwise} \end{cases}$

2: $G \Leftarrow$ random cell grid of I // *random horizontal and vertical lines set*

3: $s \Leftarrow |G|$

4: $w \Leftarrow \{-0.5, 0, 0.5\}$

5: $B \Leftarrow \bigcup_{i=1}^{s} \left| G_i * w \right|$ // *pixel gaps*

6: **return** $\sum_{x \in B} \mathrm{f}(x)$

Algorithm 1: CEA_1. See that, in line 6, we count the number of gaps that are greater than an empirical value based on training performed on the whole database.

Require: Image I.

1: $P \Leftarrow \mathcal{P}(I)$

2: $V \Leftarrow \{\max(P_i) - \min(P_i)\}_{i=0}^{|P|}$

3: **return** $\mathrm{mean}(V)$

Algorithm 2: CEA_2. In line 1 $\mathcal{P}$ is a *non-exclusive partitioning* of an image into overlapping square patches of a constant size with small joint bars. The functions min, max return the minimal and the maximal pixel value in the patch.

Require: Image I_p of person p of size $n \times k$, factor t.

1: $C \Leftarrow \bigcup_{1,1 \leq i,j \leq n,k} I_{i,j} - t \geq 0$

2: **return** $\dfrac{|C|}{|I|}$

Algorithm 3: CEA_3. $I_{i,j} - t$ is positive if the pixel belongs to a valley and negative for a ridge.

Require: Image I of size $n \times k$, filter ω.

$T \Leftarrow 4$
For $t = 1$ to T **do**
  For $i = 1$ to n **do**
    For $j = 1$ to k **do**
      $M_{i,j} \Leftarrow \sum_{m=1}^{5} \sum_{n=1}^{5} \omega_{m,n} I_{2i+m,2j+n}$
  $I \Leftarrow M$
return $\sigma(I)$

Algorithm 4: CEA_{4A}.

Require: Image I of size $n \times k$, filter ω.

$T \Leftarrow 4$
For $t = 1$ to T **do**
  For $i = 1$ to n **do**
    For $j = 1$ to k **do**
      $M_{i,j} \Leftarrow \sum_{m=1}^{5} \sum_{n=1}^{5} \omega_{m,n} I_{2i+m,2j+n}$
  $I \Leftarrow M$
$G_x = \dfrac{\partial}{dx} I$
$G_y = \dfrac{\partial}{dy} I$
return $\sigma(|G_x| + |G_y|)$

Algorithm 5: CEA_{4B}.

Require: Image I of size $n \times k$, factor f_1.

let $\mathrm{f}(x) = \begin{cases} 1 & \text{if } x > e^{f_1} \\ 0 & \text{otherwise} \end{cases}$
$M \Leftarrow \mathrm{fft}(I)$
return $\sum_{1,1 \le i,j \le n,k} f(M_{i,j})$

Algorithm 6: CEA_5.

Require: Image I_p of person p of size $n \times k$, factor a_p, factor f_1.

1: **let** $\mathrm{f}(x) = \begin{cases} 1 & \text{if } x > a_p f_1 \\ 0 & \text{otherwise} \end{cases}$

2: $M \Leftarrow \sum_{1,1 \leq i,j \leq n,k} \mathrm{f}(I_{i,j})$

3: **return** $\frac{M}{|I|}$

Algorithm 7: CEA_{7AA}.

Require: Image I_p of person p of size $n \times k$, factor f_1.

1: **let** $\mathrm{f}(x) = \begin{cases} 1 & \text{if } x > f_1 \cdot \max(I) \\ 0 & \text{otherwise} \end{cases}$

2: $M \Leftarrow \sum_{1,1 \leq i,j \leq n,k} \mathrm{f}(I_{i,j})$

3: **return** $\frac{M}{|I|}$

Algorithm 8: CEA_{7AB}.

Require: Image I_p of person p of size $n \times k$, factor a_p, factor f_1, factor r.

1: **let** $\mathrm{f}(x) = \begin{cases} 1 & \text{if } x > a_p f_1 \\ 0 & \text{otherwise} \end{cases}$

2: $S \Leftarrow$ subset of r random rows of I

3: $M \Leftarrow \sum_{1,1 \leq i,j \leq n,k} \mathrm{f}(S_{i,j})$

4: **return** $\frac{M}{|I|}$

Algorithm 9: CEA_{7CA}.

Require: Image I_p of person p of size $n \times k$, factor f_1, factor r.

1: **let** $\mathrm{f}(x) = \begin{cases} 1 & \text{if } x > f_1 \cdot \max(I) \\ 0 & \text{otherwise} \end{cases}$

2: $S \Leftarrow$ subset of r random rows of I

3: $M \Leftarrow \sum_{1,1 \leq i,j \leq n,k} \mathrm{f}(S_{i,j})$

4: **return** $\frac{M}{|I|}$

Algorithm 10: CEA_{7CB}.

2.3 Classifiers Description

SVM

A Support Vector Machine (SVM) is a set of supervised learning methods, used for the classification. SVM performs classification by constructing an n-dimensional hyperplane that optimally separates the data into two categories: *fake* or *real*. Viewing vectors V_{ver} of the training set[6] as two clusters of vectors in an n-dimensional space, SVM constructs a separating hyperplane in that space that maximizes the *margin* between the two data sets. To calculate the margin, two parallel hyperplanes are constructed, one on each side of the separating hyperplane in such a way that maximizes the distance between the data sets. We use a 10-fold cross validation in order to overcome the possible over-fitting problem (see Chapter 5). In our case the 10-fold cross-validation looses only 10% of the data, but is about 80 times more efficient than leave-one-out cross-validation (LOOCV). We conclude that our statistical model does not suffer from the over-fitting problem.

k-NN

The k-nearest neighbors algorithm (k-NN) is a method for classification objects based on closest training examples in the feature space. k-NN is a type of lazy learning where the function is only approximated locally and all the computation is deferred until the classification. The k-nearest neighbors algorithm is amongst the simplest of all machine learning algorithms. A fingerprint is classified by a majority vote of its neighbors in CEA space, with the object being assigned to the most common class amongs its k nearest neighbors. We choose k to be an odd number, thus avoiding tied votes. k-NN classifier is usually used as one of the heuristics of feature extraction in pattern analysis. To utilize the k-NN computational efficiency we apply it as a high level classifier and compare the error rates of low computationally intensive high level classifiers with the high computationally intensive ones.

Classification Tree

Classification trees are used to predict membership of objects in the classes of a categorical dependent variable from their measurements on one or more predictor variables. Classification tree analysis is one of the main techniques used in data mining. The goal of our classification trees is to predict responses on a categorical dependent variable $\{real, fake\}$. As such, the technique have

[6]See Section 2.1 for V_{ver} and n definitions

much in common with the traditional methods of Discriminant Analysis. In Chapter 5 we compare the results of a classification tree with that of linear discriminant analysis. The flexibility of classification trees makes them an attractive analysis alternative in cases where theoretical and distributional assumptions of the traditional methods are not met, and this typically is the case with fingerprints. But as an exploratory technique, or as a technique of last resort when traditional methods fail, classification trees are, in the opinion of many researchers, unsurpassed [24].

Discriminant Analysis

One of the simplest techniques to solve our problem is Linear Discriminant Analysis (LDA). The problem is defined as follows: find an hyper-plane in the hyper-space of V_{ver}s, where projection of the train data is well-separated into two clusters [23]. Despite the fact that LDA is mostly used as a remedy for curse dimensionality, decreasing the features set for the classification technique, we use it as an independent classification technique. It is simple and less computationally intensive comparing to SVM and classification tree. We show in Chapter 5 that LDA results in our case are comparable to those of SVM, and classification tree.

Chapter 3

Smart ACL model

3.1 Design

Compared to ACL, Smart ACL solves a different problem: given a fingerprint image and *the identity of the fingerprint owner*, the layer's output is an indication whether the input image is a spoof. Informally, Smart ACL uses the previously extracted characteristics with relevant identity information for the verification purposes.

Smart ACL's design is similar to the ACL's. Yet, Smart ACL differs from the ACL by using the declared identity of the fingerprint owner. Smart ACL improves the correctly classified *fakes* rate by training the classifier on the ratio of CEA output for *real* and *fake* fingerprints of all enrolled identities.

In the enrollment step, human's fingerprint is preprocessed with each of the involved CEAs. Each CEA yields a scalar v_i, which is aggregated into the resulting vector $V_{tr} = v_i|_{i=1}^{n}$. V_{tr} is saved in database (or on a smartcard).

During the verification step, the fingerprint in question is preprocessed with the same set of CEAs. Lets denote a result vector V_{ver}, $|V_{ver}| = n$. The classification is performed on the vector

$$\left\{ \frac{V_{veri}}{V_{tri}} \right\}_{i=1}^{n}.$$

3.2 ACL vs. Smart ACL

ACL defines a *single* cut-off training vector for all the enrolled persons, while the Smart ACL defines one such vector *for each* enrolled person. Therefore the resolution of Smart ACL is expected to be higher. Contrary to other anti-counterfeit applications, in our case we *do know* person's identity at the verification step.

Chapter 4

Experiment Scope

The database consist of heterogeneous population, including pianists and those occupied in manual labour, young and elderly people. It includes 784 images of 24 people, ages 18 to 85, male and female. 11 people are above 40 years old; 14 people are occupied in manual labour.

Each fingerprint was captured by the L SCAN 100 CrossMatch optical scanner. The scanner dynamic range is 8 bits, 256 gray scales maximum and the scanning resolution is 500 dpi. The scanner contains an optical scattering sensor: only the light scattered by the contacting finger ridges is received by the camera while all the other light is absorbed by passing through the glass surface. The scanner complies to international AFIS standard—the FBI's Electronic Fingerprint Transmission Specification 7.1. The German Federal Office for Information Security (BSI) has certified this scanner for its outstanding image quality. Large scale voter registration programs in Africa used this scanner; 5,000 units were installed in the e-passport enrollment system in Germany. Figure 1.1 shows some examples of scanned fingerprints.

The images of the fingerprints were acquired using a specially written application, *CaptureEssentials*. The application is written mainly in Visual C++, using the scanner's dynamically-linked libraries. The application has the capabilities of choosing best acquisition parameters, like contrast, and provides automatic means of acquisition multiple images from a single person. More details are provided in the Appendix A.

Each spoof was prepared by Matsumoto et al. concept spoof [6]. Figure 1.1 shows the examples of the spoofs we have prepared. Artificial fingers have been created from the casts using gelatin, commonly used for confectionery. Matsumoto et al. termed these artificial fingers as *gummy*. Appendix B provides more details on the modeling technique. All commercial AFAS systems tested in [6] enrolled a gummy fingerprint. FAR of verification gummy fingerprints in their tests ranged from 68%–100% (for detailed

description see [6] and [18]). As widely recognised, the gummy spoofs are the most difficult to detect, thus many researches use them in their tests (For details see [25], [26]).

The other advantages of using Matsumoto et al. gummy spoofs in our tests are their availability and reduced as well as possession of less amount of artifacts than in other spoofing methods. The process of preparation of gummy spoofs is simple enough to be performed at home. However it takes over 2 hours to prepare one spoof. Comparing to other spoofs, Matsumoto gummies do not suffer from aliasing (especially compared with gummies in the report [8]).

Chapter 5

ACL Results

5.1 Weak Bayesian classifiers

Table 5.1 shows the result of classification of the raw CEAs. The error rates in the table may be used as a reference for further comparison to the high level classification techniques results. We see that the only requirement for CEA is that it can classify with FRR and FAR rates below 30%. CEA_5 used in our experiments as a noisy heuristic, in order to check the stability of the different high level classification methods.

5.2 SVM

The best result has been achieved with 5 independent CEAs. Correctness rate is 0.9330. Error rate is 0.0670. Sensitivity is 0.9072. Specificity is 0.9588. We see a definite improvement compared to a weak Bayesian classifiers.

We have tried several different kernels: linear, quadratic, polynomial with different degrees. The recommended kernel function is the Radial Basis Function (RBF) with $\sigma = 1$. The only method used to find a separating plane was the Least Squares method.

It should be noted that we put a special emphasis on the higher specificity, as this is a basic requirement for the secured anti spoofing algorithms. Relatively low sensitivity may lead to rejects of authentic fingerprints, but usually this acceptable in high security applications.

$\overline{\text{CEA}_i}$	FRR (%)	FAR (%)
Alg1	20.71	24.74
Alg2	28.28	24.74
Alg3	24.75	20.62
Alg4A	22.22	17.01
Alg4B	15.66	11.34
Alg5	33.33	36.60
Alg7AA	38.89	36.08
Alg7AB	30.30	26.80
Alg7CA	27.27	19.59
Alg7CA	29.80	17.00

Table 5.1: $\overline{\text{CEA}}$ Weak Bayesian classifiers, if used alone. As we can see most algorithms generate unacceptably high FRR and FAR rates. Algorithm 4B gives the best results.

5.3 k-NN

Table 5.3 represents results of applying k-NN method with different metrics and k parameter. It can be seen that best results have been achieved for $k = 5$ (while bigger k is not practical in embedded system), and the metrics do not have a significant effect on the results. Thus we suggest using the *Manhattan* metric, as computationally simple. Applying k-NN with noisy CEA, like Algorithm 5, dramatically increases the error rates.

k-NN is less reliable than SVM, as may be concluded from the observation of sensitivity and specificity of the both methods. k-NN is less computationally intensive than SVM, but in SVM the computational load is split into two phases: training and testing. The former may be done on server and the latter on the embedded device. Yet in k-NN all the computation is done during the testing phase. Overall, SVM seems to give better results, but is harder to implement in embedded systems.

5.4 Decision Tree

The recommended criterion for choosing a split is Gini's diversity index. As we can see, classification tree manages the noisy CEA, like CEA_5, better than other classifiers. Additionally, the decision tree overcomes the curse dimensionality problem, by positioning the best heuristics in the upper decision nodes, and pruning. The decision tree method produces an error rate that

Statistical measures	LDA	SVM
Correct rate	0.8814	0.9330
Error rate	0.1186	0.0670
Sensitivity	0.9072	0.9072
Specificity	0.8557	0.9588
PositivePredictiveValue	0.8627	0.9565
NegativePredictiveValue	0.9022	0.9118

Table 5.2: Statistical measures of LDA and SVM classifiers in ACL. SVM has a notably better specificity, however LDA is much simpler to implement.

is comparable to other classifiers: 0.06505. In Figure 5.1 we see the example of a decision tree, built with all available CEAs. CEAs with the lowest error rates placed in the upper nodes of the decision tree, as expected (For CEAs error rates see Table 5.1).

The decision tree may be pruned to its best pruning level in order to reduce the computational intensity during the authentication phase. It can handle the trade off between computational efficiency and reliability on-the-fly. In Figure 5.2 we see that only 8 decision nodes required for a minimum error rate tree, and the minimum error rate is 0.935. We also see, that the solid line does not cross the minimum error threshold as the number of leaves increases. Therefore the tree converges.

The decision tree is simpler to understand and interpret, than SVM. It performs well with large data in a short time. Yet, the decision tree can become overly complex, and it has low stability: alteration of even a single training point may lead to a complete change of the tree layout.

5.5 Linear Discriminant Analysis

The best result is achieved with 5 independent CEAs. Correct rate is 0.8814. Error rate is 0.1186. Sensitivity is 0.9072. Specificity is 0.8557. We see a definite improvement compared to the weak Bayesian classifiers (Table 5.1). The observed results are comparable with those of SVM and other classifiers. [1]

$$C(I) = \begin{cases} 1 & \text{if } I \in \{\text{False Positive, False Negative}\} \\ 0 & \text{if } I \in \{\text{True Positive, True Negative}\} \end{cases} \tag{5.1}$$

[1]From the difference in statistical measures in Table 5.2 follows, that this is not the Occam's razor principle case (for details see [27]).

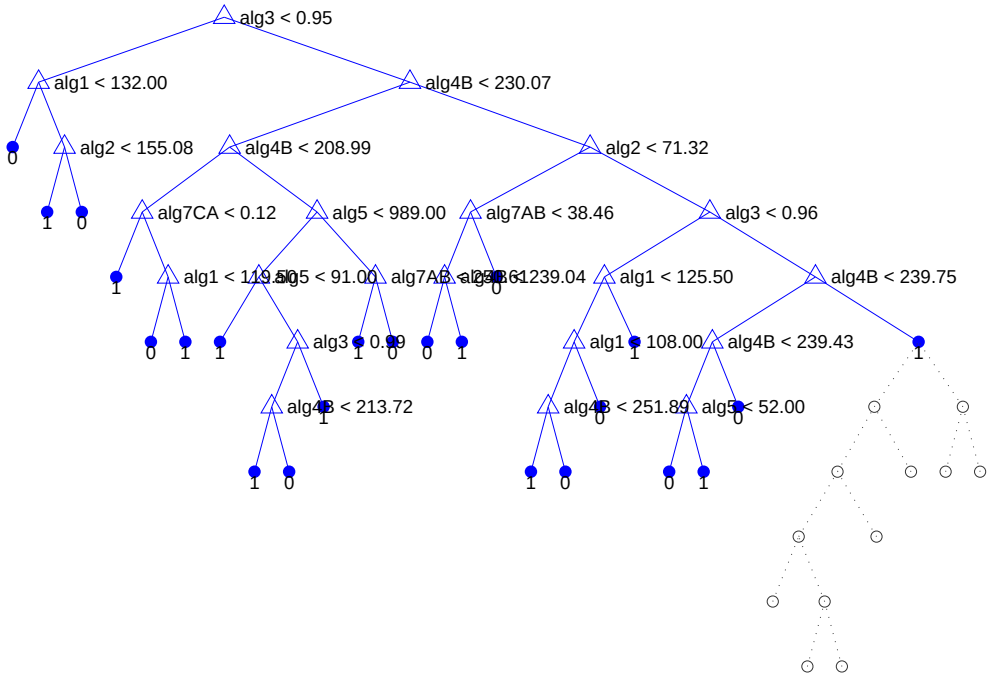


Figure 5.1: Decision tree. Part of shown the tree is pruned. Note Algorithm 5 is placed close to leaves, since it is a noisy classifier.

		k		
Metrics	Analysis	1	3	5
Eucleadian	sensitivity	0.7320	0.7835	0.8144
	specificity	0.9220	0.9356	0.9424
Manhatten	sensitivity	0.7577	0.7732	0.7835
	specificity	0.9305	0.9441	0.9492

Table 5.3: Comparison of sensitivity and specificity in k-NN classifier for various k-NN parameters.

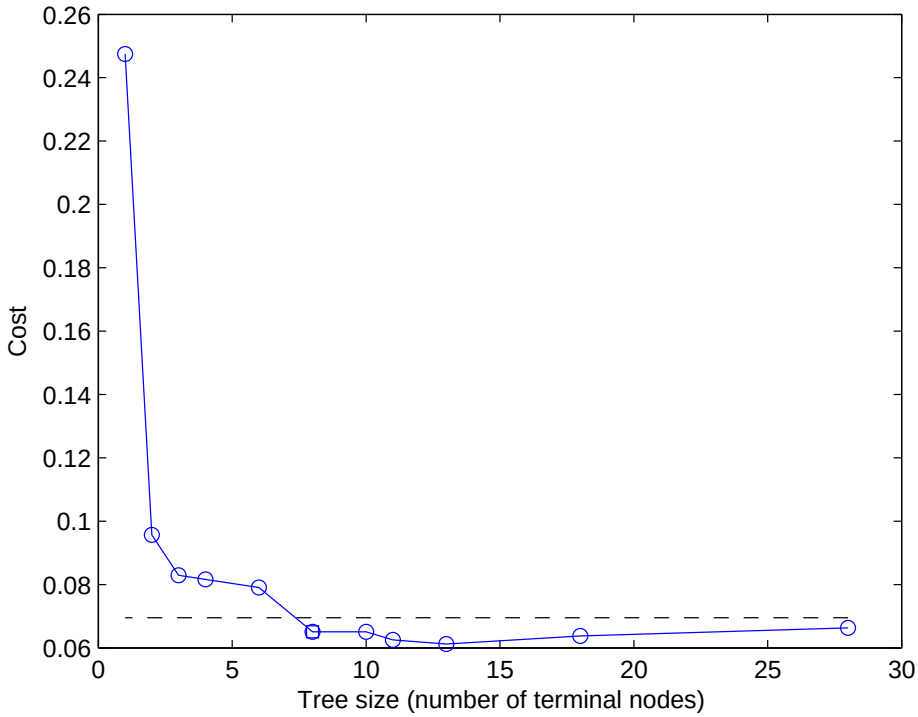


Figure 5.2: Decision tree error rate graph. The solid line shows the estimated cost for each tree size, the dashed line marks one standard error above the minimum error rate, and the square marks the smallest tree under the dashed line. The solid line is below the dashed line, proving the tree convergence. The minimal error rate is 0.06505, and the best pruning level is 3.

Chapter 6

Smart ACL Results

6.1 Weak Bayesian classifiers

Figure 6.1 and Figure 6.2 show $\overline{\text{CEA}}$s—weak Bayesian classifiers. Noteworthy to remind, that $\overline{\text{CEA}}$s used solely as heuristics for higher level classification. The plots show the distribution functions of human fingerprints (blue line) and fake fingerprints (red line). The green line is the classification threshold of the weak Bayesian classifiers.

Table 6.3 shows $\overline{\text{CEA}}$s error rates. Comparing results in Table 6.3 and results shown in Table 5.1 reveals that Smart ACL weak Bayesian classifiers are better than ACL weak Bayesian classifiers.

6.2 SVM

The best result is achieved with 5 independent CEAs. Correctness rate is 0.9439. The recommended kernel function is a quadratic or RBF with $\sigma = 1$. We see a definite improvement compared to the Smart ACL weak Bayesian classifiers (Tables 6.1 and 6.3). Table 6.1 shows an improvement in sensitivity and specificity measures compared to those of SVM-based ACL. Remembering that (FAR $= 1 -$ Specificity), for Smart ACL with SVM classifier we get FAR $= 0.0101$, and for ACL we achieve FAR $= 0.0412$. Hence, Smart ACL reduces the FAR by factor 4.

6.3 k-NN

As we saw in Section 5.3, best results for k-NN-based ACL are FRR $= 0.1856$ and FAR $= 0.0576$ using Eucleadean metrics and $k = 5$. With Smart ACL

Statistical measures	ACL SVM	Smart ACL SVM
Correct rate	0.9330	0.9439
Error rate	0.0670	0.0561
Sensitivity	0.9072	0.8969
Specificity	0.9588	0.9899
PositivePredictiveValue	0.9565	0.9886
NegativePredictiveValue	0.9118	0.9074

Table 6.1: Statistical measures of SVM classifier in ACL and Smart ACL

we get significantly better FRR, with little compromising of FAR: FRR = 0.1340, FAR = 0.0606 for *Manhatten* metrics with $k = 3$. k-NN is the only Smart ACL classifier that improves FRR w.r.t. ACL.

6.4 Decision Tree

The Decision Tree with Smart ACL gives similar results as ACL (see Section 5.4).

6.5 Linear Discriminant Analysis

The best result is achieved with 5 independent CEAs. Correctness rate is 0.9031. Error rate is 0.0969. Sensitivity is 0.8454. Specificity is 0.9596. We see a definite improvement comparing to the Smart ACL weak Bayesian classifiers (Table 6.3).

Table 6.2 shows the difference between statistical measures of LDA and SVM. LDA error rates are higher than in SVM. However, LDA is less computationally intensive than SVM. Hence the trade off between computational efficiency and reliability.

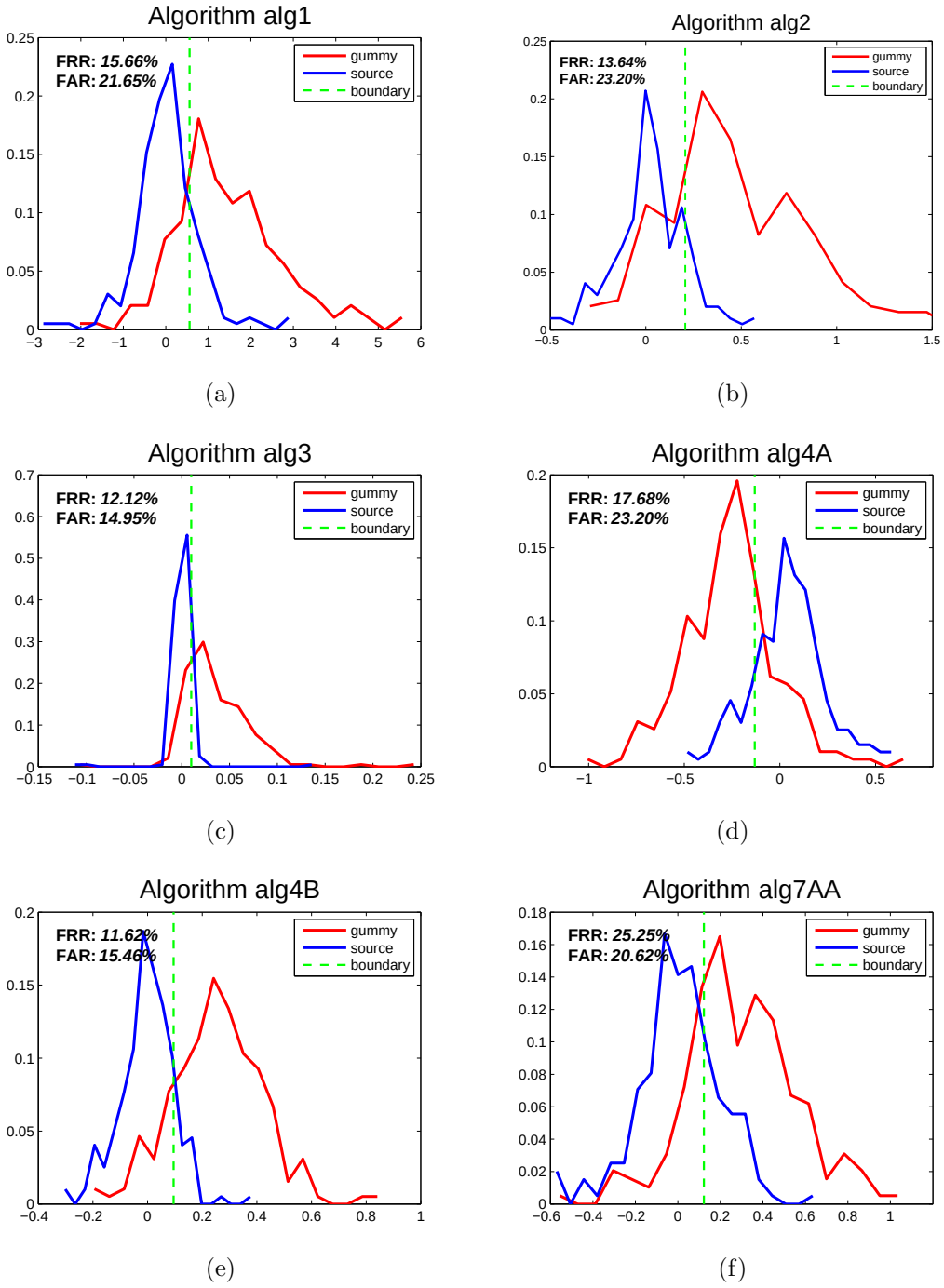


Figure 6.1: CEA algorithms performance. Blue line is the original fingerprints, red line—counterfeited ones. Green dashed line is the splitting hyperplane of the corresponding $\overline{\text{CEA}}$. For all the algorithms except algorithm 4a the points to the right of the green dashed line are treated as gummy, to the left—original fingerprints. Algorithm 4b is inverted.

Statistical measures	LDA	SVM
Correct rate	0.9031	0.9439
Error rate	0.0969	0.0561
Sensitivity	0.8454	0.8969
Specificity	0.9596	0.9899
PositivePredictiveValue	0.9535	0.9886
NegativePredictiveValue	0.8636	0.9074

Table 6.2: Comparison of LDA and SVM classifiers in Smart ACL. SVM has better sensitivity and specificity, therefore reducing LDA's error rate by almost factor of 1.7.

$\overline{\text{CEA}}_i$	FRR (%)	FAR (%)
Alg1	17.68	19.07
Alg2	13.64	23.20
Alg3	12.12	14.95
Alg4A	17.68	23.20
Alg4B	11.62	15.46
Alg5	34.34	21.13
Alg7AA	25.25	20.62
Alg7AB	27.27	16.49
Alg7CA	26.77	19.07
Alg7CB	23.74	23.20

Table 6.3: Error rates of $\overline{\text{CEA}}$s in Smart ACL

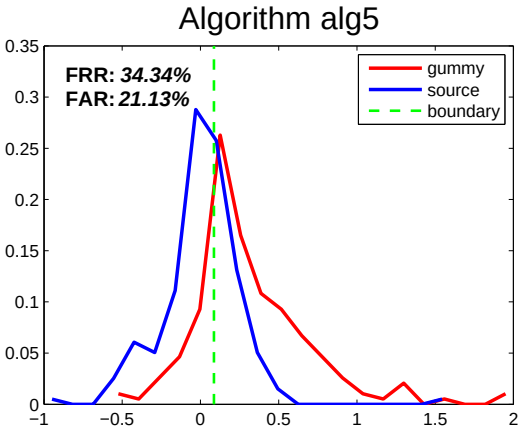


Figure 6.2: CEA_5. This is an example of *bad* $\overline{\text{CEA}}$. FRR is unacceptably high (34%), and separability is low. Using this CEA with SVM technique leads to the curse dimensionality problem, and thus downgrading the whole classification system. However, a hierarchical clustering tree effectively drops this CEA by putting it at the bottom of the classification tree (at leaves).

Chapter 7

Conclusion

In this paper a new, anti-counterfeiting layer in AFAS is defined. We present two different implementations of the layer, ACL and Smart ACL. Both solutions bear the following advantages:

Expandability—new features may be easily added;

Extensibility—other spoofing methods may be detected;

Embeddability—algorithms are mostly time- and space-efficient, and thus suitable for embedded applications;

Reliability—most classifiers yield high specificity.

Simulation and results were presented for a database of real and counterfeited fingerprints. Comparative analysis between the different techniques reveals the following findings:

- Increased number of features leads to curse of dimensionality. If new features are added to the system we recommend to apply LDA before the main classifier in order to reduce the dimensionality.
- There are various classifiers with different detection capabilities, that may be chosen for each specific application.
- Most classifiers enable fine-tuning with the trade-off between sensitivity and specificity, although high specificity with reasonable sensitivity seems to be the most suitable fingerprint identification systems.
- k-NN was used as a classifier, however it may instead be used as a feature (CEA), as it is computationally effective.

While we have demonstrated the efficiency of our method, we still plan to improve it by including additional information that reflects the best classification method for each person, as opposed to the regular averaging approach. Preliminary results we have suggested that the performance of the system could indeed be substantially increased.

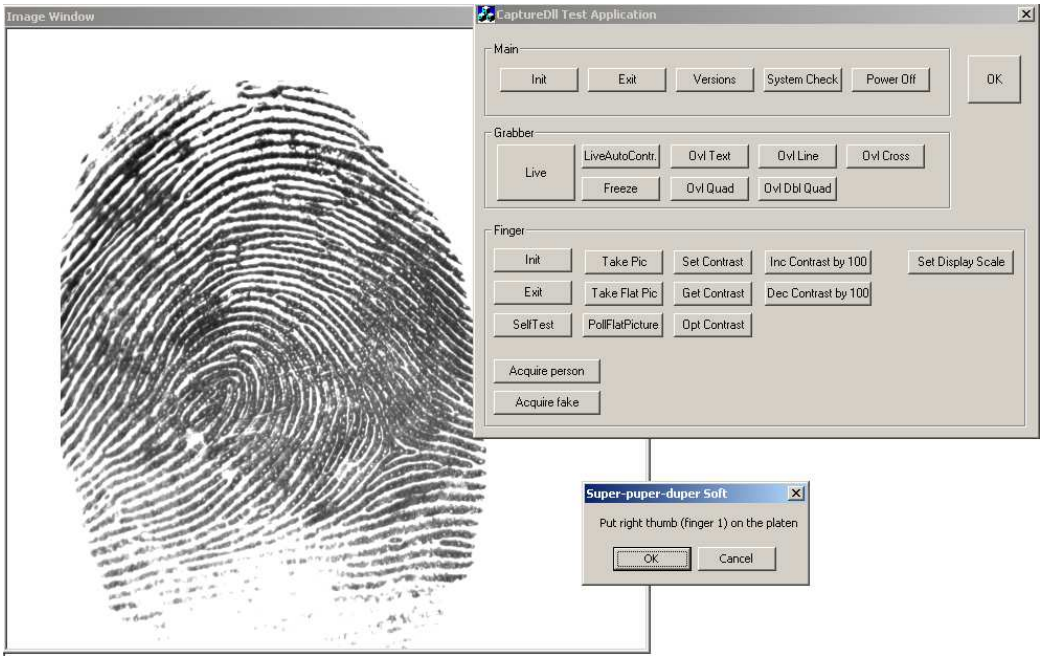


Figure A.1: CaptureEssentials—screenshot of a fingerprint acquisition

Appendix A

CaptureEssentials

The program listed below aims to acquire fingerprint images using the L SCAN 100 CrossMatch optical scanner. Figure A.1 shows the screenshot of common usecase—acquisition of a finger.

```
void CTestCaptureDllDlg::OnAcqPerson()
{
  PersonParams pp;
  char *sides[] = {"right", "left"};
  char *fingernames[] = {"thumb", "forefinger", "middle",
```

```
                        "ring", "minimus"};
  if (pp.DoModal() == IDOK) // The user selected OK button, not cancel
  {
    for (int side = 0; side < 2; side++) {
      for (int finger = 0; finger < 5; finger++) {
        int res = GetOneFinger(fingernames[finger], finger+1,sides[side],
                  pp.m_from, pp.m_step, pp.m_to, (const char*)pp.m_dir,
                  (const char*)pp.m_name);
        if (res != 0)
          goto stop;
      }
    }
  }
stop:
  ;
}

int CTestCaptureDllDlg::GetOneFinger(char *fingername, int finger_num,
                    char *side, int from, int step, int to,
                    const char *dir, const char *name, int trial) {
  char buf[1000] = "";
  char trial_buf[1000] = "";
  char filename[1000];
  char dir_tocreate[1000] = "";
  char trial_filename_buf[1000] = "";
  if (trial != -1) {
    sprintf(trial_filename_buf, "_%02d", trial);
    sprintf(trial_buf, " - trial %d", trial);
  }
  sprintf(buf, "Put %s %s (finger %d) on the platen%s",
      side, fingername, finger_num, trial_buf);
  sprintf(dir_tocreate, "%s\\%s", dir, name);
  CreateDirectory(dir_tocreate, NULL);
  if (MessageBox(buf,"Super-puper-duper Soft", MB_OKCANCEL) == IDOK) {
    for (int contr = from; contr <= to; contr += step) {
      ILM_FINGER_setContrast(contr);
      sprintf(buf, "Don't move finger! Contrast=%d", contr);
      GrabberOvltext(1, 0, buf, 20, false, true);
      sprintf(filename, "%s\\%s_%c_%d_%04d%s.bmp", dir_tocreate, name,
          toupper(side[0]), toupper(finger_num), contr,
          trial_filename_buf);
      FingerTakeFlatpic(filename, true);
      ILM_GRABBER_live( m_ImgWnd.m_hWnd, DisplayImage( true ));
    }
    sprintf(buf, "Don't move finger! Contrast=OPTM");
    GrabberOvltext(1, 0, buf, 20, false, true);
    ILM_FINGER_optimizeContrast( OPT_CONTRAST );
    sprintf(filename, "%s\\%s_%c_%d_%4s%s.bmp", dir_tocreate, name,
        toupper(side[0]), finger_num, "OPTM", trial_filename_buf);
```

```
    FingerTakeFlatpic(filename, true);
    ILM_GRABBER_live( m_ImgWnd.m_hWnd, DisplayImage( true ));
    GrabberOvltext(400, 1, buf, 20, false, false);
  }
  else
    return -1;
  return 0;
}


void CTestCaptureDllDlg::OnAcqFake()
{
  FakeParams fp;
  char *sides[] = {"right", "left"};
  char *fingernames[] = {"thumb", "forefinger", "middle",
                         "ring", "minimus"};
  if (fp.DoModal() == IDOK)  // The user selected OK button, not cancel
  {
    for (int trial = 0; trial < fp.m_numoftrials; trial++) {
      int side = fp.m_hand;
      int finger = fp.m_finger;
          int res = GetOneFinger(fingernames[finger], finger+1,
                    sides[side], fp.m_from, fp.m_step, fp.m_to,
                    (const char*)fp.m_dir, (const char*)fp.m_name,
                    trial);
          if (res != 0)
            goto stop;
    }
  }
stop:
  ;
}
```

Appendix B

Gummy Preparation

As were mentioned in Chapter 4 we use the Matsumoto at al. concept spoof preparing [6]. We adjusted the materials for molds and artificial fingers so that the materials can be accessed in Israel. The material used for molds is a "Morph-plast"[1]; the material used for artificial fingers is gelatine powder made from fish bones[2]. The main difficulty in artificial fingers preparing is bubbling of the gelatine. Matsumoto at al. recommend repeatedly heating and cooling the gelatine by the microwave oven. We observed that microwave heating increases the bubbles in the fish-bones gelatine. The solution is repeatedly heating and cooling of the gelatine with bain-marie on the gas stove. The other difficulty is finding the proper temperature for the warming gelatine. The gelatine has to be a liquid, but not too warm as for melting the plastic mold.

[1] may be ordered at eBay for £2

[2] may be bought in most grocery stores in Israel

Bibliography

[1] R. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.

[2] M. Kákona. Drawbacks of biometric methods. Technical report, ICZ, 2001. Available online at `http://home.i.cz/kakl/biometrics/DrawbacksofBiometricMethods.htm`.

[3] H.C. Lee and R.E. Gaensslen. *Advances in Fingerprint Technology*. Elsevier, New York, 1991.

[4] S. B. Lee and S. Tsutsui. *Intelligent biometric techniques in fingerprint and face recognition*. CRC Press, Inc., Boca Raton, FL, USA, 1999.

[5] S. Jain. Digital watermarking techniques: A case study in fingerprints and faces. In *Proceedings of Indian conference on computer vision, graphics, and image processing*, pages 139–144, 2000.

[6] T. Matsumoto, H. Matsumoto, K. Yamada, and S. Hoshino. Impact of artificial gummy fingers on fingerprint systems. In *Proceedings of SPIE: Optical Security and Counterfeit Deterrence Techniques IV, 2002*, volume 4677, pub-SPIE:adr, 2002. pub-SPIE.

[7] Ton van der Putte and Jeroen Keuning. Biometrical fingerprint recognition: don't get your fingers burned. In *Proceedings of the fourth working conference on smart card research and advanced applications on Smart card research and advanced applications*, pages 289–303, Norwell, MA, USA, 2001. Kluwer Academic Publishers.

[8] starbug. How to fake fingerprints? Technical report, Chaos Computer Club e.V, October 2004. Available online at `http://www.ccc.de/biometrie/fingerabdruck_kopieren.xml?language=en`.

[9] K. A. Nixon, R. K. Rowe, M. S. Ennis, and S. P. Corcoran. Spoof detection using multispectral fingerprint imaging without enrollment. In *Proceedings of Biometrics Symposium (BSYM2005)*, 2005.

[10] David Zhang and Anil K. Jain, editors. *Biometric Authentication, First International Conference, ICBA 2004, Hong Kong, China, July 15-17, 2004, Proceedings*, volume 3072 of *Lecture Notes in Computer Science*. Springer, 2004.

[11] David Zhang and Anil K. Jain, editors. *Advances in Biometrics, International Conference, ICB 2006, Hong Kong, China, January 5-7, 2006, Proceedings*, volume 3832 of *Lecture Notes in Computer Science*. Springer, 2006.

[12] Arun Ross, Anil K. Jain, and Jian Z. Qian. Information fusion in biometrics. *Lecture Notes in Computer Science (A longer version appears in Pattern Recognition Letters, Vol. 24, Issue 13, pp. 2115-2125, September, 2003.*, 2091:354–359, 2001.

[13] Lin Hong and Anil Jain. Integrating faces and fingerprints for personal identification. *IEEE transactions on pattern analysis and machine intelligence*, 20:1295–1307, 1998.

[14] Anil K. Jain, Lin Hong, and Yatin Kulkarni. A multimodal biometric system using fingerprint, face, and speech. Technical Report MSU-CPS-98-32, Department of Computer Science, Michigan State University, East Lansing, Michigan, November 1998. Posted also at AVBPA99.

[15] Benoit Duc, Elizabeth Saers Bigiin, C Josef Bigiin D, Gilbert Maitre, and Stefan Fischer. Fusion of audio and video information for multi modal person authentication. *Pattern Recognition Letters*, 18:835–843, 1997.

[16] K. K. Veeramachaneni, L. A. Osadciw, and P. K. Varshney. Adaptive multimodal biometric fusion algorithm using particle swarm. In B. V. Dasarathy, editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5099 of *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, pages 211–221, April 2003.

[17] G. Parziale and E. Diaz-Santana. The surround imager$^{\mathrm{TM}}$: a multi-camera touchless device to acquire 3d rolled-equivalent fingerprints. In *Proceedings of International Conference on Biometrics*, pages 244–250, 2006.

[18] S.A.C. Schuckers. Spoofing and anti-spoofing measures. *Information Security Technical Report*, 7(4):56–62, 2002.

[19] L. Thalheim and J. Krissler. Body check: Biometric access protection devices and their programs put to the test. In *in c't Magazin für Computer und Technik*, page 114, 2002.

[20] Yaroslav Bulatov, Sachin Jambawalikar, Piyush Kumar, and Saurabh Sethia. Proc. 1st internat. conf. on biometric authentication (icba). In Zhang and Jain [10], pages 753–759.

[21] P. Varchol and D. Levický. Using of hand geometry in biometric security systems. *Radioengineering*, 16(4):82–87, 2007.

[22] Ruud M. Bolle, Jonathan H. Connell, Sharath Pankanti, Nalini K. Ratha, and Andrew W. Senior. *Guide to Biometrics*. SpringerVerlag, 2003.

[23] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. John Willey & Sons, New Yotk, 1973.

[24] Thomas Hill and Pawel Lewicki. *Statistics: Methods and Applications : a Comprehensive Reference for Science, Industry, and Data Mining*. StatSoft, Inc., 2006.

[25] Wei-Yun Yau, Hoang-Thanh Tran, Eam Khwang Teoh, and Jian-Gang Wang. Fake finger detection by finger color change analysis. In Seong-Whan Lee and Stan Z. Li, editors, *ICB*, volume 4642 of *Lecture Notes in Computer Science*, pages 888–896. Springer, 2007.

[26] O.G. Martinsen, S. Clausen, J.B. Nysaether, and S. Grimnes. Utilizing characteristic electrical properties of the epidermal skin layers to detect fake fingers in biometric fingerprint systems a pilot study. *Biomedical Engineering, IEEE Transactions on*, 54(5):891–894, 2007.

[27] Imad Y. Hoballah and Pramod K. Varshney. An information theoretic approach to the distributed detection problem. *IEEE Transactions on Information Theory*, 35(5):988–994, 1989.